

Towards Practical RDF Datasets Fusion

Yanbin Liu¹, François Scharffe², and Chunguang Zhou¹

¹ College of computer science and technology,
Jilin University, China,

² Semantic Technology Institute,
University of Innsbruck, Austria
`francois.scharffe@uibk.ac.at`

Abstract. In this paper, we describe our ongoing work on RDF datasets fusion. We first define what RDF datasets fusion is and list some key challenges around this issue. Based on the problems analysis, we present ways towards solutions for each problem. We detail an experimental fusion algorithm, which takes into account both the similarity of literal contents and the graph structure of the dataset, and test this algorithm on small-scale RDF datasets. Our approach gives the user an important role for defining the input of the algorithm. At last, this paper lists the remaining issues and future works to achieve a practical RDF datasets fusion system.

1 Introduction

The Semantic Web is an evolution of the Web allowing machines to process data. Its foundation lies in the availability of good quality structured data, represented as RDF datasets. Different datasets managed by different organizations may offer the same contents, for example, the Musicbrainz³ and Jamendo⁴ datasets overlap deeply, as they both describe musical data.

Many linked-data datasets are built from RDF mappings of existing relational databases, with interlinks added to other datasets.⁵ For example, locations in the Jamendo dataset are linked to the Geoname dataset.⁶ However, as stated in [1], the source databases might not be sufficiently complete and also contain errors. This is not acceptable in cases where quality data is needed, for example in the case of a commercial applications.

One solution might be to fusion information from many data sources in order to constitute a richer dataset. Alternative to interlinking solutions which let datasets independent and provide links between similar entities, the fusion of two datasets results in the creation of a new enriched dataset. This approach can

³ <http://www.musicbrainz.org>

⁴ <http://www.jamendo.com>

⁵ An up-to-date overview of the linked datasets cloud is available at <http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

⁶ <http://www.geonames.org/ontology>

be used to enrich a existing reference dataset with some other smaller datasets made available through GRDDL⁷ transformations, for example.

We believe our approach share with the interlinking approach a high dependency to the domains to be linked. Said differently, a fusion algorithm performing well on integrating two given datasets as high chances to perform bad on two other datasets from another domain. Considering two existing works [2, 3], we can observe that each of them does particular assumptions resulting on the use of domain specific heuristics to interlink the datasets. In order to be able to design a system performing well on various domains, we propose to place the user at the center of the fusion task, allowing her to specify inputs of the fusion algorithm.

This paper is organized as follows. In Section 2, we define more formally the fusion problem, outline the challenges it poses, and draw the differences with interlinking based approaches. In Section 3, we present our first steps towards a RDF dataset fusion system. Finally, in Section 5 we see what are the remaining challenges towards a practical system.

2 Fusion problem and challenges

Before detailing the solution, we first define the fusion problem and outline the challenges it poses.

2.1 Problem definition

We consider two RDF datasets G_1 and G_2 respectively. Their resources are described by the same one ontology o . G_1 and G_2 describe many resources, represented by r_i^1 and r_i^2 . The RDF dataset fusion definition is as follows:

Definition 1. *The fusion problem is to find and merge the pairs (r_i^1, r_i^2) iff r_i^1 in G_1 and r_i^2 in G_2 describe the same real world object. If we add the data of r_i^2 to r_i^1 , G_1 is called the source dataset and G_2 the extension dataset.*

For example, we want to find that $r_i^1 = \text{http://data.linkedmdb.org/page/film/133}$ to $r_i^2 = \text{http://dbpedia.org/page/Chinatown-\%28film\%29}$, actually describe the same movie. The fusion problem is simplified as merged resource graphs G_1 and G_2 are based on a same ontology. This means that after the fusion procedure, each resource has the same structure and the node is marked by the literal content or URI according to the ontology. A fusion algorithm computes the similarity of any two resources in G_1 and G_2 and select the most similar resource pairs to merge.

We suppose that the two datasets are available locally. In the linked data context, it can be expected that links to external datasets are included, leading to missing information in the local cache. We address this issue together with other challenges in Section 2.2 below.

⁷ <http://www.w3.org/2004/01/rdxh/spec>

2.2 Challenge and solution

The fusion problem can be partly solved using the results of works in other related areas (see Section 4) but has its own characteristics. We list below the set of potential challenges arising in order to design a practical general-purpose RDF dataset fusion system. We associate when possible a solution to each challenge.

1. Generally, the data scale is very large in RDF datasets, comparing a resource in the source dataset with the whole resources in the extension dataset might thus lead to tractability issues. Heuristics can be defined to reduce the problem complexity. For example in [2] SPARQL and keyword queries are used to get a resources list from the extension dataset. The algorithm just need to compare the source resource with the resources in the return list.
2. After extracting literal contents, we need a fuzzy string comparison algorithm to support the matching task. Rigorous string comparison algorithms cannot find “Johann Sebastian Bach” and “J.S.Bach” are the same person. We designed and implemented a sequence alignment algorithm to solve this problem. Sequence alignment is a powerful string comparison technique using dynamic programming to produce global alignments, it has been successfully used to find the similarity of DNA sequences [4]. Here, we can use the return values to represent the similarity of two strings. The algorithm description is as follows:

Given two sequences, such as a and b , a_i and b_i mean the i -th character in a and b respectively. Construct a score matrix S , in which

$$S(i, j) = \text{MAX} \begin{cases} S(i-1, j) + g \\ S(i, j-1) + g \\ S(i-1, j-1) + p(i, j) \end{cases},$$

$$\text{here, } p(i, j) = \begin{cases} c & \text{if } a_i = b_i \\ -c & \text{else} \end{cases},$$

g and c are parameters. $S(i, j)$ stands for the similarity of a 's substring from the beginning to i -th character and b 's substring from beginning to j -th character. The last value of matrix S is the similarity of string a and b .

3. RDF datasets can be in different languages while describing similar resources. For example the dataset <http://www.scharffe.fr/pub/dist2008/bach-1.rdf.xml> in English and <http://www.scharffe.fr/pub/dist2008/bach-2.rdf.xml> in German are about the same resource, they contains musical works from J.S. Bach.⁸ If we want to fusion these two datasets, we need to solve the language problem at first. We plan use the Google Translate API⁹ on this matter.
4. Similar with the previous item, some matched literal contents cannot be detected by the string comparison algorithm, while they are semantically equivalent. In the example illustrated Figure 1, a genre of the musical work in one dataset is “Clavier Work”, while it is “Piece for other keyboards” in the

⁸ http://en.wikipedia.org/wiki/Johann_Sebastian_Bach

⁹ <http://code.google.com/p/google-api-translate-java>

other. Here a string similarity algorithm does not help. An external thesaurus such as WordNet [5] will allow to identify that “Work” and “Piece” are near synonyms in a music context. The same for “Clavier” and “Keyboards”.

5. In some cases, neither string comparison nor lookup in an external thesaurus helps to detect that two properties are similar. In the case of object properties, or data properties referencing to a structured vocabulary like SKOS,¹⁰ It may happen that an object at a different position in the taxonomy hierarchy was assigned to the property. For example, a music piece in one dataset has the genre “Vocal Music” while it has the genre “Air” in the other dataset. Checking a music genres taxonomy reveals that the first is a direct parent of the second in the hierarchy. The taxonomy can be used to compute a similarity according to the distance between the compared objects.
6. Another issue arises with object properties that cannot directly be compared because they reference an external dataset. Objects usually cannot be compared based on their URIs only. A lookup has thus to be performed to find more informations on the objects. This can be done by dereferencing the object URI or performing a SPARQL *DESCRIBE* query.¹¹
7. The fusion task might vary considerably depending on the datasets domain. A naive approach has to compute a combined similarity on the resources properties, leading to an imprecise and computationally demanding procedure. Letting the user tuning the fusion algorithm in order to select relevant properties and eventually to assign external resources for a specific fusion task will considerably improve the procedure. In the example illustrated Figure 1, the two objects could easily be matched on the “opus” property, as it is a unique identifier for a musical work. The procedure performance would thus be greatly improved if the user was able to indicate this fact.

Next in Section 3, we study an example illustrating the challenges introduced in this section and detail our solution algorithm based on this example.

3 Initial experiment

In this section, we present the input, output and detail of the fusion algorithm. We perform the fusion of two datasets describing J.S. Bach compositions and works. At the time of writing, we are able to select the properties relevant for the similarity computation and compute a global similarity for two given objects.

A reasonable approach to find similarities between the musical composition objects in the example datasets is to compare the compositions URIs fragments and title properties. If these comparison values are similar enough, the algorithm detects a resource pair and finally fusion the objects. We give in the following part the formal specification of the algorithm. Considering the example illustrated Figure 1, we try to find out similarities between the musical compositions pair (G_1, G_2) in order to merge the two datasets.

¹⁰ <http://www.w3.org/2004/02/skos/>

¹¹ <http://www.w3.org/TR/rdf-sparql-query/#describe>

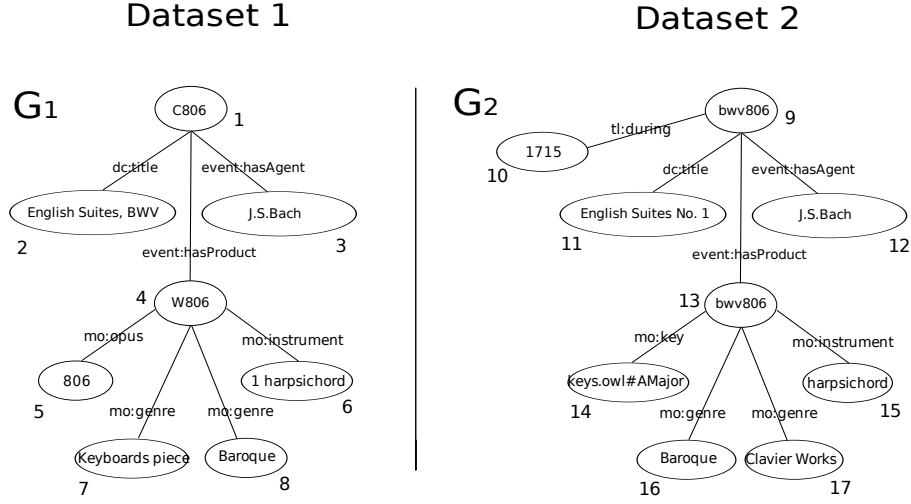


Fig. 1. Example datasets

3.1 Input and output

The input of fusion algorithm takes as input two datasets described under a same ontology and the user input. The user input is a set of heuristics improving the algorithm performance: the set of properties to be considered during the matching process, and external resources to be used to compare certain properties. The first dataset will be considered as the source dataset and the second as the extension dataset that will be used to extend the source dataset as the result of the fusion operation. The formal description is as follows:

$$INPUT = \{Source\ G_1(R_1), Extension\ G_2(R_2), User\ input\}$$

R_1 and R_2 being the set of resources in datasets G_1 and G_2 respectively.

The algorithm creates the resulting graph according to the input datasets. After finding all similar resources pairs $(R_1(x), R_2(y))$, the algorithm merges them into a new graph G_{1+} being an extension of the source graph G_1 where each matched resource R_1 is enriched with properties p_i of R_2 not included in R_1 and such that the similarity between $R_1(x)$ and $R_2(y)$ is greater than a given threshold t . When a resource matches more than once in the other graph, only the resources pair having the highest similarity is fused. Formally, the output of the fusion operation can be described as follows

$$OUTPUT : G_{1+} = \{G_1 \cup P/p_i \in P \text{ iff } p \in R_x \wedge p \notin R_1 \wedge sim(R_1, R_x) > t \\ \wedge \forall n | sim(R_1, R_n) > t, x = max(sim(R_1, R_n))\}$$

This approach is rather simplistic, a more sophisticated approach would give the user the possibility to choose the strategy to adopt when constructing the

fusioned graph: which properties should be included, what to do when a property appears more than once, etc.

We will next describe the algorithm we have implemented so far.

3.2 Fusion algorithm

Considering the two datasets in Figure 1, the algorithm models the graph G_1 and G_2 at first, and each ellipse is a resource or a literal and the edges are the properties.

Holding the graphs, at first, we compute the similarity values for all possible pairs, for example (G_1, G_2) is one of them. The algorithm enumerates all the possible matching properties of (G_1, G_2) with the similar value computed by the string match algorithm showed in Table 1. The compared properties are *dc:title*, *event:hasAgent*, *event:hasProduct*, *mo:genre* and *mo:instrument* in the example. Our current implementation uses a naive approach where the user inputs the properties to be considered in the matching phase. The normalized values are obtained by dividing each value by the maximal absolute value. The maximal absolute value is 14 in Table 1. The range of normalized value is [-1, 1].

Table 1. Property similarity of (G_1, G_2)

Resource pair	Value (Normalized)
S(1, 9)	3 (0.214)
S(2,11)	14 (1.000)
S(3,12)	8 (0.571)
S(4,13)	3 (0.214)
S(6,15)	9 (0.643)
S(7,16)	1 (0.071)
S(7,17)	1 (0.071)
S(8,16)	7 (0.500)
S(8,17)	1 (0.071)

Next, we select the candidate property pairs to compute the graph similarity according to the value, (1, 9), (2, 11), (3, 12), (4, 13), (6, 15), (7, 17) and (8, 16) are selected to set \mathbb{L} , because their values are the most relevant for the graph similarity computation. For example, the algorithm selects (7, 17) and (8, 16) within *mo:genre* because $S(8, 16)$ is the biggest in the set $\{S(7, 16), S(7, 17), S(8, 16), S(8, 17)\}$. It will be selected at first, and then all others including 8 and 16 will be marked as invalid, so the second valid candidate is $S(7, 17)$. We assume that we have extracted the literal content corresponding to the URI fragment of each resource, so we can compare the similarity of two resources directly. Here, the similarity of (7, 17) should be computed using an external thesaurus such as Wordnet, this point has been referred in challenge section, our system has not integrated this part now.

And then, we compute the similar value of (G_1, G_2) iteratively. In i -th iteration,

$$S^i(G_1, G_2) = (S^i(1, 9) + S^i(2, 11) + S^i(3, 12) + S^i(4, 13) + S^i(6, 15) + S^i(7, 17) + S^i(8, 16))/7.$$

$$S^{i+1}(1, 9) = S^i(1, 9) + (S^i(2, 11) + S^i(3, 12) + S^i(4, 13))/3.$$

$S^{i+1}(4, 13) = S^i(1, 9) + S^i(4, 13)$ (The first time), (2, 11) and (3, 12) are computed by the same way.

$$S^{i+1}(4, 13) = S^i(4, 13) + (S^i(6, 15) + S^i(7, 17) + S^i(8, 16))/3 \text{ (The second time)}$$

$S^{i+1}(6, 15) = S^i(4, 13) + S^i(6, 15)$, (7, 17) and (8, 16) are computed by the same way. at the end of each iteration, we normalize the new similar values. The iteration will stop if $S(G_1, G_2)$ has not change or it has finished the run times. This process is similar as the one described in [6]. The iteration operation takes into account the information of the graph structure, but it often occurs that the dataset graph structure is simple. The example shown here is itsel too simple to show the advantage of this approach. We will extend this part in future work.

Finally, we select the pairs whose similarity measure is the highest, there is only (G_1, G_2) in this example. We use a threshold to avoid matching between pairs which are dissimilar, here we set it to 0.5. According to the matched pair (G_1, G_2) , the fusion of G_1 and G_2 will be made by our approach, and merged them into a new resource G_{1+} showed in Figure 2, it is the result of our algorithm. The resource 10 and 14 are the ones from the extension dataset, which are added to the source dataset.

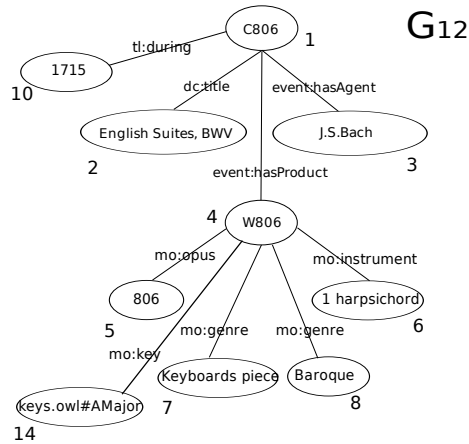


Fig. 2. Fusion result

The formal description of the matching algorithm is as follows:

Algorithm 1 Fusion

- 1: Construct the graphs of input datasets.
 - 2: Extract the suitable properties from the starting resource.
 - 3: **for all** (G_i, G_j) **do**
 - 4: Compute the set \mathbb{L} according to string match algorithm.
 - 5: **while** runs $i < \text{iterative times}$ AND $S(G_i, G_j)$ is not convergent **do**
 - 6: $S^i(G_i, G_j) = \sum_{z=1}^N S^i(\mathbb{L}_z)/N$ { \mathbb{L}_z is the z -th element of set \mathbb{L} , N is the size of \mathbb{L} .}
 - 7: **for all** $\mathbb{L}_m \in \mathbb{L}$ **do**
 - 8: $S^{i+1}(\mathbb{L}_m) = S^i(\mathbb{L}_m) + \sum_{z=1}^N S^i(\mathbb{L}_z^m)/N$
 - 9: $S^{i+1}(\mathbb{L}_z^m) = S^i(\mathbb{L}_z^m) + S^i(\mathbb{L}_m)$
 - 10: **end for**{ \mathbb{L}_m is the resource with property((1, 9) and (4, 13) in example)in \mathbb{L} , and \mathbb{L}^m is its property set. }
 - 11: Normalize set \mathbb{L} .
 - 12: **end while**
 - 13: **end for**
 - 14: Select the result pairs based on graph similar values, for example (G_1, G_2) is a pair of them.
 - 15: make and output the fusion result from the selected pair.
-

4 Related work

Research on data interlinking finds its origins in record matching techniques for relational databases. Recent surveys of this field are available in [7, 8]. Ontology-based approaches can be categorized in three different though closely related fields depending on the basic considerations and goals of the interlinking methods: ontology integration systems include tools [9, 1] to translate and fusion data described under heterogeneous ontologies. Dedicated interlinking algorithms [2, 3], and dataset fusion techniques as introduced here. While they are not directly applicable, ontology matching [10] and merging [11] are certainly to be considered.

While we consider in this paper datasets described under a unique ontology, the KnoFuss architecture [9] tackles the data interlinking problem whether or not the datasets are described under the same ontology. It is based on a generic component-based approach allowing to select the best appropriate method for a given interlinking task. In [2], Raimond *etal* consider more closely the interlinking task in a linked-data context, using an algorithm close to the similarity flooding algorithm in [6]. URI dereference is used in this approach to lookup for more information allowing to increase the confidence in entities similarity and output *owl:same_as* links. In [3], Jaffri *etal* tackle the problem of author disambiguation across DBLP and DBpedia. The Names domain is analyzed in details, and flaws of the two linked datasets identified. This paper shows as an indirect conclusion that each domain needs a particular treatment for the linkage of relevant datasets. This raises the need for customizable systems where the user can interact with the linking algorithms.

5 Conclusion and future work

In this paper, we have described the RDF dataset fusion problem. We identified the challenges corresponding to this problem and proposed solutions to tackle these challenges. We then proposed a fusion operator taking as an input two datasets to be merged and a set of user inputs used to guide the matching algorithm. We described our current fusion algorithm using a sequence matching algorithm and computing a similarity through a flooding process. The work presented in this paper is still at an early stage, but already open paths toward extensions. While we do not explicitly distinguish here between the data matching part and the fusion, a general framework would be needed to separate each part of the process in separate components. This would allow to reuse matchers for either datasets fusion or interlink. It would also be fruitful to reuse existing formalisms for the description of the results of the matching. Finally, it would be useful to keep track of the matching process in the resulting dataset. A vocabulary is needed describing the result graph and the matchers used to generate it.

In order to achieve these aforementioned goals, we plan in the short term to:

- Implement the solutions proposed in the challenges part of the paper: usage of SPARQL and keyword queries to reduce the comparison space, usage of external resources such as WordNet or SKOS vocabularies for matching terms, URI dereferencing and SPARQL *DESCRIBE* queries for resources lookup, and usage of translation tools.
- Define the user interface allowing to select relevant properties to be included in the similarity computation, and allowing to assign external resources to guide the computation of each property similarity.
- similar to the previous item, give the possibility to select the parameters of the output graph: select, merge, or duplicate properties.
- Test the system on various datasets and analyze the results.
- Develop a framework for RDF datasets fusion and interlink.

Acknowledgment

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

References

1. Nikolov, A., Uren, V., Motta, E., de Roeck, A.: 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)
2. Raimond, Y., Sutton, C., Sandler, M.: Automatic interlinking of music datasets on the semantic web. In: Proceedings of the Linking Data On the Web workshop at WWW'2008

3. Jaffri, A., Glaser, H., Millard, I.: Uri disambiguation in the context of linked data. In: Proceedings of the Linking Data On the Web workshop at WWW'2008
4. Rivasa, E., R.Eddy, S.: A dynamic programming algorithm for rna structure prediction including pseudoknots. *Journal of Molecular Biology* **285** (1999) 2053–2068
5. Fellbaum, C.: *WordNet: An Electronic Lexical Database* (Language, Speech, and Communication). The MIT Press (May 1998)
6. Melnik, S.: Similarity flooding: a versatile graph matching algorithm. In: Proc. 18th International Conference on Data Engineering (ICDE), San Jose (CA US). (2002)
7. Elmagarmid, A., Ipeirotis, P., Verykios, V.: Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering* **19**(1) (January 2007) 1–16
8. W.E, W.: Overview of record linkage and current research directions. Technical Report 2006-2, Statistical Research Division. U.S. Census Bureau (2006)
9. Thor, A., Rahm, E.: Moma - a mapping-based object matching system. In: 3rd Biennial Conference on Innovative Data Systems Research, Asilomar, California (2007)
10. Euzenat, J., Shvaiko, P.: *Ontology matching*. Springer-Verlag, Heidelberg (DE) (2007)
11. Scharffe, F.: Dynamerge: A merging algorithm for structured data integration on the web. In: DASFAA 2007 International Workshop on Scalable Web Information Integration and Service. (2007)