Julian Eberius, Christopher Werner, Maik Thiele, Katrin Braunschweig, Lars Dannecker, Wolfgang Lehner

DeExcelerator: A Framework for Extracting Relational Data From Partially Structured Documents

Erstveröffentlichung in / First published in:

CIKM'13: 22nd ACM International Conference on Information and Knowledge Management. San Francisco, 27.10.–01.11.2013. ACM Digital Library, S. 2477 – 2480. ISBN 978-1-4503-2263-8.

DOI: https://doi.org/10.1145/2505515.2508210

Final edited form was published in "CIKM'13: 22nd ACM International Conference on Information and Knowledge Management. San Francisco 2013",
S. 2477–2480. ISBN: 978-1-4503-2263-8
https://doi.org/10.1145/2505515.2508210

DeExcelerator: A Framework for Extracting Relational Data From Partially Structured Documents

Julian Eberius[1] Christopher Werner[1] Maik Thiele[1] Katrin Braunschweig[1]
Lars Dannecker[2] Wolfgang Lehner[1]

[1]Database Technology Group
Technische Universität Dresden, Germany
firstname.lastname@tu-dresden.de

[2]SAP AG
Dresden, Germany
lars.dannecker@sap.com

ABSTRACT

Of the structured data published on the web, for instance as datasets on Open Data Platforms such as `data.gov`, but also in the form of HTML tables on the general web, only a small part is in a relational form. Instead the data is intermingled with formatting, layout and textual metadata, i.e., it is contained in partially structured documents. This makes transformation into a true relational form necessary, which is a precondition for most forms of data analysis and data integration. Studying `data.gov` as an example source for partially structured documents, we present a classification of typical normalization problems. We then present the DeExcelerator, which is a framework for extracting relations from partially structured documents such as spreadsheets and HTML tables.

Categories and Subject Descriptors

H.2 [**Database Management**]: Miscellaneous

Keywords

Spreadsheets, Normalization, Extracting Relational Tables

1. INTRODUCTION

There are several new sources for relational data on the web. One interesting example are Open Data Platforms such as `data.gov` or `data.gov.uk`, on which government agencies publish datasets. Most of the files on these OD platforms are optimized for human consumption, such as spreadsheets, HTML and PDF, as shown in [1]. To reuse the wealth of structured data contained in these datasets in a different context or to find relations between datasets, a human analyst will often have to transform these datasets manually, e.g., to load them into a statistics tool. Although there is some support for performing this task semi-automatically (see Section 5), there exists no fully automatic extraction of relational data from such documents.

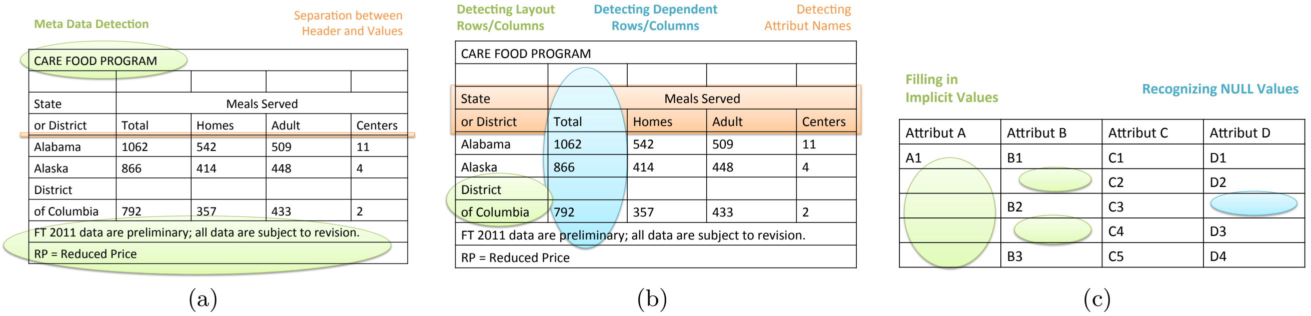
In this paper, we aim at transforming partially structured documents, e.g. spreadsheets or HTML pages, into first normal form relations without any user interaction. To this end, we will at first present a set of typical denormalizations and irregularities that appear in spreadsheets and web tables alike in Section 2. These features are usually introduced to enhance readability, but constraint the reuse of these documents. We will then present the *DeExcelerator*, a framework for normalizing a partially structured document into one or more relational tables in Section 3, and give the results of a preliminary user study in Section 4. Then, we will discuss related work as well as the demonstration scenario in Sections 5 and 6, respectively.

2. SPREADSHEET NORMALIZATION

By manually studying a corpus of real-world Open Data published as Excel spreadsheets on the platform `data.gov` (see Section 4), we identified a set of typical denormalizations that often appear in spreadsheets. Our goal was to transform the corpus of documents from this platform into a set of tables that can be handled by an off-the-shelf relational database management system. While we originally studied spreadsheets to identify these characteristics, other tables created for human consumption, e.g. web tables on Wikipedia, show most of the identified characteristics, as well. Note, that most of these denormalizations are not introduced because of lack of database know-how, but because spreadsheets are designed to be comprehended visually by a human, not to be processed by a DBMS.
We found the following (non-exhaustive) list of challenges for spreadsheet normalization. Examples for all challenges are shown in Figure 1.

- **Table Search (TS)**: While spreadsheet software usually offers the possibility to save multiple different sheets in one document with some form of logical separation, users often just copy multiple tables into one physical sheet for convenience.
- **Metadata Extraction (ME)**: Metadata that is essential to the interpretation of the table, e.g., information regarding provenance or time of validity is usually just copied into the cells of the spreadsheet (see Figure 1a). Some of this data might be highlighted visually, e.g. a table title might be underlined, but other metadata might appear in the next cell, without any clear separation.

Final edited form was published in "CIKM'13: 22nd ACM International Conference on Information and Knowledge Management. San Francisco 2013", S. 2477–2480. ISBN: 978-1-4503-2263-8
https://doi.org/10.1145/2505515.2508210


Meta Data Detection
Separation between Header and Values
Detecting Layout Rows/Columns
Detecting Dependent Rows/Columns
Detecting Attribut Names
Filling in Implicit Values
Recognizing NULL Values

CARE FOOD PROGRAM				
State	Meals Served			
or District	Total	Homes	Adult	Centers
Alabama	1062	542	509	11
Alaska	866	414	448	4
District				
of Columbia	792	357	433	2
FT 2011 data are preliminary; all data are subject to revision.				
RP = Reduced Price				

(a)

CARE FOOD PROGRAM				
State	Meals Served			
or District	Total	Homes	Adult	Centers
Alabama	1062	542	509	11
Alaska	866	414	448	4
District				
of Columbia	792	357	433	2
FT 2011 data are preliminary; all data are subject to revision.				
RP = Reduced Price				

(b)

Attribut A	Attribut B	Attribut C	Attribut D
A1	B1	C1	D1
		C2	D2
	B2	C3	
		C4	D3
	B3	C5	D4

(c)

Figure 1: Spreadsheet Normalization Challenges

- **Layout Element Removal (LER)**: Many authors add cells to improve legibility, even if this distorts the column/row structure of the table. Data columns might be separated by empty columns to create visual padding, or rows might be inserted to simulate line breaks in cells with long text content (see Figure 1b).
- **Header Recognition (HR)**: While attribute names will often be highlighted visually using bold face or colors, spreadsheets offer no machine readable distinction between data and schema information. Sometimes the attribute name might be the first cell of a data column, as in CSV files, but this simple heuristic will fail most of the time with real-world spreadsheets (see Figure 1b).
- **Type Recognition (TR)**: Spreadsheets often contain type information on a cell-by-cell basis. If the previous challenges have been solved successfully so that clean, self-contained columns have been identified, one might expect that this challenge becomes trivial. Still, many spreadsheets will contain string annotations inline invalidating type information the spreadsheet tool might be able to provide, while tables extracted from the web have no type information at all.
- **Value Extrapolation (VE)**: Since spreadsheets often contain multidimensional data in one denormalized table, many authors skip repeated values that occur because of denormalization for visual clarity. In many cases, simply filling neighboring empty cells with previous values might solve this problem. Still, in some cases empty cells might also symbolize NULL-values and not implied values (see Figure 1c).
- **Dependent Row/Column Removal (DR)**: Adding derived rows and columns such as *Total* cells and summation rows that are automatically updated is a main selling point for spreadsheet software. When the data is published including these derived columns it contains redundant information (see Figure 1c).

Notice that even when all these defects have been removed, the resulting table might still be only in first normal form. Automatic decomposition of the resulting table is out of scope for this demonstration paper.

3. THE DE-EXCELERATOR

The DeExcelerator is a framework for transforming partially structured documents, i.e., documents that contain some relational data, into a first normal form relation that can be imported into a relational database. It implements a pipeline of abstract extraction phases, each one cleaning or removing one of the artifacts and denormalizatons described in Section 2. The phases and their order correspond to the challenges given in there.

Before the first phase there is an *ingestion* step, in which input files are transformed into a generic representation. As the minimal, common representation of our problem space we chose a two dimensional array of strings, with optional cell-level metadata attached. It is simple enough to be used with HTML tables as input, which do not have much information attached to them except the structure of row and cell tags, but also allows to capture the metadata available in spreadsheets. After the ingestion step there will be cells containing metadata text, headers or data values at any position in the matrix, as shown in Figure 1.

For each of the further phases, the DeExcelerator contains concrete implementations of the abstract extraction operations, which are based on our study of real-world datasets, as well as on table extraction techniques from the literature (see Section 5). Extraction heuristics implemented in the DeExcelerator operate on the string matrix only, but may use the cell-level metadata attached by the ingestion step, e.g., to use color information defined on the original spreadsheet cells as evidence for the header recognition. All implemented heuristics will return a *transformed matrix*, e.g. a matrix where cells containing textual metadata have been removed, as well as a *confidence* value. It may also attach new metadata to the cells of the matrix. The confidence value is used by the DeExcelerator to decide on the extraction output in case of conflicting results, e.g., two different sets of attribute names.

For space reasons, we can not give all the heuristics currently implemented in the DeExcelerator. Notice however, that the source code of the framework and its operators is available (see Section 6). To give the reader a better understanding of the style of our heuristics, we will provide some selected examples.

Consider the problem of *header-detection*, in which we want to find the separation between table header and data, e.g., the red line in Figure 1a. The DeExcelerator currently implements the following list of heuristics, whose output is combined to decide on a separator.

- Date: As soon a date cell appears, the header ends one row above it.
- Background color: A change in background color (in a spreadsheet) signals the start of the data portion.

Final edited form was published in "CIKM'13: 22nd ACM International Conference on Information and Knowledge Management. San Francisco 2013", S. 2477–2480. ISBN: 978-1-4503-2263-8
https://doi.org/10.1145/2505515.2508210

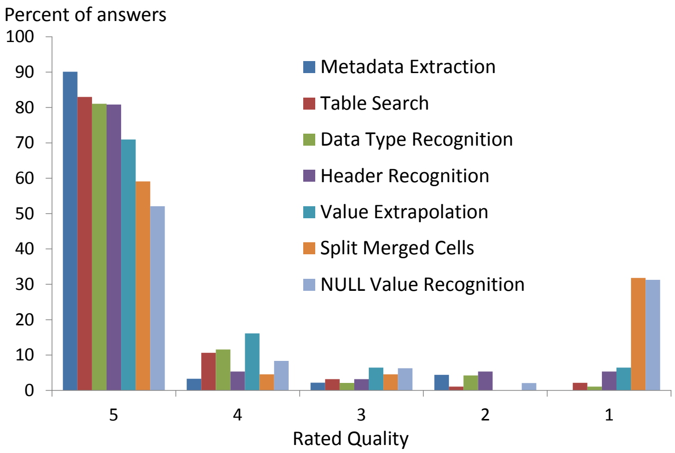


Figure 2: Extraction Success from Complete Success (5) to Unsuccessful (1), Table Difficulty from Hard (5) to Easy (1)

- Number sequence: The start of a numeric sequence in one column signals the start of the data segment.
- Number: A number in a cell is also evidence for the start of the data section, albeit a weaker one.
- Different string: A sequence of cells in a column with the same value signals the start of the data block. Such sequences are usually a product of denormalization, e.g. they are keys in some hierarchical dimension.
- String length: The same logic can be applied to several cells in one column with equal length. These are usually constant length identifier codes.

Similar lists of heuristics exist for every other phase.

4. EVALUATION

We conducted a preliminary evaluation of the DeExcelerator using about 2,000 Excel files from `data.gov`, which contained about 4,500 sheets. To evaluate the relevance of our challenge classes, we counted how often each challenge was encountered. For instance, the 4,500 sheets contained about 5,000 tables, which indicates the necessity of the *table search* step. The DeExcelerator could split header and data segment, which is the most important step for creating a relational table, in 78% of the cases. As another example, in about 83% of the tables the header identified was not found in the first row of the matrix, which indicates that header recognition is a non-trivial problem.

For space reasons, we can not present the full evaluation in this paper, but we want to highlight another aspect: We conducted an user study, in which 10 database students where asked to rate the success of the various extraction and cleaning steps for 50 sheets from our evaluation set, where every sheet was rated by two students. The students rated the success of every phase on a scale from 1 to 5, as well as the perceived difficulty of normalizing the original table on the same scale. As Figure 2 shows, the test set contained a good mixture of difficult and easy tables, with most tasks being solved in a good manner for a fully automatic system.

5. RELATED WORK

There are two classes of related work: first, there is a number of tools build to enable a user to extract and clean relational data from source documents, for example Wrangler[4] and Google Refine[3]. Both systems offer tooling to help a user working manually on a document cleaning project. In contrast, DeExcelerator is a predefined pipeline that can be applied to large heterogeneous documents collections automatically. The user can be involved by creating new implementations of the extraction steps.

The second category of related work covers table recognition algorithms. They exist for various types of input, e.g. web tables, web lists, PDF files and even images, and use a large variety of heuristic, learning-based and even many visual techniques. An extensive overview of these approaches is given in [6]. The DeExcelerator implements many previously published heuristics for table recognition, and defines an abstract pipeline of extraction steps specifically tuned for transforming spreadsheets and web tables into relational tables. An especially active area of research is concerned with extracting relational tables from large HTML corpora and interpreting the meaning of their attributes, e.g., [5] and [2] among many others. While these approaches focus more on specifics of identifying relational tables in large corpora, and semantic annotation, respectively, the DeExcelerator focuses on syntactic artifacts that occur in tables meant for human consumption. So the DeExcelerator could be seen as a preprocessing step to these techniques.

6. DEMONSTRATION SCENARIO

In this interactive demonstration, users will be able to challenge the DeExcelerator with their own Excel Spreadsheets, or use some of the packaged datasets from `data.gov` as input. We created a Web interface which visualizes the steps of the extraction pipeline and shows the input document as it transforms step by step into a relational table. The interface also displays the results of the heuristics implemented in each step, and thus allows the user to understand what is going on in each respective step. The DeExcelerator website[1] features a screencast, a link to the actual GUI, as well as a link to the source code.

7. REFERENCES

[1] K. Braunschweig, J. Eberius, M. Thiele, and W. Lehner. The State of Open Data - Limits of Current Open Data Platforms. In *WWW'12 Web Science Track*, 2012.

[2] M. J. Cafarella, A. Y. Halevy, Y. Zhang, D. Z. Wang, and E. Wu. Uncovering the Relational Web. In *WebDB*, 2008.

[3] D. Huynh and S. Mazzocchi. Google refine.

[4] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: interactive visual specification of data transformation scripts. In *CHI'11.*, pages 3363–3372, New York, NY, USA, 2011. ACM.

[5] J. Wang, H. Wang, Z. Wang, and K. Zhu. Understanding Tables on the Web. In P. Atzeni, D. Cheung, and S. Ram, editors, *Conceptual Modeling*, volume 7532 of *Lecture Notes in Computer Science*, pages 141–155. Springer Berlin Heidelberg, 2012.

[6] R. Zanibbi, D. Blostein, and R. Cordy. A survey of table recognition: Models, observations, transformations, and inferences. *Int. J. Doc. Anal. Recognit.*, 7(1):1–16, Mar. 2004.

[1] `http://wwwdb.inf.tu-dresden.de/edyra/DeExcelerator/`