Data Harvesting and Cleaning

Andrea Denger, Christian Kaiser, and Selver Softic
Group 4

Institute for Information Systems and Computer Media (IICM),
Graz University of Technology
A-8010 Graz, Austria

22 May 2013

Abstract

This survey comprehends main principles of data harvesting and cleaning and offers a short and compact outline over these two fields in theoretical and practical manner. The Harvesting part is focused on scraping rather then crawling, since scraping tables from HTML and PDF is a main point of matter of this survey. Hand in hand with literature study and tools introduction, short practical examples are included and discussed within the chapter about data harvesting. The examples chapter at the end of survey elaborates these tasks more precisely. The Data Cleaning chapter starts with some basic definitions and the literature study from the field of data cleaning and concludes the survey with the description of several promising off-line and on-line tools from science and software industry as well as discussing problems and issues from this field. Same as in the case of data harvesting, the examples chapter delivers practical insights on this problem. At the end some conclusions has been drawn regarding the elaborated topics.

Contents

Chapter 1

Introduction

Open data communities all over the world deal with the problem to get access to data which should be public, for example data of a cities public transport system or annual reports of companies owned by the government. As the companies are often not willing to publish raw data, the data is published in, for example, PDF format. "Government sites in particular are famous for publishing PDFs instead of raw data"[Rotter, 2013].Journalists and the open data communities therefore have to harvest the given data and additionally transform the data to raw data in order to ensure the possibility of running statistics or visualizations on it. This survey comprehends main principles of data harvesting and cleaning and offers a short and compact outline over these two fields in theoretical and practical manner including selected examples in a tutorial-style. The Harvesting chapter is divided into a motivation section including the literature study, a section about web crawling, a section about scraping, a short section about an introduction into tools and techniques and a conclusion section dealing with challenges and problems. The Data Cleaning chapter is divided into a motivation section including the literature study, a section about data quality, a section about the data cleaning process, a section each about an introduction into On-line and Off-line tools and services and a conclusion section dealing with challenges and problems. The examples chapter at the end of survey elaborates tasks more precisely in a tutorial-style about crawling a Twitter feed, extracting tables from PDF files and cleaning data using the OpenRefine[1] Tool.

[1]http://openrefine.org/

Chapter 2

Data Harvesting

2.1 Motivation and Definition

Data harvesting, known as scraping or mining, is a process to transform raw data into "useful" information. On the one hand, harvesting seems to hold the promise of reduced data acquisition and updated costs. On the other hand, online publishers spend time and effort protecting data from appropriation and unauthorized use from various types of harvester.

> "Web harvesting describes the process of gathering and integrating data from various heterogeneous web sources." [Gatterbauer, 2009]

> "'Scraping' data basically means to retrieve data from the web, stored in a less convenient format like HTML tables, and copy it into a format you can use such as a CSV file or database."[1]

Figure 2.1: Data / Web Harvesting.[2]

The internet is a huge data store which can be accessed from all over the world and there has been noted a rapid expansion of the web what causes an enormous growth of information. Information can vary from text, media or any other binary information. Information on web pages can be displayed in different formats and therefore posing difficulties in its harvesting which has led to increased difficulty in the extraction of "useful" and "potential" information. Web scraping therefore confronts this problem by harvesting explicit information from a number of websites for knowledge discovery and easy access. It is therefore important to realize that the web offers unprecedented challenge and opportunity to data harvesting.

[1]Ben Morris, http://www.sciprogblog.com/2012/01/scraping-data-with-python.html, online 2013-05-18
[2]http://www.flickr.com/photos/dff-jisc/4332033087/, online 2013-05-15

Loginworks Softwares[3] describes challenges for data harvesting and Web scraping in their blog[4] in the following way:

- *Huge amount of information.* Some information is found on the internet and it can range from one aspect to the other. Usually this information is more than what is needed. Therefore it is a great concern in getting the required information that is relevant for a personal purpose too.
- *Wide and diverse coverage of web information.* In the web almost all topics are covered. This is an opportunity to get the variety of information. Nevertheless it is still a challenge of getting information on a particular target from the wide and diverse audience.
- *All types of data are available on the web.* Information is usually stored in many formats like texts, multimedia, spreadsheets, structured tables and so on and so forth. Harvesting such kind of information is a task that may consume resources in terms of personnel, time and financial.
- *Most of the data is linked.* Information on the web is linked from one website to the other with several hyperlinks. When it comes to harvesting information from such sites that make the majority in the internet today, it is likely to mismatch information.
- *Most of the data is redundant.* The issue is that collected information that is found on a large number of web sites may be similar. This is because of banner advertisements, copyright notices, navigation panels and many others. It is therefore important to engage in web scraping as to solve such a kind of problem.
- *Deep web and surface web.* Surface data can be regarded as the data which will get by use of browser. There is more information that is protected from public users. This information may be more beneficial than other information which can be regarded as surface data. The web is ever dynamic.
- *It is a virtual society.* The Internet can be regarded as a virtual society based on the following reasons. The internet is never only about products and services, data, however it is about interactions about people, organizations and various automatic systems too. This usually poses a great challenge when it comes to harvesting of such data.

2.2 Web Crawler

2.2.1 Definition

A *Web crawler*, known as robot or spider, is a computer program that browses the World Wide Web in a methodical, automated manner to create an index of the data it is looking for. The first web robot appears to be the Wanderer by Matthew Gray, a physics student at Massachusetts Institute of Technology (MIT), written in the spring of 1993. Gray's World Wide Web Wanderer was designed to track the growth of the then-infant Web.

> "In Spring of 1993, I wrote the Wanderer to systematically traverse the Web and collect sites. I was initially motivated primarily to discover new sites, as the Web was still a relatively small place. The Wanderer was the primary tool for collection of data to measure the growth of the web. It was the first automated Web agent or "spider". The Wanderer was first functional in spring of 1993 and performed regular traversals of the Web from June 1993 to January 1996 [Gray, 1995]."

Manning et al. list the desiderata for Web crawlers in two categories. Firstly, features that web crawlers must provide and secondly, they should provide [Manning, Raghavan, and Schütze, 2008].

Features a crawler **must** provide [Manning, Raghavan, and Schütze, 2008 p. 443]:

[3] www.loginworks.com/

[4] www.loginworks.com/blogs/web-scraping-blogs/174-web-scraping-the-solution-to-data-harvesting

- *Robustness:* The Web contains servers that create spider traps, which are generators of web pages that mislead crawlers into getting stuck fetching an infinite number of pages in a particular domain. Crawlers must be designed to be resilient to such traps. Not all such traps are malicious; some are the inadvertent side-effect of faulty website development.

- *Politeness:* Web servers have both implicit and explicit policies regulating the rate at which a crawler can visit them. These politeness policies must be respected.

Features a crawler **should** provide [Manning, Raghavan, and Schütze, 2008 p. 444]:

- *Distributed:* The crawler should have the ability to execute in a distributed fashion across multiple machines.

- *Scalable:* The crawler architecture should permit scaling up the crawl rate by adding extra machines and bandwidth. Performance and efficiency: The crawl system should make efficient use of various system resources including processor, storage and network bandwidth.

- *Quality:* Given that a significant fraction of all web pages are of poor utility for serving user query needs, the crawler should be biased towards fetching "useful" pages first.

- *Freshness:* In many applications, the crawler should operate in continuous mode: it should obtain fresh copies of previously fetched pages. A search engine crawler, for instance, can thus ensure that the search engine's index contains a fairly current representation of each indexed web page. For such continuous crawling, a crawler should be able to crawl a page with a frequency that approximates the rate of change of that page.

- *Extensible:* Crawlers should be designed to be extensible in many ways - to cope with new data formats, new fetch protocols, and so on. This demands that the crawler architecture be modular.

The process is called *Web crawling* or spidering. Many sites, in particular search engines, use spidering as a means of providing up-to-date data. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. Crawlers can also be used for automating maintenance tasks on a Web site, such as checking links or validating HTML code. Also, crawlers can be used to gather specific types of information from Web pages, such as harvesting e-mail addresses, for example for spam.

The process of web harvesting can be divided into three subsequent tasks [Gatterbauer, 2009]:

- data or information retrieval, which involves finding relevant information on the Web and storing it locally. This task requires tools for searching and navigating the Web, i.e., crawlers and means for interacting with dynamic or deep web pages, and tools for reading, indexing and comparing the textual content of pages;

- data or information extraction, which involves identifying relevant data on retrieved content pages and extracting it into a structured format. Important tools that allow access to the data for further analysis are parsers, content spotters and adaptive wrappers;

- data integration which involves cleaning, filtering, transforming, refining and combining the data extracted from one or more web sources, and structuring the results according to a desired output format.

In general, a Web crawler starts with a list of URLs to visit, called the seeds. As the crawler visits URLs, hyperlinks on the page are identified and added to the list of URLs to visit next, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies.

2.2.2 Purposes

Some purposes of using Web crawlers are to monitor content change, visualize relations or identify clusters, for example a friend-network of a social network. Another purpose is in data collections, for example tweets using the Tag #tugraz. Crawlers can be scheduled, for example to run on a daily or monthly basis. Some On-line Crawlers, for example Scraperwiki[5], provide an API, for example to retrieve the crawled data JSON formatted.

There are several uses for web crawlers. Basically a Web crawler is used to collect information out on the Internet. Web search engines commonly use Web crawlers to collect information about what is available on public web pages. Their primary purpose is to collect data to provide relevant web sites if Internet surfers search for a keyword on the site. A related use is Web archiving, where large sets of web pages are periodically collected and archived for posterity. An example for web archiving is the Internet archive[6], a non-profit digital library with the stated mission of "universal access to all knowledge". Another use is web data mining, where web pages are analyzed for statistical properties, or where data analytics is performed on them. An example would be Attributor[7], a provider of digital content protection for the publishing industry **olston2010we**

Linguists may use a web crawler to perform a textual analysis to scour the Internet to determine what words are commonly used today. Market researchers may use a Web crawler to determine and assess trends in a given market.[8]

On the one hand Web crawler can adopt a pull model where the web is proactively searched for new or updated information, or it is tried to establish a convention and a set of protocols enabling content providers to push content of interest to the collectors. A general purpose is that Web crawler gather as many pages as possible from a particular set of URLs. According to Peshave [Peshave, 2005] there are two different techniques Web crawlers are able to go. A focused crawler is designed to only gather documents on a specific topic. Hence the goal of the focused crawler is to selectively seek out pages that are relevant to this pre-defined set of topics. As the size of the Web is growing it has become imperative to parallelize the crawling process in order to finish downloading the pages in a reasonable amount of time. A distributed Web crawling, it is a distributed computing technique whereby Internet search engines employ many computers to index the Internet via Web crawling.

2.3 Scraping

2.3.1 Data Scraping vs. Regular Parsing

> "The key element that distinguishes data scraping from regular parsing is that the output being scraped was intended for display to end-user, rather than as input to another program, and is therefore usually neither documented nor structured for convenient parsing[9]."

Data scraping is a technique in which a computer program extracts data from human-readable output coming from another program. Screen scraping is normally associated with the programmatic collection of visual data from a source, instead of parsing data as in web scraping.

2.3.2 PDF Scraping

Extracting information from PDF documents is called PDF scraping.

> "When you're looking for data to use in your own applications, you may find it's not always in the format you want. Government sites in particular are famous for publishing PDFs instead of raw data which is easier to parse[10]."

[5]http://www.scraperwiki.com
[6]http://archive.org/about/, online: 2013-05-18
[7]http://www.attributor.com/, online: 2013-05-18
[8]http://www.wisegeek.org/what-is-a-web-crawler.htm, online: 2013-05-18
[9]http://en.wikipedia.org/wiki/Data_scraping
[10]http://readidea.com/magazines/issue4/scraperwiki.html, online 2013-05-20

Some problems with PDF scraping will be:

- `hard to format`, for example tables of annual reports contain "null" values
- `correct data-type`, for example integers parsed as strings
- `large volume`, for example data tables which exceed one PDF page in length
- `rate of change`, for example for daily measurement data, need flexibility

PDFMiner[11] is a tool for extracting information from PDF documents. Unlike other PDF-related tools, it focuses on gathering and analyzing text data. PDFMiner allows to obtain the exact location of text in a page, as well as other information such as fonts or lines. It includes a PDF converter that can transform PDF files into other text formats (such as HTML). It has an extensible PDF parser that can be used for other purposes than text analysis.

The command line tool *pdf2txt.py* extracts text contents from a PDF file. It extracts all the text that are to be rendered programmatically, i.e. text represented as ASCII or Unicode strings. It cannot recognize text drawn as images that would require optical character recognition. It also extracts the corresponding locations, font names, font sizes, writing direction (horizontal or vertical) for each text portion. You need to provide a password for protected PDF documents when its access is restricted. You cannot extract any text from a PDF document which does not have extraction permission. Not all characters in a PDF can be safely converted to Unicode.

Example (pdf2txt.py):

```
\$ pdf2txt.py —P mypassword —o output.txt secret.pdf
```

Listing 2.1: command line extract a text from an encrypted PDF file

The command line tool *dumppdf.py* dumps the internal contents of a PDF file in pseudo-XML format. This program is primarily for debugging purposes, but it's also possible to extract some meaningful contents (such as images).

Example 1 (dumppdf.py):

```
\$ dumppdf.py —a foo.pdf
```

Listing 2.2: command - line dump all the headers and content, except stream objects

Example 2 (dumppdf.py):

```
\$ dumppdf.py —T foo.pdf
```

Listing 2.3: command - line dump the table of contents

Slate[12] is a Python package that simplifies the process of extracting text from PDF files. It depends on the PDFMiner package. Slate provides one class, PDF. PDF takes a file-like object and will extract all text from the document, presentating each page as a string of text

```
>>> with open('example.pdf') as f:
...     doc = slate.PDF(f)
...
>>> doc
[..., ..., ...]
```

[11]http://www.unixuser.org/ euske/python/pdfminer/index.html
[12]https://pypi.python.org/pypi/slate, online: 2013-05-18

```
>>> doc[1]
'Text from page 2...'
```

Listing 2.4: PDF - each page as a string of text

If the pdf is password protected, pass the password as the second argument:

```
>>> with open('secrets.pdf') as f:
...     doc = slate.PDF(f, 'password')
...
>>> doc[0]
"My mother doesn't know this, but..."
```

Listing 2.5: PDF - password protected

The *AddToIt Document Engine*[13] was designed with flexibility and performance in mind. Some of the features of the AddToIt Document Engine include:

- Retrieval of documents and data from multiple sources including websites
- Transformation of documents from unstructured to structured formats, e.g. PDF to XML
- Normalization of data within documents into a standard data form
- Document and data storage for fast retrieval from a database
- Delivery of data in multiple formats via multiple means (FTP/HTTP)

The AddToIt Document Engine is collecting data from HTML and PDF documents found on the web from within databases on disk or from email. It then transforms them into an XML document representation and normalizes the data. It stores the data in document and data form in the AddToIt database. Users of the data receive the data from the database on request (e.g. through a web interface) or it is delivered to them at regular intervals.

Tesseract[14] is a open source OCR engine available under the Apache 2.0 license[15]. Combined with the Leptonica Image Processing Library[16] it can read a wide variety of image formats and convert them to text in over 60 languages.

2.3.3 HTML Scraping

Beautiful Soup[17] is a Python library for parsing HTML documents (including having malformed markup, i.e. non-closed tags, so named after Tag soup). It creates a parse tree for parsed pages that can be used to extract data from HTML, so this library is useful for web scraping — extracting data from websites.

Three features make it powerful:

1. Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what is needed.
2. Beautiful Soup automatically converts incoming documents to Unicode and outgoing documents to UTF-8. No thinking about encodings, unless the document does not specify an encoding and Beautiful Soup cannot autodetect one. If so, the original encoding has to be specified.

[13]http://www.addtoit.com/screen-scraping_index.html
[14]http://code.google.com/p/tesseract-ocr/, online: 2013-05-18
[15]http://www.apache.org/licenses/LICENSE-2.0
[16]http://leptonica.com/
[17]http://www.crummy.com/software/BeautifulSoup/

3. Beautiful Soup sits on top of popular Python parsers like lxml[18] (The lxml XML toolkit is a Pythonic binding for the C libraries libxml2 and libxslt.) and html5lib[19] (html5lib is a pure-python library for parsing HTML. It is designed to conform to the WHATWG HTML specification, as is implemented by all major web browsers.), allowing everybody to try out different parsing strategies or trade speed for flexibility.

2.4 Tools and Techniques

2.4.1 Scraper Wiki

In 2003, the founders of *ScraperWiki* were the first people in the world to write computer programs to download Government information and make it more accessible to citizens.*ScraperWiki*[20] was founded in 2009 by Julian Todd and Aidan McGuire. They received initial funding from British TV station Channel 4. After fostering an active community of open data coders and data journalists, ScraperWiki won the *Knight News Challenge*[21] in 2011. In 2012 ScraperWiki closed a million dollar round of investment led by EV, a UK-based fund manager, which provides venture and growth capital of up to £2million to new or existing SMEs located throughout the UK and trading across technology and non-technology sectors, to improve their platform for coders and build out our business offering.[22]

Figure 2.2: ScraperWiki Logo.[23]

ScraperWiki is a screen scraping tool. It is a technology platform that allows programmers to write and schedule screen scrapers and store the data they generate. It is an online tool that makes finding and storing data simpler and more collaborative. The main use of the ScraperWiki website is providing a place for programmers and journalists to collaborate on analyzing public data.

2.4.2 pdftohtml

Pdftohtml[24] is a program that converts pdf documents into html and generates its output in the current working directory. The conversion process is performed using CLI (Windows Command Line Interpreter). Pdftohtml supports extracting images from the PDF document, complete document conversion. Furthermore it runs in

[18]http://lxml.de/
[19]https://github.com/html5lib/html5lib-python#readme
[20]https://scraperwiki.com/
[21]http://thenextweb.com/uk/2011/06/23/scraperwiki-a-uk-open-data-project-wins-280k-award/
[22]http://www.prweb.com/releases/2012/2/prweb9149117.htm
[23]https://scraperwiki.com/, online 2013-05-18
[24]http://pdftohtml.sourceforge.net/, online 2013-05-18

debug mode. It is a cross-platform application which works on Windows, Mac, and Linux. Pdftohtml was developed by Gueorgui Ovtcharov and Rainer Dorsch. It is based and benefits from Derek Noonburg's xpdf[25] package.

```
pdftohtml [options]  [ ]
  -f              : first page to convert
  -l              : last page to convert
  -q              : don't print any messages or errors
  -h              : print usage information
  -help           : print usage information
  -p              : exchange .pdf links by .html
  -c              : generate complex document
  -i              : ignore images
  -noframes       : generate no frames
  -stdout         : use standard output
  -zoom           : zoom the pdf document (default 1.5)
  -xml            : output for XML post-processing
  -hidden         : output hidden text
  -nomerge        : do not merge paragraphs
  -enc            : output text encoding name
  -dev            : output device name for Ghostscript (png16m, jpeg etc)
  -v              : print copyright and version info
  -opw            : owner password (for encrypted files)
  -upw            : user password (for encrypted files)
```

Listing 2.6: Usage of pdftohtml[26]

2.4.3 pdftable

Pdftable[27] is a python module and command line utility that analyzes XML output from the program pdftohtml in order to extract tables from PDF files and output them as CSV data. It makes it easier to automate the process of parsing tabular data contained within reports, ledgers, or other data sets that are only published in PDF. The Python script takes the XML input and extracts the founded tables to a CSV file via the package *pdftable.py*[28] one file for each table type.

The *pdftohtml.exe*[29] converts the given page of the downloaded PDF file to XML. The Command is shown in the next line.

```
pdftohtml -f 74 -l 75 -xml -stdout file.pdf| python pdftable -f file\%d.csv
```

Listing 2.7: Command on pdftohtml.exe

2.4.4 htmltable

There are some Python packages for parsing HTML tables available.

htmltable.py[30] is a small Python HTML parser to extract tables from HTML files. On the one side it extracts all tables from an HTML file and nested tables (tables in a table) are also supported. On the other side it flattens HTML tables into plain-text tables.

[25]http://www.foolabs.com/xpdf/, online 2013-05-18
[27]http://pdftable.sourceforge.net/, online 2013-05-20
[28]https://pypi.python.org/pypi/pdftable/1.0, online 2013-05-20
[29]http://pdftohtml.sourceforge.net/, online 2013-05-20
[30]https://github.com/sangjinhan/htmltable

```
html = open('sample.html').read()
extractor = HTMLTableExtractor()
tables = extractor.get_tables(html)
```

Listing 2.8: Usage:'tables' will be a list of tables. Each table is represented as a list of list.

Some notes on htmltable.py are that there is no external dependency. It works with the standard Python library. And it may incorrectly work against broken HTML files.

table_parser.py[31] is a simple HTML table parser. It turns tables (including nested tables) into arrays.

2.5 Challenges and Problems

There are many challenges in data harvesting. Whether data should be harvested from the Web or local databases. Many different business will deal with data harvesting. For example business out of real estate, journalism or in the field of the automotive industry. For example with eBay being one of the world's most leading ecommerce marketplace that envelopes a huge portfolio of businesses from all over the world, a lot of storeowners browse the web on a daily basis in search of product details and bid prices. Using web scraping service on an eBay store or ecommerce business will help to increase efficiency and productivity.

An important research problem is the optimal combination of automation (high recall) and human involvement (high precision). At which stages and in which manner a human user must interact with an otherwise fully automatic web harvesting system for optimal performance (in terms of speed, quality, minimum human involvement, etc.) remains an open question.

Problem scenarios are:

- Large volume of Web pages / PDFs
- High rate of changes on web pages
- Scanned PDFs need Optical Character Recognition (OCR)[32] to transform the images to data.

Web scraping may be against the terms of use of some websites. The enforceability of these terms is unclear. Crawlers can retrieve data much quicker and in greater depth than human searchers, so they can have a crippling impact on the performance of a site. Needless to say, if a single crawler is performing multiple requests per second and/or downloading large files, a server would have a hard time keeping up with requests from multiple crawlers.

Web site owners use the /robots.txt file to give instructions about their site to web robots.

```
User-agent: *
Disallow: /
```

Listing 2.9: Example /robots.txt

"User-agent: *" represents that this section applies to all robots.

"Disallow: /" submits to the robot should not visit any pages on the site.

There are two important considerations when using /robots.txt:

- robots can ignore your /robots.txt. Especially malware robots that scan the web for security vulnerabilities, and email address harvesters used by spammers will pay no attention.
- the /robots.txt file is a publicly available file. Anyone can see what sections of your server you don't want robots to use.

[31]https://github.com/niksite/table-parser/blob/master/table_parser.py

[32]http://en.wikipedia.org/wiki/Optical_character_recognition, online 2013-05-20

Chapter 3

Data Cleaning

3.1 Motivation and Definition

"Dirty Data" is a problem dated with the occurrence of first computers. Under this term useless "garbage data" is considered produced by computing process. Hand in hand with increasing number of implementations of customer relationship management (CRM) systems and data warehouses the problem of "Dirty Data" emerged and data hygiene became an important issue. Nowadays data cleaning became an integral part of data processing and maintenance. Meanwhile a broad range of methods intending to enhance the accuracy and usability of data has been developed. Within the series of reports made 2001 by Cutter Consortium about organizations that implement data warehouses for the purposes of business intelligence various sources of "Dirty Data" has been identified by interviewed candidates. The report has shown that most common sources of unclean data in practice come from:

- data missing from database fields
- lack of company wide coding standards for data
- poor data entries which includes typos, misspellings, abbreviations and duplicates
- poorely documented or old legacy data

Within the scientific community many authors has been investigating the issues around the sources of "Dirty Data". In their comprehensive survey on data cleaning [Müller and Freytag, 2003] distinguish three types of data anomalies: syntactical, semantic, and coverage anomalies.

As synctatical anomalies can be considered:

- Lexical errors - which are discrepancies between the structure of the data items and the specified format.
- Domain format errors - which specify errors where the given value for an attribute does not conform with the anticipated domain format.
- Irregularities - concerned with the non-uniform use of values, units and abbreviations

In their work authors sorted out following semantic anomalies subgroups:

- Integrity constraint violations - which describe tuples (or sets of tuples) that do not satisfy one or more of the integrity constraints.
- Contradictions - which are values within one tuple or between tuples that violate some kind of dependency between the values.
- Duplicates - two or more tuples representing the same entity from the mini-world.

- Invalid tuples - which represent by far the most complicated class of anomaly found in data collections. Hereby are meant tuples that do not display anomalies of the classes defined above but still do not represent valid entities from the mini-world.

Finally the third group of data anomalies which treats the coverage aspects distinguishes:

- Missing values - as the result of omissions while collecting the data.
- Missing tuples - result from omissions of complete entities existent in the mini-world that are not represented by tuples in the data collection.

Taking all this aspects into account most appropriate and most cited definition for data cleaning is given by [Rahm and Do, 2000]:

Data cleaning, also called data cleansing or scrubbing, deals with detecting and removing errors and inconsistencies from data in order to improve the quality of data.

3.2 Data Quality

In order to specify the measurement of data cleaning process success a set of metrics was necessary to be defined. Those definition has been delivered by previous works on this topic like [Naumann, 2002; Tonn, 2000]. All data outcomes of cleaning process which satisfy these criteria are considered to be of a high quality. This means that data quality is defined as an aggregated value over a set of quality criteria as stated in Naumann, 2002.

Starting with the quality criteria defined in [Naumann, 2002; Tonn, 2000], [Müller and Freytag, 2003] describe the set of criteria which are affected by comprehensive data cleansing and define how to assess scores for each one of them for an existing data collection. According to findings in Müller and Freytag, 2003, in order to measure the quality of a data collection, scores have to be assessed for each of the quality criteria. This assessment can be used then to detect the necessity of data cleansing for a data collection as well as to measure success of a performed data cleansing process. Other field of application the authors also describe is optimization of data cleaning by specifying priorities for each of the criteria which in turn influences the execution of data cleaning methods affecting the specific criteria.

3.3 Data Quality Criteria

The quality criteria defined for comprehensive data cleaning in [Müller and Freytag, 2003] form a hierarchy. Figure 3.1 shows the resulting hierarchy.

Those criteria has been defined by [Naumann, 2002; Müller and Freytag, 2003; Motro, 1989] as follows:
Accuracy in [Naumann, 2002] is defined as the quotient of the number of correct values in the data collection and the overall number of values. Accuracy can be also seen as an aggregated value over the quality criteria *Integrity*, *Consistency*, and *Density*.
Integrity is used as defined in Motro, 1989. It is further divided into the criteria *Completeness* and *Validity* and therefore again an aggregated value over quality criteria.
Completeness is defined as the quotient of entities from a set S being represented by a tuple in T and the overall number of entities in S.
Validity is the quotient of entities from S being represented by tuples in T and the total amount of tuples. For instance the percentage of tuples in T representing (valid) entities from S.
Consistency concerns the syntactical anomalies as well as contradictions. It is further divided into *Schema conformance* and *Uniformity* forming another aggregated value over quality criteria.
Schema conformance is the quotient of tuples in T conform to the syntactical structure defined by schema R and the overall number of tuples in T.
Uniformity is directly related to irregularities, for instance the proper use of values within each attribute.

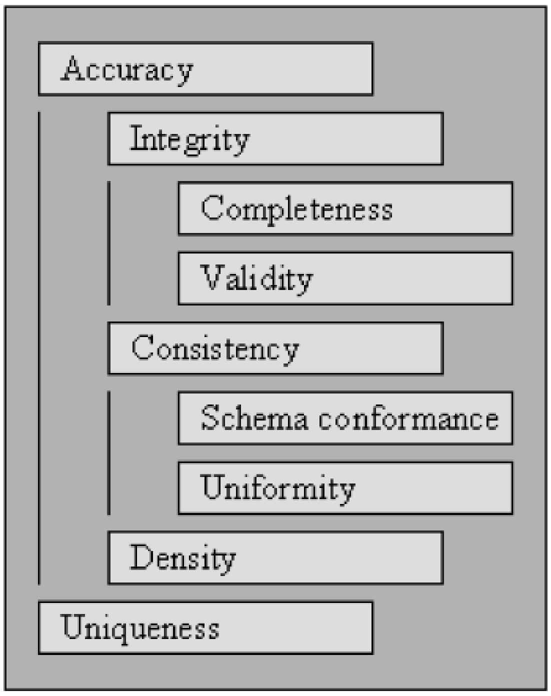
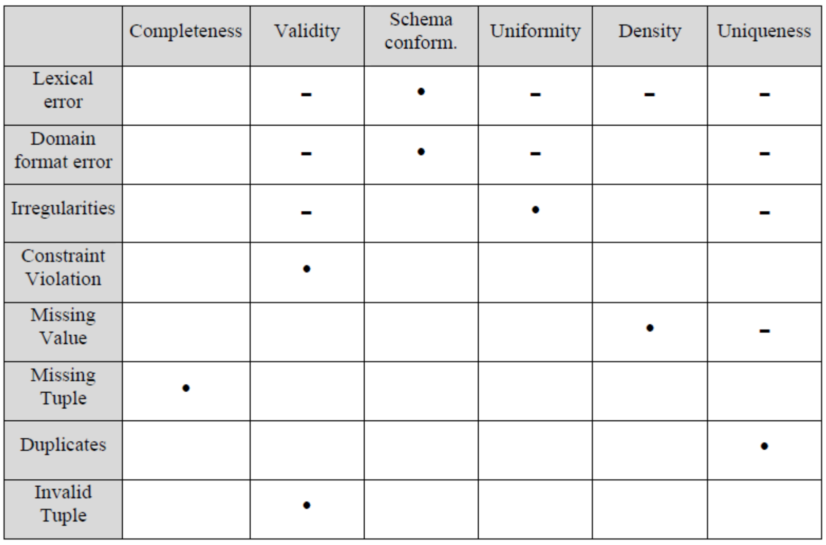


Figure 3.1: Hierarchy of data quality criteria as described in [Müller and Freytag, 2003].

Density is quotient of missing values in the tuples of S and the number of total values that known existing for a represented entity.
Uniqueness is the quotient of tuples representing the same entity in the mini-world and the total number of tuples in T. A collection that is unique does not contain duplicates.
Figure 3.2 lists the defined quality criteria and anomalies affecting them. Each point indicates direct downgrading of the quality criteria while dash indicates that the occurrence of this anomaly impedes the detection of other anomalies downgrading the quality criteria.



	Completeness	Validity	Schema conform.	Uniformity	Density	Uniqueness
Lexical error		-	•	-	-	-
Domain format error		-	•	-		-
Irregularities		-		•		-
Constraint Violation		•				
Missing Value					•	-
Missing Tuple	•					
Duplicates						•
Invalid Tuple		•				

Figure 3.2: How Data anomalies affecting data quality criteria from [Müller and Freytag, 2003].

3.4 Data Cleaning process

Müller and Freytag, 2003 define data cleaning process as "the entirety of operations performed on existing data to remove anomalies and receive a data collection being an accurate and unique representation of the mini-world". Generally seen it is usually a semi-automatic or automatic process which includes actions of formating, adaptations, validations, derivation of missing values as well removing contradictions between the entry values. Mostly this process finishes by merging and eliminating the duplicates as well by identification of outliers. The outcome of this process should be easier manageable and better fitting data.

3.4.1 Main Steps

According to Maletic and Marcus, 2000; Raman and Hellerstein, 2001 data cleaning process can be subdivided in three major steps:

- Auditing data to identify the types of errors and anomalies which have influence on data quality.
- Choosing appropriate methods to deal with identified problems.
- Applying the methods to the data.

Hereby the last two steps are considered as specification and execution of data cleaning work flow. Additionally to those three major steps Müller and Freytag, 2003 are adding the fourth "Post-processing or control step". their view of data cleaning process looks as follows:

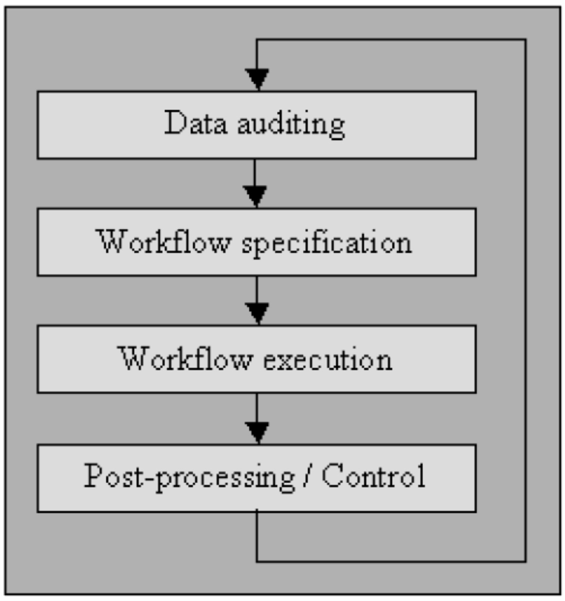


Figure 3.3: Data cleaning process steps by [Müller and Freytag, 2003].

Data cleaning process respecified by [Müller and Freytag, 2003] can be seen in Figure 3.3 and it is considered as endless loop where each iteration delivers a better quality of data.

3.4.2 Popular Methods

In this section we intend to give a short overview for the most popular methods used nowadays in the field of data cleaning:

Parsing is one of the most often and most applied techniques. Parsing in data cleaning is performed for the detection of syntax errors. A parser for is a program that decides for a given string whether it is an element of the defined context or not.

Data transformation intends to map the data from their given format into the format expected by the application ([Abiteboul et al., 1999]). The transformations affect the schema of the tuples as well as the domains of their values.

Integrity Constraint Enforcement - In general, integrity constraint enforcement ensures the satisfaction of integrity constraints after transactions modifying a data collection by inserting, deleting, or updating tuples have been performed. The two different approaches are *integrity constraint checking* and *integrity constraint maintenance*. I*ntegrity constraint checking* rejects transactions which violate some integrity constraint. *Integrity constraint maintenance* is concerned with identifying additional updates to be added to the original transaction to guarantee that the resulting data collection does not violate any integrity constraint. Latest findings in research on this topic has shown that integrity constraint enforcement is applicable without limitations only to some degree in data cleaning.

Duplicate elimination - data cleaning introduced several approaches for duplicate elimination. Every duplicate detection method proposed requires an algorithm for determine whether two or more tuples are duplicate representations of the same entity. Generally, for efficient duplicate detection every tuple has to be compared to every other tuple using this duplicate detection method.

Statistical Methods - [Maletic and Marcus, 2000] described the application of statistical methods in data cleaning. These methods can be used for the auditing of data as well as the correction of anomalies. Detection and elimination of complex errors representing invalid tuples go beyond the checking and enforcement of integrity constraints. They often involve relationships between two or more attributes that are very difficult to uncover and describe by integrity constraints. This can be viewed as a problem in outlier detection. Within this process the minorities of tuples and values that do not conform to the general characteristics of a given data collection are discovered.

3.5 Off-line Tools

This section is meant to deliver fast overview over most promising state of the art off-line tools for data cleaning which has been reviewed within this survey.

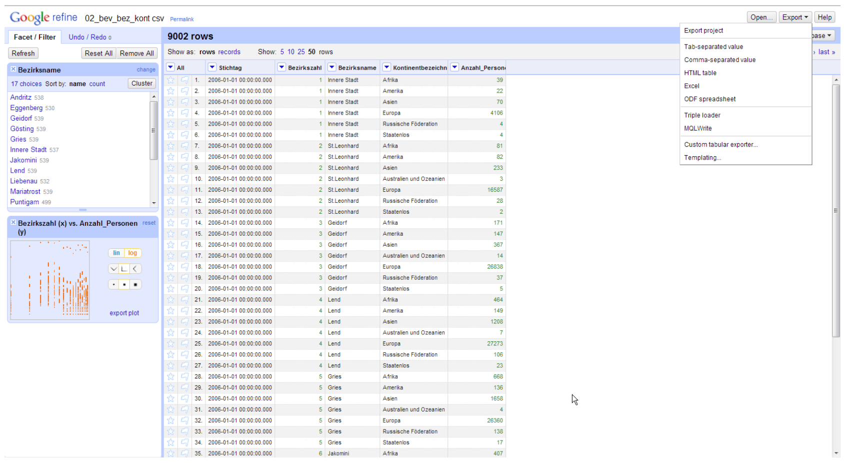
3.5.1 OpenRefine

OpenRefine (former known as Google Refine) is hosted as an open source project[1] and it is a standalone desktop application provided by Google for data cleanup and transformation to other formats. It is similar to spreadsheet applications but offers many functions that can be found in a database (filters, facets, regular expressions). An impression of OpenRefine user interface can be seen in figure 3.4.

OpenRefine operates in rows of data in the cell/column manner very similar to Microsoft Excel or database tables. One table represents one project, where user can filter the rows using facets that define the filter criteria. One of the important features of OpenRefine represents the action history which is stored withing the project. Although the OpenRefine represent a web application that runs on server, the software can be downloaded and run off-line on the local host. OpenRefine supports also transformation, normalizing of data as well different types of conversions. It is also possible to parse data from web, by using the URL fetch feature and to add data from the web to dataset by fetching it via web services for instance in JSON[2] format. OpenRefine inte-

[1]http://openrefine.org

[2]http://www.json.org/



9002 rows

Show as: rows records Show: 5 10 25 50 rows

All	Stichtag	Bezirkszahl	Bezirksname	Kontinentbezeichn	Anzahl_Person
1.	2006-01-01 00:00:00.000	1	Innere Stadt	Afrika	39
2.	2006-01-01 00:00:00.000	1	Innere Stadt	Amerika	22
3.	2006-01-01 00:00:00.000	1	Innere Stadt	Asien	70
4.	2006-01-01 00:00:00.000	1	Innere Stadt	Europa	4106
5.	2006-01-01 00:00:00.000	1	Innere Stadt	Russische Föderation	4
6.	2006-01-01 00:00:00.000	1	Innere Stadt	Staatenlos	4
7.	2006-01-01 00:00:00.000	2	St.Leonhard	Afrika	81
8.	2006-01-01 00:00:00.000	2	St.Leonhard	Amerika	82
9.	2006-01-01 00:00:00.000	2	St.Leonhard	Asien	233
10.	2006-01-01 00:00:00.000	2	St.Leonhard	Australien und Ozeanien	3
11.	2006-01-01 00:00:00.000	2	St.Leonhard	Europa	16587
12.	2006-01-01 00:00:00.000	2	St.Leonhard	Russische Föderation	28
13.	2006-01-01 00:00:00.000	2	St.Leonhard	Staatenlos	2
14.	2006-01-01 00:00:00.000	3	Geidorf	Afrika	171
15.	2006-01-01 00:00:00.000	3	Geidorf	Amerika	147
16.	2006-01-01 00:00:00.000	3	Geidorf	Asien	367
17.	2006-01-01 00:00:00.000	3	Geidorf	Australien und Ozeanien	14
18.	2006-01-01 00:00:00.000	3	Geidorf	Europa	26838
19.	2006-01-01 00:00:00.000	3	Geidorf	Russische Föderation	37
20.	2006-01-01 00:00:00.000	3	Geidorf	Staatenlos	5
21.	2006-01-01 00:00:00.000	4	Lend	Afrika	464
22.	2006-01-01 00:00:00.000	4	Lend	Amerika	149
23.	2006-01-01 00:00:00.000	4	Lend	Asien	1208
24.	2006-01-01 00:00:00.000	4	Lend	Australien und Ozeanien	7
25.	2006-01-01 00:00:00.000	4	Lend	Europa	27273
26.	2006-01-01 00:00:00.000	4	Lend	Russische Föderation	106
27.	2006-01-01 00:00:00.000	4	Lend	Staatenlos	23
28.	2006-01-01 00:00:00.000	5	Gries	Afrika	668
29.	2006-01-01 00:00:00.000	5	Gries	Amerika	136
30.	2006-01-01 00:00:00.000	5	Gries	Asien	1658
31.	2006-01-01 00:00:00.000	5	Gries	Australien und Ozeanien	4
32.	2006-01-01 00:00:00.000	5	Gries	Europa	26360
33.	2006-01-01 00:00:00.000	5	Gries	Russische Föderation	138
34.	2006-01-01 00:00:00.000	5	Gries	Staatenlos	17
35.	2006-01-01 00:00:00.000	6	Jakomini	Afrika	407

Figure 3.4: Data cleaning with OpenRefine.

grates augmentation of datasets with data from Freebase[3] as well contributing data to Freebase using Schema Alignment feature. This involves reconciliation, mapping string values in cells to entities in Freebase. One of the fancy features are also scatter plot based query capabilities. OpenRefine data import is supported from following formats:

- TSV (Tab Separated Values), CSV (Comma Separated Values)
- Text file with custom separators or columns split by fixed width
- XML (Extensible Markup language)[4]
- RDF (Resource Description Framework) triples (RDF/XML and Notation3 serialization formats)[5]
- JSON (JavaScript Object Notation)
- Google Spreadsheets, Google Fusion Tables
- If input data is in a non-standard text format, it can be imported as whole lines, without splitting into columns, and then columns extracted later with Refine's tools.
- Archived and compressed files are supported (.zip, .tar.gz, .tgz, .tar.bz2, .gz, or .bz2) and Refine can download input files from a URL.
- To use web pages as input, it is possible to import list of URLs and then invoke a URL fetch function.

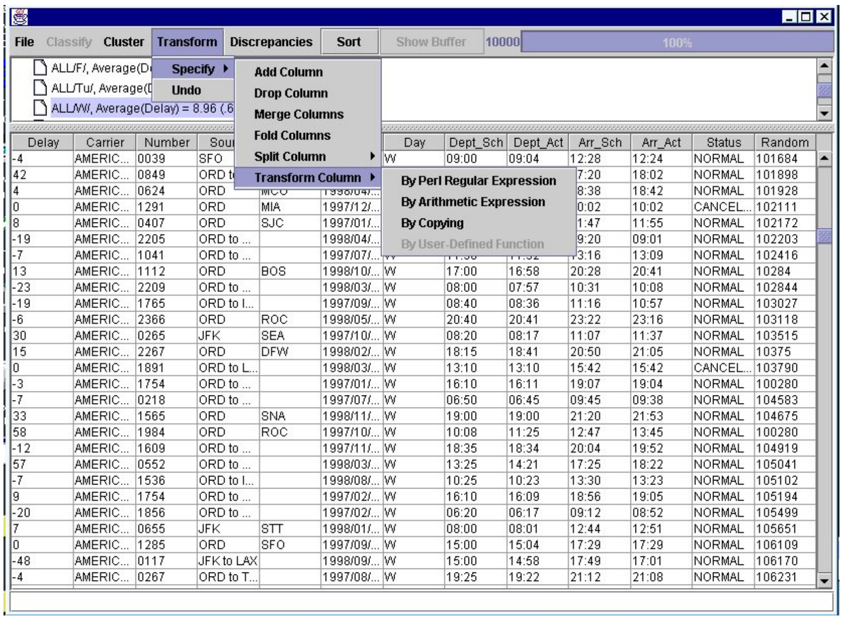
Export is supported as:

- TSV (Tab Separated Values), CSV (Comma Separated Values)
- Microsoft Excel
- HTML table
- ODF Spreadsheet[6]
- Templating exporter: it is possible to define custom template for outputting data, for example as MediaWiki table.
- Whole OpenRefine projects in native format can be exported as a .tar.gz compressed archive.

[3]http://www.freebase.com
[4]http://www.w3.org/XML/
[5]http://www.w3.org/RDF/
[6]http://en.wikipedia.org/wiki/OpenDocument

3.5.2 Potter's Wheel A-B-C

Potter's Wheel A-B-C[7] is an interactive tool for data cleaning and data analysis implemented by the University of Berkeley which tightly integrated operations like: data transformation, discrepancy detection, and data analysis. Users build stepwise transformations by adding or undoing transforms, in an intuitive manner through a spreadsheet-like interface. At the same time synchronously to these actions the effect of a transform is shown at once on records visible on screen. The system automatically checks for discrepancies in the current transformed version of the data, and flags them to the user as they are found. Users can specify transforms through graphical operations, or by specifying examples of the desired effect, and need not worry about specifying complex regular expressions, grammars, or custom scripts. Discrepancy detection is done in a highly customizable manner. Figure 3.5 shows Potter's Wheel A-B-C in action. Last version that was released is 1.3 and dates from year 2000.



ALL/F/, Average(D
ALL/Tu/, Average(
ALL/W/, Average(Delay) = 8.96 (.6

Delay	Carrier	Number	Sour			Day	Dept_Sch	Dept_Act	Arr_Sch	Arr_Act	Status	Random
-4	AMERIC...	0039	SFO			W	09:00	09:04	12:28	12:24	NORMAL	101684
42	AMERIC...	0849	ORD t						7:20	18:02	NORMAL	101898
4	AMERIC...	0624	ORD	MCO	1998/04/...				8:38	18:42	NORMAL	101928
0	AMERIC...	1291	ORD	MIA	1997/12/...				0:02	10:02	CANCEL...	102111
8	AMERIC...	0407	ORD	SJC	1997/01/...				1:47	11:55	NORMAL	102172
-19	AMERIC...	2205	ORD to ...		1998/04/...				9:20	09:01	NORMAL	102203
-7	AMERIC...	1041	ORD to ...		1997/07/...				3:16	13:09	NORMAL	102416
13	AMERIC...	1112	ORD	BOS	1998/10/...	W	17:00	16:58	20:28	20:41	NORMAL	10284
-23	AMERIC...	2209	ORD to ...		1998/03/...	W	08:00	07:57	10:31	10:08	NORMAL	102844
-19	AMERIC...	1765	ORD to I...		1997/09/...	W	08:40	08:36	11:16	10:57	NORMAL	103027
-6	AMERIC...	2366	ORD	ROC	1998/05/...	W	20:40	20:41	23:22	23:16	NORMAL	103118
30	AMERIC...	0265	JFK	SEA	1997/10/...	W	08:20	08:17	11:07	11:37	NORMAL	103515
15	AMERIC...	2267	ORD	DFW	1998/02/...	W	18:15	18:41	20:50	21:05	NORMAL	10375
0	AMERIC...	1891	ORD to L...		1998/03/...	W	13:10	13:10	15:42	15:42	CANCEL...	103790
-3	AMERIC...	1754	ORD to ...		1997/01/...	W	16:10	16:11	19:07	19:04	NORMAL	100280
-7	AMERIC...	0218	ORD to ...		1997/07/...	W	06:50	06:45	09:45	09:38	NORMAL	104583
33	AMERIC...	1565	ORD	SNA	1998/11/...	W	19:00	19:00	21:20	21:53	NORMAL	104675
58	AMERIC...	1984	ORD	ROC	1997/10/...	W	10:08	11:25	12:47	13:45	NORMAL	100280
-12	AMERIC...	1609	ORD to ...		1997/11/...	W	18:35	18:34	20:04	19:52	NORMAL	104919
57	AMERIC...	0552	ORD to ...		1998/03/...	W	13:25	14:21	17:25	18:22	NORMAL	105041
-7	AMERIC...	1536	ORD to I...		1998/08/...	W	10:25	10:23	13:30	13:23	NORMAL	105102
9	AMERIC...	1754	ORD to ...		1997/02/...	W	16:10	16:09	18:56	19:05	NORMAL	105194
-20	AMERIC...	1856	ORD to ...		1997/02/...	W	06:20	06:17	09:12	08:52	NORMAL	105499
7	AMERIC...	0655	JFK	STT	1998/01/...	W	08:00	08:01	12:44	12:51	NORMAL	105651
0	AMERIC...	1285	ORD	SFO	1997/09/...	W	15:00	15:04	17:29	17:29	NORMAL	106109
-48	AMERIC...	0117	JFK to LAX		1998/09/...	W	15:00	14:58	17:49	17:01	NORMAL	106170
-4	AMERIC...	0267	ORD to T...		1997/08/...	W	19:25	19:22	21:12	21:08	NORMAL	106231

Figure 3.5: Doing transformations with Potter's Wheel A-B-C.

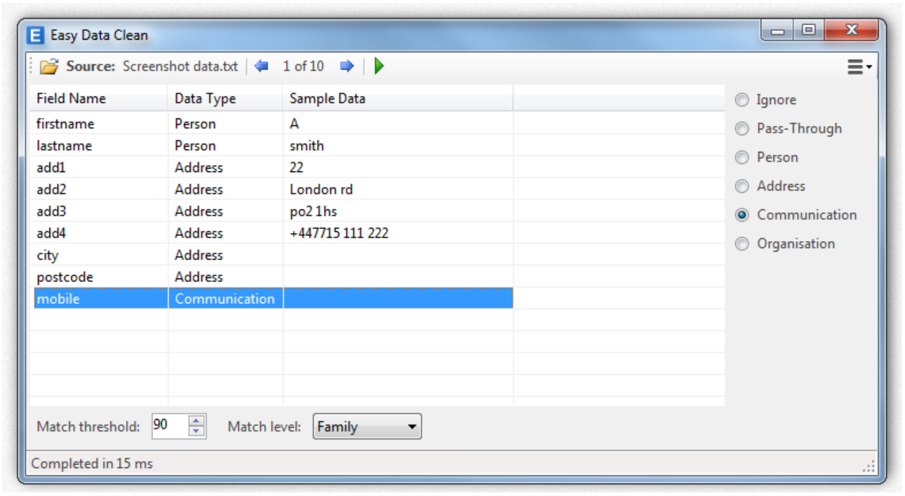
3.5.3 Easy Data Clean

Easy Data Clean[8] is commercial tool for data cleaning that works exclusively with data sheets and offers very small range of functionality that resides mostly on outliers and entity recognition like Person, Address, Communication within spread sheet data. It is primary meant for pre-processing mailing files with commercial contacts. According to the makers of this tool Easy Data Clean supports CSV's and Excel spreadsheets at present. Other formats will come along as updates are released. At the moment Easy Data Clean is for UK data only, covering England, Ireland, Scotland, Wales and the Channel Islands. Although a very few information is available Easy Data Clean seems to be simple small data cleaning tool with very special purpose.

3.5.4 DataCleaner

DataCleaner[9] is a data quality analysis application and a solution platform. It resides on a strong data profiling engine, which is extensible and thereby adds data cleansing, transformations, enrichment, deduplication, matching and merging are possible. Data Cleaner supports various sets of sources and formats like TSV, CSV,

[7]http://control.cs.berkeley.edu/abc/
[8]http://easydataclean.com/
[9]http://datacleaner.org/



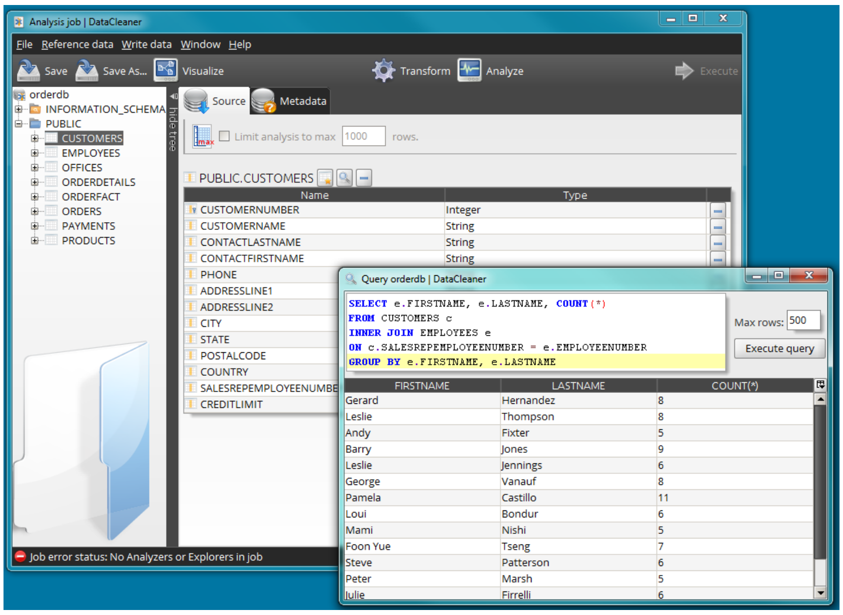
Field Name	Data Type	Sample Data
firstname	Person	A
lastname	Person	smith
add1	Address	22
add2	Address	London rd
add3	Address	po2 1hs
add4	Address	+447715 111 222
city	Address	
postcode	Address	
mobile	Communication	

Match threshold: 90 Match level: Family

Completed in 15 ms

Figure 3.6: Easy Data Clean user interface.

Microsoft Excel Spreadsheets, Databases (Oracle, MYSQL, PostgresSQL, Microsoft SQL), Mongo DB, Couch DBB, XML, Text Formated Files, OpenOffice Documents, Microsofi Access, DBase, SAS, Sybase, H2 as many others. Each project is treated as analyzing job which is supported with different kind of analyzers like boolean, number, character set, value distribution analyser and several others. Beside analyzers very important part of Data Cleaner are transformation and query capabilities (see figure 3.7).



Limit analysis to max 1000 rows.

PUBLIC.CUSTOMERS

Name	Type
CUSTOMERNUMBER	Integer
CUSTOMERNAME	String
CONTACTLASTNAME	String
CONTACTFIRSTNAME	String
PHONE	
ADDRESSLINE1	
ADDRESSLINE2	
CITY	
STATE	
POSTALCODE	
COUNTRY	
SALESREPEMPLOYEENUMBE	
CREDITLIMIT	

```
SELECT e.FIRSTNAME, e.LASTNAME, COUNT(*)
FROM CUSTOMERS c
INNER JOIN EMPLOYEES e
ON c.SALESREPEMPLOYEENUMBER = e.EMPLOYEENUMBER
GROUP BY e.FIRSTNAME, e.LASTNAME
```

Max rows: 500

FIRSTNAME	LASTNAME	COUNT(*)
Gerard	Hernandez	8
Leslie	Thompson	8
Andy	Fixter	5
Barry	Jones	9
Leslie	Jennings	6
George	Vanauf	8
Pamela	Castillo	11
Loui	Bondur	6
Mami	Nishi	5
Foon Yue	Tseng	7
Steve	Patterson	6
Peter	Marsh	5
Julie	Firrelli	6

Job error status: No Analyzers or Explorers in job

Figure 3.7: DataCleaner in action.

DataCleaner is tool also designed to clean up and optimize commercial CRM Systems like Salesforce[10] or SugarCRM[11] and it is widely used within enterprises. One of the most impressive features are various possibilities to visualize results as shown in figure 3.8 for more profound analytics.

[10]http://www.salesforce.com/
[11]http://www.sugarcrm.com/

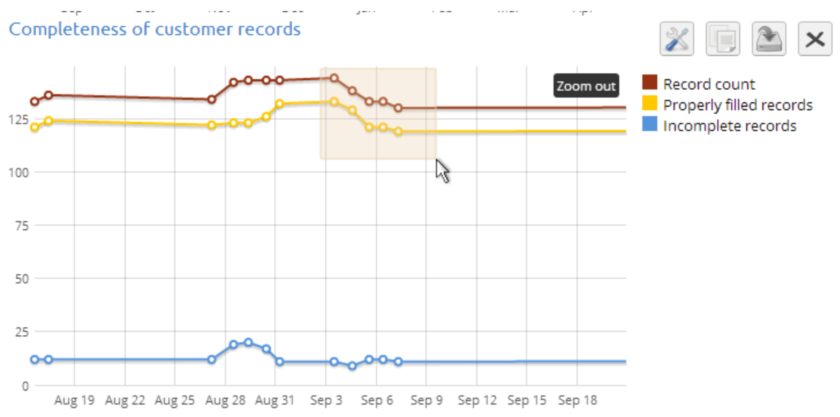


Figure 3.8: DataCleaner results visualization.

3.6 Online Tools and Services

In this section we introduce shortly some alternatives to off-line cleaning tools exposed as web applications or web based services designed for the same purpose.

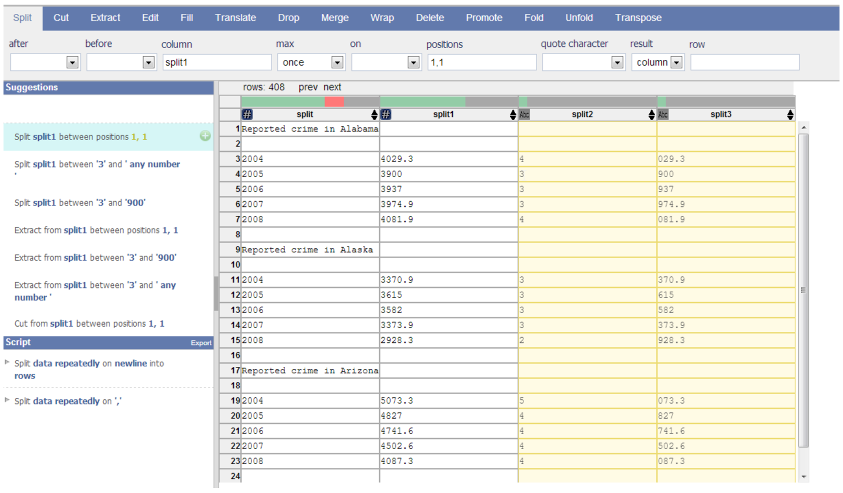
3.6.1 Wrangler

Data Wrangler[12] is a web-based service from Stanford University's Visualization Group which allows interactive cleaning and transformation of data for further analysis from Microsoft Excel (CSV, TSV), R, Protovis, and some other text based formats formats. With Wrangler, users can specify transformations by building up a sequence of basic transforms. Actions emerge as users select data. Based upon context of interaction Wrangler suggests the user applicable transforms. To ensure relevance, Wrangler enumerates and ranks possible transforms using a model which reflects user input with the frequency, diversity, and specification difficulty of applicable transform types available. To convey the effects of data transforms, Wrangler provides short natural language descriptions which users can refine via interactive parameters and visual previews of transformation results. These techniques enable users to rapid assessment of transforms. During the transformation in between steps are recorded in a script to facilitate reuse and provide documentation of data origins. Hereby Wrangler's interactive history viewer supports review, refinement, and annotation of results. Wrangler is able to infer regular expressions from example selections and supports mass editing, uses semantic roles, and provides natural language descriptions of transforms. Underlying the Wrangler interface is a declarative data transformation language consisting out of following functionalities:

- Map transforms
- Lookups and joins
- Reshape transforms
- Positional transforms
- Sorting
- Aggregation (for instance sum, min, max, mean, standard deviation)
- Key generation
- Schema transforms

In figure 3.9 a sample screen shot from Data Wrangler interface is shown together with the history panel on the left.

[12] http://vis.stanford.edu/wrangler/



after	before	column	max	on	positions	quote character	result	row
		split1	once		1,1		column	

Suggestions:

- Split **split1** between positions **1, 1**
- Split **split1** between '3' and ' **any number** '
- Split **split1** between '3' and '900'
- Extract from **split1** between positions **1, 1**
- Extract from **split1** between '3' and '900'
- Extract from **split1** between '3' and ' **any number** '
- Cut from **split1** between positions **1, 1**

Script:

- Split **data repeatedly** on **newline** into **rows**
- Split **data repeatedly** on ','

	split	split1	split2	split3
1	Reported crime in Alabama			
2				
3	2004	4029.3	4	029.3
4	2005	3900	3	900
5	2006	3937	3	937
6	2007	3974.9	3	974.9
7	2008	4081.9	4	081.9
8				
9	Reported crime in Alaska			
10				
11	2004	3370.9	3	370.9
12	2005	3615	3	615
13	2006	3582	3	582
14	2007	3373.9	3	373.9
15	2008	2928.3	2	928.3
16				
17	Reported crime in Arizona			
18				
19	2004	5073.3	5	073.3
20	2005	4827	4	827
21	2006	4741.6	4	741.6
22	2007	4502.6	4	502.6
23	2008	4087.3	4	087.3
24				

Figure 3.9: Data Wrangler interface in use.

3.6.2 Data8

Data8[13] is a solutions provider from UK with a range of data cleansing, postcode lookup and data validation services. Online data cleansing service from Data8 include:

- Gone Away Suppression - removing non-responders from mailings.
- PAF Cleansing - improving data's address quality.
- Telephone Appending - finding telephone numbers from customer data.
- Postcode Lookup - capturing address details quickly and accurately.

3.6.3 AddressDoctor

AddressDoctor[14] AddressDoctor is a bunch of commercial tools for addresses disambiguation. Beside off-line solutions this company offers also a web service solution. AddressDoctor's Web Services offer users the opportunity to correct addresses in the Cloud without installing software. AddressDoctor operates its Web Services on high-availability clusters using the powerful AddressDoctor correction technology. Equipped with a SOAP interface, these Web Services can be quickly integrated into a user's own applications, CRM systems, online shops and websites .Addresses are corrected in very short time and transmitted over secure (SSL) links. Users automatically benefit from the latest reference data and up-to-the-minute supplemental content such as geo codes or consumer segmentation.

3.7 Challenges and Problems

Data cleaning process is applied mostly with different intension and within different areas of the data integration and management process. It is defined as the sequence of operations intending to enhance to overall data quality of a data collection. However, current research delivered only a rough description of the procedure, therefore data cleaning as it is remains highly domain dependent task. There is still wide set of problems and challenges in data cleaning that are not solved up to now by the existing approaches. Most important of them are described more into detail in following subsections:

[13]http://www.data-8.co.uk/
[14]http://www.addressdoctor.com/

3.7.1 Error Correction

Error Correction is most challenging problem within data cleaning. In this domain of problems belong: the correction of values to eliminate domain format errors, constraint violations, duplicates and invalid tuples. In many cases the available information and knowledge is insufficient to determine the correct modification of data tuples to remove these anomalies. Currently the only practical solution for this problem is deleting affected tuples. This deletion of tuples in consequence leads to a loss of information if the tuple is not invalid as a whole. One of the approaches how loss of information can be avoided would be by keeping the tuple in the data collection and masking the erroneous values until appropriate information for error correction is available.

3.7.2 Maintenance of Cleaned Data and Impact on Costs

Cleaning data is a time consuming and expensive task. After having performed data cleaning and achieved a data collection free of errors one does not want to perform the whole data cleaning process again after some of the values in data collection change. Only the part of the cleaning process which is affected by the changed value should be re-started. This affection can be determined by analyzing the cleaning history. In order to avoid re-running unaffected steps cleaning history is kept not only for tuples that have been corrected, but also for those that have been verified as correct within the cleaning process. After one of the values in the data collection has changed, the cleaning work flow has to be repeated for those tuples that contains the changed value as part of their cleaning lineage. Respecting these procedures can contribute to easier maintenance and decrease data cleaning costs.

Chapter 4

Selected Examples

A survey paper turns out to be more useful for a reader if the paper includes real-life examples. Therefore in the current chapter we provide example-solutions for ficitve use-cases. Use-case "A" deals with crawling of twitter-messages of the hash-tag "#tugraz" and is provided in the subsection "Crawling Twitter Feed". Use-case "B" deals with the problem of gathering data made public as PDF files what is known as "scraping".

> "When you're looking for data to use in your own applications, you may find it's not always in the format you want. Government sites in particular are famous for publishing PDFs instead of raw data which is easier to parse." [Rotter, 2013]

According to this problem an example, how to transform a table extracted from a PDF file to CSV format, is shown using an annual report PDF file from "Holding Graz - Kommunale Dienstleistungen GmbH"[1]. As the data in the CSV still is "dirty" it will be shown how to clean it using the tool "OpenRefine" in the subsection "OpenRefine" within the section Data Cleaning.

4.1 Data Harvesting

4.1.1 Crawling Twitter Feed

The following is an example which deals with crawling of twitter-messages of the hash-tag "#tugraz". For easy reuse without the need of installing tools locally an on-line web-crawler development website called "Scraperwiki"[2] was chosen. The functionality to write code on-line either in Python, PHP or ruby is provided. The possibility to pay for writing a crawler is provided too, however this is not in the paper focus. For this example "write code" and afterwards "Python" as the language to write in was chosen. The code used was found on the Scraperwiki website [Poitras, 2013] and was customized to deal with tweets about the hash-tag "#tugraz".

The listing4.1 shows the code. Lines 1-3 import some packages used. In lines 7-11 customizations can be done by changing the following variable-values:

- `QUERY`: The keywords to search for, or, because twitter uses hash-tags to mark keywords or topics in a Tweet, a hash-tag.[Twitter, 2013b]
- `RESULTS_PER_PAGE`: The number of results per page.
- `RESULT_TYPE`: Specifies what type of search result is preferred to receive. Valid values: "recent", "popular" or "mixed", where "mixed" is a mixture of popular and most recent tweets. [Twitter, 2013a]
- `NUM_PAGES`: The number of pages. The script stops when all pages are filled with data.

[1]http://www.holding-graz.at/holding-graz/unternehmen/geschaeftsbericht.html, online 2013-05-13
[2]https://scraperwiki.com/

- `ENTITIES`: When set to either true, t or 1, each tweet will include a node called "entities,". This node offers a variety of metadata about the tweet in a discrete structure, including: urls, media and hashtags. [Twitter, 2013a]

Lines 13-15 builds and sends the query to the Twitter-API. Lines 16-31 try to receive the response and parse it. Line 28 saves the gathered information to the in-built database of the Scraperwiki website.

```
 1 import scraperwiki
 2 import simplejson
 3 import urllib2
 4
 5 # Change QUERY to your search term of choice.
 6 # Examples: 'newsnight', 'from:bbcnewsnight', 'to:bbcnewsnight'
 7 QUERY = '#tugraz'
 8 RESULTS_PER_PAGE = '100'
 9 RESULT_TYPE = 'recent'
10 NUM_PAGES = 14
11 ENTITIES = 'true'
12
13 for page in range(1, NUM_PAGES+1):
14     base_url = 'http://search.twitter.com/search.json?q=%s&rpp=%s&page=%s&
           result_type=%s&include_entities=%s' \
15          % (urllib2.quote(QUERY), RESULTS_PER_PAGE, page, RESULT_TYPE,
                 ENTITIES)
16     try:
17         results_json = simplejson.loads(scraperwiki.scrape(base_url))
18         for result in results_json['results']:
19             #print result
20             data = {}
21             data['id'] = result['id']
22             data['text'] = result['text']
23             data['from_user'] = result['from_user']
24             data['created_at'] = result['created_at']
25             data['geo'] = result['geo']
26             data['entities'] = result['entities']
27             print data['from_user'], data['text'], data['geo'], data['entities
                 ']
28             scraperwiki.sqlite.save(["id"], data)
29     except:
30         print 'Oh dear, failed to scrape %s' % base_url
31         break
```

Listing 4.1: Use-case "A" - crawling a twitter feed

4.1.2 Extracting Tables from PDF

As the last example showed how to crawl Twitter-messages the following example deals with a difficult and probably everyday problem of, for example, journalists, when they use published government data. The problem is, that the data is not available in a convenient format, for example: copying and pasting of a table out of a PDF file probably leads to blank rows, null values or wrong data-types like integer represented as string. Copying some tables out of a PDF file by hand can be time consuming whereas a script may save time. The following shows an example how to scrape a balance-report table out of the 2011 annual report of the "Holding Graz - Kommunale Dienstleistungen GmbH" which is placed on the pages 74 and 75. Firstly a binary program is used to transform given pages of the PDF file to XML. In the program call "file.pdf" is the PDF file and should be changed to the file-name of the downloaded file. The integer value after the "-f" command is the starting page, the integer after the "-l" command is the last page, all pages in-between including first and last page will be transformed. The program is called "pdftohtml"[3] and is used in the following way to transform pages 74 to 75:

```
pdftohtml -f 74 -l 75 -xml -stdout file.pdf
```

Listing 4.2: Use-case "B" - command line: transform PDF page to XML - part 1

Secondly the XML has to be parsed which is done with a Python package called "pdftable"[4]. The package detects tables in XML files, parses the table data and finally outputs the data to a CSV file - one file for each different table detected. If a table is split over 2 pages it detects this one as 2 different tables and therefore outputs 2 CSV files. As the package was written for Python 2.7 some changes for Python 3 had to be made. The package takes the XML produced in part 1 as input, therefore the whole process is combined into a single command:

```
pdftohtml -f 74 -l 75 -xml -stdout file.pdf | python pdftable -f file%d.csv
```

Listing 4.3: Use-case "B" - command line: PDF table to CSV file - part 2

4.2 Data Cleaning

Data Cleaning is a process to transform a given dataset to a cleaned version of it. Data Cleaning is used to eliminate empty rows, null values, merge similar columns which where divided while harvesting, filter or sort the dataset, edit the data to eliminate misspellings, transform it to another format, etc. Theory and available Tools are already described in the Data Cleaning chapter, therefore this part shows how to use one of those tools: OpenRefine[5].

4.2.1 OpenRefine

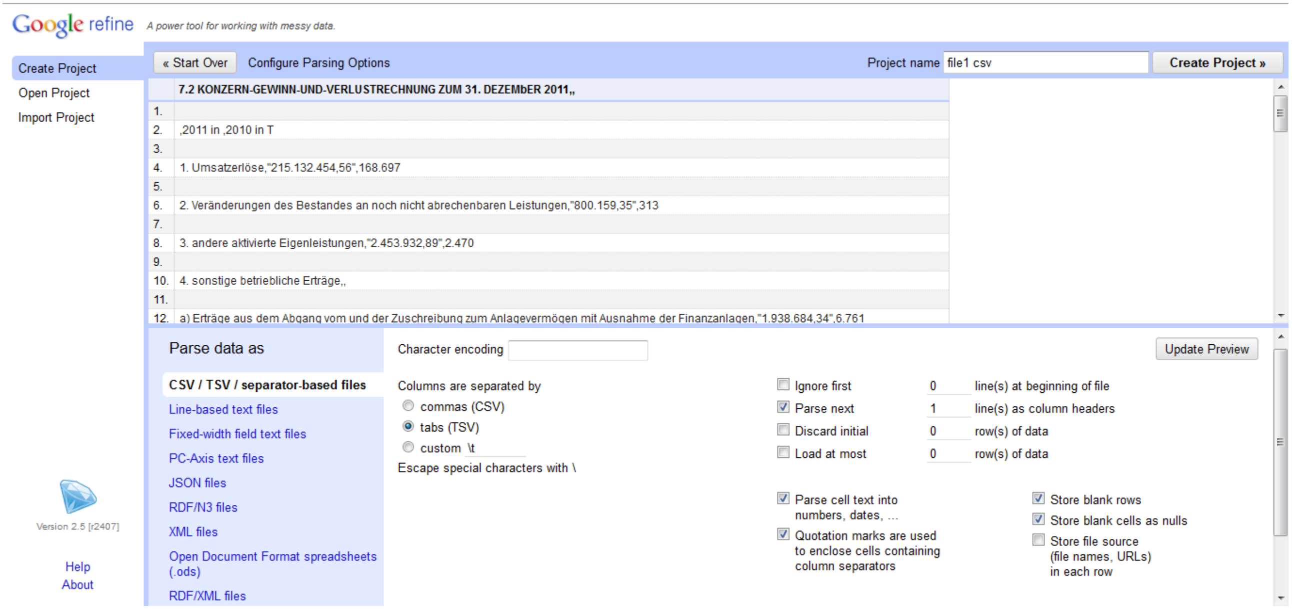
To use OpenRefine the latest stable version (current stable version is 2.5 and still has the old branding Google Refine, will change with next version) is downloadable as ZIP-compressed file. Unzip it and run the executable file. The program is launched and the landing-page is shown in the standard web-browser. The program runs locally, therefore no need for fear of data-loss. "Get data from This Computer" is chosen.

Annual Report - Graz Holding

This example deals with the CSV files produced in subsection 4.1.2. The purpose is to eliminate blank rows and header-rows at first to be able to perform filtering tasks in advance. At the beginning the file is loaded, here for simplicity just "file1.csv" of the 2 available files is chosen, but it is possible to load more than one file simultaneously. Clicking on the "Next" button leads the user to the data-parsing view, shown in figure 4.1. At the data-parsing view useful features are provided, in the example the following features are used (from left to right):

- `Parse data as`: As a CSV file was imported, "CSV / TSV / separator-based files" was chosen.
- `Columns are separated by`: As a CSV file was imported, "commas (CSV)" was chosen.
- `Parse next`: As the extracted table starts with header-titles at row 1 and 3 selecting this option was chosen and the value was set to 3.

[3] http://pdftohtml.sourceforge.net/, online 2013-05-05
[4] http://pdftable.sourceforge.net/, online 2013-05-20
[5] http://openrefine.org/, online 2013-05-10



Configure Parsing Options

Project name: file1 csv

	7.2 KONZERN-GEWINN-UND-VERLUSTRECHNUNG ZUM 31. DEZEMbER 2011,,
1.	
2.	,2011 in ,2010 in T
3.	
4.	1. Umsatzerlöse,"215.132.454,56",168.697
5.	
6.	2. Veränderungen des Bestandes an noch nicht abrechenbaren Leistungen,"800.159,35",313
7.	
8.	3. andere aktivierte Eigenleistungen,"2.453.932,89",2.470
9.	
10.	4. sonstige betriebliche Erträge,,
11.	
12.	a) Erträge aus dem Abgang vom und der Zuschreibung zum Anlagevermögen mit Ausnahme der Finanzanlagen,"1.938.684,34",6.761

Parse data as
- **CSV / TSV / separator-based files**
- Line-based text files
- Fixed-width field text files
- PC-Axis text files
- JSON files
- RDF/N3 files
- XML files
- Open Document Format spreadsheets (.ods)
- RDF/XML files

Character encoding

Columns are separated by
- commas (CSV)
- [✓ tabs (TSV)]
- custom \t

Escape special characters with \

- ☐ Ignore first 0 line(s) at beginning of file
- ✓ Parse next 1 line(s) as column headers
- ☐ Discard initial 0 row(s) of data
- ☐ Load at most 0 row(s) of data
- ✓ Parse cell text into numbers, dates, ...
- ✓ Quotation marks are used to enclose cells containing column separators
- ✓ Store blank rows
- ✓ Store blank cells as nulls
- ☐ Store file source (file names, URLs) in each row

Figure 4.1: Data Parsing after loading a CSV file with OpenRefine.

- `Store blank rows`: The dataset contains blank rows, therefore this option was unselect.

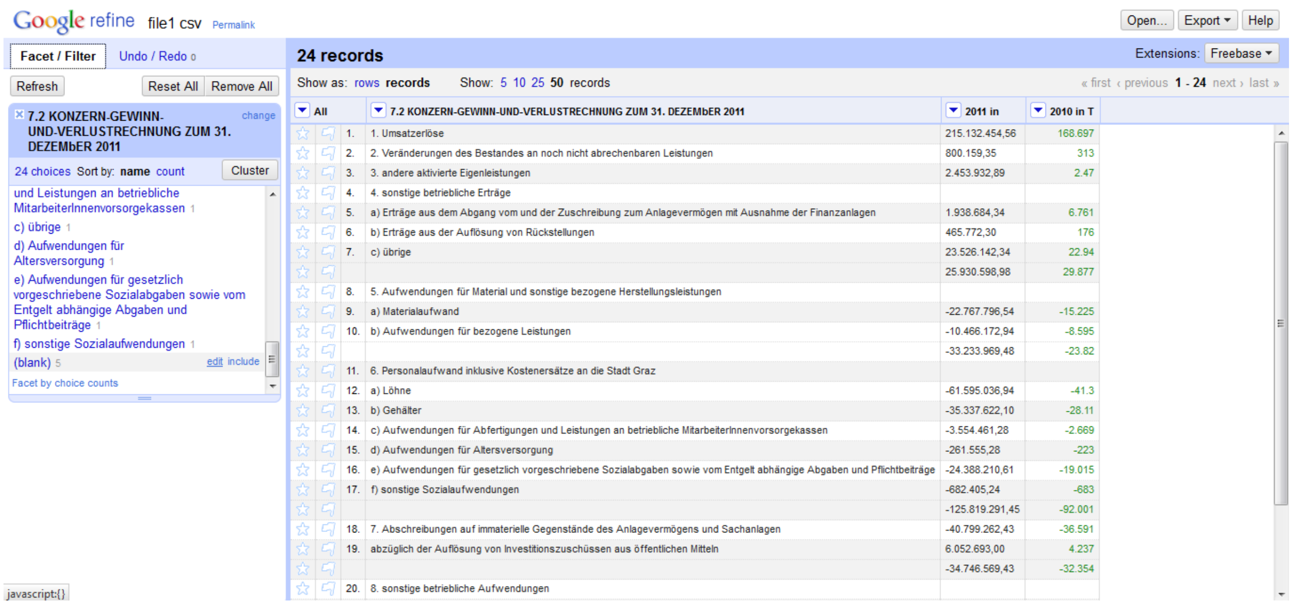
Clicking on "Create Project" lead us to the working area. It is recognized that there are rows with a blank cell in column 0, in the example called "7.2 KONZERN-GEWINN-UND-VERLUSTRECHNUNG ZUM 31. DEZEMbER 2011", and a sum in column 1 and 2, in the example called "2011 in" and "2010 in T". Those values are the aggregated sums of the regarding section of the balance report. To edit the cell-name of those sum-rows in column 0 the following has to be done:

- Clicking the drop-down button at column 0 opened up a menu.
- Firstly "Facet" and secondly "Text Facet" in the sub-menu had to be chosen. The Facets available had been opened on the left side of the screen in the "Facet/Filter" box.
- Scrolling down until the entry "(blank)5" and mouse-hovering showed an edit-button to click on as shown in figure 4.2. The edit button was clicked.
- Entering a new Name for the cells, for example: Section Sum and clicking on the "Apply" button lead to a change in all affected cells.

Once this is done filtering the data to just display rows with negative values is the next task. Therefore the drop-down button at column "2011 in" is clicked and Text filter from the menu is chosen. A new box was opened in the Facet/Filter area. Selecting the regular expression option and typing "^" into the text-field filters the data-set to display negative values. To save the changed table, exporting in various formats is provided.

Open Data sample with Graz Residents Data

As the example of the annual report of Holding Graz was rather small due to the number of rows for this example data of Graz city is used. The data is provided on the website "http://data.graz.gv.at", the purpose of the group in charge is to make public data available for the public. The used dataset about habitants of the Graz-districts divided into citizenship is downloadable as a CSV file (source: http://data.graz.gv.at/daten/package/die-grazer-bev-lkerung-nach-bezirk-und-staatsabgeh-rigkeit). The dataset has 124528 rows of data, including monthly data about the years 2006 until 2013. A journalists job could be to show the trend of amount of habitants that live in Lend or Gries with french citizenship. To eliminate typical season-related up- and downturns just data of the month January is used. The following list shows how to do it:



7.2 KONZERN-GEWINN-UND-VERLUSTRECHNUNG ZUM 31. DEZEMbER 2011

24 choices Sort by: name count

und Leistungen an betriebliche MitarbeiterInnenvorsorgekassen 1
c) übrige 1
d) Aufwendungen für Altersversorgung 1
e) Aufwendungen für gesetzlich vorgeschriebene Sozialabgaben sowie vom Entgelt abhängige Abgaben und Pflichtbeiträge 1
f) sonstige Sozialaufwendungen 1
(blank) 5

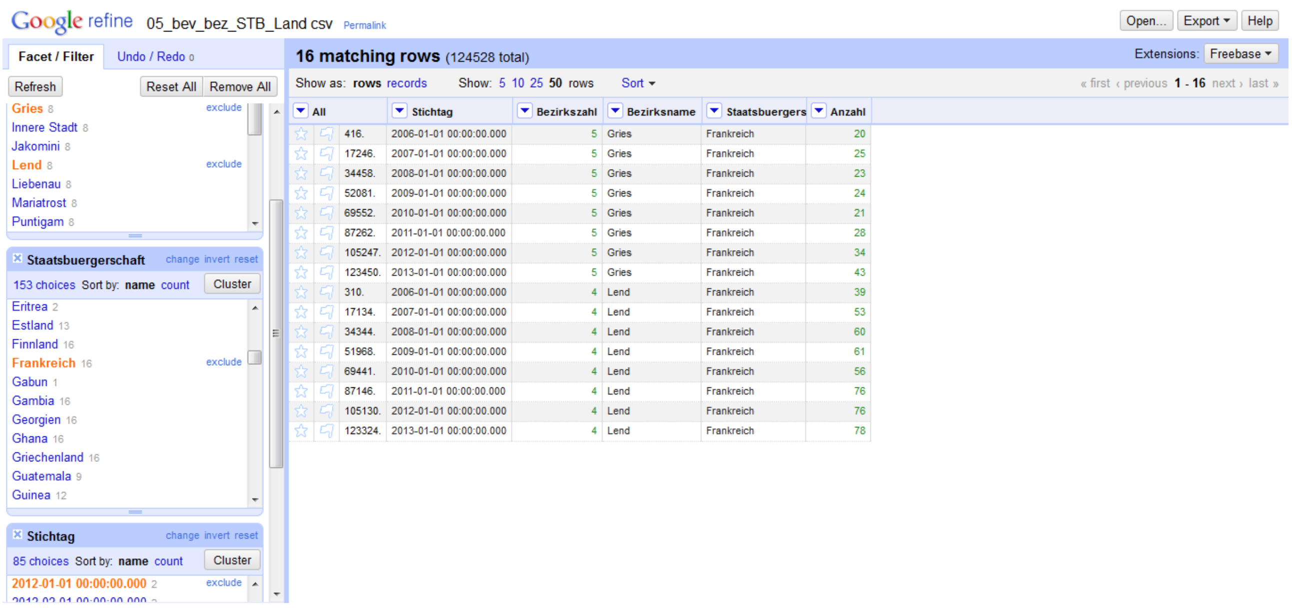
24 records

Show as: rows **records** Show: 5 10 25 **50** records

All	7.2 KONZERN-GEWINN-UND-VERLUSTRECHNUNG ZUM 31. DEZEMbER 2011	2011 in	2010 in T
1.	1. Umsatzerlöse	215.132.454,56	168.697
2.	2. Veränderungen des Bestandes an noch nicht abrechenbaren Leistungen	800.159,35	313
3.	3. andere aktivierte Eigenleistungen	2.453.932,89	2.47
4.	4. sonstige betriebliche Erträge		
5.	a) Erträge aus dem Abgang vom und der Zuschreibung zum Anlagevermögen mit Ausnahme der Finanzanlagen	1.938.684,34	6.761
6.	b) Erträge aus der Auflösung von Rückstellungen	465.772,30	176
7.	c) übrige	23.526.142,34	22.94
		25.930.598,98	29.877
8.	5. Aufwendungen für Material und sonstige bezogene Herstellungsleistungen		
9.	a) Materialaufwand	-22.767.796,54	-15.225
10.	b) Aufwendungen für bezogene Leistungen	-10.466.172,94	-8.595
		-33.233.969,48	-23.82
11.	6. Personalaufwand inklusive Kostenersätze an die Stadt Graz		
12.	a) Löhne	-61.595.036,94	-41.3
13.	b) Gehälter	-35.337.622,10	-28.11
14.	c) Aufwendungen für Abfertigungen und Leistungen an betriebliche MitarbeiterInnenvorsorgekassen	-3.554.461,28	-2.669
15.	d) Aufwendungen für Altersversorgung	-261.555,28	-223
16.	e) Aufwendungen für gesetzlich vorgeschriebene Sozialabgaben sowie vom Entgelt abhängige Abgaben und Pflichtbeiträge	-24.388.210,61	-19.015
17.	f) sonstige Sozialaufwendungen	-682.405,24	-683
		-125.819.291,45	-92.001
18.	7. Abschreibungen auf immaterielle Gegenstände des Anlagevermögens und Sachanlagen	-40.799.262,43	-36.591
19.	abzüglich der Auflösung von Investitionszuschüssen aus öffentlichen Mitteln	6.052.693,00	4.237
		-34.746.569,43	-32.354
20.	8. sonstige betriebliche Aufwendungen		

Figure 4.2: Data editing with OpenRefine.

- CSV file has to be downloaded and loaded with OpenRefine, for details see 4.2.1.
- In the parsing view "custom", value is ";" was chosen for "the Columns are separated by". `Parse next` was selected, the corresponding value is 1. Project was created.
- The drop-down button at column called "Berzirksname" was clicked and the term "Facet" in the menu and the term "Text Facet" in the submenu were chosen. Text facet was opened up in "Facet/Filter" box.
- Searching for "Gries" and mouse-over the word made the buttons "edit" and "include" visible. Clicking on "include" and the same for the district "Lend" added the terms to the filter.
- Opening the "Text Facet" for "Staatsbuergerschaft" too and clicking or "include" "Frankreich" added this part to the filter.
- Opening the "Text Facet" for "Stichtag" too and including every date which corresponds to January by clicking on the "include" button added this part to the filter.
- Finally clicking on the dropdown button of "Bezirksname" and choosing "Sort" from the menu, followed by choosing sorting values as "text" and the option "a-z" lead to the result shown in figure 4.3 and can be exported in various formats.


05_bev_bez_STB_Land csv

16 matching rows (124528 total)

Gries 8
Innere Stadt 8
Jakomini 8
Lend 8
Liebenau 8
Mariatrost 8
Puntigam 8

Staatsbuergerschaft

153 choices Sort by: name count

Eritrea 2
Estland 13
Finnland 16
Frankreich 16
Gabun 1
Gambia 16
Georgien 16
Ghana 16
Griechenland 16
Guatemala 9
Guinea 12

Stichtag

85 choices Sort by: name count

2012-01-01 00:00:00.000 2

All	Stichtag	Bezirkszahl	Bezirksname	Staatsbuergers	Anzahl
416.	2006-01-01 00:00:00.000	5	Gries	Frankreich	20
17246.	2007-01-01 00:00:00.000	5	Gries	Frankreich	25
34458.	2008-01-01 00:00:00.000	5	Gries	Frankreich	23
52081.	2009-01-01 00:00:00.000	5	Gries	Frankreich	24
69552.	2010-01-01 00:00:00.000	5	Gries	Frankreich	21
87262.	2011-01-01 00:00:00.000	5	Gries	Frankreich	28
105247.	2012-01-01 00:00:00.000	5	Gries	Frankreich	34
123450.	2013-01-01 00:00:00.000	5	Gries	Frankreich	43
310.	2006-01-01 00:00:00.000	4	Lend	Frankreich	39
17134.	2007-01-01 00:00:00.000	4	Lend	Frankreich	53
34344.	2008-01-01 00:00:00.000	4	Lend	Frankreich	60
51968.	2009-01-01 00:00:00.000	4	Lend	Frankreich	61
69441.	2010-01-01 00:00:00.000	4	Lend	Frankreich	56
87146.	2011-01-01 00:00:00.000	4	Lend	Frankreich	76
105130.	2012-01-01 00:00:00.000	4	Lend	Frankreich	76
123324.	2013-01-01 00:00:00.000	4	Lend	Frankreich	78

Figure 4.3: Data Filtering with OpenRefine.

Bibliography

Abiteboul, Serge et al. [1999]. “Tools for data translation and integration”. *IEEE Data Engineering Bulletin* 22.1 (1999), pages 3–8. http://sites.computer.org/debull/99mar/99MAR-CD.pdf#page=5 (cited on page 17).

Gatterbauer, W. [2009]. *Encyclopedia of Database Systems*. DBLP, 2009, pages 3472–3473 (cited on pages 3, 5).

Gray, Matthew [1995]. “Measuring the Growth of the Web June 1993 to June 1995” (1995). http://www.mit.edu/people/mkgray/growth/ (visited on 05/18/2013) (cited on page 4).

Maletic, Johnatan I. and Andrian Marcus [2000]. “Data cleansing: Beyond integrity analysis”. In: *Proceedings of the Conference on Information Quality*. 2000, pages 200–209 (cited on pages 16, 17).

Manning, C.D., P. Raghavan, and H. Schütze [2008]. *Introduction to Information Retrieval*. An Introduction to Information Retrieval. Cambridge University Press, 2008. ISBN 9780521865715 (cited on pages 4, 5).

Motro, Amihai [1989]. “Integrity is validity plus completeness”. *ACM Transactions on Database Systems (TODS)* 14.4 (1989), pages 480–502. ISSN 0362-5915. http://doi.acm.org/10.1145/76902.76904 (cited on page 14).

Müller, Heiko and Johann-Christoph Freytag [2003]. *Problems, Methods and Challenges in Comprehensive Data Cleansing*. Technical Report HUB-IB-164. Humboldt-Universität zu Berlin, Institut für Informatik, 2003. http://www.dbis.informatik.hu-berlin.de/fileadmin/research/papers/techreports/2003-hub_ib_164-mueller.pdf (cited on pages 13–16).

Naumann, Felix [2002]. *Quality-driven query answering for integrated information systems*. Berlin, Heidelberg: Springer-Verlag, 2002. ISBN 3-540-43349-X (cited on page 14).

Peshave, Monica [2005]. “How Search Engines Work and a Web Crawler Application” (2005) (cited on page 6).

Poitras [2013]. *Poitras, Twitter API - ScraperWiki*. May 13, 2013. http://scraperwiki.com/scrapers/twitter_api_7/ (visited on 05/13/2013) (cited on page 24).

Rahm, Erhard and Hong Hai Do [2000]. “Data Cleaning: Problems and Current Approaches”. *IEEE Bulletin of the Technical Committee on Data Engineering* 23.4 (2000), pages 3–13. http://sites.computer.org/debull/A00DEC-CD.pdf (cited on page 14).

Raman, Vijayshankar and Joseph M. Hellerstein [2001]. “Potter’s Wheel: An Interactive Framework for Data Cleaning and Transformation”. In: *Proc. International Conference on Very Large Data Bases (VLDB)*. 2001. http://www.vldb.org/conf/2001/P381.pdf (cited on page 16).

Rotter, Martha [2013]. *TUTORIAL: Data Scraping for N00bs*. May 14, 2013. http://readidea.com/magazines/issue4/scraperwiki.html (visited on 05/14/2013) (cited on pages 1, 24).

Tonn, Daniel [2000]. “Data Cleansing Verfahren für Data Warehouses”. Master’s thesis. Berlin: Humboldt University, 2000 (cited on page 14).

Twitter [2013a]. *GET search - Twitter Developers*. May 18, 2013. `http://dev.twitter.com/docs/api/1/get/search` (visited on 05/18/2013) (cited on pages 24, 25).

Twitter [2013b]. *Using hashtags on Twitter*. May 18, 2013. `http://support.twitter.com/articles/49309-using-hashtags-on-twitter#` (visited on 05/18/2013) (cited on page 24).