New Directions in Linked Data Fusion

Jan Michelfeit[1] and Jindřich Mynarz[2]

[1] Faculty of Mathematics and Physics, Charles University in Prague, Czech Rep.
michelfeit@ksi.mff.cuni.cz,
[2] University of Economics, Prague, Czech Republic

Abstract. When consuming Linked Data from multiple sources, or from a data source after deduplication of entities, conflicting, missing or outdated values must be dealt with during data fusion in order to increase the usefulness and quality of the data. In this poster, we argue that the nature of Linked Data in RDF requires a more sophisticated approach to data fusion than the current Linked Data fusion tools provide. We demonstrate where they fall short on a real case of public procurement data fusion when dealing with property dependencies and fusion of structured values, and we propose new data fusion extensions to address these problems.

Keywords: Data Fusion, Linked Data, RDF, Data Integration

1 Introduction

The value of Linked Data lies in the ability to link pieces of data. A data integration process applied to the data can provide a unified view on data and simplify the creation of Linked Data consuming applications. Nevertheless, conflicts emerge during the integration. Deduplication reveals different URIs representing the same real-world entities (identity conflicts), and conflicting values appear due to errors, missing, or outdated pieces of information (data conflicts).

Resolution of these conflicts is a task for data fusion. It *combines multiple records representing the same real-world object into a single, consistent and clean representation* [1]. In the context of Linked Data represented as RDF, real-world objects are represented as *resources*. A set of RDF triples describing a resource, a *resource description*, corresponds to a "record". Conflicts are resolved, and low-quality values purged to get a clean representation of a resource. This is typically realized by *fusion functions* such as Latest, Vote, or Average. Tools implementing Linked Data fusion include, e.g., Sieve [2], or LD-FusionTool[3] which we develop as part of the UnifiedViews ETL framework[4] [3].

In this poster, we demonstrate how errors can be introduced in the fused data when there are dependencies between RDF properties, or when fusing the common pattern of structured values (e.g., address of an entity). We propose how to deal with these cases by extending the data fusion process with the notion of *dependent properties* and *dependent resources*.

[3] https://github.com/mifeet/LD-FusionTool
[4] successor of the ODCleanStore framework, where LD-FusionTool originated

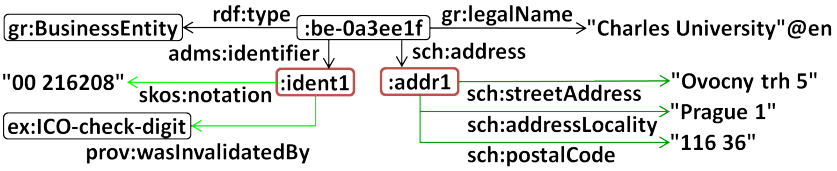

Fig. 1. Sample representation of a business entity. Red boxes denote dependent resources (structured values), green arrows denote groups of dependent properties.

2 Motivating Example

We demonstrate the need for new data fusion capabilities on a real scenario with public procurement data extracted from an XML-based API and RDFized using the UnifiedViews framework. The extracted data needs to be deduplicated and fused in order to obtain high-quality data before further analytical processing.

Fig. 1 shows how a business entity (BE) is represented in RDF. It has a legal name, address, and official identifier, which may be marked as syntactically invalid. The extracted data contains many copies of the same BE because of duplication in the source dataset. Simple merge of matched BEs would result in data conflicts due to misspellings and errors in the dataset or mismatches in the generated `owl:sameAs` links. Our goal is to fuse BEs so that each has a single legal name, address, and identifier, choosing the best possible values.

Property dependencies. We encounter the first problem with the state-of-the-art Linked Data fusion tools when fusing addresses. The tools resolve each property independently which can result in the selection of, e.g., a town from one address in the input and a postal code from another one. Such result could be incorrect, however, because the postal code is related to the town. We need to introduce dependency between properties to obtain a correct fused result.

Fusing structured values. Both address and identifier can be regarded as structured values of a BE. We will refer to the main resource (e.g., BE) as a *parent resource* and to the resource representing the structured value (e.g., address) as a *dependent resource*. Currently, structured values need to be fused separately. One way of achieving this is generating `owl:sameAs` links among structured values based on their properties, e.g., match addresses based on similarity of street, and town. This approach has two drawbacks: it doesn't guarantee that a BE will have only a single address after fusion, and the error of automatically generated `owl:sameAs` links accumulates. Another way is generating `owl:sameAs` links between dependent resources that belong to the same parent resource. This approach may lead to errors when two parent resources point to the same dependent resource, e.g., two different BEs point to the same address. All addresses for the two BEs would incorrectly be merged in such case.

We argue that a smarter approach considering structured values in resource descriptions could (1) overcome the outlined problems with the separate fusion of structured values, (2) reduce the overhead of additional linking, fusion, and validation, (3) gracefully handle blank nodes, where linking may not be practical.

3 Extending Linked Data Fusion

In this section, we propose how to extend data fusion to improve on the issues demonstrated in Section 2. Let there be a set U (RDF URI references), a set B (blank nodes) and a set L (literals). A triple $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ is an *RDF triple* and we refer to its components as subject, predicate, and object, respectively. Let $g \in U$ be a graph name. We regard a triple (s, p, o) that belongs to a *named graph* g as a *quad* (s, p, o, g).

3.1 Property Dependencies

Independent fusion of properties is not always sufficient, as we demonstrated in Section 2. What we want is to keep the values of dependent properties together in the fused result if the values occurred together in the input data. Let us call a set of input quads sharing the same subject s and graph name g before resolution of identity conflicts an input group $IG_{s,g}$. Furthermore, let $d(p_1, p_2)$ denote that there is a dependency between properties p_1 and p_2.

Definition 1. *A fused result R from input quads I* satisfies property dependencies *if and only if $\forall p_1, p_2 \in U$ such that $d(p_1, p_2)$: all quads $(s, p, o, g) \in R$ such that $p = p_1 \vee p = p_2$ are derived*[5] *from the same input group in I.*

We chose to define input groups based on subject and graph because it covers two common scenarios: (1) fusing data from multiple sources (input quads can have different graph names), and (2) fusion after deduplication of a single source (quads will have different subjects before resolution of identity conflicts).

Here is how a basic data fusion algorithm can be extended to produce results satisfying property dependencies. The input of the algorithm includes these dependencies – we assume it is given as an equivalence relation d. We also assume the input resource description contains all quads for all mutually dependent properties. The extended algorithm consists of the following high level steps:

1. Find equivalence classes $\mathcal{C}$ of the equivalence relation d.
2. For every class of dependent properties $C \in \mathcal{C}$:
   (a) Let I_C be all input quads with one of the properties in C.
   (b) For every nonempty input group $I_{s,g}$ in I_C, let $O_{s,g}$ be the fused result of the basic data fusion algorithm applied on $I_{s,g}$.
   (c) Select one set O_C from all sets $O_{s,g}$ of fused quads according to some fusion tool specific criterion and add O_C to the result.
3. Fuse input quads with properties that do not have any dependency using the basic data fusion algorithm.

It is straightforward to prove that the extended algorithm indeed produces results satisfying property dependencies. The criterion used in step (2c) can depend on the implementing fusion tool. In LD-FusionTool, which can assess the quality of fused quads, we select $O_{s,g}$ such that the average quality of the fused result is maximal.

[5] By *derived* we mean "selected from" for the so called *deciding* fusion functions such as LATEST, or "computed from" for *mediating* fusion functions such as AVERAGE.

3.2 Dependent Resources

Current Linked Data tools fuse resource descriptions composed of triples having the respective resource as its subject. Further triples describing structured values are not included (e.g., street is not included for a BE). This leaves a space for improvement as demonstrated in Section 2. We propose the inclusion of dependent resources reachable from the parent resource through specially annotated properties, in analogy to [4]. For resource r with resource description R, we fuse a property p annotated with fusion function DependentResource as follows:

1. Let $D_{r,p} = \{o \mid (r, p, o, g) \in R, o, g \in U\}$ be dependent resources. Recursively fuse resources in $D_{r,p}$ as if there were `owl:sameAs` links between all pairs of resources in $D_{r,p}$. Denote the fused result $F_{r,p}$.
2. Let $d \in U$ be a new unique URI. Add a new quad (r, p, d, g), and quads $\{(d, q, o, g) \mid (s, q, o, g) \in F_{r,p}\}$ to the result.

This approach produces a single fused dependent resource (e.g., a single address of a BE), and takes advantage of the locality of `owl:sameAs` links to avoid incorrect merge of dependent resources with multiple parents. A unique URI is generated in step (2) so that other parts of the RDF graph where the dependent resource may occur are not affected by its "local" fusion for one parent resource.

4 Conclusion

Our practical experience with fusion of public procurement data shows that the graph nature or RDF has its specifics that need to be addressed. State-of-the-art Linked Data fusion tools do not cover two common patterns in RDF: fusion of structured values, and dependencies between properties.

We answer this challenge with new data fusion features. We introduce the concepts of dependent properties and dependent resources, and propose how to appropriately extend data fusion. The extensions have been implemented in LD-FusionTool and successfully used to fulfill the goals of our motivational scenario.

The new data fusion features show a new direction in Linked Data fusion – taking advantage of the broader context in the RDF graph. This can be further leveraged not only in conflict resolution, but also in quality assessment.

Acknowledgement. This work was supported by a grant from the EU's 7th Framework Programme number 611358 provided for the project COMSODE.

References

[1] Bleiholder, J., Naumann, F.: Data Fusion. In: ACM Computing Surveys 41.1 (2008)
[2] Mendes, P. N., Mhleisen, H., Bizer, C.: Sieve: Linked Data Quality Assessment and Fusion. In: Proceedings of the 2012 Joint EDBT/ICDT Workshops, ACM (2012)
[3] Knap, T., et al.: UnifiedViews: An ETL Framework for Sustainable RDF Data Processing. In: The Semantic Web: ESWC 2014, Posters and Demos Track (2014)
[4] Mynarz, J., Svátek, V.: Towards a Benchmark for LOD-enhanced Knowledge Discovery from Structured Data. In: Proceedings of the Second International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data (2013).