What IS on the Menu! A Case Study in Data Cleaning and Provenance

Alok K. Shukla,Gitika Jain,Apurva V. Hari

Final Project Report

Abstract—Data Preparation - Cleaning and Wrangling; the first step of any Data Science work-flow is often most time consuming and least enjoyable of all tasks. And that is mainly because these tasks (specially cleaning) largely involve lot of manual steps - mostly repetitive and tedious, which can not be still fully automated; thus tools like Amazon Mechanical Turks [1] and CrowdFlower [2] are quite popular for these reasons. Another issue with whole Data Science work-flow is that of Provenance - lineage of datasets; the experiments are often not reproducible due to lack of proper provenance. The focus of this report is on use of Open Source tools [3] [4] [5] that can help Data Cleaning work-flows be more effective and traceable.

Index Terms—Data Cleaning, Provenance, OpenRefine, YesWorkflow, Workflow, SQL

1 Introduction

This report summarizes our experience with an end-to-end data preparation work-flow; in practice of Data Cleaning and Provenance establishment. We use tools and techniques introduced in CS598 [6]: Theory and Practice of Data Cleaning with a real world dataset [7] and document the whole work-flow along with findings. Tools used include OpenRefine [3], SQLite [5] and YesWorkFlow [4].

2 Dataset overview and initial assessment

2.1 The Dataset

The New York Public Library Rare Book Division holds over 45,000 historical menus. About half of these were collected and curated by Frank E. Buttolph [8] between 1900 and 1921. The menus date from the 1850s to the present and include menus from restaurant, railroad and steamship companies, as well as a range of other organizations.

Beginning in 2011, menus from the NYPL's collection were digitized and transcribed with the help of thousands of volunteers. Through the NYPL's What's on the Menu? [7] project, volunteers looked at digitized copies of the menus and typed in the many pieces of information included on each one, such as restaurant names, locations, dishes, prices, and dates.

- *Alok K. Shukla,*
  E-mail: alokks2@illinois.edu,
- *Gitika Jain,*
  E-mail: gitikaj2@illinois.edu,
- *Apurva V. Hari,*
  E-mail: vhari2@illinois.edu,
  University of Illinois at Urbana–Champaign.

Submitted on Aug 5, 2017.

The What's on the Menu? [7] project makes all the data from its crowd-sourced transcriptions available via bulk downloads and via an application programming interface (API). The current data set includes around 450,000 data points from the transcription project and the library's meta-data on the over 17,500 menus digitized so far.

This dataset is split into four files to minimize the amount of redundant information contained in each (and thus, the size of each file). The four data files are *Menu*, *MenuPage*, *MenuItem*, and *Dish*. These four files are described briefly here, and in detail in the appendix B.

Menu
: The core element of the dataset. Each Menu has a unique identifier and associated data, including data on the venue and/or event that the menu was created for; the location that the menu was used; the currency in use on the menu; and various other fields. Each menu is associated with some number of MenuPage values.

MenuPage
: Each MenuPage refers to the Menu it comes from, via the menu_id variable (corresponding to Menu:id). Each MenuPage also has a unique identifier of its own. Associated MenuPage data includes the page number of this MenuPage, an identifier for the scanned image of the page, and the dimensions of the page. Each MenuPage is associated with some number of MenuItem values.

MenuItem
: Each MenuItem refers to both the MenuPage it is found on – via the menu_page_id variable – and the Dish that it represents – via the

dish_id variable. Each MenuItem also has a unique identifier of its own. Other associated data includes the price of the item and the dates when the item was created or modified in the database.

Dish

A Dish is a broad category that covers some number of MenuItems. Each dish has a unique id, to which it is referred by its affiliated MenuItems. Each dish also has a name, a description, a number of menus it appears on, and both date and price ranges.

2.2 Use-cases discussion

2.2.1 Is data clean enough already?

The data is really messy for any actual practical use-case other than getting an overall sense of number of menus and dishes from a particular time period - since the date columns even though not perfectly clean can be used to get an aggregate sum. Few more -

- Analyzing the dish dataset can provide the information about the popular dishes based on how many times the dish has been appeared in a menu, when was the first/last time the dish appeared in the menu.
- Menu Page can be used to gather the information about the menu structure like height, width in previous years.
- Menu Page and Menu item together can be used to get the information about a particular menu item when and where it appeared on a menu.

2.2.2 Potential use-cases of cleaned Data

Once properly cleaned, we can discuss the possibility of working on use-cases like -

- Can be used to predict the food preferences over the time based on the event, years and location.
- How the Popularity of a dish changed over the time.
- The structure of the Menu.
- How has the median price of restaurant dishes changed over time?Are there particular types of dishes (alcoholic beverages, seafood, breakfast food) whose price changes have been greater than or less than the average change over time?
- Can we predict anything about a dish's price based on its name or description?
- There's been some work on how the words used in advertisements for potato chips are reflective of their price; is that also true of the words used in the name of the food?
- Are, for example, French or Italian words more likely to predict a more expensive dish?

2.3 The Work-Flow Environment

2.3.1 Version Control and Collaboration

We used private GitHub repository for version control and collaboration; we also used the GitHub Project Boards feature to track our progress.
Note: The repository will be made public as soon as the submission deadline is over. If its still inaccessible, please contact alokks2@illinois.edu

2.3.2 AWS Setup

In order to make our work reproducible and benchmarked; we setup a standard environment for this cleaning work flow. We used Amazon AWS's EC2 instance to perform this complete case study. Anyone interested can reproduce the same results that we got; either by creating similar EC2 instance and cloning our repository; or by logging into our instance and running the project thats already setup. Details of environment are in appendix A.

3 Data Cleaning with OpenRefine

Other than the standard textual data cleaning operations (removing unnecessary spaces, special characters, regular expression matching); we were impressed by OpenRefine's Clustering feature; it gives us ability to cluster similar text and replace it by one decided upon value, which is particularly useful in this case, where a dish can have many variants of the ways it is spelled. Following is description of operations that we performed while cleaning the data files with OpenRefine. The deliverables of this step are listed as download-able links in appendix C.
Each file is separately loaded into OpenRefine first (with UTF-8 encoding) and then cleaning is performed step by step; when the file is clean enough; we export it to another CSV and also save the operations history JSON.

3.1 Menu

File: *Menu.csv* Size: *3.2 MB* Rows: *17545*
A moderately sized dataset, could be easily cleaned with following steps. Description is given per column basis.

3.1.1 *sponsor*

1) Trim leading and trailing white spaces.
2) Collapse consecutive white spaces.
3) Convert column values to upper case
4) Remove the special characters '% # ! / ( )[ ] ?' using GREL.
5) Replace Semicolon (;) with a space and then trim leading and trailing white spaces and collapse consecutive white spaces.
6) Make a facet and perform the cluster operation using the **key-collision** method and **fingerprint** function. Merge the relevant clusters.
7) Repeat last step with **n-gram fingerprint, metaphone3, cologne-phonetic** methods.
8) Make a facet and perform the cluster operation using the **nearest neighbor method** and **levestein distance** function . Merge the relevant clusters.

9) Make a facet and perform the cluster operation using the nearest neighbor method and **PPM distance** function. Merge the relevant clusters.

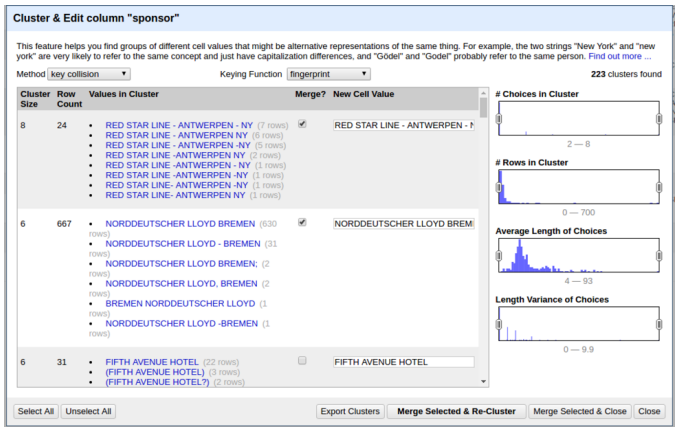

Figure 1. Clustering ”sponsor”

Repeat same steps for **event, venue, place, occasion** and **location**.

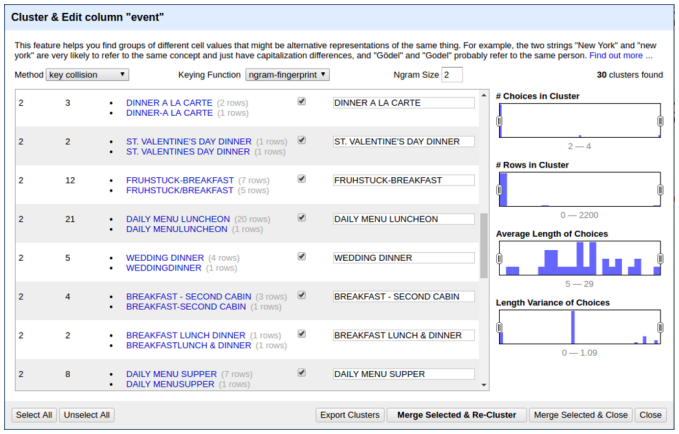

Figure 2. Clustering ”event”

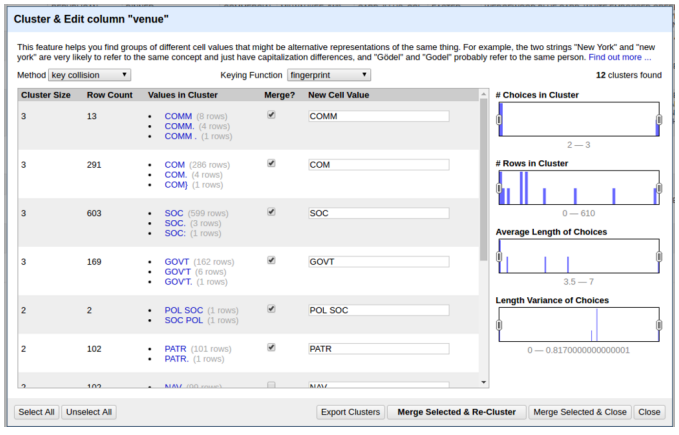

Figure 3. Clustering ”venue”

3.1.2 *physical_description*

1) Split the column values using semicolon (;) as separator, we get 7 columns as a result.
2) Rename the first column as ”physical_description_type”.
3) Using the GREL function, join only the separated columns - ”physical_description 2”, ”physical_description 3”, and ”physical_description 4” into one column, separate the value using a dash (-) character and name the column as ”physical_description_additional”. Leave the respective value as blank space if the column is null or blank.
4) In case, if the ”physical_description 2” column is empty, then make ”physical_description_additional” empty as well.

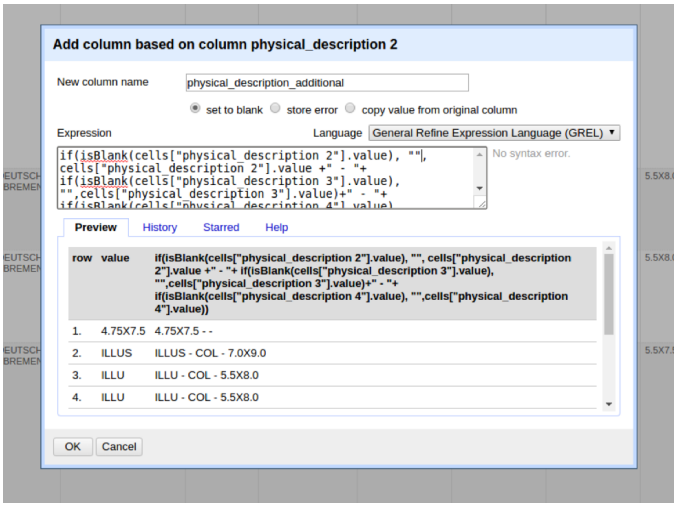

Figure 4. Cleaning ”physical_description”

3.1.3 *date*

- Convert the ”date” column values to date format ”YYYY-MM-dd”. Also, get rid of outliers where the year is “less than 1851 and more than 2012”.
- Replace the outliers with empty characters.

3.1.4 *call_number*

1) Trim leading and trailing white spaces.
2) Collapse consecutive white spaces.

Repeat same steps for **notes**.

3.1.5 *Left as-is*

id, name, keywords, language, status, page_count, dish_count.

3.2 MenuPage

File: *MenuPage.csv* Size: *4.7 MB* Rows: *66937*
No column cleaned with OpenRefine.

3.3 MenuItem

File: *MenuItem.csv* Size: *118 MB* Rows: *1332726*
The **created_at** and **updated_at** columns were cleaned as **date** column of **Menu** file.

3.4 Dish

File: *Dish.csv* Size: *26.5 MB* Rows: *423397*

The **name** column presents opportunities for clustering but the sheer volume of labels was a challenge for OpenRefine. We could only use **key-collision** clustering methods as they are linear in their time complexity; **nearest_neighbour** methods were computationally too expensive for us.

3.4.1 Configuring OpenRefine for larger datasets

By updating parameters in **refine.ini** we were able to work with Dish data file. Specifically we took following steps.

1) **Increase Heap-Size** Update *REFINE_MEMORY*
2) **Increase Form Size** To deal with following issue, update *REFINE_MAX_FORM_CONTENT_SIZE* . We eventually set it to 999999999.

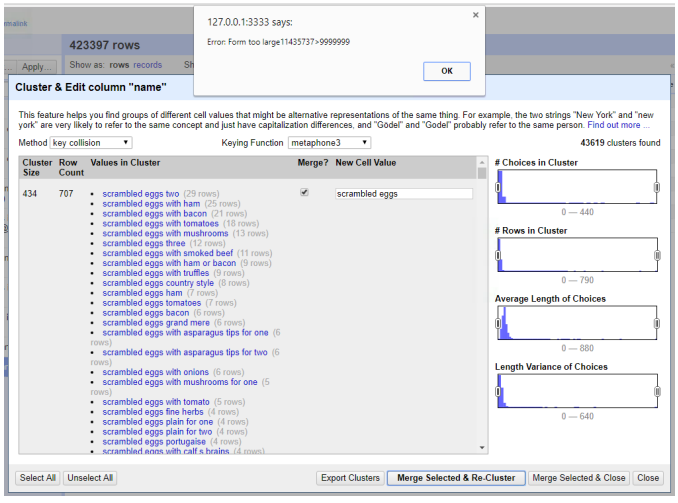

Figure 5. The Form Size problem in clustering large datasets

Details on why we couldn't use non-**key-collision** clustering method are available at this wiki. Specifically how it was computationally too expensive given our resources to use **nearest-neighbor** methods.

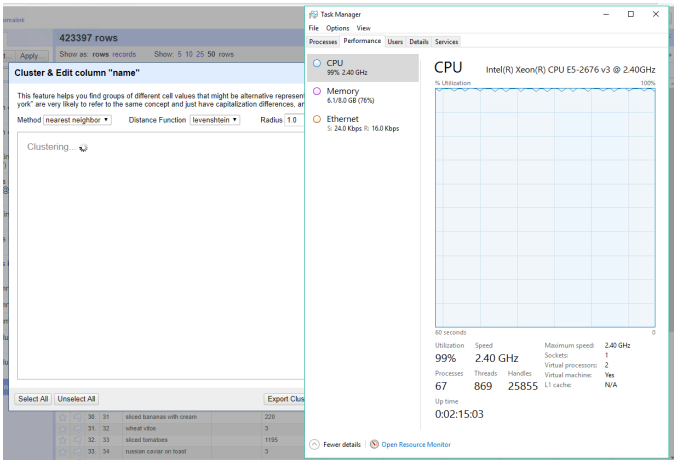

Figure 6. Trying Nearest Neighbor Clustering

Following is approximate summary of time taken in linear **key-collision** clustering operations on Dish dataset.

- **fingerprint** : *45 Seconds*
- **n-gram fingerprint, n=2** : *5 Seconds*
- **metaphone-3** : ***5 minutes***

The **date** column was also cleaned to extract year info using date utilities.

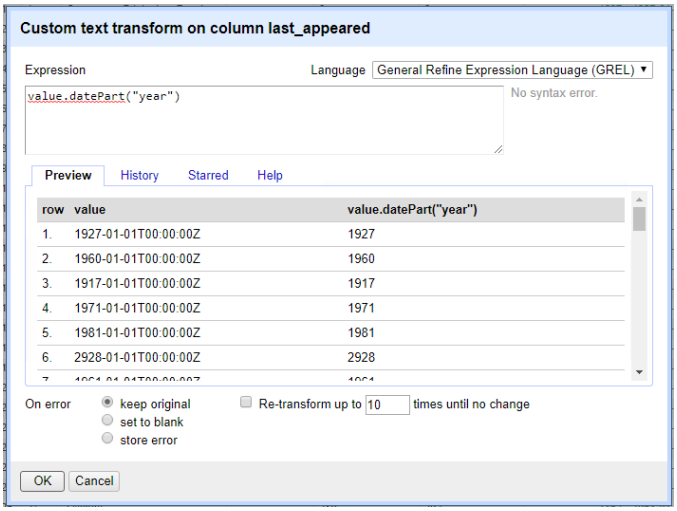

Figure 7. Extracting year from date.

4 Developing a Relational Schema with SQL

Using SQLite with SQLite Studio, we first imported CSV files into a db and then performed integrity constraints checks.

4.1 The Schema

Figure 8 shows the Entity Relationship (ER) diagram for the dataset we selected for this project. It depicts the simple relationships between the data tables.

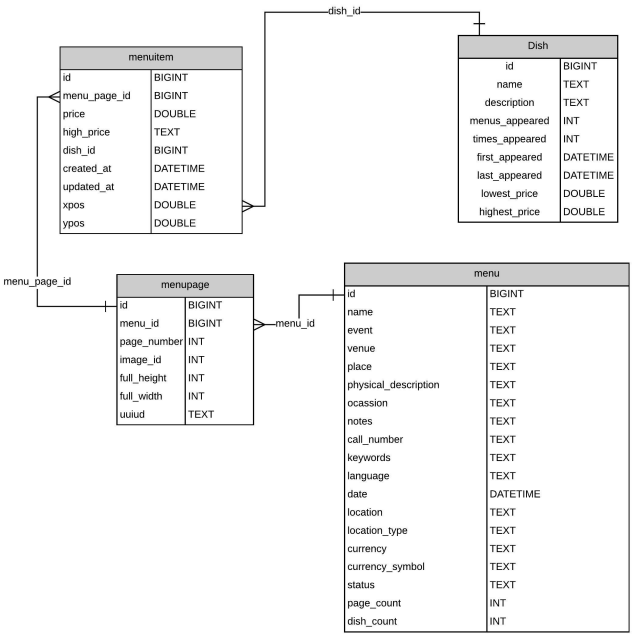

Figure 8. ER Diagram for the dataset

4.2 Integrity Constraints Check

For every table, few constraints were checked and dirty data discarded.

4.2.1 *Dish*

We implemented the following integrity constraints for the *dish* table.

1) id has to be distinct without duplicates (shown in Figure 9).

Figure 9. Integrity constraints (1/6) for dish.csv

2) id should not be NULL (also shown in Figure 9).
3) menus_appeared and times_appeared cannot be NULL (shown in Figure 10).

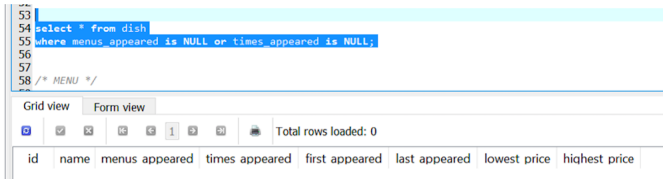

Figure 10. Integrity constraints (2/6) for dish.csv

4) first_appeared and last_appeared should be between 1851 and 2012. We wrote the following query for this task.

```
select * from dish
where
(first_appeared
not between '1851' and '2012')
and
(last_appeared
not between '1851' and '2012');
```

We found that some rows violated this constraint (not shown here). We deleted those rows to clean the dataset.

5) We also found that there are rows of data where last_appeared is 0.
We executed the following query to obtain such rows. We did not delete such rows as these dishes may still be appearing elsewhere in the dataset.

```
select * from dish
where
last_appeared = '0';
```

6) lowest_price for any dish should be less than highest_price (as shown in Figure 11).

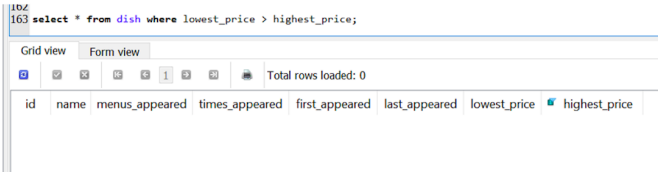

Figure 11. Integrity constraints (3/6) for dish.csv

7) lowest_price and highest_price should not be 0 (shown in Figure 12)

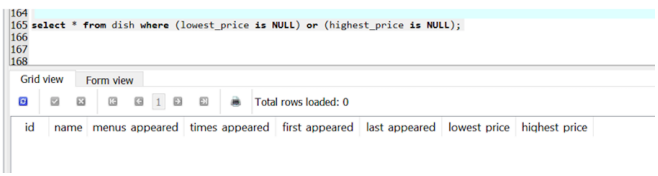

Figure 12. Integrity constraints (4/6) for dish.csv

8) We delete the *description* column as it is blank. We performed this task using the option provided by the SQLite GUI (as shown in Figurefig:dishic6).

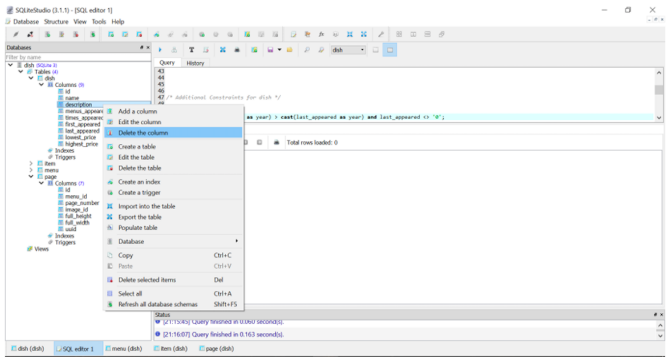

Figure 13. Integrity constraints (5/6) for dish.csv

9) first_appeared should not be greater than last_appeared. We show the query and the output of running this query in Figure 14.

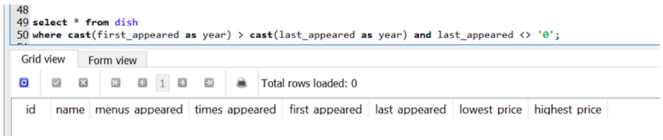

Figure 14. Integrity constraints (6/6) for dish.csv

4.3 MenuPage

We next implemented the constraints for the *menupage* table.

1) id should not be NULL. We show the SQL query and the output in Figure 15.

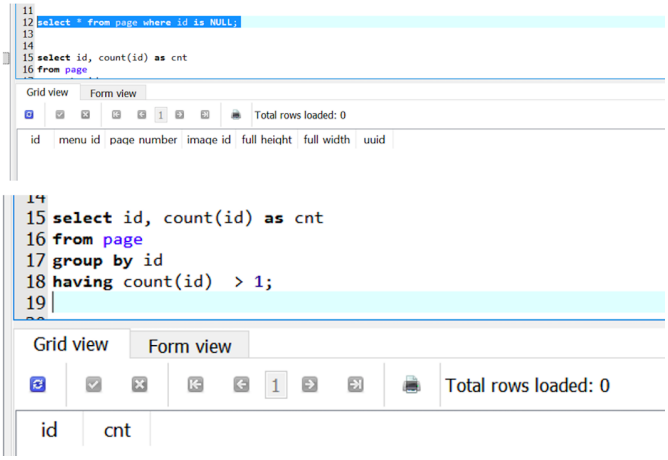

Figure 15. Integrity constraints (1/3) for menupage.csv

2) id should be unique. We show the SQL query and the output in Figure 15.

3) Page_number should not be NULL or 0. We show the queries written for this task in Figures 16 and 17.

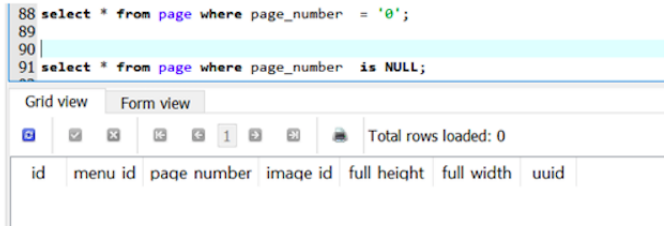

Figure 16. Integrity constraints (2/3) for menupage.csv

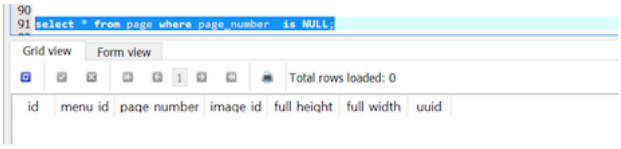

Figure 17. Integrity constraints (3/3) for menupage.csv

4.4 MenuItem

We implemented the following constraints for the *menuitem* table.

1) id is unique and not null (as shown in Figure 18).

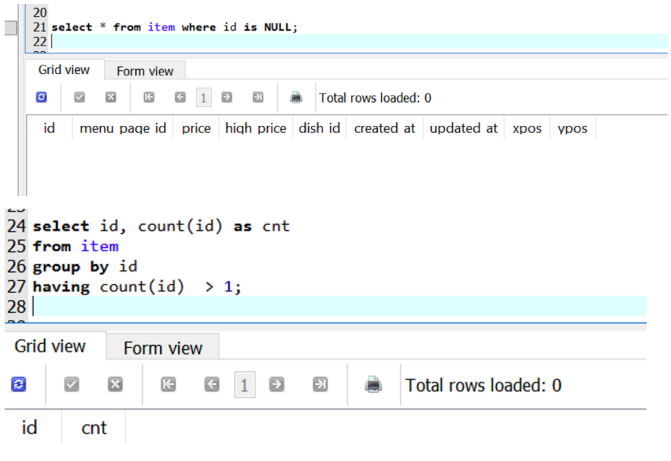

Figure 18. Integrity constraints for menuitem.csv

2) xpos and ypos values should always be between 0 and 1 (as shown in Figures 19 and 20).

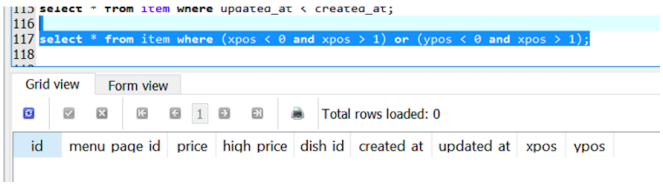

Figure 19. Integrity constraints (1/2) for menuitem.csv

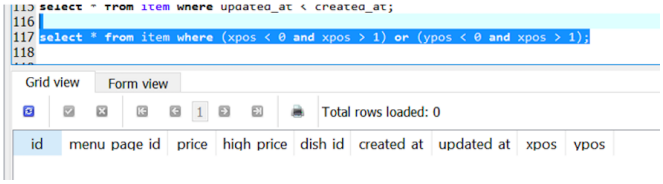

Figure 20. Integrity constraints (2/2) for menuitem.csv

3) updated_at should always be greater than created_at. There is an integrity violation based on this constraint. We show it in Figure 21.

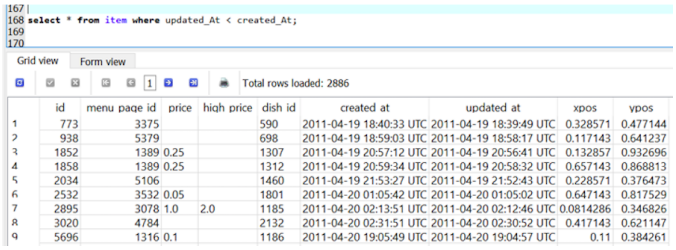

Figure 21. Integrity constraint violations for menuitem.csv

4.5 Menu

We implemented the following constraints for the *menu* table.

1) id has to be unique. Figure 22 shows the query and the output.
2) id should not be NULL. Figure 22 shows the query and the output.

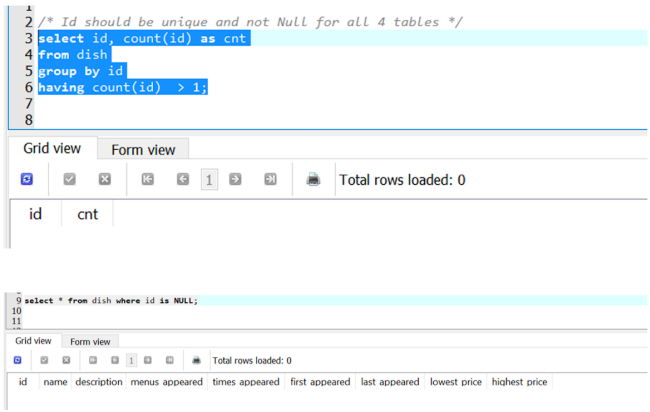

Figure 22. Integrity constraints for menu.

3) sponsor cannot have blank or NULL values. We wrote the following SQL qeury to check this constraint.

```
select * from menu
where
sponsor is NULL or '';
```

We found several records that violate this constraint with blank sponsor name. We deleted such records.

4) Delete the column language, keywords, location_type, physical_description 7 as they are blank. We deleted these columns using the SQLite GUI similar to the procedure we followed when we deleted the column description for the table dish (as shown in Figure 13).
5) page_count cannot be 0 or NULL. We wrote the following query for this constraint.

```
select * from menu
where
page_count = '0' or NULL;
```

We did not find any records that violate this constraint.

5 Creating a Work Flow Model with YesWorkflow

We used the YesWorkflow on-line editor to create the workflow graph for whole process. The actual script is linked in appendix C .

5.0.1 *The Complete Work-flow Graph*

The complete graph generated by YesWorkflow in Figure - 23 with data and operations' both graphs.

5.0.2 *Data-flow Graph*

Without the operations - Data Lineage in Figure - 24

5.0.3 *Operations Graph*

Figure - 25 is without the data - only operations' graph.

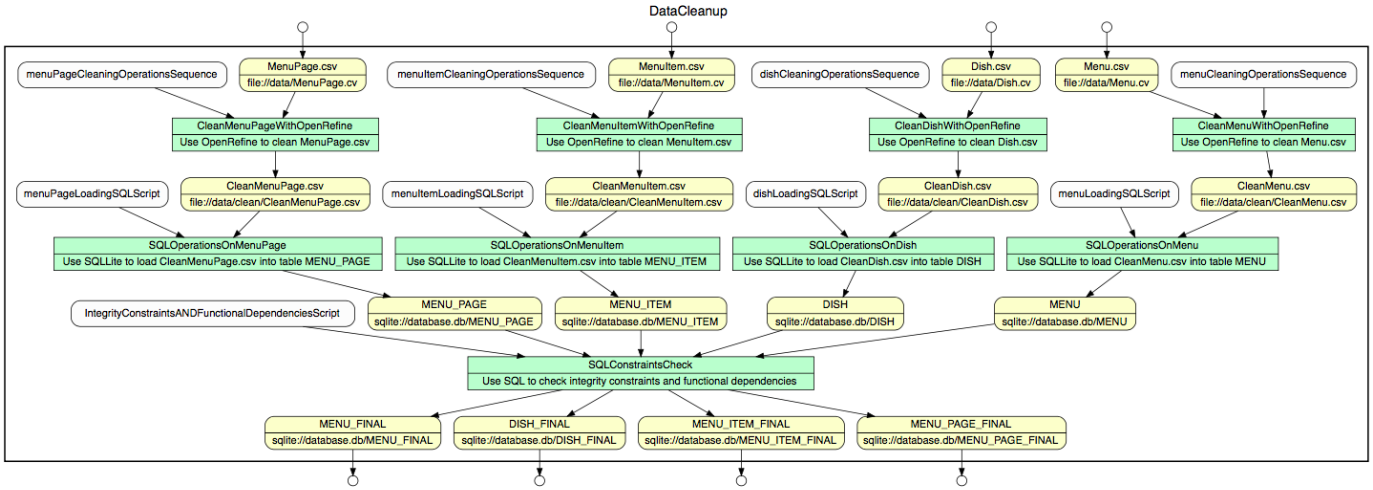

Figure 23. The Work Flow Graph

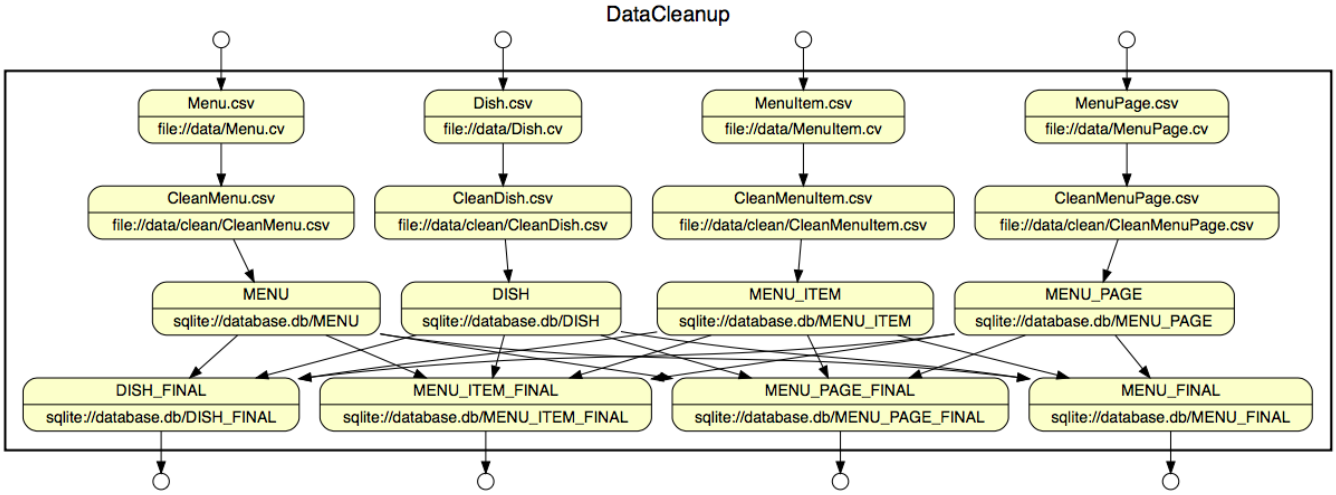

Figure 24. The Data Flow Graph

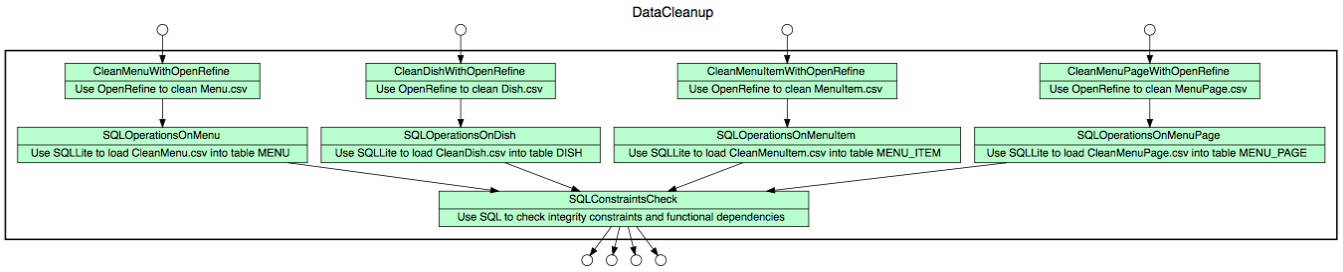

Figure 25. The Operations Graph

6 Challenges

The most challenging problem within data cleaning remains the correction of values to remove duplicates and invalid entries. In many cases, the available information on such anomalies is limited and insufficient to determine the necessary transformations or corrections, leaving the deletion of such entries as a primary solution. The deletion of data, though, leads to loss of information; this loss can be particularly costly if there is a large amount of deleted data.
When data is this huge; there is no way to completely verify the integrity of our steps - specially in case of clustering - we could not see what all clusters are suggested by OpenRefine; and we had to blindly trust its suggestions.

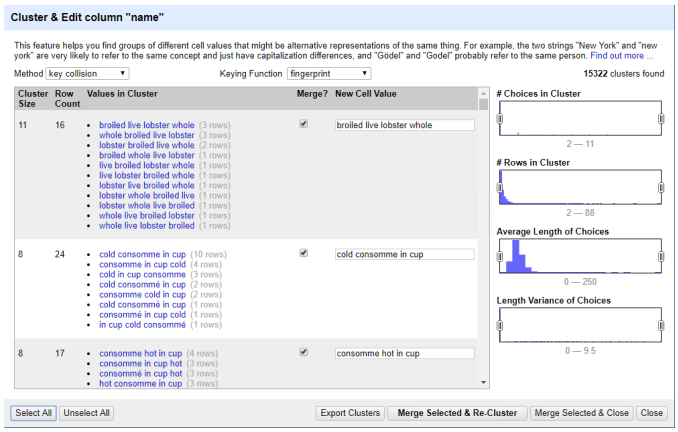

Figure 26. Too many clusters to verify!

And, we had no way of knowing this in advance; that we would be able to merge *scrambled eggs* and *eggs scrambled* in advance; what if there were others too, say *orange juice* and *juicy orange*.

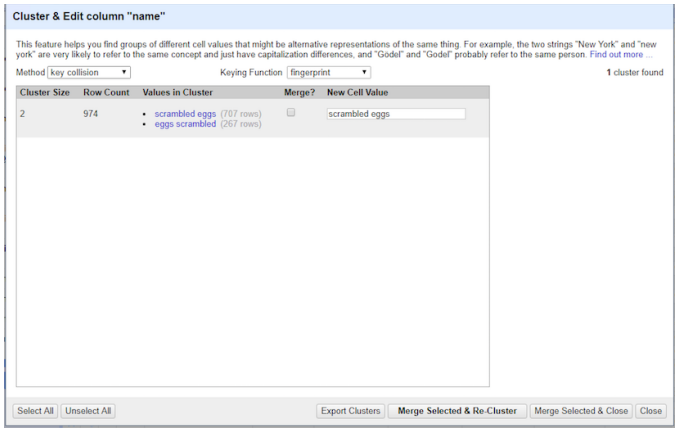

Figure 27. The lucky case of Scrambled Eggs!

These challenges validate the need of ***human in the loop*** and thats why companies like CrowdFlower [2] are booming.

7 Conclusion

Data Cleaning or Data Wrangling is not only an effective tool for removing the unwanted data "dirty" data, but also the medium to make data in our system selective, concise and appropriate in order to perform the better analysis. Based on our experience, we can say Data Cleaning is one of the most time-consuming steps in any Data Analysis.
We tried to clean the data using OpenRefine and SqlLite. It took us significant time to clean the data using OpenRefine, specially because of the way it enables its usage, i.e through GUI - with manual steps. Major portion of the cleaning was done in the Menu and Dish dataset. Identifying integrity constraints also took a significant amount of time.
Tools like OpenRefine have their limitations in terms of how much data can they handle; and thats why more people are inclined towards using scripting methods with Python, R etc but they do not provide the visibility that OpenRefine does. We are long way form one perfect solution to Data Cleaning; but with advancement in tools; a lot of tedious task burden can be transferred to machines.

8 Alternate Approach

With help and inspiration from Trevor Muñoz [9] at Curating Menus [10]; we also tested an approach to cleaning Menus Data with popular Python library pandas [11] ; the gains were visible, in terms of speed and flexibility but it lacked the ability to involve a human being as a decision maker since most of the stuff gets automated. The notebook details are available in appendix C; we run the notebook on same environment - outlined in appendix A.

9 Next Steps - Visualizing Menus Data

Data Visualization is one of the most effective techniques for gaining insights from Data; and we were fortunate enough to be taking CS498 Data Visualization class this summer. We used both D3 [12] and Tableau [13] for creating visualizations using this dataset. Links for both are present in appendix C.

Acknowledgments

The authors would like to thank Prof. Bertram Ludaescher, for this opportunity. Being a renowned expert in the field; his guidance has been invaluable. We would also like to our TA for the class Niteesh Kanungo; he has been always available for all our queries and discussions. And, a special thanks to all our fellow MCD-DS students, their community at Slack has proved to be an exceptional support structure.

REFERENCES

[1] "Amazon mechanical turk," https://www.mturk.com/mturk/welcome.
[2] "Crowdflower," https://www.crowdflower.com/.
[3] "Openrefine: A free, open source, powerful tool for working with messy data," http://openrefine.org/.
[4] B. L. et al, "Yesworkflow," https://github.com/yesworkflow-org/.
[5] "Sqlite," https://www.sqlite.org/.
[6] B. Ludaescher, "Cs598 - special topics," https://cs.illinois.edu/courses/profile/CS598, Summer 2017.
[7] N. Y. P. Library, "What's on the menu?" http://menus.nypl.org/, 2011.
[8] "Frank e. buttolph," https://en.wikipedia.org/wiki/Frank_E._Buttolph.
[9] T. Muñoz, "Borrow a cup of sugar? or your data analysis tools? more work with nypl's open data, part three," https://goo.gl/qozyDt.
[10] T. M. et. al., "Borrow a cup of sugar? or your data analysis tools? more work with nypl's open data, part three," http://curatingmenus.org/.
[11] "Python data analysis library," http://pandas.pydata.org/.
[12] M. B. et. al., "D3: Data-driven documents," http://d3js.org/.
[13] "Tableau: Make your data make an impact," https://www.tableau.com/.

Alok K. Shukla
After graduating from National Institute of Technology, Allahabad IN in 2009, I have been working with an Enterprise Software Products firm based out of New Delhi, IN. Some of the interesting use-cases I've worked with involved extensive use of Text Analytics, Process Mining, Queueing Theory and Natural Language Processing. I plan to graduate from UIUC in Summer 2018, with a specialized masters in Data Science.

Gitika Jain
I am a Lead Java Developer and Data Science Enthusiast having strong background in Object oriented architecture, analysis, design and software development using latest technologies like Java, J2EE , Scala, Akka, Restful Services, Big Data and Elastic Search technologies. I plan to graduate from UIUC in Spring 2018, with a specialized masters in Data Science.

Apurva Hari
I am currently working as a Data Analyst at Silicon Valley Bank, California.I mainly analyze large volumes of data to uncover inconsistencies and identify interesting patterns.I plan to graduate from UIUC in Fall 2018, with a masters degree in Computer Science.

APPENDIX A
THE WORK-FLOW ENVIRONMENT

A.1 Hardware Specifications

AWS EC2 Instance Specifications			
Type	**vCPUs**	**Memory**	**Storage**
t2.large	2	8 GiB	EBS

A.2 Software Specifications

OS : *Windows Server 2016 Datacenter*

Software	Version	Link
OpenRefine	2.7	https://goo.gl/ifyD28
SQLite	3.20.0	https://goo.gl/VHVoQE
SQLite Studio	3.1.1	https://goo.gl/qyAqri
YesWorkflow	Demo App	try.yesworkflow.org
Anaconda - Python 3.6	4.4.0 64-bit	https://goo.gl/UT7tKe

APPENDIX B
FILE DESCRIPTIONS

B.0.1 ***Dish.csv***

Data on individual dishes from the menu collection.
Size: 26.5 MB
Count: 423397
Cross-table reference notes: The field *id* in *Dish.csv* is referred to by the field *dish_id* in *MenuItem.csv*

1) ***id*** : a unique identifier for this dish
2) ***name*** : the name of this dish
3) ***description*** : a description of this dish; always blank as of 17 Jul 2017 release
4) ***menus_appeared*** : the number of menus this dish appears on
5) ***times_appeared*** : the number of times this dish appears (seems to be in case the dish appears multiple times or in multiple sections on a given menu)
6) ***first_appeared*** : the year this dish first appeared; sometimes 0 or 1, which should probably be treated as errors or NA
7) ***last_appeared*** : the year this dish last appeared; sometimes 2928, 0, or 1, which should probably be treated as errors or NA
8) ***lowest_price*** : the lowest price that this item was sold for
9) ***highest_price*** : the highest price that this item was sold for

B.0.2 ***Menu.csv***

Data on individual menus in the NYPL collection.
Size: 3.2 MB
Count: 17545
Cross-table reference notes: The field *id* in *Menu.csv* is referred to by the field *menu_id* in *MenuPage.csv*

1) ***id*** : a unique identifier for this menu
2) ***name*** : the name of this menu (often blank, or the name of the restaurant)
3) ***sponsor*** : the sponsor of this menu (often the name of the restaurant)
4) ***event*** : the event this menu was created for (typically the name of the meal, or the special event that this menu was created for)
5) ***venue*** : the venue where food from this menu was served
6) ***place*** : geographical description of the menu's location; often includes city/state/country, street address, or name of venue
7) ***physical_description*** : a description of the menu's physical characteristics, including paper stock, dimensions, colors, design, etc.
8) ***occasion*** : a special occasion, holiday, daily, or blank
9) ***notes*** : any special notes about this menu
10) ***call_number*** : the menu's call number within the NYPL collection
11) ***keywords*** : keywords for the menu; always blank as of 17 Jul 2017 release

12) ***language*** : language the menu is printed in; always blank as of 17 Jul 2017 release
13) ***date*** : date the menu was collected, as a string with the format YYYY-MM-DD
14) ***location*** : location where the menu was used
15) ***location_type*** :type of the location value; always blank as of 17 Jul 2017 release
16) ***currency*** : currency (possibly obsolete) charged for items on this menu
17) ***currency_symbol*** : symbol for the currency
18) ***status*** : the digitization/transcription status of this menu; one of ”complete” or ”under review”
19) ***page_count*** : number of pages in this menu
20) ***dish_count*** : number of dishes on this menu

B.0.3 ***MenuItem.csv***

Data on individual menu items from the NYPL menus.
Size: 118.5 MB
Count: 1332726
Cross-table reference notes: The field *menu_page_id* in *MenuItem.csv* is referred to by the field *id* in *MenuPage.csv*
The field *dish_id* in *MenuItem.csv* is referred to by the field *id* in *Dish.csv*

1) ***id*** : a unique identifier for this menu item
2) ***menu_page_id*** : the id of the menu page that this item appears (see Cross-table reference notes)
3) ***price*** : the price of (the smallest portion of) this item
4) ***high_price*** : the price of the largest portion of this item
5) ***dish_id*** : the id of the dish that this menu item refers to (see Cross-table reference notes)
6) ***created_at*** :the date and time that this database entry was created
7) ***updated_at*** : the most recent date that this database entry was updated
8) ***xpos*** : the x-axis position of the item on the scanned image
9) ***ypos*** : the y-axis position of the item on the scanned image

B.0.4 ***MenuPage.csv***

Data on individual pages from the NYPL menus.
Size: 4.7 MB
Count: 66937
Cross-table reference notes: The field *id* in *MenuPage.csv* is referred to by the field *menu_page_id* in *MenuItem.csv*
The field *menu_id* in *MenuItem.csv* is referred to by the field *id* in *Menu.csv*

1) ***id*** : a unique identifier for this menu page
2) ***menu_id*** : the id of the menu this page belongs to (see Cross-table reference notes)
3) ***price*** : the page number in the menu
4) ***high_price*** : a unique identifier for the scanned image of this menu page, accessible on the NYPL site
5) ***dish_id*** : the height of the menu
6) ***created_at*** :the width of the menu
7) ***updated_at*** : a(nother) unique identifier for this page/image

APPENDIX C
LINKS TO SCRIPTS/ DELIVERABLES

Note: If any of these links are broken or you are using a printed copy please contact alokks2@illinois.edu.

C.1 Original Dataset

Time-stamp: 17th July 2017, 4 P.M. Central Time.
New York Public Library’s crowd-sourced historical menus - NYPL_Dataset.zip

C.2 Complete Project Deliverables

Please use this link to download all of our deliverables including the data and this report itself.

C.2.1 ***OpenRefine***

OpenRefine operations history for all data files and accompanying screen-shots are available at this link

C.2.2 ***SQL***

Data Definition and Manipulation - all scripts are available at this link.

C.2.3 ***YesWorkflow***

The annotations script and data flow operations graphs are available here.

C.3 The Alternative approach - Pandas Notebook

The IPython notebook is hosted at same AWS instance; contact alokks2@illinois.edu if you want to run our version; otherwise you can fork it from Github.

C.4 Visualizations

C.5 Using D3.js

Please visit What IS on the Menu! for D3 visualizations.

C.6 Using Tableau

See Tableau Dashboards with cleaned data and with uncleaned.