Graph Aided Schema Alignment for High Volume File Conversion

R.H. van 't Veer, MA

January 20, 2017

Abstract

When confronted with a deluge of files to be converted to Linked Data, where are we to begin? Where most solutions focus on converting a single table or file, real-world cases often encompass a multitude of files. Few solutions exist for dealing with where full conversion is infeasible, where thousands of files, scattered over hundreds of schemas. The solution provided here explores the idea of graph-based schema comparison for scaling data conversion of thousands of files. It concludes that this approach is helpful in charting similarities and differences, provides grip and insight on a starting point, on progress and offers estimations on the required effort on transforming all data. **Keywords:** Linked Data, conversion, triplification, scaling, schema alignment

1 Introduction

When undertaking the endeavour to convert legacy tabular data into Linked Data, the main time investment most probably lies in the application of the proper vocabulary, creating an ontology and describing all classes and properties. This is because usually it is easy to create an automated or 'direct' mapping from a data table that produces linked data using a dummy prefix. The general approach is to append to a provided prefix: the table names for class names, the column names for properties, the primary key as identifier for instance data.[1] The resulting output does support graph analysis, but is lacking in semantics. Without the benefits of properly described classes and properties, the data is simply just as semantically poor as it was before conversion to Linked Data. The data may technically be RDF, but it is unlikely to contain much outgoing RDF links[1]. The

[1]http://5stardata.info/en/

brunt of manual labour involved in triplifying legacy data is in creating new classes and properties or choosing existing ones from shared vocabularies.

When we accept the assumption that the manual data modelling step is the most time-consuming part of triplification, it must follow that if we are confronted with a situation where the conversion of data must scale to hundreds or thousands of source schemas, mapping solutions augmented with manual modelling will fail, either due to the time invested in creating the semantic model, either to the manual input needed for the conversion of individual sources, or both. If conversion of thousands, or hundreds of thousands of files or tables from differing sources is called for, it becomes evident that a file-by-file conversion system that requires manual input for every single file will take too much time. If, say, a collection of 100,000 files requires one hour for every file to be processed, 2,500 full working weeks or around 50 years are required for the process to finish, depending on how many holidays are granted by your contract. Even if no more than one minute of manual labour is involved, the better part of a full year of undivided attention to manual input is required. We must aim much higher. Ideally, we wouldn't want to invest more than a few weeks - so the goal is to reduce the per-source manual mapping effort to no more than a few seconds, perhaps as few as five for our 100,000 file example.

In order to prevent making a career out of converting a large volume of sources, I will propose a method for automating the steps that precede the actual mapping. It revolves around re-using as much information from the modelling stage as possible. Yet, there are surprisingly few solutions that provide such far-reaching automation of Linked Data mapping. The only solution that I found re-using previously applied mappings is a package called Karma.[2] Karma offers handy mapping suggestions based on settings applied earlier[3], but still requires manual confirmation of the suggested mapping, not to mention further manual input for file-by-file mapping. If we are to reach our five-second goal, we need automation that does bulk transformation on a large volume of files simultaneously, on heterogeneous sources.

2 Case study: the Dutch repository of archaeological data files

If many thousands of sources are involved, chances are that a large number of source schemas are present as well, but considerably less than the number of sources to begin with. This schema variability is something we can use to our

[2]https://github.com/usc-isi-i2/Web-Karma

[3]https://youtu.be/h3_yiBhAJIc?t=155

advantage. With our test case, this is the case as well. The case in point is the national repository for archaeological data. The Netherlands can boast an extensive collection of digital archaeological data, gathered by various institutions[4] and submitted to the Data Archiving and Networked Services (DANS) organisation, itself a sub-organisation of the Royal Netherlands Academy of Arts and Sciences.[5] At the moment of writing, the DANS repository holds close to twenty-seven thousand deposited data sets on the topic of archaeology. The concept of a centralized data repository for archaeological data is unfortunately still much of a rarity,[6] let alone services that hold this much information.

However, there is the central problem to the accessibility of the data in this national repository. The Netherlands may take pride in an archaeological data repository of repute, but its access leaves much to be desired. No data contained in the repository, other than on project-level metadata, can be searched for information of interest. Thus, it does not allow searching for specific artefact or archaeological structure types, nor does it support geospatial search beyond the centroid locations on the project level. Only results from text searches on comments supplied in the project description can be provided. What is needed, is a solution that provides full semantic search capabilities on artefacts and other archaeological phenomena, that allows researchers to do deep and complete queries for very specific problems. This is to be the main aim of this research outline: to investigate a solution that opens up the full search access to the Dutch corpus of archaeological data.

The DANS archaeological repository is a collection of semi-structured data, commonly collected in the field and office through databases, spreadsheets and geospatial data tables, all flattened tabular data. Although the archaeological sector has agreed on a common data exchange format (*SIKB0102* or more popularly named the *Pakbon* protocol)[2] for the Netherlands, its uptake is still very much limited[7] and its normative model does not lend itself very well to the faults and omissions that occur in the unstructured data. Historically, data has been gathered for a long time through databases and spreadsheets that do not conform to the strict referential integrity that the SIKB0102 protocol enforces, so there is a backwards compatibility problem. The bulk of the current archaeological data simply cannot be expressed through this protocol, since it would violate most of

[4]Archaeological companies, municipal services, universities

[5]See http://knaw.nl/en/about-us/overzicht, and https://dans.knaw.nl/en/front-page?set_language=en

[6]The UK has an Archaeological Data Service, see http://archaeologydataservice.ac.uk/, but few others of its kind are found elsewhere in the world.

[7]At the time of writing, there are around 25 data sets that can be accessed as structured through this format, which was learned from conversation with Xander Wilcke and Milco Wansleeben.

its constraints.

This is hardly surprising when we take into account the variation of schemas that define the tabular data, which includes the geospatial data tables used to record the contours of excavation trenches, archaeological features and the likes. It varies not only from organisation to organisation, but also in respect to project types - Medieval town excavations with complex three-dimensional stratigraphy require a different approach than digs on Mesolithic sites that often focus on collecting flint artefacts in sampling units that conform to standardized sizes. It is also not uncommon to adapt the database schema to the specific needs of the project, resulting in single-instance schema variations. So, the DANS archaeological repository seems the ideal candidate for exploring a method to convert many thousands of sources, spread over several hundreds of schemas.

3 Graph-based schema analysis

If the amount of data schemas is a fraction of the amount of sources, there is of course the question how it can be useful to leverage the knowledge of all the schemas. For our purposes, we define the data schema as the property names - in tabular data the column name - and the data type of the property, like a numeral or a string literal. Then, we work from a set of ordered principles that structure the schemas into a graph. These are dangerous assumption to apply blindly, as will be discussed further on, but as simple principles, they provide a scaffold for the structuring of the data schemas into a graph.

1. Schemas are sorted by their properties? Reference tables should end up in a different cluster than their source file counterparts Question your assumptions. The why-question should not pop up, or it should be explained using a literature reference Edge weight for - number of needed changes - type of changes Separation of trenches from features based on table contents

2. Sources that share the same schema, contain the same types of information. No graph building has to be done for same-schema files - if all files would share the same schema, no schema graph analysis would be needed as all files could possibly be converted the same way, with simple scripting.

3. Schemas that extend from another schema will probably hold the same kind of information, but add some extra properties for whatever reason. The reason might be that is actually a different class of information, but at least that class is pretty close to its 'parent'.

4. Schema extensions can be ordered by the amount of additions to their parent. Extensions with less additions are probably closer to their parent than ones

with many, so we will optimize the usefulness of the graph by minimizing the extension difference. For extensions, we can actually build a directed graph, as the inverse of an extension (and thus its inverse direction) is deletion of one or more properties.

5. Schema property extensions are a bit tougher, but can still be of use. If two schemas of the same number of properties share all but one property, the differing property can either be a renamed property, or something altogether different. We can often tell by lexical comparison. If the property name from one schema is a substring or a lexically close substring of the other, we possibly have some kind of derived property. If not, it is more likely that it is something entirely different.

6. If all of the above fail to produce a close relative, all that rests is a set of renamed properties that quite probably share little information with their counterparts. There is a weak relationship, which can be ordered to its closest relative by the least amount of additions and deletions.

References

[1] Franck Michel, Johan Montagnat, and Catherine Faron-Zucker. A survey of RDB to RDF translation approaches and tools. Research report, I3S, May 2014. ISRN I3S/RR 2013-04-FR 24 pages.

[2] Chris Sueur. Digitaal werken in de archeologie, 2015.