Profiling CSV documents on the Web

Jürgen Umbrich and Sebastian Neumaier
Vienna University of Economics and Business
Welthandelsplatz 1; 1020 Vienna; Austria

This study reports on our findings about 74395 CSV files published on the Web as Open Data. The documents are extracted from 91 Open Data CKAN portals for which the meta data indicate a comma/character-separate-values file. Our analysis includes the inspection of the HTTP response headers, encoding detection and guessing of used delimiters. We also determine the deviation of data tables compared to a canonical form [1].

Our findings show that the majority of the CSV files adhere to the RFC4180 specification, meaning the use of *csv* as file extension, `text/csv` as the HTTP response header `content-type` , and ',' as delimiter. We also show that there exists nearly no information about the content encoding in the HTTP headers. The major observed deviations are that data tables contain rows in which one or several data cells occupy multiple columns and that one or several data cells are empty.

Contents

1 Introduction

A widespread method to manage and publish information in an intuitive manner is to represent the information in form of tabular data. In fact, many institutions handle their data in spreadsheets (e.g. MS Excel, or other OpenOffice spreadsheet) and use the tabular format to exchange data between different systems (e.g. a simpler form compared to the XML format). A simple plain text format for the exchange of such spreadsheets is the comma-separate-values format (CSV).[1] While the RFC specification of the CSV format is well defined, we observe many deviations from the originally specification. For example, various spreadsheet software exports their data into the CSV format using different delimiter symbols (in comparison to the default ',' delimiter), or there exists CSV files with unsymmetrical column numbers for different rows. We can observe a large number of such "exported" CSV-like documents with the Open Data movement and one can declare such files as rather character-separate-values than comma-separate-values.

This report attempts to analyse to which degree documents on the Web, that are defined as CSV files (e.g. via header `content-type` definition or file extension), follow the RFC4180 specification. The results will help to better understand the various deviations which provides the consumers with valuable information about what to expect when processing CSV files from the Web.

To do so, we analyse CSV files published as Open Data by CKAN portals. The CKAN Open Data portal software acts as a catalog for mostly government data and is one of the main frameworks used. A CKAN portal hosts several datasets which can consist of one to several resources. Each resource is described by some meta data, including a field to specify the format. We used those resource format information to derive a our seed list of potential CSV files.

In the following we will briefly highlight our methodology and tools to profile CSV files and report on our findings.

2 Methodology

We start with a list of potential CSV URLs and retrieve for each each URL its content and the response header information. Next, we "profile" the documents and header and extract the following information: For each document, we extract the following information:

- File Extension:
  the last three or four characters after the period in the file name or URL
- Header content-type:
  value of the `HTTP Response header` "Content-Type" field
- Header Charset
  value of the "charset" suffix in the `HTTP Response header` "Content-Type" field

[1] RFC4180: http://www.ietf.org/rfc/rfc4180.txt

- Content Encoding:
  guessing the encoding using the Python chardet library[2]

- CSV Dialect:
  A CSV dialect contains information about the used delimiter, line terminator or quote characters. We guess the dialect using the Python csvkit library[3] *(a slight modification of the original CSV library*[4]*).*

- CSV Deviation:
  A CSV file is well defined, consisting of a optional header row and several data rows, each of them with the same number of columns. Ermilov et al. define a canonical model for tabular data and a set of deviations which provide interesting insights [1]. The deviations are grouped into three categories:
  - *the table level, that is, leading whitespaces or multi-tables),...*
  - *the header level, that is, duplicate or missing header values, or a inconsistent number of header columns compared to the data columns) , ...*
  - *the data level, that is, duplicate data rows, missing or incomplete data values, ...*

3 Findings

TOTAL	SUCCESS	404	PARSER ERRORS
74395	56528 75.98%	1653 2.22%	16214 21.79%

Table 1: General statistics

We conducted the latest profiling consisting of 74395 on November, the 9th 2014. A total of 56528 could be downloaded and analysed, while we received for 1653 files a `404 NOT FOUND` HTTP status code and for 16214 documents a parser error (mainly due to wrong detected encodings or malformed formats) (cf. Table 1 and Table 8 for parser errors). Table 2 shows the extracted file extensions together with the number of documents. We can see that most documents use the ".csv" file extension as specified in the RFC. The other interesting observation is that around 4421 documents do not have a file extensions, mainly due to the reason that those documents are exposed via APIs .

3.1 HTTP Header field:

Next, we analyse the values of the `content-type` fields in the HTTP Headers for the successful downloaded documents. Table 3 shows the results for the extracted content-

[2] https://pypi.python.org/pypi/chardet

[3] http://csvkit.readthedocs.org/en/0.9.0/

[4] https://docs.python.org/2/library/csv.html

	#DOCS	
csv	**63542**	(85.41%)
	4421	(5.94%)
none	4143	(5.57%)
zip	476	(0.64%)
aspx	253	(0.34%)
html	219	(0.29%)
xls	212	(0.28%)
tsv	184	(0.25%)
ashx	177	(0.24%)
xml	111	(0.15%)
others	657	(0%)

Table 2: File extensions

types, with the RFC recommended content type in bold. The good news are that most documents are correctly identified as CSV files with the a specified content type of "text/csv".

Table 4 lists the amount of documents that had an optional charset information in the `content-type` field. A total of 48474 documents do not define a specific encoding. This is especially problematic since many of those CSV files contain specific characters (e.g. "Umlauts" as found in the German alphabet). One needs to inspect and guess the encoding without that clear declaration in the header fields.

3.2 Content Encoding:

Next, we inspect the actual content encoding of the downloaded files by guessing the right encoding based on the first hundreds of lines. Table 5 lists our guessed charsets by inspecting the first 100 lines of each documents. Naturally, we observe a mix of different encodings across the documents with the "ascii" encoding as dominating ones. Note, we did not verify the correctness of the encoding in this step and purely report on the results obtained by using the the Python library.

3.3 Usage of delimiteres:

We used the detected encoding and parse the documents to identify the delimiter character for each CSV file. The results in Table 6 show again that most documents follow the RFC specification of using the ',' as delimiter. Please note, that we did not verify the correctness of the guessed delimiters at this stage and purely report the guessed delimiters from the Python library.

	#DOCS	
text/csv	**33499**	(59.26%)
application/octet-stream	10257	(18.14%)
text/html	8320	(14.72%)
text/plain	1055	(1.87%)
application/vnd.ms-excel	843	(1.49%)
text/x-comma-separated-values	546	(0.97%)
text/comma-separated-values	467	(0.83%)
missing	430	(0.76%)
binary/octet-stream	218	(0.39%)
text/tab-separated-values	173	(0.31%)
others	720	(1.27%)

Table 3: `Content-Type` HTTP Header

3.4 Deviations

Eventually, we inspected the content of the CSV files and determined the deviation of the tables compared the canonical model as defined in [1]. To do so, we parsed the content using the guessed delimiters and encoding and inspected the table, headers and data for the various defined deviations. We added one more deviation to check if the cardinality of the data rows differ between different rows, indicated with “D-Cardinality”. An observed D-Cardinality deviation might indicate that we a) either have two tables, b) have the wrong delimiter or c) the table does not follow the strict requirements that each row has the same number of columns as specified in the RFC.

4 Conclusion

In this study, we profiled 74395 CSV-like files published on the Web as Open Data in CKAN portals. The main focus of this study is to extract some core features common to CSV files, such as, the used file extension, the defined content-type and used delimiter. In addition, we determine the content-encoding and table deviations.

Our findings show that the majority of the CSV files adhere to the RFC4180 specification, that is, the use of *.csv* as file extension, `text/csv` as the HTTP response header `content-type` , and ’,’ as delimiter. The findings also show that there exists for the majority of the documents no information about the content encoding in the HTTP headers. Regarding deviations, the major observed deviations are that data tables contain rows in which one or several data cells occupy multiple columns and that one or several data cells are empty. In addition, we detected in 46662 documents (82%) at least one type of deviation.

While this report presents some general and potentially interesting findings, further work is necessary to get a full picture of the landscape for the CSV files on the Web.

	#docs	
none	48474	(85.75%)
utf-8	6994	(12.37%)
iso-8859-1	976	(1.73%)
	30	(0.05%)
unicode (utf-8)	16	(0.03%)
utf8	14	(0.02%)
windows-1252	5	(0.01%)
windows-1250	2	(0%)
name=”03_cani_residenti_x_razza.csv”	1	(0%)
name=”01_elenco_canili.csv”	1	(0%)
others	15	(0.03%)

Table 4: Encoding specified in Content-Type HTTP Header

Firstly, we did no verify that the delimiters and encoding are correctly guessed which potentially has an impact on the deviation analysis. Secondly, we plan to extend our profiling to report on the shapes and types of tables we find on the Web and provide a more comprehensive deviation analysis. Thirdly, we will analyse the content and header information to see “what” kind of data is published and how “easy” it is for a machine to understand the internal structure and meaning of the information. Very often, the data in tables can be of tree shape, where branch nodes represent categories (e.g., “male” vs “female“), or the used headers can be mapped to a public knowledge graph (e.g. freebase, dbpedia or wikidata).

Acknowledgments This work was partially funded by the “Jubiläumsfond der Stadt Wien”.

References

[1] Ivan Ermilov, Sören Auer, and Claus Stadler. “User-driven semantic mapping of tabular data”. In: *I-SEMANTICS 2013 - 9th International Conference on Semantic Systems, ISEM ’13, Graz, Austria, September 4-6, 2013*. Ed. by Marta Sabou, Eva Blomqvist, Tommaso Di Noia, Harald Sack, and Tassilo Pellegrini. ACM, 2013, pp. 105–112. isbn: 978-1-4503-1972-0.

	#docs	
ascii	24084	(42.61%)
ibm855	9921	(17.55%)
iso-8859-2	5952	(10.53%)
utf-8	4909	(8.68%)
windows-1252	4547	(8.04%)
windows-1251	2628	(4.65%)
shift_jis	2117	(3.75%)
cp932	717	(1.27%)
iso-8859-5	713	(1.26%)
ibm866	581	(1.03%)
others	359	(0.64%)

Table 5: Detected encodings

	#docs	
,	**45527**	(80.54%)
;	5248	(9.28%)
	3049	(5.39%)
\t	2171	(3.84%)
:	475	(0.84%)
—	58	(0.10%)

Table 6: Identified delimiters

	#DOCS	
with deviations	46662	(82.55%)
T-Metadata	5096	(9.02%)
T-Multiple	4866	(8.61%)
T-Whitespace	924	(1.63%)
H-Incomplete	6984	(12.35%)
H-Multiple-column-cell	6045	(10.69%)
H-Missing	5601	(9.91%)
H-Cardinality	4051	(7.17%)
H-Multiple-header-rows	2178	(3.85%)
H-Duplicate	601	(1.06%)
D-Multiple-column-cell	38259	(67.68%)
D-Incomplete	32810	(58.04%)
D-Missing	5429	(9.60%)
D-Duplicate	4453	(7.88%)
D-Cardinality	3906	(6.91%)

Table 7: Deviations according to the definition of [1].

	#DOCS	
cannot guess dialect	4292	(7.59%)
newline inside string	2577	(4.56%)
chardet: encoding not detected	871	(1.54%)
	299	(0.53%)
new-line character seen in unquoted field - do you need to open the file in universal-newline mode?	263	(0.47%)
line contains null byte	98	(0.17%)
field larger than field limit (131072)	6	(0.01%)

Table 8: Parser error messages.