Master Thesis Proposal

Extracting Tables From Plain Text

Leonardo Hübscher

Supervisor
Prof. Dr. Felix Naumann

Advisor
Lan Jiang

Information Systems
Hasso Plattner Institute
Winter semester 2020/2021

1 Introduction

In April 2020, 59% of the world population had access to the internet and created 2,500,000 Terabytes of data per day. (Domo 2020) It is expected, that by 2025 75% of the world population will be online, resulting in a total amount of $1,7510^{11}$ Terabyte of data in existance.(Reinsel et al. 2018)

However, the data is only useful if someone gains value out of it. Data scientists are experts at developing machine learning algorithms that can extract value from high volumes of data automatically. An essential step towards developing these algorithms is the acquisition and standardization of such data. Different surveys on this topic suggest that data scientists have to deal with data wrangling between 26%(Mooney n.d.) to 80%(Press 2016) of their time. Even if a data scientist would have to spend only a fourth of his working time on data preparation, this would certainly not scale with the expected growth of data in the foreseeable future and the collected data might become worthless.

This master thesis aims at reducing the time data scientists spend with data wrangling. By automating the data preparation, data scientists should be enabled to focus on further processing.

To think about how we can automate data preparation, we need to know where data scientists obtain their data. Amongst other sources, data scientists browse open data portals to access data from organizations and governments to develop their machine learning algorithms. One possible implementation of such an open data portal is a *data lake*. Information that is available on data lakes has no exact use case at the time of uploading, thus the optimal storage schema is unknown. The lack of standardization increases the complexity of subsequent data preparation. To ensure a high inter-compatibility nevertheless, data is often stored as plain-text in the form of tables. A study showed that tabular data makes ~10% of all resources available on such a portal.(Mitlöhner et al. 2016) As the structured, two-dimensional representation of information is easy to read for humans and machines, it additionally serves as a low-level interface between these two, making it easier for data scientists to work with the data at hand. One of the most commonly used text-based file formats for sharing tables is the comma-separated values format, which was first supported by IBM back in 1972. (IBM Corporation 1972) Surprisingly, its first official specification was published whole 33 years later in 2005 as part of the request for comments specification (RFC) 4180. (Shafranovich 2005) The absence of a specification for such a long time entails several challenges that complicate the data preparation task and therefore increase the manual labor required by data scientists.

In this master thesis, we want to eliminate these challenges by automatically extracting tables from plain-text files and exporting them into a standardized format.

2 Challenge

To be able to export tables from CSV and plain-text files, we need to tackle two big challenges.

The first challenge will become clearer after looking at the request for comments specification (RFC) 4180, which specifies the comma-separated values (CSV) file format. A CSV file is a text-based file, containing multiple lines, where each line represents a record. Optionally, the first line might represent the header row of the table. Special characters delimit the records and fields or escape text accordingly. While there exists a suggestion for characters that should be used for delimiting, there is no standard specified. The RFC document states: "[...] there is no formal specification in existence, which allows for a wide variety of CSV files." (Shafranovich 2005) The lack of standardization has lead to the existence of multiple variants of the CSV specification, which are called *dialects*. On the data lake Mendeley[1], only every fifth CSV table matches the dialect of RFC 4180.[2] The main problem with having multiple dialects is that the mapping for the special characters used for delimiting and escaping does not get stored along with the CSV file. The first challenge is the correct determination of the used dialect.

The second challenge has to do with the fact that data lakes do not only host CSV files but also plain-text files. As the main purpose of a data lake is to exchange data in a machine-readable way, we assume that plain-text files contain structured information, too. However, plain-text files are more difficult as they might contain metadata, multiple tables (with different delimiters), or data that is structured by layout. Each difference makes the problem of extracting the relevant data more challenging.

In the next chapter, we will give a short overview of how other scientists tackled the challenges of guessing the delimiter and extracting tabular information from plain-text files.

3 Related Work

In this section, we want to briefly reflect on related work that already exists for the topics of table extraction and delimiter detection.

In "Design of an end-to-end method to extract information from tables" the authors present a system that extracts tabular data from financial statements while allowing for user feedback and error correction to improve the results.(e Silva et al. 2006) It splits the process of table extraction into five steps: (i) Location (ii) Segmentation (iii) Functional Analysis (iv) Structural Analysis (v) Interpretation. In the master thesis, we will be focusing on the first two steps: *Location* and *Segmentation*. The first step deals with the separation of the table from its surrounding document. The second step covers the detection of the table's physical model, such as rows and columns. They propose to

[1]`https://data.mendeley.com/`

[2]Number stated by Lan Jiang

run the *Location* step twice. Once before the *Segmentation*-step as a filter to detect non-tabular lines and once after, to eliminate non-data lines. (e Silva et al. 2006)

The same authors cover the topic of table area detection in a second paper in greater detail. On financial PDF documents, the algorithm removed 50% of the total lines and detected 99% of the table lines. They achieved high accuracy by looking at the distribution of contiguous inner spaces within a line. However, working on transformed PDF-documents differs slightly from our approach, as it allows them to incorporate a different set of features. In their future work, they suggest using the incidence of keywords as a feature for determining whether a line is part of a table. (e Silva et al. 2003)

A paper that covers the *Segmentation*-step is "Multi-Hypothesis CSV Parsing", which is based on a master thesis from 2016(Döhmen 2016). In contrast to the prior papers, they work on governmental data, thus not only being restricted to financial data. The paper deals with the identification of the dialect of a CSV document. However, its main contribution relies on changing the way of decision making. While other solutions apply multiple heuristics sequentially, they propose to have a decision tree with the goal to do not decide too early. The tree allows considering different variants of the dialect. The variants get scored by applying some quality models. When it comes to evaluation, we need to say that on the one hand side it lacks a proper dataset size, as it only consists of 64 tables. On the other hand, the paper makes the ground truth available publicly. Additionally, different solutions have been used as a baseline to compare to which might be helpful for our evaluation. As the authors published their code, a direct comparison becomes possible. In the end, the authors stress the importance of detecting the column data type. (Döhmen et al. 2017)

The authors of the paper "Wrangling messy CSV files" developed an approach that takes data types more into account. They have been able to detect the proper dialect in 97% of the cases on a dataset that consists of governmental data and files from GitHub repositories. Their approach determines the dialect of a CSV file by considering the tables' consistency, based on the row length and the data type patterns. The algorithm chooses the delimiters to reduce the number of patterns. However, the algorithm only works on a fixed set of data types and delimiters. (van den Burg et al. 2019)

The research area around Optical Character Recognition (OCR) also deals with the proper detection of tables. "A survey of table recognition" indicates that this is not directly applicable to plain-text files, since OCR can exploit the existence of structural features more intensively, whereas for plain-text documents it is not guaranteed to have visual structures. (Zanibbi et al. 2004)

In "Pytheas: Pattern-based Table Discovery in CSV Files" they propose a solution that skips determining the CSV dialect and rather focuses on the line classification and assignment of lines to tables. The main idea is to use a flexible set of *fuzzy rules* whose weight for the classification is determined by a supervised machine learning algorithm. They evaluated their approach on 4,500 files from four different open data portals, showing that their algorithm has very high precision and recall of 95%. As an interesting side note, they only discovered vertically stacked tables within their dataset, meaning that no line contained multiple tables. (Christodoulakis et al. 2020)

Two papers that deal with the *Functional Analysis*, as introduced in (e Silva et al. 2006), try to detect the schema of an extracted table. The papers "Table Extraction Using Conditional Random Fields" as well as "Schema Extraction for Tabular Data on the Web" use conditional random fields to label table row types. The later one introduces a new concept of logarithmic binning, which seems to be more promising than the first paper, even though the described features are based on web tables that indulge with more structural information due to semi-structured Hypertext Markup Language (HTML). However, both approaches require a more labor-intensive dataset, as each row needs to get labeled. (Pinto et al. 2003) (Adelfio & Samet 2013)

4 Goal

The goal of the master thesis is the development of an algorithm that can detect and extract tables from text-based files. The problem can be split into two sub-goals: one covering the extraction of tables from CSV-like files and one handling the extraction from plain-text files. While the former covers the proper detection of a tables dialect, the latter deals with the separation of text into tabular and non-tabular segments. In this section, we want to discuss the required preparations, assumptions, and the evaluation of our approach.

As a prerequisite, we will need to acquire and prepare a dataset that we can work with. We already have a dataset at hand from prior work, which was fetched from the data lake Mendeley and contains a great variety of files. We first need to partition the files by determining whether they comply with RFC 4180 (*non-faulty* tables) or not (*faulty* tables). This allows us to analyze the faulty tables in greater detail and gather learnings that we can apply when designing our algorithm.

After fulfilling the prerequisites, we can start working on the defined sub-goals. While doing that, we have to make several assumptions, which are listed below. The first list contains assumptions that make the problem more generic.

1. The algorithm should be able to handle files that have multiple, vertically stacked tables.
2. A table might consist of multiple delimiters.
3. A delimiter might consist of multiple characters.
4. Tables can be structured using layout instead of CSV-like delimiter.
5. The algorithm should handle styling, such as borders, properly.
6. The approach should work with all typical table sizes.

The second list contains restrictions that make the problem more specific.

1. The encoding of the text files is known beforehand.
2. The algorithm does not correct errors that are already present in the source document.
3. As the input files are static, scalability is not a big concern. We will aim for a solution that can take up to several seconds.

Both lists are subject to change during the work on the master thesis.

After developing an approach, we need to evaluate it. We might either use the same dataset as before or create a second one, to increase the variety of input even further. Both datasets would require a more in-depth analysis to make the evaluation comprehensible. To evaluate the approach, we also need to label data to build up our ground truth. For each line of a plain-text file, we need metadata that includes (i) a line type, which can be one of: no table/ header/ data/ other (ii) field delimiters (iii) escape characters. The development of a tool might be required to help with the labeling process. However, we might also be able to use existing labeling tools, such as *DeExcelarator*[3] from TU Dresden.

The evaluation itself should cover the accuracy and precision of both goals individually and in combination. We will compare the results to a baseline that might consists of Sniffer[4], Messy[5] and Multi-Hypothesis CSV Parsing(Döhmen et al. 2017). Besides analyzing the key metrics, we want to investigate error cases, to derive further learnings for future work.

If our solution can hold up with the other approaches or even outperform them, we might implement a short demo website as well. The website should allow for easy testing by uploading a plain-text file and getting the extracted tables returned as CSV in an uniform way. The website should help future scientists to evaluate the application without the need to setup up anything and might also serve to collect more difficult plain-text examples.

In case we have some spare time left after tackling the main goal, we might also work on an addition, which deals with the row type detection of tables. We could imagine applying the approach from "Schema Extraction for Tabular Data on the Web" (Adelfio & Samet 2013) to our extracted plain-text tables.

4.1 Milestones

The master thesis has to be written within six months, which equals to roughly 26 weeks. A possible schedule is shown in Table 1. It does not include the proposed demo website nor the row type detection, as we consider them not to be an essential part of the master thesis. Tasks may overlap time-wise.

[3]`https://wwwdb.inf.tu-dresden.de/research-projects/deexcelarator/`

[4]`https://docs.python.org/3/library/csv.html`

[5]`https://messytables.readthedocs.io/en/latest/`

Estimted effort in weeks	**Task**
2	Preparation of the existing Mendeley dataset by partitioning the tables into faulty/ non-faulty. High-level analysis of the dataset to gain insights.
3	Implementation of an approach from related work to classify whether a line contains flowing text with graphical components.
2	Implementation of the evaluation tool.
4	Implementation of a novel approach to determine the CSV dialect.
2	Time buffer for primary goal. Might be used for the additional row type detection.
2	Crawling a second dataset and analyzing it.
3	Evaluation
8	Write master thesis

Table 1: Time estimation for milestones

References

Adelfio, M. & Samet, H. (2013), ‘Schema extraction for tabular data on the web’, *Proceedings of the VLDB Endowment* **6**, 421–432.

Christodoulakis, C., Munson, E. B., Gabel, M., Brown, A. D. & Miller, R. J. (2020), ‘Pytheas: Pattern-Based Table Discovery in CSV Files’, *Proc. VLDB Endow.* **13**(12), 2075–2089.
URL: *https://doi.org/10.14778/3407790.3407810*

Döhmen, T. (2016), ‘Multi-Hypothesis Parsing of Tabular Data in Comma-Separated Values (CSV) Files’, (September).
URL: *https://homepages.cwi.nl/ boncz/msc/2016-Doehmen.pdf*

Döhmen, T., Mühleisen, H. & Boncz, P. (2017), Multi-Hypothesis CSV Parsing, *in* ‘Proceedings of the 29th International Conference on Scientific and Statistical Database Management’, SSDBM ’17, Association for Computing Machinery, New York, NY, USA.
URL: *https://doi.org/10.1145/3085504.3085520*

Domo (2020), ‘Data Never Sleeps 8.0’.
URL: *https://www.domo.com/learn/data-never-sleeps-8*

e Silva, A. C., Jorge, A. & Torgo, L. (2003), Automatic Selection of Table Areas in Documents for Information Extraction, *in* F. M. Pires & S. Abreu, eds, ‘Progress in Artificial Intelligence’, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 460–465.

e Silva, A., Jorge, A. & Torgo, L. (2006), 'Design of an end-to-end method to extract information from tables', *Document Analysis and Recognition* **8**, 144–171.

IBM Corporation (1972), 'IBM FORTRAN Program Products for OS and the CMS Component of VM/370 General Information'.
URL: *http://bitsavers.trailing-edge.com/pdf/ibm/370/fortran/GC28-6884-0_IBM_FORTRAN_Program_Products_for_OS_and_CMS_General_Information_Jul72.pdf*

Mitlöhner, J., Neumaier, S., Umbrich, J. & Polleres, A. (2016), Characteristics of Open Data CSV Files, pp. 72–79.

Mooney, P. (n.d.), '2018 Kaggle Machine Learning & Data Science Survey'.
URL: *https://www.kaggle.com/paultimothymooney/2018-kaggle-machine-learning-data-science-survey*

Pinto, D., McCallum, A., Wei, X. & Bruce Croft, W. (2003), 'Table Extraction Using Conditional Random Fields', *SIGIR Forum (ACM Special Interest Group on Information Retrieval)* (SPEC. ISS.), 235–242.
URL: *https://people.cs.umass.edu/ mccallum/papers/crftable-sigir2003.pdf*

Press, G. (2016), 'Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says', *Forbes Tech* pp. 4–5.
URL: *https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#65a725686f63 https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-sc*

Reinsel, D., Gantz, J. & Rydning, J. (2018), 'The Digitization of the World - From Edge to Core. IDC White Paper', *IDC White Paper* **2018**(May).
URL: *https://www.seagate.com/files/www-content/our-story/trends/files/dataage-idc-report-final.pdf*

Shafranovich, Y. (2005), Common Format and MIME Type for Comma-Separated Values (CSV) Files, RFC 4180, RFC Editor.
URL: *https://www.rfc-editor.org/rfc/rfc4180.txt*

van den Burg, G. J. J., Nazábal, A. & Sutton, C. (2019), 'Wrangling messy CSV files by detecting row and type patterns', *Data Mining and Knowledge Discovery* **33**(6), 1799–1820.
URL: *https://doi.org/10.1007/s10618-019-00646-y*

Zanibbi, R., Blostein, D. & Cordy, J. R. (2004), 'A survey of table recognition', *Document Analysis and Recognition* **7**(1), 1–16.
URL: *https://doi.org/10.1007/s10032-004-0120-9*