Proceedings of the I-SEMANTICS 2012 Posters & Demonstrations Track, pp. 38-42, 2012.

Lightweight Transformation of Tabular Open Data to RDF

Miel Vander Sande, Laurens De Vocht, Davy Van Deursen,
Erik Mannens, and Rik Van de Walle

Ghent University - IBBT
Department of Electronics and Information Systems - Multimedia Lab
Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium
firstname.lastname@ugent.be

Abstract. Currently, most Open Government Data portals mainly offer data in tabular formats. These lack the benefits of Linked Data, expressed in RDF graphs. In this paper, we propose a fast and simple semi-automatic tabular-to-RDF mapping approach. We introduce an efficient transformation algorithm for finding optimal relations between columns based on ontology information. We deal with multilingual diversity between data sets by combining translators and thesauri to create context. Finally, we use efficient string and lexical matching approaches in a learning loop to ensure high performance with good precision.

1 Introduction

Every government in the world possesses a huge number of public data sets. Today, the idea of Open Government Data is uprising, where data sets are made freely available on the Web. This way, governments can boost their transparency, create economic value and facilitate governmental participation.

Most portals (e.g., data.gov.be) mainly offer downloadable spreadsheets or tabular records in an XML or CSV like format. Although these initiatives are considered Open Data, they do not reach their full potential. The content in many different sets overlaps, but there are no links identifying this. Also, the data may be machine readable, but not machine understandable, which makes data integration the responsibility of the developer. These obstructions can be tackled with Linked Open Data, where data sources are reformulated in the graph-structured RDF standard. However, current data sets are still structured in classic hierarchical schema's, while graph patterns introduce a whole new way of organising data. Hierarchical patterns can easily be translated into a simple graph, but this is not always the optimal choice. In this paper, we propose lightweight automatic mapping approach for optimally restructuring tabular data as RDF.

2 Related Work

A popular solution is the RDF extension for Google Refine[1]. The transformation structure can be edited, links with other datasets can automatically be resolved and the result is generated in RDF. Although the reconciliation results are very good, the restructuring of the data is fully manual. It does not use structural knowledge extracted from ontologies or other Linked Data sources to automatically suggest an optimal graph. Only supported by search in selected ontologies, the user needs to explicitly specify which Class or Property map each field, requiring many manual operations. Also, Refine is a pure offline application, decreasing accessibility and means to collaborative editing.

More related web-oriented approaches are database abstraction or extraction systems (e.g., D2RServer, Virtuoso, Triplify). However, these transformation methods are based on database schemas and manually defined mappings. The former requires the data source to be a relational database, or at least imported into one. The latter again requires full manual editing without automated assistance. Our approach is fully compliant with these systems, since it can serialize its output as mapping rules. Finally, simple converters like Any23 do not consider any ontology information and create triples using column labels defined in a CSV file. This results into RDF with little semantics.

The closest related work is done in the KARMA [1] framework. A CRF (Conditional Random Fields) machine learning model is trained using following process in an interactive loop. Each column in the dataset is assigned a semantic type from a selected ontology. This assignment is based on user input and formerly learned types. A graph is formed using these types and structural information described in the ontology. Finally, a Steiner tree algorithm is used to extract the right relations. Although this approach achieves very good results in type assignment and tree selection, high precision is only achieved after a considerable amount of manual input.

We propose a more lightweight iteration process that replaces the existing CRF model with string, lexical, data type and context analysis. By using this analysis, we can present a possible mapping before input by the user is required. Note that we accept low precision in the first iteration, to achieve high precision, by fully exploiting user input, in later iterations. The CRF approach is also very memory consuming and copies almost all the data. Our analysis is based on far lighter analysis, resulting in higher performance. Also, stored data are restricted to ontology concepts and data types, which is significantly less. Finally, KARMA learns learns semantic types based on the data format, which are only helpful for data sets in a similar domain. We cover a broader domain of data sets, since we rely more on generic information stored in the fields and ontology, refined by user input.

[1] http://lab.linkeddata.deri.ie/2010/grefine-rdf-extension/

3 Overall approach and basic assumptions

In Linked Data, ontologies define the way data should be structured. Therefore, we will base this approach on the defined concepts and relations. Considering Open Government Data, we make the following assumptions:

- Most data sources only provide column labels and data rows
- These sources cover a wide variety of domains, resulting in heterogeneous data between sets
- Ontologies are available, are well described in OWL. The considered ontologies are average in size
- Ontologies are written in English, while data sets are multilingual
- Since this approach is used in an interactive loop with the user, low precision is acceptable in the first iterations; only if it results into simpler and faster mapping (considering public servant workload).

This approach takes two inputs: a tabular data set and an ontology which we want to match. Serialised mapping rules are constructed in three steps: concept matching, context construction and tree extraction.

3.1 Concept matching

For each column header, we will extract matching concepts from the ontology using header and data type information, as shown in figure 1. During this process, we keep a candidate list containing each retrieved concept with a confidence score between 0 and 1.

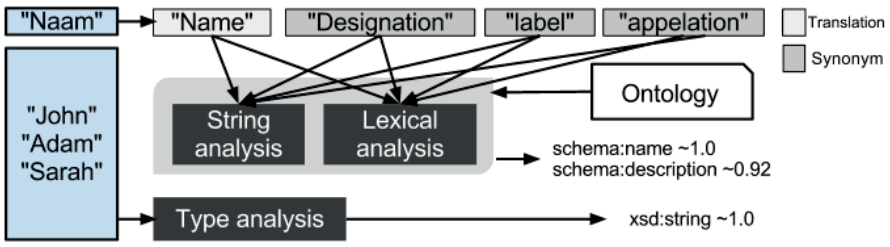

Fig. 1. Example of context extraction for a column 'naam'(dutch)

We start by translating our field label to English using the Microsoft Translator API and MyMemory[2]. The results are combined to form a unique result set. Each entry in this set will be the input of a thesaurus service, giving us a unique set of synonyms. Next, each translation or synonym is matched to the names of all classes and properties using the string and linguistic methods described in [2]. The former calculates the Jaro-Winkler distance with threshold

[2] http://www.microsofttranslator.com/dev/ http://mymemory.translated.net/

0.81 and the Smith-Waterman distance with threshold 10. The latter performs the Jiang-Conrath measure with threshold 1.0, which uses WordNet *Synsets* to score semantic relations between two nouns. The retrieved concept is then added to the candidate list, together with a normalised combination of these results as score. Then, we measure the compatibility of the column's data with the data types in the XMLSchema[3] name space. We sort them in order of compatibility and then look for the best match. The indices of this match and its preceding items indicate the compatibility score of each type.

3.2 Context graph construction

Based on the retrieved concepts, we construct a directed weighted labelled context graph in a three step process, as shown in figure 2. Firstly, we add the field label as a node and add all compatible (compatibility score greater than 0) data type properties as edges with their domain as source. The weight is calculated from their range's compatibility score and, if present in the candidate list, their similarity score. High compatibility and/or similarity result in a low edge weight. Secondly, we add all the classes from the candidate list. Since super classes might have a fitting relation to the field node, we copy each edge with a superclass as source, to the current class with a slightly higher weight (unless the current class is a candidate. E.g., *ex:PersonName* in figure 2). For completeness, we add a *subClassOf* relation between the super classes. Thirdly, we add object properties to connect the current nodes in the graph, based on their domain and range. When an object property is in the candidate list, we use its confidence score to determine the edge weight. If not, the edge gets a default weight.

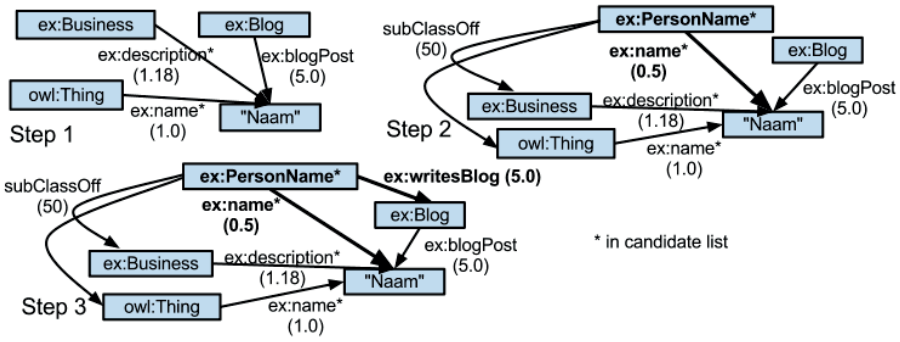

Fig. 2. Example of graph construction for a column 'name'

[3] http://www.w3.org/2001/XMLSchema

3.3 Tree extraction

Finally, we look for optimal paths between the different fields. First, we merge all the different context graphs into one and keeping the lowest edge weights. Second, we use a tree algorithm for finding paths in the graph. For extracting a tree, we use the *Steiner Tree algorithm* as described by Craig A. Knoblock et al. This algorithm finds the minimum-weight spanning tree in a graph between a subset of nodes, called Steiner Nodes. It extends the *minimal spanning tree* algorithm to dynamically add nodes if they provide a shorter path. The result, as shown in figure 3, determines our final mapping.

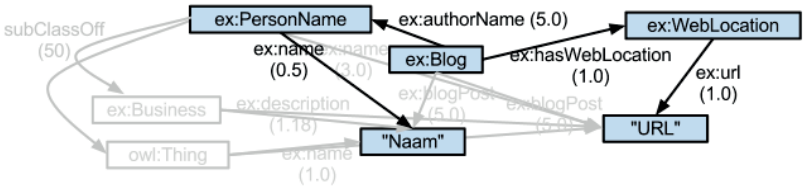

Fig. 3. Extracted steiner tree from merged graph

4 Discussion

We introduced a fast automatic mapping approach for transforming plain tabular data optimally into RDF. It uses a combined method of translation services and thesauri to deal with multilingual data sets. Furthermore, string, lexical and data type analyses is used to match ontology concepts to the different columns. For each column, a context graph is constructed based on the retrieved concepts. Finally, a possible mapping is selected by merging all context graphs and finding a tree connecting all columns.

In future work, an evaluation can be performed by comparing the result again manually defined mappings. This can be done in two ways: the different amount of manual operations and the precision/recall difference between the two. Furthermore, using this approach in a semi-automated loop, could dramatically increase the precision. Corrections by the user can be used to improve the tree selection process, or introduce machine learning for better concept matching.

References

1. Craig A. Knoblock, Pedro Szekely, Jos Luis Ambite, Shubham Gupta, Aman Goel, Maria Muslea, Kristina Lerman, and Parag Mallick. Interactively mapping data sources into the semantic web. In *Proceedings of the 1st Int. Workshop on Linked Science 2011 in Conjunction with the 10th Int. Semantic Web Conference*, 2011.
2. Feiyu Lin and Andrew Krizhanovsky. Multilingual ontology matching based on wiktionary data accessible via sparql endpoint. *CoRR*, abs/1109.0732, 2011.