Evaluation of Open Source Data Cleaning Tools: Open Refine and Data Wrangler

Per Larsson
plarsson@cs.washington.edu

June 7, 2013

Abstract

This project aims to compare several tools for cleaning and importing data. Open Refine and Data Wrangler are evaluated and compared. In conclusion non of them are mature tools yet. Open Refine is not suitable for large data sets but it works better for Data Wrangler.

1 Introduction

Today a lot of useful data is stored in different ad-hoc formats. Examples include scientific data, web server logs and network traffic logs. This data is often interesting to run queries on or use together with analysis tools but their format poses a challenge. One often wish that the data was given in some other format or stored in a relational database such as PostgreSQL. Another challenge is that this kind of data can be high volume.

There exists a number of computer programs which can read ad-hoc data and rewrite it into a more convenient format which is more clean and easily parsable.

This report tries to evaluate some of the open source tools (presented in the following subsections) that exists for this purpose. Since the data analyst often can be a domain expert rather than a programmer these tools must be reasonable user friendly for a non-programmer. Are these tools sufficient? Are they easy to use? Are they practical for very large data sets?

Open Refine describes itself as "Open Refine is a power tool for working with messy data, cleaning it up, transforming it from one format into another, extending it with web services, and linking it to databases like Freebase.". It is open source, written in Java and available to download for free from the website[3].

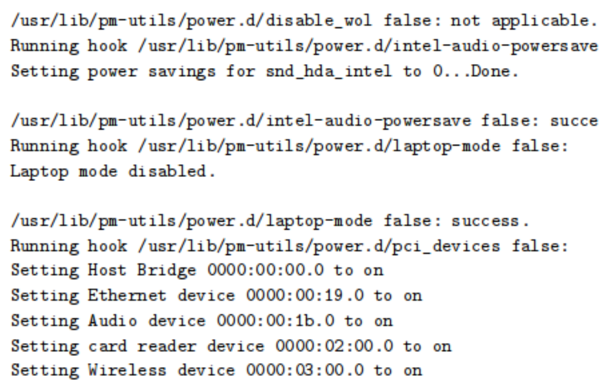
Data Wrangler is a tool similar to Open Refine developed at Stanford in Javascript an Python.

2 Method

The goal of the method is to answer the following question: *How effective is the tool in taking data from it's original ad-hoc format into a format that is importable into a relational database?*, this practically means that the tools will read the data and transform it into a comma-separated-values-file (`.csv`) which is importable into most RDBMS:es like PostgreSQL. This includes evaluating tool data reading, tool usage and tool data

CPU	64-bit Intel, 2 cores, 3 GHz
RAM Memory	4GB
Web Browser	Google Chrome

Figure 1: *Test computer set up*



```
/usr/lib/pm-utils/power.d/disable_wol false: not applicable.
Running hook /usr/lib/pm-utils/power.d/intel-audio-powersave
Setting power savings for snd_hda_intel to 0...Done.

/usr/lib/pm-utils/power.d/intel-audio-powersave false: succe
Running hook /usr/lib/pm-utils/power.d/laptop-mode false:
Laptop mode disabled.

/usr/lib/pm-utils/power.d/laptop-mode false: success.
Running hook /usr/lib/pm-utils/power.d/pci_devices false:
Setting Host Bridge 0000:00:00.0 to on
Setting Ethernet device 0000:00:19.0 to on
Setting Audio device 0000:00:1b.0 to on
Setting card reader device 0000:02:00.0 to on
Setting Wireless device 0000:03:00.0 to on
```

Figure 3: *Format of /var/log/pm-powersave.log*

export. A normal standard PC will be used to perform the tests (see figure 2).

The tools are tested one after another and a short summary was written for each step. Each step is tested with three different real-life data sets with varying attributes.

2.1 Data sets

Billion Triples Challenge 2010 Dataset (BTC) [2] contains 3.2bn lines of N-Quads (figure 2.1), approximately 27GB, spread out over 317 files. This dataset will effectively test the maximum size that the tools are capable of handling.

As mentioned in the introduction a common application for data cleaning tools is log files. Two common Linux log files will be used for evaluation; `/var/log/kern.log` (figure 2) and `/var/log/pm-powersave.log` (figure 3). Smaller than the BTC they have more challenging formats.

3 Results

3.1 Open Refine

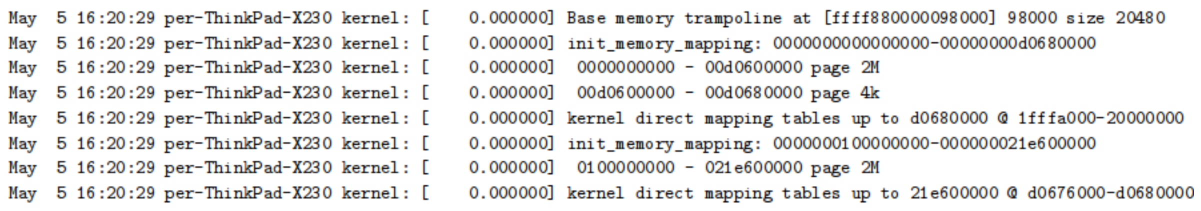
3.1.1 Data import

Open Refine can be run on the local computer but it is also easy to run it on a remote server since it starts a web server and the user access it through a web user interface.

When importing data the user must select which data file he or she wishes to process and upload it to the server, whether it runs locally or not. This means that even though the data file is already on the local computer, it has to be copied over to the server. A new Refine "project" is then created from the data.

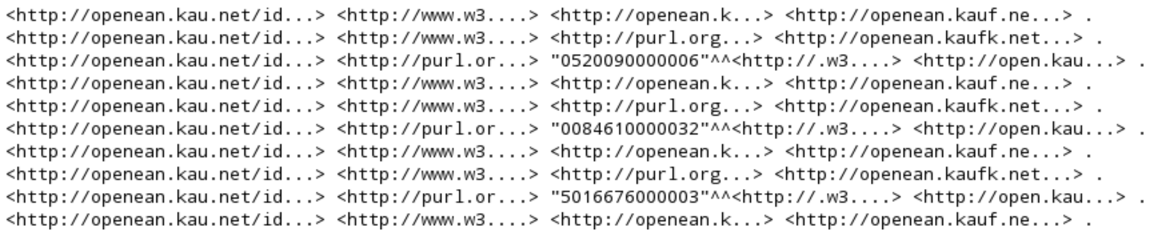
Refine does not work very well with big data. When tested with files of 4100MB and 2100MB the server simply throws a `java.lang.OutOfMemoryError` error which gives a hint that the amount of working memory might be an issue. Files of 512MB (about 2482653 BTC triples) works better and it takes 48 seconds to upload the file and create the project and it is then ready to be edited. However, a simple split operation splitting the rows where there's a space (approximately at 4-5 places per row) takes over 15 minutes to perform. Since the user is supposed to work with the data in real time this is unacceptable.

Importing `/var/log/kern.log` (13373 lines), which is smaller, works better. Open Refine automatically suggests creating several columns depending on repeted patterns. For example it suggests splitting the lines between the date, the computer hostname (`per-ThinkPad-X230`) and



```
May  5 16:20:29 per-ThinkPad-X230 kernel: [    0.000000] Base memory trampoline at [ffff880000098000] 98000 size 20480
May  5 16:20:29 per-ThinkPad-X230 kernel: [    0.000000] init_memory_mapping: 0000000000000000-00000000d0680000
May  5 16:20:29 per-ThinkPad-X230 kernel: [    0.000000]  0000000000 - 00d0600000 page 2M
May  5 16:20:29 per-ThinkPad-X230 kernel: [    0.000000]  00d0600000 - 00d0680000 page 4k
May  5 16:20:29 per-ThinkPad-X230 kernel: [    0.000000] kernel direct mapping tables up to d0680000 @ 1fffa000-20000000
May  5 16:20:29 per-ThinkPad-X230 kernel: [    0.000000] init_memory_mapping: 0000000100000000-000000021e600000
May  5 16:20:29 per-ThinkPad-X230 kernel: [    0.000000]  0100000000 - 021e600000 page 2M
May  5 16:20:29 per-ThinkPad-X230 kernel: [    0.000000] kernel direct mapping tables up to 21e600000 @ d0676000-d0680000
```

Figure 2: *Format of* `/var/log/kern.log`



```
<http://openean.kau.net/id...> <http://www.w3....> <http://openean.k...> <http://openean.kauf.ne...> .
<http://openean.kau.net/id...> <http://www.w3....> <http://purl.org...> <http://openean.kaufk.net...> .
<http://openean.kau.net/id...> <http://purl.or...> "0520090000006"^^<http://.w3....> <http://open.kau...> .
<http://openean.kau.net/id...> <http://www.w3....> <http://openean.k...> <http://openean.kauf.ne...> .
<http://openean.kau.net/id...> <http://www.w3....> <http://purl.org...> <http://openean.kaufk.net...> .
<http://openean.kau.net/id...> <http://purl.or...> "0084610000032"^^<http://.w3....> <http://open.kau...> .
<http://openean.kau.net/id...> <http://www.w3....> <http://openean.k...> <http://openean.kauf.ne...> .
<http://openean.kau.net/id...> <http://www.w3....> <http://purl.org...> <http://openean.kaufk.net...> .
<http://openean.kau.net/id...> <http://purl.or...> "5016676000003"^^<http://.w3....> <http://open.kau...> .
<http://openean.kau.net/id...> <http://www.w3....> <http://openean.k...> <http://openean.kauf.ne...> .
```

Figure 4: *Format of* `Billion Triples Challenge 2010` *(URLs have been shortened at the dots). As one can see the data is inconsistently bad formatted with two* $\wedge\wedge$ *instead of a space at some points.*

the actual values into their own columns.

3.1.2 Tool usage

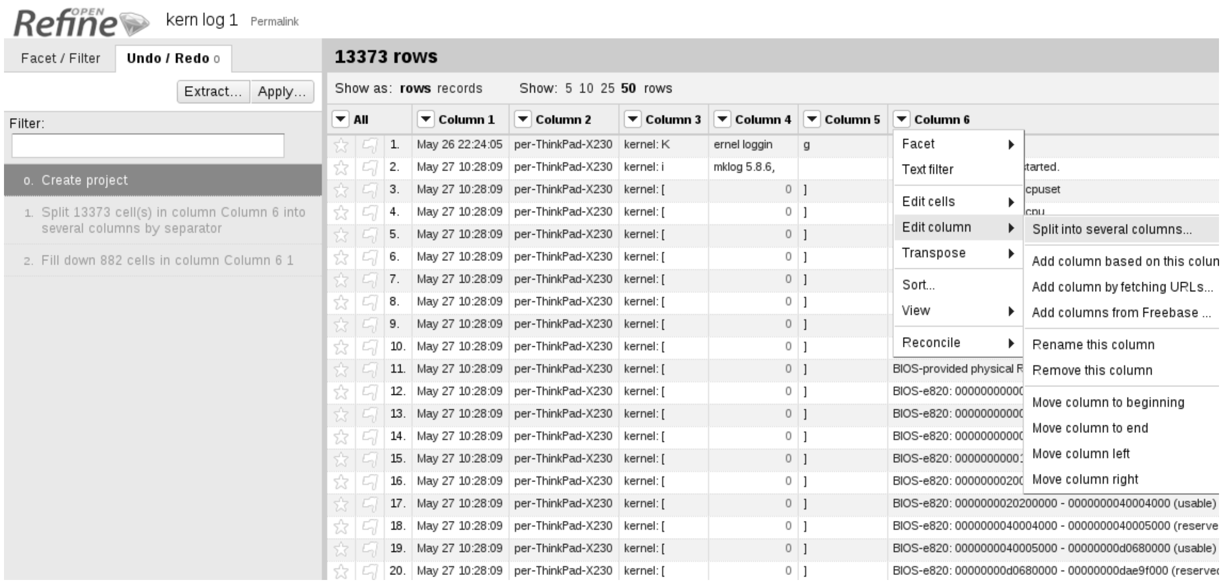
First manipulation of the BTC dataset (figure 2.1 was attempted. In order for the tool to be interactive and not too slow between the operations a much smaller subset taking only 16MB (81585 lines) was used. Refine originally gives us the whole line in a single column. This column can be splut into several using a regular expression. In this case `[\'']{1}\s{1}[\''<]{1}` was used. The rows in figure 2.1 have a bad format, different from most of the other rows in the dataset. The carets instead of a space is a special case that must be manually detected. In most cases the given regular expression works and only leaves a `<` at the beginning of the first column and `> .` at the end of the last column. These are easy to remove with 2 more operations.

Refine manages to do a good job with `kern.log`. The lines are split after the first 15 characters to separate the date in it's own column. A new split is performed at `[` and `]`. Remaining data is split using the regular expression `(/:[^$])|(/s{2})`, that is, split either where there are two spaces in a row (indentation) or a comma that is not the last character. See figure .

Next Refine was tested with `pm-powersave.log`. Refine does not handle data where the record data is stored over several lines. There's no way to create rows from a single column.

3.1.3 Data export

To export the data we simply click "Export", select the `.csv` file format (importable by PostgreSQL) and download it to our local disk.



0. Create project

1. Split 13373 cell(s) in column Column 6 into several columns by separator

2. Fill down 882 cells in column Column 6 1

13373 rows

All	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
1.	May 26 22:24:05	per-ThinkPad-X230	kernel: K	ernel loggin	g	
2.	May 27 10:28:09	per-ThinkPad-X230	kernel: i	mklog 5.8.6,		tarted.
3.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	cpuset
4.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	cpu
5.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	
6.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	
7.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	
8.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	
9.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	
10.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	
11.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	BIOS-provided physical R
12.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	BIOS-e820: 00000000000
13.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	BIOS-e820: 00000000000
14.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	BIOS-e820: 00000000000
15.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	BIOS-e820: 00000000001
16.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	BIOS-e820: 00000000200
17.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	BIOS-e820: 0000000020200000 - 0000000040004000 (usable)
18.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	BIOS-e820: 0000000040004000 - 0000000040005000 (reserve
19.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	BIOS-e820: 0000000040005000 - 00000000d0680000 (usable)
20.	May 27 10:28:09	per-ThinkPad-X230	kernel: [	0	]	BIOS-e820: 00000000d0680000 - 00000000dae9f000 (reservec

Figure 5: Screenshot of Open Refine processing `kern.log`

3.2 Data Wrangler

3.2.1 Data import

Data Wrangler can work with data in two ways. For configuration one can simply paste the data into it's web interface. If one has more data that practical to paste into a text box then one can export the operations as python code and process arbitrary amounts of data. There are therefor no size limitations to the data being processed.

3.2.2 Tool usage

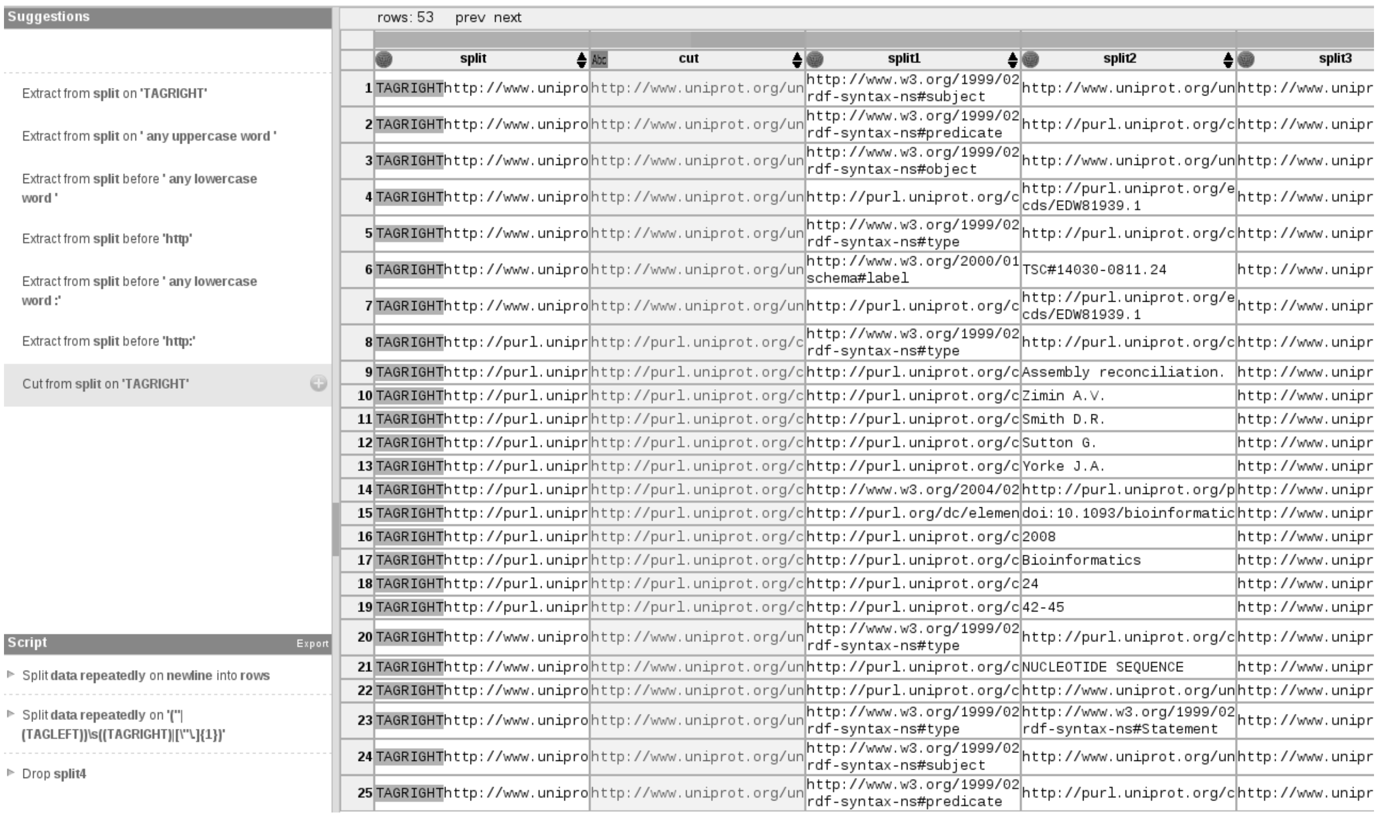
As mentioned above one can chose to use the web interface directly on the data or use the web interface to export a python script describing the operations performed and then run it on several files. The web interface is using Javascript and therefor has some performance issues and only supports 1000 rows but one can use it to configure Data Wrangler on a subset of the data and then apply the configuration on the whole data set. There are also some issues, like the web interface interprets the tagged links in the BTC data to be HTML tags and thus does not show them. Therefor they need to be searched for and replaced by for example `TAGLEFT` and `TAGRIGHT`.

3.2.3 Data export

Data Wrangler has a button where one can choose to export the data in `.csv`.

4 Conclusion

Open Refine is not suitable for processing large data sets. It is built using a server-client architecture which does allow you to run the program on a powerfull server machine but becomes inpractical when the server runs on the client machine since we need to copy the file. Files of size



Extract from **split** on **'TAGRIGHT'**

Extract from **split** on **' any uppercase word '**

Extract from **split** before **' any lowercase word '**

Extract from **split** before **'http'**

Extract from **split** before **' any lowercase word :'**

Extract from **split** before **'http:'**

Cut from **split** on **'TAGRIGHT'**

- Split **data repeatedly** on **newline** into **rows**
- Split **data repeatedly** on **'("|(TAGLEFT))\s((TAGRIGHT)|[\"\.]{1})'**
- Drop **split4**

	split	cut	split1	split2	split3
1	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://www.w3.org/1999/02 rdf-syntax-ns#subject	http://www.uniprot.org/un	http://www.unipr
2	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://www.w3.org/1999/02 rdf-syntax-ns#predicate	http://purl.uniprot.org/c	http://www.unipr
3	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://www.w3.org/1999/02 rdf-syntax-ns#object	http://www.uniprot.org/un	http://www.unipr
4	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://purl.uniprot.org/c	http://purl.uniprot.org/e cds/EDW81939.1	http://www.unipr
5	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://www.w3.org/1999/02 rdf-syntax-ns#type	http://purl.uniprot.org/c	http://www.unipr
6	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://www.w3.org/2000/01 schema#label	TSC#14030-0811.24	http://www.unipr
7	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://purl.uniprot.org/c	http://purl.uniprot.org/e cds/EDW81939.1	http://www.unipr
8	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://www.w3.org/1999/02 rdf-syntax-ns#type	http://purl.uniprot.org/c	http://www.unipr
9	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://purl.uniprot.org/c	Assembly reconciliation.	http://www.unipr
10	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://purl.uniprot.org/c	Zimin A.V.	http://www.unipr
11	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://purl.uniprot.org/c	Smith D.R.	http://www.unipr
12	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://purl.uniprot.org/c	Sutton G.	http://www.unipr
13	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://purl.uniprot.org/c	Yorke J.A.	http://www.unipr
14	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://www.w3.org/2004/02	http://purl.uniprot.org/p	http://www.unipr
15	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://purl.org/dc/elemen	doi:10.1093/bioinformatic	http://www.unipr
16	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://purl.uniprot.org/c	2008	http://www.unipr
17	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://purl.uniprot.org/c	Bioinformatics	http://www.unipr
18	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://purl.uniprot.org/c	24	http://www.unipr
19	TAGRIGHThttp://purl.unipr	http://purl.uniprot.org/c	http://purl.uniprot.org/c	42-45	http://www.unipr
20	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://www.w3.org/1999/02 rdf-syntax-ns#type	http://purl.uniprot.org/c	http://www.unipr
21	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://purl.uniprot.org/c	NUCLEOTIDE SEQUENCE	http://www.unipr
22	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://purl.uniprot.org/c	http://www.uniprot.org/un	http://www.unipr
23	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://www.w3.org/1999/02 rdf-syntax-ns#type	http://www.w3.org/1999/02 rdf-syntax-ns#Statement	http://www.unipr
24	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://www.w3.org/1999/02 rdf-syntax-ns#subject	http://www.uniprot.org/un	http://www.unipr
25	TAGRIGHThttp://www.unipro	http://www.uniprot.org/un	http://www.w3.org/1999/02 rdf-syntax-ns#predicate	http://purl.uniprot.org/c	http://www.unipr

Figure 6: Screenshot of Data Wrangler processing a subset of BTC

512MB or smaller could be processed on the test machine but the waiting times for this file size made it impractical for trial and error use. The amount of data that can be processses is probably limited by the amount of working memory.

Data Wrangler is more promising, although still in beta phase. Since you can export the operations to Python code you can process large data sets. It is also much smarter since it suggests operations when the user for example highlights text, making life easier for users that don't know regular expressions.

5 Related work

There's been some research in investigating what is called data cleaning and quality tools. Research papers that I have read on the topic has been focusing on the issue where data from several sources, with different formats, needs to be combined in a single database and thus data needs to be cleaned from inconsistency and bad formatting. [6] provides a good overview of some approaches and problems related to data cleaning. Data cleaning in this paper is described as "... detecting and removing errors and inconsistencies from data in order to improve the quality of data", which is not quite the same as the aim of my project.

I have not found any papers that tries to do the same thing as me.

References

[1] PADS. http://www.padsproj.org/. Retrieved 5/16 2013.

[2] Billion Triples Challenge 2010. http://cass-mt.pnnl.gov/btc2010/ Retrieved 5/16 2013.

[3] Open Refine. http://code.google.com/p/google-refine/. Retrieved 5/16 2013.

[4] Data Wrangler. http://vis.stanford.edu/wrangler/. Retrieved 5/16 2013.

[5] Kandel, Sean, et al. "Wrangler: Interactive visual specification of data transformation scripts." PART 5———Proceedings of the 2011 annual conference on Human factors in computing systems. ACM, 2011.

[6] Rahm, Erhard, and Hong Hai Do. "Data cleaning: Problems and current approaches." IEEE Data Engineering Bulletin 23.4 (2000): 3-13.