

RDF-AI: an Architecture for RDF Datasets Matching, Fusion and Interlink

François Scharffe

Semantic Technology Institute, University of Innsbruck, Austria
francois.scharffe@uibk.ac.at

Yanbin Liu, Chunguang Zhou

College of Computer Science and Technology, Jilin University, China

Abstract

With the recent publication of large quantities of RDF data, the Semantic Web now allows concrete applications to be developed. Multiple datasets are effectively published according to the linked-data principles. Integrating these datasets through interlink or fusion is needed in order to assure interoperability between the resources composing them. There is thus a growing need for tools providing datasets management. We present in this paper RDF-AI, a framework and a tool for managing the integration of RDF datasets. The framework includes five modules for pre-processing, matching, fusing, interlinking and post-processing datasets. The framework implementation results in a tool providing RDF datasets integration functionalities in a linked-data context. Evaluation of RDF-AI on existing datasets shows promising results towards a Semantic Web aware datasets integration tool.

1 Introduction

The Semantic Web is an evolution of the Web allowing machines to process data. Its foundations lies in the availability of structured data described using ontologies. Web datasets are structured data sources following the Semantic Web standards, and maintained by a single entity. Different datasets managed by different entities may offer similar contents. For example two datasets containing musical data, Musicbrainz¹ and Jamendo² overlap deeply. Many of the resources they describe refer to the same real-world objects. Overlaps between datasets will become usual as more and more Web datasets are published.³

On of the fundaments of the Semantic Web is the use of Uniform Resource Identifiers (URIs) to identify objects. The use of URIs assures that real world objects

can be identified and referred to unambiguously. If many Web datasets describe resources using different URI schemes there is a need to indicate that two resources refer to the same real world object, even though they have a different URI. The following three approaches can be considered, ordered by increasing distributivity. **Merging datasets** together in order to have a unique URI assignment scheme. While it is not feasible at Web scale, merging datasets can be in some cases useful. We also consider this approach in the system described in this paper. **URIs equivalence servers** provide centrally maintained lists of equivalent resources. This approach is followed in [Bouquet *et al.*, 2008]. **Equivalence lists attached to datasets** are published by the datasets maintainers. Each dataset refer in this approach to other datasets containing similar resources. This approach is followed in [Jaffri *et al.*,].

In each of the aforementioned approach, equivalences between resources need to be given in order to either fusion datasets or build the equivalence lists. Given the large size of some datasets, it is not realistic to consider constructing resources equivalences manually. A more reasonable approach consists in using a matcher to automatically detect them. We propose in this paper RDF-AI, a framework and a tool for automatically matching RDF datasets.

RDF-AI takes in input two datasets and generates in output either a new dataset resulting from the fusion of the two input datasets, or a list of correspondences between equivalent resources of the two datasets. RDF-AI architecture is modular, allowing to use any matching algorithm able to take RDF graphs as an input and to output alignments specified in the ontology alignment format.⁴

The contributions of this paper are as follows:

- An architecture for matching Web datasets
- A tool, RDF-AI, implementing this architecture
- A new resources matching algorithm

This paper is organized as follows. In Section 2, we overview related existing works. In Section 3, we present the system architecture and detail in Section 4 RDF-AI

¹<http://www.musicbrainz.org>

²<http://www.jamendo.com>

³The linked datasets cloud gives an idea of the number of datasets available <http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

⁴<http://alignapi.gforge.inria.fr/format.html>

implementation and algorithms. In section 5, we test and evaluate the system on two pairs of datasets. Finally, in Section 6 we conclude and present new perspectives opened by the work presented in this paper.

2 Related work

We have a look in this section over related approaches and techniques related to the integration of RDF datasets. Our study is organized along two axes. We first overview the various approaches developed for similar problems, and then see which techniques are used to solve these problems.

Detecting the similarity between records inside and between relational data-bases is a well studied area. Two similar problems can be distinguished: a set of database records is analysed to detect duplicates in order to perform data cleansing; two sets of database records are analysed to detect similar records and perform record linkage. Record linkage can then be used to perform databases fusion. These areas have been largely studied both theoretically [Fellegi and Sunter, 1969], and technically (see [Elmagarmid *et al.*, 2007; Winkler, 2006] for recent surveys).

Matching RDF datasets is also closely related to the well studied area of ontology matching [Euzenat and Shvaiko, 2007]. Matching ontologies includes matching instances, but in this case only instances from a specific ontology are generally considered. RDF datasets matching on the contrary deals most of the time with datasets described using many ontologies.

Another area can be distinguished under the terms of identity recognition, instance unification or URI equivalence mining. Two approaches have been recently considered for the management of URI equivalence: The OKKAM project [Bouquet *et al.*, 2008] tries to tackle the issue by proposing Entities Name Servers that act as resources directories around common identifiers. Each real world entity is provided with a specific identifier, which is then linked to its various URIs. The approach in [Jaffri *et al.*, 2008] uses Consistent Reference Services finding equivalent URIs using equivalence lists assigned to every datasets. In both approaches there is a need for a system such as the one we present in this paper identifying equivalent resources.

These three areas make use of various techniques to perform the matching. String comparison techniques are necessary in all cases, sometimes using fuzzy methods [Chaudhuri *et al.*, 2003]. We perform string matching in RDF-AI using a sequence alignment algorithm based on dynamic programming [Rivasa and R.Eddy, 1999]. Other techniques relevant to the particular problem of RDF datasets matching are described below.

Specificities of matching datasets in a linked-data environment need to be considered. The distributed nature of resources makes data not necessarily known at the time of matching. Evidence must be acquired on-demand by dereferencing resources URIs, or by querying for their description using a SPARQL *Describe* query. A

recent work on Web data interlinking and fusion [Raimond *et al.*,] exploits the graph and the Web based nature of Web datasets. The matching evidence is propagated through other resources via object properties, in a similar manner to the similarity flooding algorithm proposed in [Melnik, 2002]. This approach also tackles the problem of having large datasets available behind a SPARQL endpoint: a solution to reduce the size of the matching space by using an external query service is proposed.

Another set of techniques perform equivalence mining using ontology axioms [Hogan *et al.*, 2007; Saïs *et al.*, 2007; Nikolov *et al.*,]. In [Hogan *et al.*, 2007] inverse functional properties are used to find out about resources equivalence. Two resources are declared equivalent if they are both subject of an inverse functional property which range to the same object. The L2R method [Saïs *et al.*, 2007] uses a purely logic based approach using a set of predefined string equivalence logical facts, which are then combined with ontology axioms in order to deduce new facts about resources equivalence. A forward chaining algorithm is then used to propagate similarities. This method was recently combined with a numerical approach using string matching techniques [Saïs *et al.*, 2008]. In the Knofuss architecture [Nikolov *et al.*,], ontologies are used to perform a consistency checking on the dataset resulting from a fusion process.

Before describing in Section 4 the details of the matching algorithm, we present in the next section an architecture for Web datasets matching, fusion and interlink.

3 System architecture

RDF-AI architecture is composed of five independent modules allowing to pre-process, match, fusion, interlink and post-process RDF datasets. Inter-modules communication is realized using standard representation formalisms. The architecture overview picture is not given here for space reasons, it is available at <http://www.scharffe.fr/pub/ir-kr-2009/rdf-ai-architecture.pdf>

The pre-processing module performs operations on the datasets in order to prepare them for the process. The matching module takes two datasets as an input and returns an alignment between them. The interlinking module takes an alignment in input and returns a graph containing a set of *owl:sameAs* statements between resources of the two input graphs. The fusion module takes an alignment in input and returns a graph containing a new dataset resulting of merging the two input graphs. The post-processing module takes in input a graph resulting from the fusion module, check its consistency and process it for publication. Each module input includes a set of parameters given by the user.

Modules being independent from each other they can easily be interchanged. This feature is particularly important for allowing to use various matchers. We present in the following each module in detail.

Preprocessing This module is concerned with preprocessing operations preparing the source datasets for the matching process. Inputs are G_1 and G_2 the datasets to be matched and a set of parameters. Outputs are G'_1 and G'_2 the processed input graphs and a report.

The list of possible operations for this module is given below. The user selects which operation needs to be performed using the module input parameters.

Checking The module checks if the input datasets are consistent with regard to their ontologies. It also checks that every resource is typed. This phase might trigger other operations if checks fail.

Materialization This operation consists in materializing RDF triples. For example materialization of inverse or transitive properties.

Translation This operation consists in translating given properties from one language to another.

Ontology evolution This operation consists in adapting a dataset to the other in the case one ontology is a more recent version of the ontology used for the other dataset.

Properties transformations These operations consist in modifying properties values of one or both datasets to prepare them for the matching process. For example, our system implements a foaf:name transformations changing “lastname, firstname” into “firstname lastname”.

The pre-processing module output two datasets corresponding to the two input datasets modified by the module operations. The new datasets are then passed to the matching module that will be used to detect similarities between their datasources.

Matching The matching module takes in input two datasets and returns an alignment between their resources. Inputs are G'_1 and G'_2 the two datasets resulting from the preprocessing step. Parameters are given according to the matching system. They are used to tune the system by indicating properties relevance in the matching process. The output of the matching module is an alignment given in the alignment format [Euzenat, 2004] extended to represent more expressive correspondences [Euzenat *et al.*, 2007]. An alignment is represented as a set of *cells* containing correspondences between the resources of the two input datasets. The *measure* property of a cell indicates the degree of confidence the matcher gives to the correspondence between these two resources. The alignment format is the standard format for representing matching algorithms output in the ontology alignment evaluation initiative⁵. Using this format allows various matchers to be utilized in RDF-AI. The system then process the alignment in order to either interlink or fuse the datasets. These operations are described in the following two sections.

⁵OAEI: <http://oei.ontologymatching.org>

Interlink The interlink module takes in input an alignment between two datasets and outputs a named graph containing a set of interlinking primitives. Input is an alignment, which format was described in Section 3. The parameter of the module is a threshold above which a link will be created between two resources in the alignment. This threshold is compared to the *measure* property of the alignment. The output of the interlinking process is named graph containing a set of links between equivalent resources using the *owl:sameAs* property. The graph is described using the TriG syntax⁶, as well as the Void vocabulary for describing Web datasets, and properties from the ontology alignment vocabulary⁷.

A named graph containing a set of interlinks is typed as a *void:Linkset*. Each linkset is given two *void:target* properties referring to the datasets interlinked in this linkset. The *align:fromAlignment* and *align:threshold* properties refer to the alignment from which the linkset is generated and the threshold used during the generation. The linkset can then be used in linked-data applications.

Fusion The fusion module takes in input the alignment and the two original datasets and returns a new dataset corresponding to the result of merging them. The module takes an alignment as input, as well as parameters. The set of parameters allow the user to control how the fusion is performed. A *source dataset* and an *extension dataset* are given: the source dataset will be extended with properties that the extension dataset does not include. Properties appearing in both datasets are either fused or duplicated according to the user configuration. The output of the fusion module is a new graph resulting from this fusion process.

We will describe the detail of the fusion module implementation in RDF-AI in Section 4. Further processing of the the resulting graph is performed in the post-processing module.

Post processing This module is concerned with processing of the linkset or the fused graph. It checks inconsistencies that may appear as a result of the fusion, for example breaking an ontology axiom. Input is a dataset G_3 , and a report is generated in output indicating the results of checking the inconsistencies in the dataset.

We describe next in Section 4 the implementation of the architecture presented in this section.

4 Implementation

In this section, we detail the implementation of RDF-AI on an illustrating example which involves the preprocessing, matching, interlinking, fusion and output phases. We run the system on two datasets describing J.S. Bach⁸ musical compositions and works.

⁶<http://www4.wiwiw.fu-berlin.de/bizer/TriG>

⁷<http://semanticweb.org/wiki/VoiD>, <http://www.omwg.org/TR/d7/>

⁸<http://en.wikipedia.org/wiki/Bach>

Dataset 1 Dataset 2

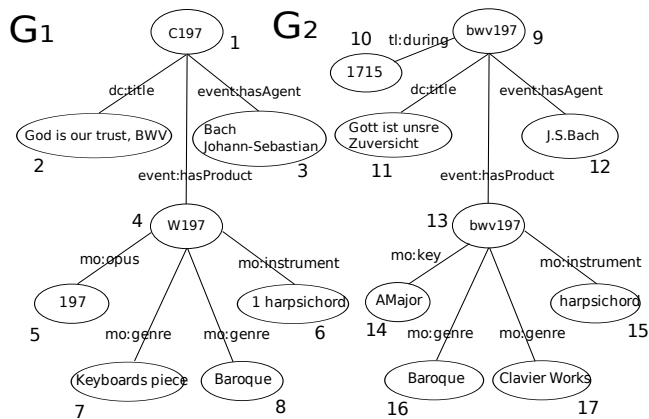


Figure 1: Example datasets

Preprocessing In the preprocessing module, RDF-AI integrates functionalities mentioned in Section 3 of this paper. RDF-AI uses the Jena framework⁹ to load the ontologies and RDF files. The preprocessing step includes the following operations:

1. Translation function: the system translates literal content for the properties given in the parameters. The implementation of this translation function is performed using the Google Translate API.¹⁰
2. Name reordering: RDF-AI can automatically adjust family and given names order. In the implementation, we cache a list of surnames¹¹ and use this data to harmonize names. In our example, “Bach Johann Sebastian” will be reordered to “Johann Sebastian Bach”.

Consider the two datasets represented in Figure 1. Each box corresponds to a resource, each ellipse to a literal, and edges to properties. The parameters for the preprocessing step are shown in the following code snippet, where the first parameter named “translation” makes RDF-AI translate the German literal content into English for the *dc:title* property in G_2 , and the second parameter automatically adjusts the person name to the format “given name, family name” in G_1 dataset.

After this step, we obtain the modified graphs G'_1 and G'_2 , nodes 3 and 11 were modified according by the pre-processing operations. This step homogenize the datasets in order to optimize the efficiency of the matching algorithm described below.

Detailed matching RDF-AI allows to reduce the number of resources to compare during the matching

phase. An initial query is used to return only those resources having a certain property value. In the Bach example, the matching space can be reduced to only those compositions having the same value for the “*tl:during*” property. This approach makes the fusion of large-scale datasets more efficient.

The matcher computes similarity values for resources in the datasets graphs according to the similarity of their comparable properties. RDF-AI actually includes two similarity computation algorithms to be selected by the user: a fuzzy string matching algorithm based on the sequence alignment algorithm [Rivasa and R.Eddy, 1999] and a word relations algorithm. There are two implementations to the latter: a synonyms comparison algorithm based on WordNet [Fellbaum, 1998] and a taxonomical similarity algorithm based on SKOS.¹² These two algorithms can be used in combination in the case more evidence is necessary to compute the similarity values.

Continuing the example, giving the graphs G'_1 and G'_2 , we compute the similarity values for all possible pairs. Because the example graph is small, we do not need to use the initial SPARQL query reducing the matching space. The parameters are shown in the file. The “*string comparison*” value of the “*method*” field denotes that this property is compared using the string comparison algorithm, “*SKOS*” is the taxonomical similarity algorithm based on SKOS and “*WordNet*” the synonym comparison algorithm based on WordNet.

The algorithm enumerates every possible matching properties of (R_1, R_2) with the similarity value computed by the similarity matching algorithm. The normalized values of the string comparison algorithm are obtained by dividing each value by the maximal absolute value. The returned value of the word relation algorithm belongs to $[0, 1]$, it differs from the string comparison algorithm because it is unsuited to quantitatively compute the semantic dissimilarity of two words. However, this difference will not affect the results.

Next, the algorithm selects the candidate property pairs to compute the resources similarity according to the normalized value. In the example, (2, 11), (3, 12), (4, 13), (6, 15), (7, 17) and (8, 16) are selected as their values are the most relevant for the graph similarity computation. The algorithm selects (7, 17) and (8, 16) for *mo:genre* because $S^0(8, 16)$ is the biggest in the *mo:genre* set $\{S^0(7, 16), S^0(7, 17), S^0(8, 16), S^0(8, 17)\}$. It will be selected at first, and then all others matching pairs including 8 or 16 will be marked as invalid. The second valid candidate is thus $S^0(7, 17)$.

Then, the algorithm computes the similarity value of the resources (R_1, R_2) . This method considers the co-affectation of resources at different levels:

$$S(4, 13) = (S^0(4, 13) + S^0(6, 15) + S^0(7, 17) + S^0(8, 16))/4 = 0.625$$

$$S(1, 9) = (S^0(2, 11) + S^0(3, 12) + S(4, 13))/3 = 0.732$$

$$S(R_1, R_2) = S(1, 9) = 0.732$$

⁹<http://jena.sourceforge.net>

¹⁰<http://code.google.com/apis/ajaxlanguage/>

¹¹Obtained from <http://www.surnamefinder.com>

¹²<http://www.w3.org/2004/02/skos>

Finally, the algorithm selects the the highest similarity measure and include it in the alignment, it is (R_1, R_2) in this example.

The overall space complexity of the matching algorithm is in $O(n)$, for n resources. It corresponds to the size occupied to store the dataset graph. The time complexity is in $O(n^2)$ in the worst case.

In the rest of the process, RDF-AI uses the alignment output in order to either generate a linkset or fuse the two datasets.

Interlinking and fusion The interlink module generates a linkset as a named graph including a set of *owl:sameAs* statement, according to the input alignment. The only parameter here is the threshold, set by the user, which is used to trigger the correspondences in the alignment to be included in the linkset. The correspondence is outputted as an *owl:sameAs* triple if its *measure* property value is bigger than the threshold, otherwise, this pair will not appear in the output graph. In this example, we set the threshold 0.5, and the *measure* statement value of (R_1, R_2) is 0.732, it will thus appear in the output linkset. Here, The URI of the matched resource of “c197” expressed by R_1 in Figure 1 is <http://bach1.example.org/composition/composition-197>, and R_2 is

<http://bach2.example.org/composition/composition-bwv197>.

The fusion process fuses datasets according to the alignment, the original datasets graphs and a set of user parameters. It outputs the fused graph as a new dataset.

In this example, G_1 is the source dataset and G_2 is the extension dataset, the *namespaces* in the *add* node are the properties added to the source dataset, and the *namespace* in the *merge* node are the property merged into the source dataset from the extension dataset. In this case, the merged property value is kept from the source graph. The fused graph is the final result, the example output is shown Figure 2. Node 10, 11 and 14 are added, node 17 is merged to the *mo:genre* property, node 16 is ignored in Figure 1, because its property has same value with the node 8.

In order to evaluate the quality of the matches returned by RDF-AI, we have evaluated the tool on two pairs of overlapping datasets. We present the results of this evaluation next in Section 5.

5 Experimental results

In order to evaluate the performance of RDF-AI in matching and fusing RDF datasets, we have tested the system on the following dataset pairs from two different domains:

1. AKT EPrints archive and Rexa datasets¹³.
2. The works of Johann Sebastian Bach in two different datasets¹⁴.

¹³ <http://eprints.aktors.org>, <http://www.rexa.info>

¹⁴ <http://www.scharffe.fr/pub/ir-kr-2009/>

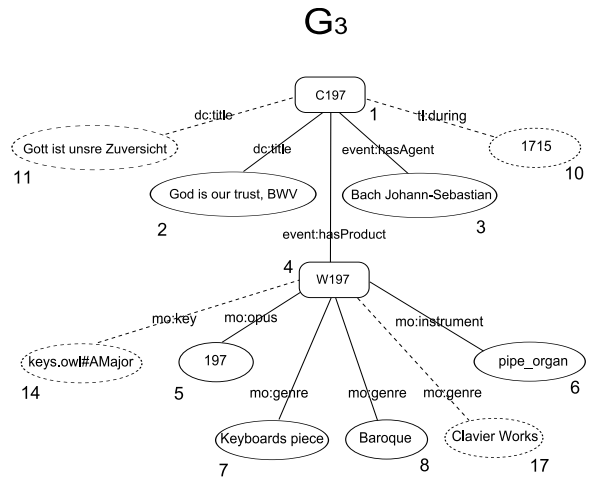


Figure 2: Fused graph

EPrints and Rexa are both described using the same ontologies. Authors are described using FOAF, publications are described using the Opus ontology¹⁵. The EPrints dataset contains 314 resources and Rexa 2103 resources. RDF-AI loads them and run according to the configuration file specifying the resources to match and the operation to be performed: matching, fusion, interlink. RDF-AI obtains on these datasets a precision of 95.9% which is higher than the one obtained by the KnoFuss architecture (92%) on the same datasets [Nikolov *et al.*,]. The optimal configuration uses the name ordering function in the preprocessing module, compared with the 92% not using the function, the precision is much improved.

Two datasets on Bach musical works datasets are mainly described using the Music ontology as well as the timeline and events ontologies (See Section 1). In this experiment, the system matches resources of the composition class, which contains 771 resources in the first dataset, and 800 of the second. The optimal precision is 97.5%. The optimal configuration uses the translation function for the *dc:title* property, from German into English. If we do not use this function, the precision drops to 87.3%. The change is significant, and is easily understandable as the title is the most important property to distinguish the different objects. Without the translation, there are significantly more underived resources, the reason is that the similarity value becomes less than the fixed threshold. If we replace WordNet with a string comparison algorithm for the *mo:genre* computation, the precision only drops from 0.1 point. This small improvement is due to the fact that the evidence gained by computing other properties is enough to find the matches. However, the word relation algorithm significantly improves the similarity value in the alignment through the *measure* property of correspondences.

¹⁵ http://knoesis.wright.edu/library/ontologies/\swetodblp/august2007/opus_august2007.rdf

This evaluation of the system shows promising results for matching Web datasets. However, the system would need to be evaluated on larger datasets that cannot be entirely loaded at runtime. We discuss this issue together with others and conclude in the following section.

6 Conclusion and future work

We have presented in this paper RDF-AI, an architecture and a tool tackling part of the fundamental problem of Web datasets interoperability. The architecture provides the basic workflow and specifies data exchange formats at every steps of the matching, interlinking and fusion process. RDF-AI was successfully evaluated on matching two pairs of Web datasets. User input is required at each step in order to configure the tool optimally. We are actually improving it so that the user input is reduced to the minimum. We particularly plan to work on an algorithm automatically acquiring the datasets structure. We are currently implementing some of the missing functionalities of RDF-AI: usage of ontology axioms in order to derive new matches, and consistency checking of the output of the fusion process.

Acknowledgment

This research works are supported by European Commission under grant No.TH /Asia Link/010 (111084) and under the EASAIER project (EU-FP6-IST-033902).

References

- [Bouquet *et al.*, 2008] Paolo Bouquet, Heiko Stoermer, and Barbara Bazzanella. An Entity Naming System for the Semantic Web. In *Proceedings of the 5th European Semantic Web Conference (ESWC2008)*, LNCS, June 2008.
- [Chaudhuri *et al.*, 2003] Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti, and Rajeev Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 313–324, New York, NY, USA, 2003. ACM.
- [Elmagarmid *et al.*, 2007] A.K. Elmagarmid, P.G. Ipeirotis, and V.S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, January 2007.
- [Euzenat and Shvaiko, 2007] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag, Heidelberg (DE), 2007.
- [Euzenat *et al.*, 2007] Jérôme Euzenat, François Scharffe, and Antoine Zimmermann. D2.2.10: Expressive alignment language and implementation. Project deliverable 2.2.10, Knowledge Web NoE (FP6-507482), 2007.
- [Euzenat, 2004] Jérôme Euzenat. An API for Ontology Alignment. In Frank van Harmelen, Sheila McIlraith, and Dimitri Plexousakis, editors, *The Semantic Web - ISWC 2004: Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004. Proceedings*, volume 3298, pages 698–712. Springer, 2004.
- [Fellbaum, 1998] Christiane Fellbaum. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May 1998.
- [Fellegi and Sunter, 1969] Ivan P. Fellegi and Alan B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.
- [Hogan *et al.*, 2007] Aidan Hogan, Andreas Harth, and Stefan Decker. Performing object consolidation on the semantic web data graph. In *In Proceedings of 1st I3: Identity, Identifiers, Identification Workshop*, 2007.
- [Jaffri *et al.*,] Afraz Jaffri, Hugh Glaser, and Ian Millard. Uri disambiguation in the context of linked data. In *Proceedings of the Linking Data On the Web workshop at WWW'2008*.
- [Jaffri *et al.*, 2008] Afraz Jaffri, Hugh Glaser, and Ian Millard. Managing uri synonymy to enable consistent reference on the semantic web. In *IRSW2008 - Identity and Reference on the Semantic Web 2008 at ESWC*, 2008.
- [Melnik, 2002] Sergey Melnik. Similarity flooding: a versatile graph matching algorithm. In *Proc. 18th International Conference on Data Engineering (ICDE)*, San Jose (CA US), 2002.
- [Nikolov *et al.*,] Andriy Nikolov, Victoria Uren, Enrico Motta, and Anne de Roeck. Handling instance coreferencing in the knofuss architecture. In *Proceedings of the workshop: Identity and Reference on the Semantic Web at 5th European Semantic Web Conference (ESWC 2008)*.
- [Raimond *et al.*,] Yves Raimond, Christopher Sutton, and Mark Sandler. Automatic interlinking of music datasets on the semantic web. In *Proceedings of the Linking Data On the Web workshop at WWW'2008*.
- [Rivasa and R.Eddy, 1999] Elena Rivasa and Sean R.Eddy. A dynamic programming algorithm for rna structure prediction including pseudoknots. *Journal of Molecular Biology*, 285:2053–2068, 1999.
- [Saïs *et al.*, 2007] Fatiha Saïs, Nathalie Pernelle, and Marie-Christine Rousset. L2r: A logical method for reference reconciliation. In *AAAI*, pages 329–334, 2007.
- [Saïs *et al.*, 2008] Fatiha Saïs, Nathalie Pernelle, and Marie-Christine Rousset. Combining a logical and a numerical method for data reconciliation. *Journal of Data Semantics*, 12, 2008.
- [Winkler, 2006] W.E Winkler. Overview of record linkage and current research directions. Technical Report 2006-2, Statistical Research Division. U.S. Census Bureau, 2006.