**CS513: Theory & Practice of Data Cleaning Project**

End-to-End Data Cleaning Workflow

MCS-DS program at the University of Illinois at Urbana-Champaign – Fall 2018

Abstract: This report describes a data cleaning workflow using the New York Public Library Rare Books Division historical menus dataset has an example to demonstrate various cleaning techniques. This project will use the following software and tools to clean and organize the dataset: OpenRefine, SQLite, and YesWorkFlow.

**Team members**

Brad Ballard - bjb3@illinois.edu

Dhanendra Singh - disingh2@illinois.edu

Jacob Rettig - jrettig2@illinois.edu

**1. Dataset Overview and Initial Assessment**

For the project, we use tools introduced in CS513 to clean and prepare a sample dirty dataset. Along the way we will document the steps from dirty to clean. The sections that follow describe this process in detail: (2.) Data cleaning with OpenRefine, (4.) Develop Rational Database Schema, and (5.) Create a Workflow Model.

An initial assessment of the New York Public Library dataset (referred to now forward as ‘NY menus’ dataset) is over 45,000 historical menus. The majority of these were organized by Frank E. Buttolph (Ref1) around 1900-1921. The dates on the menus range from the 1850s to 2010s. The data contains information on restaurant menu, but also other organizations like railroad or shipping companies. The dataset was digitized in 2011 via the “What’s on the Menu?” project (Ref2). So far 17,500 the libraries’ historical menus have been digitized. We are using the June 17, 2017 version for this project.

The data is in four files: Menu, MenuPage, MenuItem, and Dish. The ‘Menu.csv’ file has a unique id number, location information, venue, currency used, and other description-based information. The ‘MenuPage.csv’ file contains the id, plus an additional unique mean\_id, image\_id, height, width, and another with other image related information. The ‘MenuItem.csv’ file contains the id, plus an additional\_page\_id, dish\_id, and other price related information for each dish. The ‘Dish.csv’ file contains the id, name of the dish, description, first/last appearance, and various price information.

A detailed description of each file’s columns follows:

*Menu.csv*

**id**: unique id for this menu

**name**: name on menu, name of the restaurant, or blank

**sponsor**: sponsor, often the name of the restaurant

**event**: name of the meal or the event the menu was created for

**venue**: location where the food is served

**place**: often includes city, state, country, address, or name of venue

**physical description**: paper stock, dimensions, colors, design, etc. of menu

**occasion**: special occasion, holiday, daily, or blank

**notes**: additional details about the menu

**call number**: number within the NYPL collection

**keywords:** keywords on menu

**language**: language the menu is printed in

**date**: date the menu was collected, formatted as a string as YYYY-MM-DD  
**location:** where the menu was used  
**location type**: type of the location value

**currency**: money type charged for items on this menu

**currency symbol**: symbol for the currency  
**status:** the digitization status of this menu – complete or under review

**page count**: number of pages on the menu  
**dish count:** number of dishes on the menu

*MenuPage.csv*

**id**: unique designator for this menu item  
**menu id**: specific id for menu  
**page number**: page number in the menu  
**image id**: a unique id for the scanned image of this menu, accessible on the NYPL site

**full height**: height of menu  
**full width**: width of menu  
**uuid**: another unique id for this page/image

*MenuItem.csv*

**id**: unique designator for this menu item  
**menu page id**: id of the menu page that this item appears

**price**: the cost of the smallest portion of this item  
**high price**: cost of the largest portion of this item  
**dish id**: designator id of the dish that this menu item refers  
**created at**: date/time that this database entry was created  
**updated at**: the most recent date the database entry was updated  
**xpos**: x-axis position of the item on the scanned image  
**ypos**: y-axis position of the item on the scanned image

*Dish.csv*

**id**: a unique designator for this dish

**name**: the name of this dish

**description**: a description of this dish, always blank

**menus appeared**: number of menus this dish appears on

**times appeared**: number of times this dish appears (including additional sections)

**first appeared**: year this dish first appeared (also can be: 0, 1, or NA)

**last appeared**: year this dish last appeared (also can be: 0, 1, 2928, or NA)

**lowest price**: lowest price that this item was sold for

**highest price**: highest price that this item was sold for

Some hypothetical use cases for a dataset like this could be – How has the composition of restaurants changed over time? The density, clusters, or price changes? Can a model be made to predict food prices in certain areas? How have consumer food preferences changed over time? Can a model be made to estimate food prices based on ingredients/description?

The problem with answering all these questions is - the dataset is quite messy and need to be organized. For example, the date columns seem full of repeat or missing dates. However, some files are cleaner than others; we are not choosing to clean MenuPage.csv, as it is clean enough.

How clean does the dataset need to be? You can sort the columns and get some sense of the composition of the data, but there will need to be some prep work to answer practical questions. Having the data broken into multiple files makes it hard to compare across sheets. However, this format should be a straight forward to import into the SQL database.

Progress throughout his project can be tracked at our github repository (Ref3) and this associated webpage (Ref4). This platform was mostly used to aid in our team collaboration, but also provides a way to share our data cleaning techniques with a wider audience outside of this class.

**2. Data cleaning with OpenRefine**

OpenRefine, formally called Google Refine, is an open source desktop application the has many helpful features for data cleaning. It behaves like a database with rows and cells under columns which are similar to relational database tables. An OpenRefine project itself consists of one table. I plan to clean each of the four data file separately, some more than others as needed.

The major OpenRefine feature that will be helpful in cleaning our dataset is the clustering feature. The option allows the user to cluster similar text and replace it with a more standardized description. The common problem this solves the many variants of spelling but reference the same object.

The following subsections will describe the step-by-step process from input file to output file. We will use UTF-8 encoding.

***### Input File: Menu.csv #####################***

*For column: sponsor*

(1) Trim leading and trailing white spaces

(2) Collapse consecutive white spaces

(3) Convert all column values to upper case

(4) Remove special characters using GREL (%, #, !, /, (, ), [, ], ?)

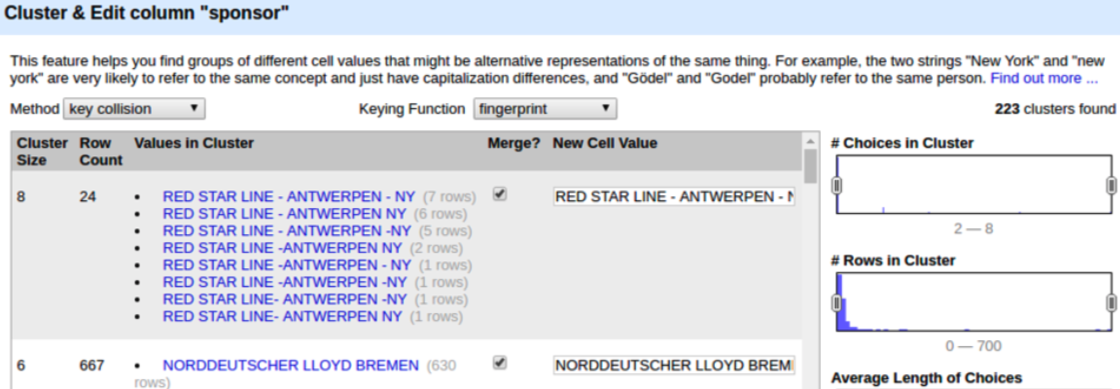
(5) Replace “;” with a space instead and then trim leading/trailing white spaces

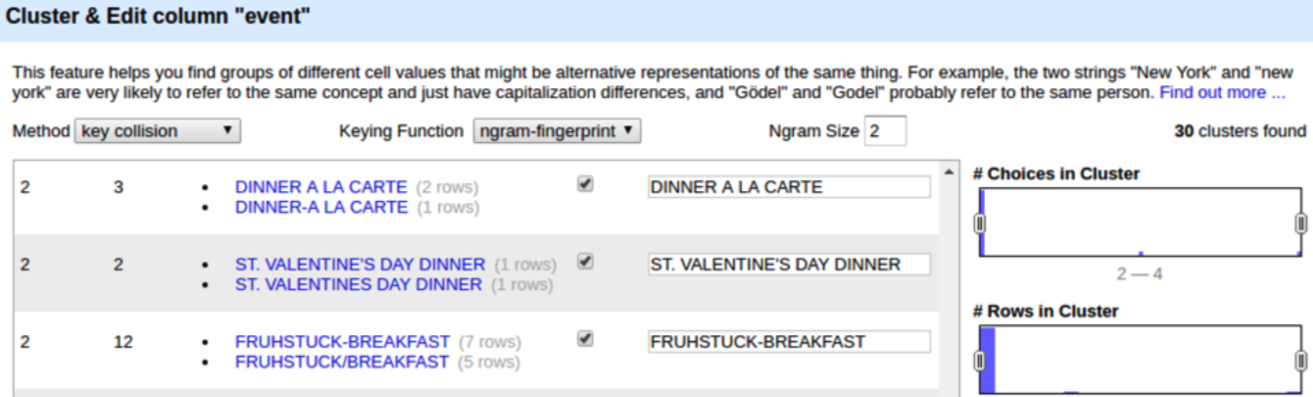
(6) Make a facet and perform the cluster operation using the key-collision method and fingerprint function. Next merge the selected clusters.

(7) Repeat the previous step with n-gram, fingerprint, meta-phone3, and cologne-phonetic methods.

(8) Create a new facet. Next, use the cluster operation nearest neighbor method and levestein distance function. Then, merge the selected clusters.

(9) Create a new facet. Next, use the cluster operation nearest neighbor method and PPM distance function. Then, merge the selected clusters.





*For column: physical\_description*

(1) Slit the columns using ‘;’

(2) Then rename the first column: physical\_description\_type

(3) Use GREL to join ‘physical\_description 2’, ‘physical\_description 3’, and ‘physical\_description 4’ into one column named: physical\_description\_additional.

(4) Use ‘ - ‘ to separate the values from the different columns (Remember the space before and after the dash). Note: if the cell in the column is empty, leave that value as a blank space. For example, if a column is blank, make the new column in ‘physical\_description\_additional’ blank also.

*For column: date*

(1) Convert date format to YYY-MM-DD

(2) Remove date outliers, where the year is either less than 1851 or greater than 2012. Make these flagged cells blank.

*For column: call\_number*

(1) Trim leading/trailing white spaces

(2) Collapse consecutive white spaces

Unchanged columns: id, name, keywords, language, status, page\_count, dish\_count

***> Output File: CMenu.csv***

***### Input File: MenuPage.csv ##################***

File not cleaned in OpenRefine

Unchanged columns: id, menu\_id, page\_number, image\_id, full\_height, full\_width, uuid

***> Output File: CMenuPage.csv***

***### Input File: MenuItem.csv ##################***

*For column: created\_at*

(1) Convert date format to YYYY-MM-DD

(2) Remove date outliers, where the year is either less than 1851 or greater than 2012. Make these flagged cells blank.

*For column: updated\_at*

(1) Convert date format to YYYY-MM-DD

(2) Remove date outliers, where the year is either less than 1851 or greater than 2012. Make these flagged cells blank.

***> Output File: CMenuItem.csv***

***### Input File: Dish.csv ##################***

*For column: name*

(1) Use key-collision to cluster values. Note: cannot do nearest\_neighbours cluster method because the time required to do this too computationally expensive

***> Output File: CDish.csv***

**3. (Optional) Clean data further with an alternative tool**

We choose not to do this optional part.

“If you find the certain steps are not well suited for OpenRefine, consider applying an alternative, more suitable solution, or another too such as Trifacta Data Wrangler, Tableau. Document you choice and consider the same questions as in Step 2.”

**4. Develop Rational Database Schema**

The next step in our data cleaning workflow is to import the cleaned data into a database. We are choosing to use SQLite as it is open source, simple, and straightforward to work with. When first designing a database, we must think about integrity constraints and the relational schema.

[discuss integrity constraints, schema code to import into database]

[INSERT SCREENSHOTS]

**5. Create a Workflow Model**

YesWorkFlow is a solution to annotating your data workflow. It’s easy to script and doesn’t require you to re-write any of your existing code. Simply add a special (YW) comments to your existing script. Later, these comments are used to declare how the data was transformed step-by-step.

[discuss your YW process]

[INSERT SCREENSHOTS]

**6. (Optional) Develop provenance queries**

We choose not to do this optional part.

“Develop provenance queries (in Datalog/DLV) that show on which inputs and intermediate data and steps the outputs of your workflow depend.”

**7. Conclusion**

Any scientist, engineer, or researcher spends a substantial amount of time cleaning data for their research. Further, at conferences or when publishing research papers, often detailing steps documenting where you got the data from and how it was cleaned are required. These steps are required in order to potentially replicate your research/models later. Tools described in this paper aim to help document these cleaning steps in a simple and easy way.

In this project we used OpenRefine and SQLite for the cleaning steps. The challenges we faced generally is handling all the duplicate names or invalid/NA entries. Without much information about how to fill in the gaps, we often had to delete these entries. One tries to avoid deleting data if possible, but in our case, we felt it an appropriate solution.

Due to the size of the datasets in this project, it is hard to verify integrity of each clustering step. In some cases, we were not able to see all the clusters that were suggested by OpenRefine and had to rely on its decision on those.

Another problem is the slow run time and manual man hours - imputing GREL commands, reviewing cluster groups, documenting steps – data cleaning still requires a lot of time even given tools like these. OpenRefine, like excel and some other tools, has limitations in the size of datasets it can work with. This is an example of where other open source programming languages help, such as Python or R. There is no ‘silver bullet’ in data cleaning. It remains a piecewise blend of various tool as the data moves through a line of steps.

**References**

[Ref1] https://en.wikipedia.org/wiki/Frank\_E.\_Buttolph

[Ref2] http://menus.nypl.org

[Ref3] https://github.com/bradjballard/dc-workflow

[Ref4] https://bradjballard.github.io/dc-workflow

**Appendix**

here

[INSERT SCREENSHOTS]