Tabular Data Cleaning and Linked Data Generation with Grafterizer

Dina Sukhobok[1], Nikolay Nikolov[1], Antoine Pultier[1], Xianglin Ye[1], Arne Berre[1], Rick Moynihan[2], Bill Roberts[2], Brian Elvesæter[1], Nivethika Mahasivam[1], Dumitru Roman[1]

[1] SINTEF, Forskningsveien 1a, 0373 Oslo, Norway
{dina.suhobok,lynnye1988mail,nivemaham}@gmail.com
{nikolay.nikolov,antoine.pultier,arne.j.berre,brian.elvesater,dumitru.roman}@sintef.no
[2] Swirrl IT LTD, Springbank Raod, MacFarlane Gray House, FK7 7WT Stirlingshire, United Kingdom
{rick.m,bill}@swirrl.com

Abstract. Over the past several years the amount of published open data has increased significantly. The majority of this is tabular data, that requires powerful and flexible approaches for data cleaning and preparation in order to convert it into Linked Data. This paper introduces Grafterizer – a software framework developed to support data workers and data developers in the process of converting raw tabular data into linked data. Its main components include Grafter, a powerful software library and DSL for data cleaning and RDF-ization, and Grafterizer, a user interface for interactive specification of data transformations along with a back-end for management and execution of data transformations. The proposed demonstration will focus on Grafterizer's powerful features for data cleaning and RDF-ization in a scenario using data about the risk of failure of transport infrastructure components due to natural hazards.

Keywords: open data, linked data, tabular data cleaning and preparation, data transformation

1 Introduction

The potential gains from data analysis and knowledge discovery in the future is estimated to billions and even trillion dollars[3,4]. In order to get a broader view on a given problem and benefit from knowledge discovery from data, analysts need access to large amounts of data. This leads to an increase in demand for using data sources such as open data. Open data is commonly published in tabular formats, which are widely adopted and familiar to data workers. Nevertheless, only a small amount of published datasets are actually used for various reasons,

[3] http://www.irishexaminer.com/lifestyle/features/dell-chief-executive-says-data-is-the-next-trillion-dollar-opportunity-370608.html
[4] http://www.idc.com/getdoc.jsp?containerId=prUS40560115

primarily due to the lack of simple approaches to interconnect the data from various tables. Linked data can alleviate some of these problems by providing a set of standards for representing and connecting the data, therefore enabling data to be discovered and used by various applications [1].

To generate valid linked data we need data to be in a shape that it is easy to manipulate and convert to RDF ('RDF-ize'). Raw data in most cases contain a number of common data quality issues, such as missing values, invalid values, duplicate records, etc. Solving data quality issues is especially important when integrating heterogeneous data sources that should be addressed together with schema-related data transformations [2]. Data preparation provides a standard way of structuring data, which makes it easier to extract needed values [3]. Hence, data publishers first need to deal with data quality issues before data can be mapped to RDF. This process is usually considered as one of the most time- and cost-consuming – according to some sources it takes up to 80% of the time [4]. We therefore need a unifying framework for data cleaning and RDF-ization, powerful enough to cope with common data cleaning problems, simple enough that non-programmers can use it, while at the same time flexible enough that data developers can easily work with it. A common framework for cleaning and RDF-ization can also simplify collaboration between users collaboratively working on the same data transformation (e.g., one doing the data cleaning, the another one the RDF-ization). In this demonstration we introduce Grafterizer as an example of such a framework.

2 The Grafterizer Framework

The main steps of transformation of raw tabular data into linked data supported by Grafterizer are depicted in Figure 1.

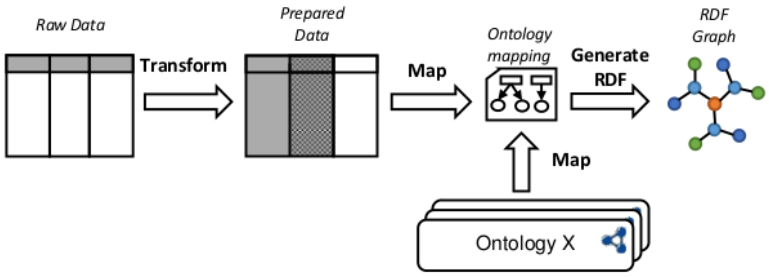

Fig. 1. Generating a semantic RDF graph from tabular data

Grafterizer is based on Grafter[5] – a library and DSL for producing linked data graphs from tabular data, which provides extensive support for data cleaning and powerful ETL data transformations, suitable for handling large datasets.

[5] `http://grafter.org/`

The Grafter library is implemented in Clojure – a functional programming language that runs on the JVM. Grafter's properties give it a number of advantages: the Java virtual machine environment makes it possible to use numerous available libraries, and, Clojure, as a functional programming language, makes it natural to treat data structures as streams of immutable values.

The data transformation process is realized through a pipeline abstraction, i.e., each step of transformation is defined as a pipe. Thereby, a function performs simple data conversion on its input and the output of one pipe is the input of another. Grafter also provides a broad set of features for creating and managing linked data graphs out of these data.

Grafter's primary target are software developers, whereas Grafterizer makes the process of creating pipelines and ontology mappings for graphs more accessible. It provides a set of functions, that can be used to solve the most common data quality issues in a fast and effective way. A summary of the tabular transformation functions is given in Table 1.

Table 1. Summary of tabular transformation functions supported by Grafterizer

Scope	Name	Short Description
Rows	Add Row	Create a new record in a dataset.
	Take/Drop Rows	Extract/delete selected row (sequence of rows)
	Shift Row	Change row's position inside a dataset
	Filter Rows	Filter rows for matches, regexes, empty values etc
	Remove Duplicates	Remove similar rows based on certain column or set of columns
Entire dataset	Sort Dataset	Sorts dataset by given column names in given order
	Reshape dataset	Restructure a dataset
	Group and Aggregate	Group values by column or multiple columns and perform aggregation (get minimal, maximum or average value, count or sum values in every group) on the rest of columns
Columns	Add Column Derive Column	Add a column with a manually specified value, or as a result of some computations performed on other columns
	Take/Drop Columns	Take/drop selected column
	Shift Column	Arbitrarily change column's order.
	Merge/Split Columns	Split or merge columns using custom separator
	Rename Columns	Change columns headers
	Map Columns	Apply function to all values in a column

Due to the large number of possible types of data quality issues, the operations on data are not limited to the functions listed above. Grafterizer makes it possible for users to define their own functions on data and involve them in the Grafterizer transformation pipeline, which makes transformations more flexible. Custom functions provide also a good way to encapsulate data modifications.

After data quality issues are solved, the dataset can be transformed to RDF. The RDF triple patterns that should appear in the resulting linked data are designed by the user, whereby the pattern for the subjects, predicates and ob-

jects of triples is specified through a mapping procedure. During the mapping process column headers are mapped to RDF nodes in order to produce a set of triples that corresponds to each data row. Grafterizer supports reuse of existing RDF ontologies by providing a searchable catalog of vocabularies and makes it possible to manage individual namespace prefixes. Each column in a dataset can be mapped as a URI node with namespace prefix assigned by user or literal node with a specified datatype. Grafterizer also provides support for error handling when casting to datatypes. Furthermore, users may assign condition(s) under which a node, a triple or entire sub-graph should be generated.

Related Work. At present there is no *unifying framework* for tabular data cleaning and linked data generation that targets both data developers and data workers (non-developers with spreadsheet knowledge level) and is flexible enough at the same time. Some methods of converting tabular data into linked data involve intermediate steps of importing raw tabular data to a relational database and transforming the resulting data into RDF format afterwards[5]. The set of recommendations[6] and advanced tools make a good base for using such methods. With the Grafterizer approach, this step is omitted – data are brought to semantic form directly from the raw tabular presentation[7]. OpenRefine[8] with the RDF Refine plugin[9] is relevant in the context of Grafterizer. OpenRefine functionalities are tightly-coupled to the service core, which hinders distribution and prevents its use "as-a-service". Furthermore, OpenRefine uses a memory-intensive multi-pass approach to data transformation functions, which is designed to operate with small to medium data volumes. Other platforms provide a broad and powerful functionality in tabular data cleaning (e.g. Trifacta Wrangler[10]), but do not support RDF data publication.

3 Demonstration Outline

Grafterizer will be demonstrated in a real case for data cleaning and RDF-ization in the context of InfraRisk[11] — a project developing a framework to identify and track the impact of natural hazards on infrastructure networks (e.g. roads, rails).

During the demonstration, visitors will learn how to solve data quality issues and transform tabular data into a linked data graph. The scenario demonstrated will cover uploading raw data into Grafterizer's interactive user interface, constructing and testing a pipeline to transform that data (with the help of provided standard functions and embedded custom Clojure code), and generating a linked data graph out of an RDF mapping.

Within the demonstration several groups of data quality issues are addressed. For example, data rows that contain no useful information will be filtered out,

[6] `http://www.w3.org/TR/rdb-direct-mapping/`
[7] `http://www.w3.org/TR/csv2rdf/`
[8] `http://openrefine.org/`
[9] `http://refine.deri.ie/`
[10] `https://www.trifacta.com/products/wrangler/`
[11] `http://www.infrarisk-fp7.eu/`

values of rows that contain badly formatted data will be cleaned-up, and missing identifiers for each data row will be produced. An example of Grafterizer's transformation pipeline and preview results are shown in Figure 2. For the

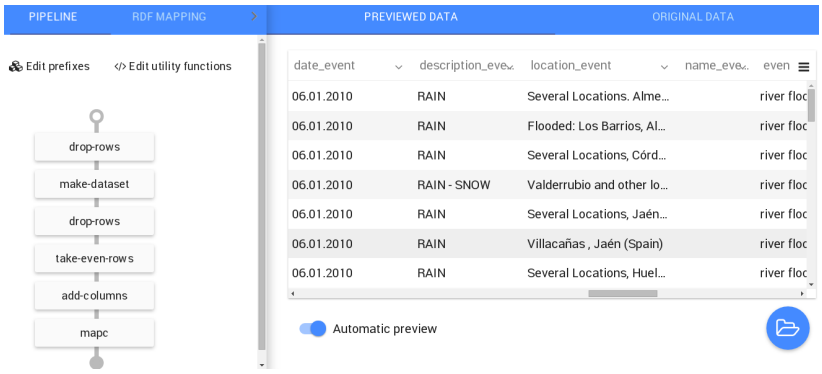

Fig. 2. Pipeline and transformation preview

RDF-ization process visitors will learn how to build an RDF mapping, manage namespace prefixes, link nodes to columns, specify conditions on triple generation, and cast cell values to literal data types.

Grafterizer is currently deployed within the DataGraft platform [6], available at `https://datagraft.net/`. A video showing Grafterizer in action can be found at `https://youtu.be/zAruS4cEmvk`.

Acknowledgements This work was partly funded by the European Commission within the following research projects: DaPaaS (FP7 610988), SmartOpenData (FP7 603824), InfraRisk (FP7 603960), and proDataMarket (H2020 644497).

References

1. Bizer, Christian, Tom Heath, and Tim Berners-Lee. "Linked data-the story so far." Semantic Services, Interoperability and Web Applications: Emerging Concepts (2009): 205-227.
2. Rahm, E., Do, H.H.: Data Cleaning: Problems and Current Ap- proaches. IEEE Bulletin on Data Engineering 23:4, 2000.
3. Wickham, Hadley . "Tidy Data." Journal of Statistical Software, 59.10 (2014): 1 - 23. Web. 1 Mar. 2016.
4. BTamraparni Dasu and Theodore Johnson. "Exploratory Data Mining and Data Cleaning (1 ed.)".2003. John Wiley & Sons, Inc., New York, NY, USA.
5. M. G. Skjveland, E. H. Lian, and I. Horrocks. Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web Data. In The Semantic Web, ISWC 2013, volume 8219 of LNCS, 2013.
6. D. Roman, N. Nikolov, A. Putlier, D. Sukhobok, B. Elvester, A. Berre, X. Ye, M. Dimi-trov, A. Simov, M. Zarev, R. Moynihan, B. Roberts, I. Berlocher, S. Kim, T. Lee, A. Smith, and T. Heath. DataGraft: One-Stop-Shop for Open Data Management. Technical Report, January 2016. Available at http://www.semantic-web-journal.net/system/files/swj1285.pdf.