Phase 1 Change Report

Points received for initial Phase-1 submission : 88

Our team chooses the following Phase-1 option: B

Phase 1 Remarks (TA Feedback)	Student Comments / Summary
Identity and describe dataset: [-5] No narrative description of the dataset.;	Added narrative description of the dataset to describe the context of the dataset
Use cases: [-5] Your answer does not satisfy definition of U0.;	Updated U0 to be more properly fitting of the definition
Data quality list: [-2] Mentioned use case but no discussion on the connection between use cases and data quality issues.	Added remarks for connections between use cases and data quality issues

Theory and Practice of Data Cleaning Project Phase 1

Mike Zhou: mikez2@illinois.edu
Sung Yoo Kim: sungyoo2@illinois.edu

Identify a Dataset

The New York Public Library's Menu dataset is an effort to digitize and categorize the various dishes and menus that have appeared in the collection of menus with the New York Public Library. The data set is important in its ability to provide historical context to when a dish was first introduced into New York City and when dishes fell out of style.

We are using the NYPL Menus dataset and specifically the dish.csv file which is a dataset of various dishes appearing in the New York Public Library's collection of menus. Each entry is a dish with the number of menus the dish appeared on, number of times the dish appeared in general, first and last appearance years of the dish, and highest and lowest prices for the dish.

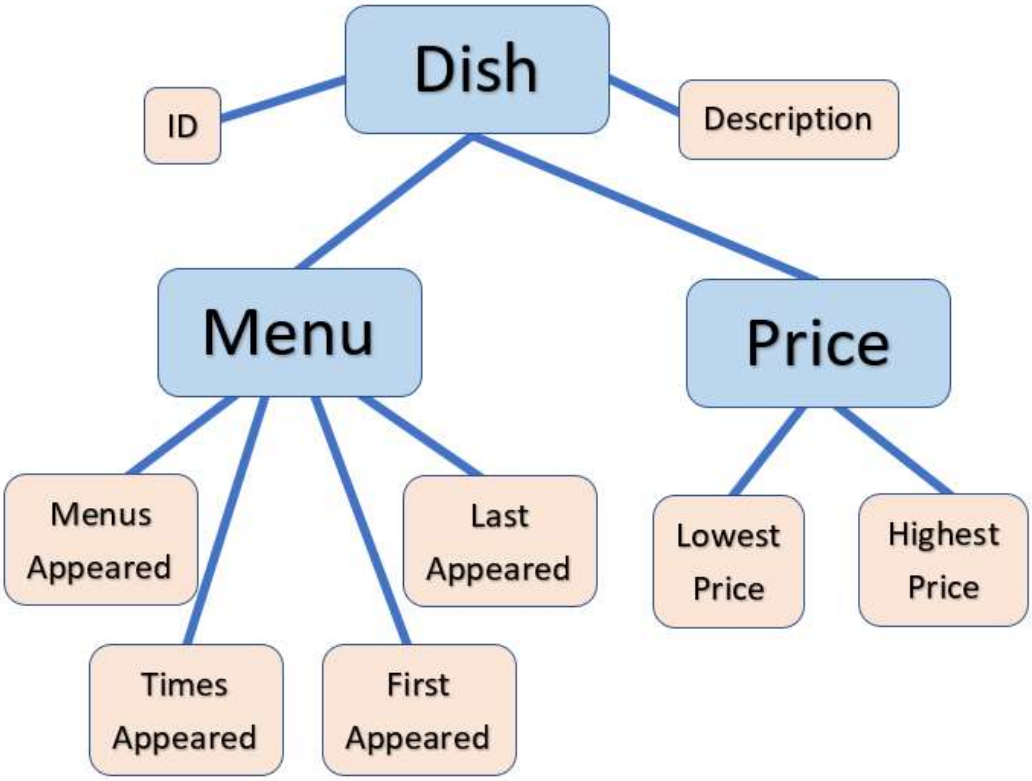
Develop a target use case

An example of a U0 use case is measuring the maximum value for the highest price column, which would determine the highest cost of any single dish in New York City. The reason that this is a U0 case is because data cleaning on the highest price column is not necessary to get the maximum value, and doing any data cleaning would not change the maximum value either.

The U1 use case would be to clean up the first and last appearance years to count the number of unique dishes present during each year, measuring how many new dishes appeared vs how many old dishes died out. This requires the year columns to be cleaned up especially for a couple of dishes considered commonplace and don't have the correct first or last appearance year set.

An example of a U2 use case is aggregating the dish names to determine dish/ingredient popularity so that similar dishes are bucketed together. e.g. Baked Oysters and Oysters Rockerfellers, as well as translating names that are similar such as pineapple and "ananas". This would be extremely hard with standard data cleaning as it is a semantics issue versus a syntax or constraint violation.

Describe the dataset



This is the NYPL Labs project file for the dishes.csv. This looks through a certain dish name and gives it a certain id and description. Then it dives into the menus the dish has appeared, times appeared, first appeared, and last appeared. In addition, it also lists the lowest and highest price for the dish based on the price listed in the menu.

This dataset is a collection from all the menus that the NYPL has been able to collect. This dataset is updated twice a month and is continually updated. The table is created with an id number for each name of a dish then continues on with the description, menu descriptions, and then the price description for the columns. I believe that the spatial extent of the data is from around the world since it is the collection of menus, but most seem to be from New York. The temporal extent of the data is from the 1840s to the present year and the data includes the dates of when the dish first appeared and last appeared. Before cleanup, there are a total of 428086 dish entries that we need to get fixed.

Obvious Data Quality Problems

There are obvious data quality problems that are easy to spot. For example, the description column is not needed. It does not contain any information about any of the dish names. It can be omitted. In addition, they include misspelled names which can lead to duplicates in the data. This is important for our U1 use case as having very closely related dishes appear more than once will change the result of the search.

Also null cases which have a 1 in the first appeared years column and a 0 or missing value in both the lowest and highest price columns. There are also dates that have not occurred like 2928 which can be determined as a typo. The first and last year appeared columns are important to our U1 use case as we are trying to measure how many unique dishes are present in any given year. Dirty data in this column would change the number of unique dishes that are present.

We can clean this up to the best of our ability with OpenRefine to help us tackle the problem of U1. With 428086, there are a lot of errors that can be omitted and fixed before we tackle our problem.

Devise an Initial Plan

1. The dataset is the New York Public Library Menus dish dataset that lists the dishes found in NYPL menus, their frequency in appearance, first and last appearance and highest and lowest prices. The U1 case we will try to clean for is to count the number of unique dishes per year.
2. The profiling of D has the quality problems of fixing the years which include false or null values and fixing certain dish names. With the years and the names of the dishes cleaned up we should be able to address the problem of U1 which is to determine the number of unique dishes for a given year.
3. We can address these changes with regular expressions in OpenRefine or with OpenRefine features. There are other options like loading the file into a Python file or using SQLite to create a new table based on the entries. However, OpenRefine should be sufficient in being able to clean the data before we have to check for any integrity constraints.
4. We should be able to determine the changes we make and how much the dataset has improved by looking at the OpenRefine changes. OpenRefine is helpful in tracking the number of changes which will help determine the difference between our datasets. We will use SQlite or Datalog to determine the integrity constraints violations between the two datasets.

5. We will use open refine’s recipe file to be able to document the changes being made to the dataset D. In addition any integrity constraint checks we do can be saved as SQL or Datalog format.

Tentative Assignment of Tasks

1. Sung Yoo Kim - OpenRefine cleaning of data, SQL/Datalog(Queries), and documentation
2. Mike Zhou - SQL/Datalog(Queries), workflow model, and documentation

Theory and Practice of Data Cleaning Project Phase 2

Mike Zhou: mikez2@illinois.edu
Sung Yoo Kim: sungyoo2@illinois.edu

Data Cleaning Performed

The data cleaning steps for OpenRefine on the menu.csv file included:

1. Trim leading & trailing whitespaces from strings
2. Remove the description column
3. Remove consecutive whitespaces in the columns
4. Change the columns of menus_appeared, times_appeared, first_appeared, last_appeared, lowest price, and highest price to number values
5. Remove rows with first_appeared as 0 and 1
6. Remove rows with last_appeared as 0 and 2928
7. Change values in times_appeared that were -1,-2,-3, & -6 to positive values
8. Change values in menus_appeared that were 0 to 1
9. Combine rows with same name of dishes using key-collision fingerprint and n-gram fingerprint (n-gram size of 2)

Take a look at the inner workflow in the workflow section to get a better idea of the OpenRefine steps.

Rationale

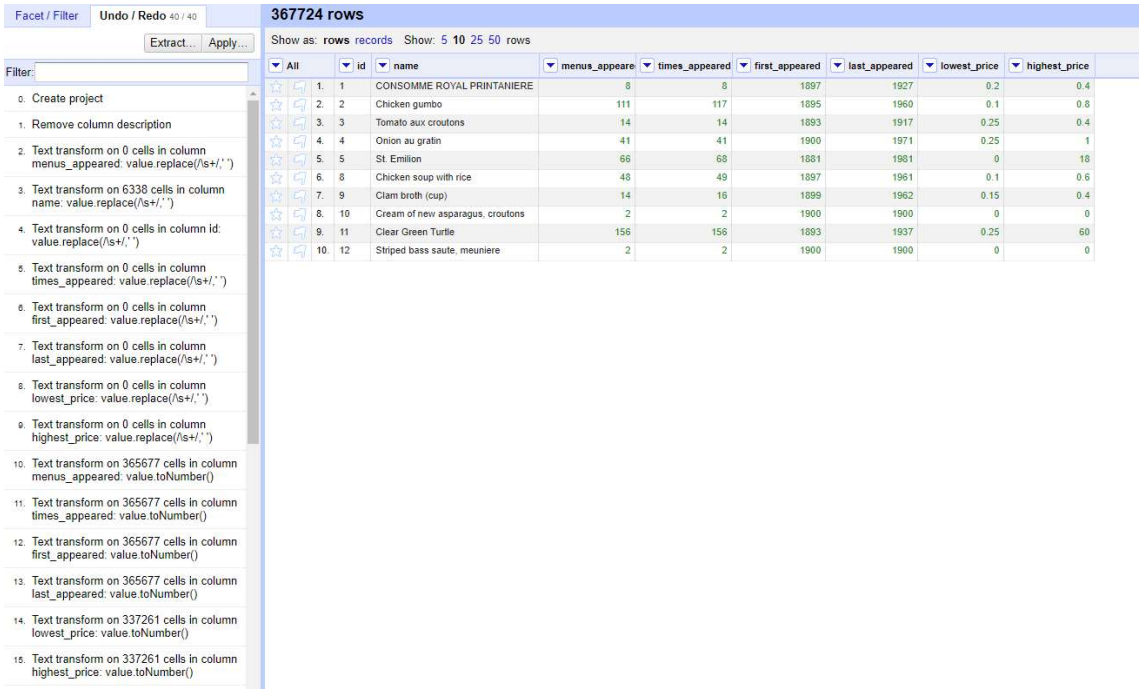
The trim of leading and trailing whitespaces and the consecutive whitespaces were one of the first steps in cleaning the dataset. This helped support our U1 case since this can help when comparing strings due to uncertain whitespaces in the data. We ensured this was not the case by excluding all the unnecessary whitespaces.

Removing the description column did not help our U1 case but the entire column had no data. It was unnecessary and did not contribute anything to our data since the column was empty.

The removing and changing of the values in first_appeared, last_appeared, times_appeared, and menus_appeared was crucial since this is the data we were going to be looking at for our U1 case of determining the number of dishes for a given year. This helped us filter out using the openRefine facets and filters to make sure that values that did not make sense were included.

Combining the rows with the same name of dishes using the key-collision fingerprint and n-gram fingerprint was helpful since it was helping our U1 case with certain duplicates that could exist in our data. However, as stated in our U2 case which is that there was too much data cleaning to be done, the 360,000 different names were not all able to fit into the

openRefine facet to be combined together. So in order to combine certain rows, we had to facet and filter to a max of 10,000 dish names and work with that. This was repeated using different facets and filters to try and get as many same dish names together. The other detection methods like metaphone 3 and cologne-phonetic were adding dish names that were different so those methods were not included in our OpenRefine cleaning process.



Facet / Filter | Undo / Redo 40 / 40

Extract... Apply...

Filter:

0. Create project
1. Remove column description
2. Text transform on 0 cells in column menus_appeared: value.replace(/\s+/,' ')
3. Text transform on 6338 cells in column name: value.replace(/\s+/,' ')
4. Text transform on 0 cells in column id: value.replace(/\s+/,' ')
5. Text transform on 0 cells in column times_appeared: value.replace(/\s+/,' ')
6. Text transform on 0 cells in column first_appeared: value.replace(/\s+/,' ')
7. Text transform on 0 cells in column last_appeared: value.replace(/\s+/,' ')
8. Text transform on 0 cells in column lowest_price: value.replace(/\s+/,' ')
9. Text transform on 0 cells in column highest_price: value.replace(/\s+/,' ')
10. Text transform on 365677 cells in column menus_appeared: value.toNumber()
11. Text transform on 365677 cells in column times_appeared: value.toNumber()
12. Text transform on 365677 cells in column first_appeared: value.toNumber()
13. Text transform on 365677 cells in column last_appeared: value.toNumber()
14. Text transform on 337261 cells in column lowest_price: value.toNumber()
15. Text transform on 337261 cells in column highest_price: value.toNumber()

367724 rows

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

All		id	name	menus_appeare	times_appeared	first_appeared	last_appeared	lowest_price	highest_price
1.	1	CONSOMME ROYAL PRINTANIERE	8	8	1897	1927	0.2	0.4
2.	2	Chicken gumbo	111	117	1895	1960	0.1	0.8
3.	3	Tomato aux croutons	14	14	1893	1917	0.25	0.4
4.	4	Onion au gratin	41	41	1900	1971	0.25	1
5.	5	St. Emilion	66	68	1881	1981	0	18
6.	8	Chicken soup with rice	48	49	1897	1961	0.1	0.6
7.	9	Clam broth (cup)	14	16	1899	1962	0.15	0.4
8.	10	Cream of new asparagus, croutons	2	2	1900	1900	0	0
9.	11	Clear Green Turtle	156	156	1893	1937	0.25	60
10.	12	Striped bass saute, meuniere	2	2	1900	1900	0	0

16. Remove 60181 rows
17. Remove 170 rows
18. Remove 11 rows
19. Text transform on 2217 cells in column menus_appeared: value.toNumber()
20. Text transform on 2217 cells in column times_appeared: value.toNumber()
21. Text transform on 2217 cells in column first_appeared: value.toNumber()
22. Text transform on 2217 cells in column last_appeared: value.toNumber()
23. Text transform on 2009 cells in column lowest_price: value.toNumber()
24. Text transform on 2009 cells in column highest_price: value.toNumber()
25. Mass edit 545 cells in column name
26. Mass edit 90 cells in column name
27. Mass edit 814 cells in column name
28. Mass edit 139 cells in column name
29. Mass edit 655 cells in column name
30. Mass edit 100 cells in column name
31. Mass edit 2393 cells in column name
32. Mass edit 174 cells in column name
33. Mass edit 590 cells in column name
34. Mass edit 101 cells in column name
35. Mass edit 138 cells in column name
36. Mass edit 45 cells in column name
37. Mass edit 419 cells in column name
38. Mass edit 98 cells in column name
39. Mass edit 132 cells in column name
40. **Mass edit 29 cells in column name**

Document Data Quality Changes

The efforts made in data cleaning include:

Changes	Description	# of Changes
Columns Deleted	Description column was deleted	1 column
Rows Deleted	Remove 0 & 1 in first_appeared and remove 0 & 2928 in last_appeared	60362 rows
Rows Changed/Combined	Changed times_appeared to positive for negative values, changed 0 to 1 in menus_appeared, and combining same name values	About 6500 rows changed

To check the quality of the data changes, the original and cleaned CSV files were imported into SQL using sqlite and come queries were run to count the number of ICVs and to see where the data was improved.

This is the schema used to import the clean dataset into sql.

```sql
CREATE TABLE dish(
  "id" TEXT,
  "name" TEXT,
  "menus_appeared" INTEGER,
  "times_appeared" INTEGER,
  "first_appeared" INTEGER,
  "last_appeared" INTEGER,
  "lowest_price" REAL,
  "highest_price" REAL
);
```

The following is a table of the queries ran against the dish table before and after cleaning.

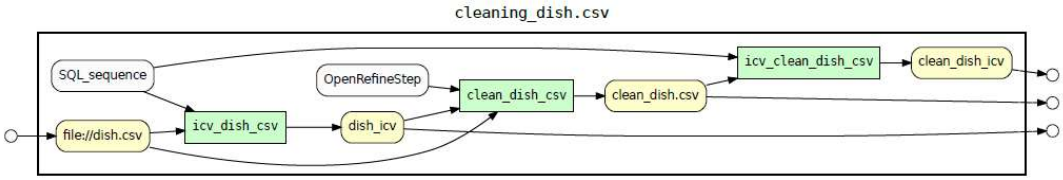
Query	Description	Original ICV Count	Cleaned ICV Count
`select * from dish where first_appeared < 1624;`	Finds the dishes that have a first_appeared before the founding of NYC (1624)	60177	0
`select * from dish`	Finds the dishes that had a last appeared after the current year	179	0

`where last_appeared > 2021;`	(2021)		
`select * from dish where first_appeared > last_appeared;`	Finds the dishes that first appeared after they last appeared.	6	0

Create a Workflow Model

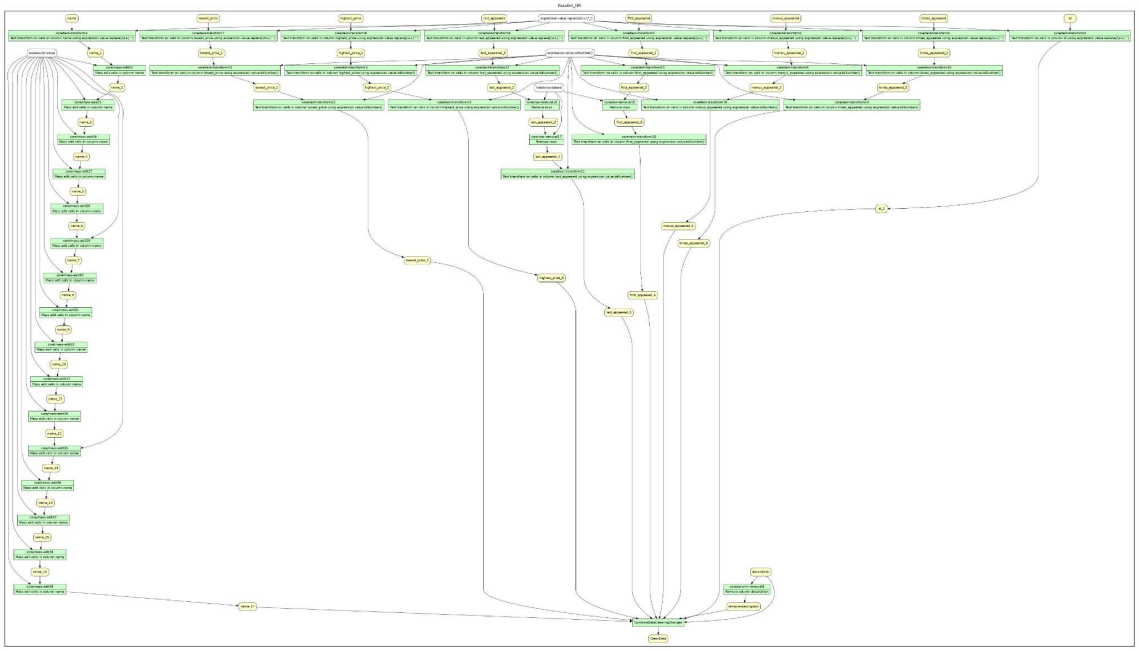
Overall WorkFlow (Outer Workflow)

Using YesWorkflow, the overall workflow was written as an annotation file and the resulting file was passed through GraphVis to get a workflow diagram of the overall process.



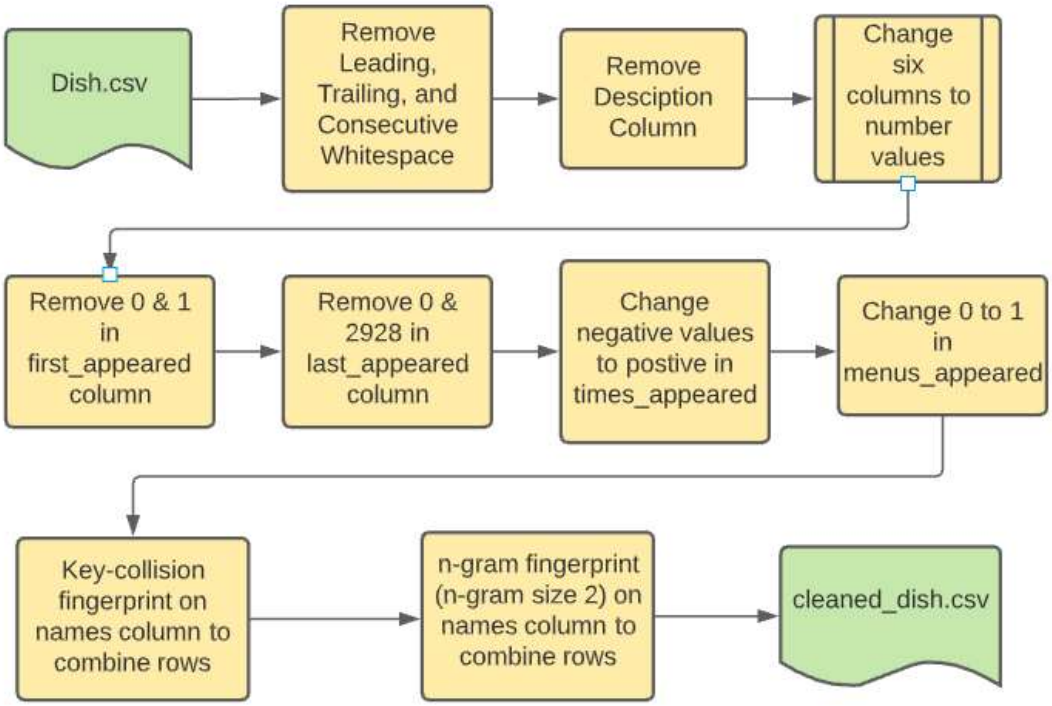
Cleaning WorkFlow (Inner Workflow using YesWorkflow)

The OpenRefine steps json was then passed through the Openrefine to Yesworkflow model tool which created the inner workflow visualization. As the image is quite large, a copy of the image PNG alongside the GraphViz files are included with the submitted package.



Inner Workflow (Simplified)

Here is a simplified version of the inner workflow made without YesWorkflow based on the cleaning steps described before.



Conclusions & Summary

In conclusion the cleaning of dish.csv, we learned that there were a lot of data issues that were readily apparent from ICVs. OpenRefine proved to be an excellent tool in its ability to clean the large dataset efficiently and combine similar items which definitely helped our U1 use case described. Creating the outer YesWorkflow diagram was easy with the annotation tooling and the inner YesWorkflow diagram is extremely complicated, which shows the importance of documenting provenance when each cleaning step is a single click.

Sung Yoo Kim contributions included working on the openRefine cleaning of the data and documenting the changes. It also included the simplified diagram of the inner workflow. Also worked on the report.

Mike Zhou contributions were the ICV checks on the raw and clean datasets and the workflow models using YesWorkflow and GraphVis. Also worked on the report and final documentation.