CS_513
Summer 2022
Final Project Phase I

Paul Bromen (pbromen2) | Tony Petrotte (adp12) | Isaac Aruldas (aruldas2)

1.) THE DATASET

D = Food_Inspections.csv

For our project we will be using the Chicago food inspections data provided by the professor. We selected it because it's interesting, (who doesn't want to know what restaurants might get you sick?), and it provided cleaning opportunities for timeline, spatial and open field text data.

2.) USE CASES

Main Use Case (U_1)

Our U1 main use case will be for a fictional app called Helph that combines yelp like crowdsourcing review, and food inspection information. It will allow users to look up health code violations for nearby restaurants. The "Food Inspections.CSV" dataset will be the primary source for violation information. Right now, this data is in an open text field and will need cleaning to be usable.

Minor Use Case 1 (U_0)

U0 minor use case of this data that would not require cleaning is the spatial location of restaurants. As the latitudinal and longitudinal columns look clean and immediately usable. This data would be used to connect health inspections to specific restaurant locations.

Minor Use Case 2 (U_2)

U2 minor use case for which this data would not be usable for even at its most sanitized would-be knowledge as to which of these restaurants would be delicious or popular. This can't be captured from this data and could only be inferred by users through crowd or expert sourced reviews, which we would obtain from other data sources and marry to the Food Inspection information we are cleaning for this project.

3.) DESCRIBE THE DATASET

Dataset column names are as follows:

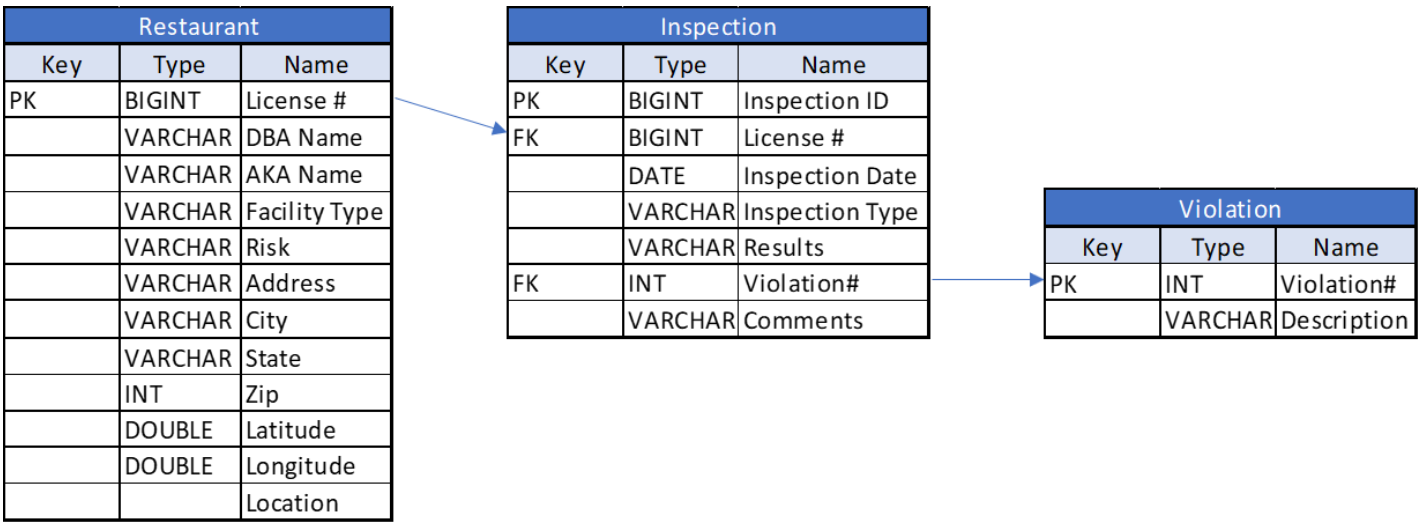
Inspection ID	DBA Name	AKA Name	License #	Facility Type
Risk	Address	City	State	Zip
Inspection Date	Inspection Type	Results	Violations	Latitude	Longitude	Location

Potential Data Structures:

Structure 1: One potential way of structuring the data for our use case would be to simply create a singular table that contains only the data we need for our use case. An example of such a structure would look like the ERD to the right:

Food Inspections	
Key	Inspection ID
Key	DBA Name
Key	AKA Name
Key	Licensee #
Key	Facility Type
Key	Risk
Key	Address
Key	City
Key	State
Key	Zip
Key	Inspection Date
Key	Inspection Type
Key	Results
Key	Violations
Key	Latitude
Key	Longitude
Key	Location

Structure 2: Another potential way of structuring the data for our use case would be a more Database centric approach. This structuring would separate the data into tables in a way that we could easily use them in a standard database and take advantage of the organization of a database as well as the power SQL queries. This would also help us keep an organization schema for the issues encountered in the Violations (covered further in the section 4). An example of such a structure would look like the ERD below:



Data Narrative:

The data we are cleaning is compiled from Chicago Food Inspections of restaurants, grocery stores, schools and other facilities that are regulated by municipal health officials. These are primarily located in the city of Chicago, although there are some facilities outside this jurisdiction. These inspections are carried out by the Chicago Department of Health, which is responsible for ensuring that food sold in the city is safe.

The most important column is “Violations” which is open text and lists infractions cited by inspectors during their visit.

Other columns included numerical identifiers like “Inspection ID” which refers to the individual inspection and “Licensee #” which refers to inspected organizations' food establishment license on file with the city. “DBA Name” and “AKA Name” are the organizations name. These generally line up, but occasionally will have differences. Typically, the “DBA Name” refers to the legal name and the “AKA Name” is customer facing, i.e., what would be on the sign out front.

“Facility Type” refers to the kind of organization that is being inspected. The most common types are restaurants, grocery stores, and schools.

The “Risk” column refers to how often establishments are inspected for example risk 1 establishments are inspected twice a year, while risk 3 establishments are inspected every other year.

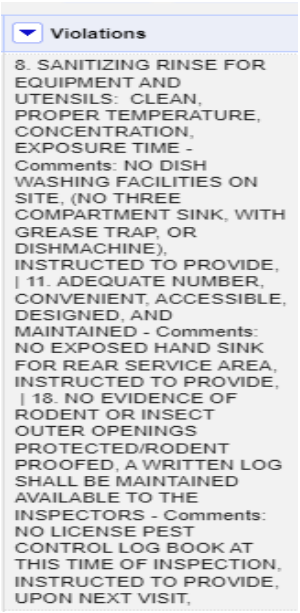
“Address”, “City”, “State”, “Zip” are all columns used for ascertaining the street address of the establishment. “Inspection Data” refers to the time of the inspection, placing the inspections on a temporal timeline. Newer inspections should supersede older inspections as an integrity constraint for example.

“Inspection Type” is the reason the inspection was undertaken. Inspections completed as a result of complaints will be especially notable for our app.

“Results” represent 7 outcomes that an inspection can have. There are several categories in this column that indicate an inspection didn’t take place.

“Latitude”, “Longitude” and “Location” place the establishment in the geographic coordinate system.

4.) LIST OBVIOUS DATA QUALITY PROBLEMS

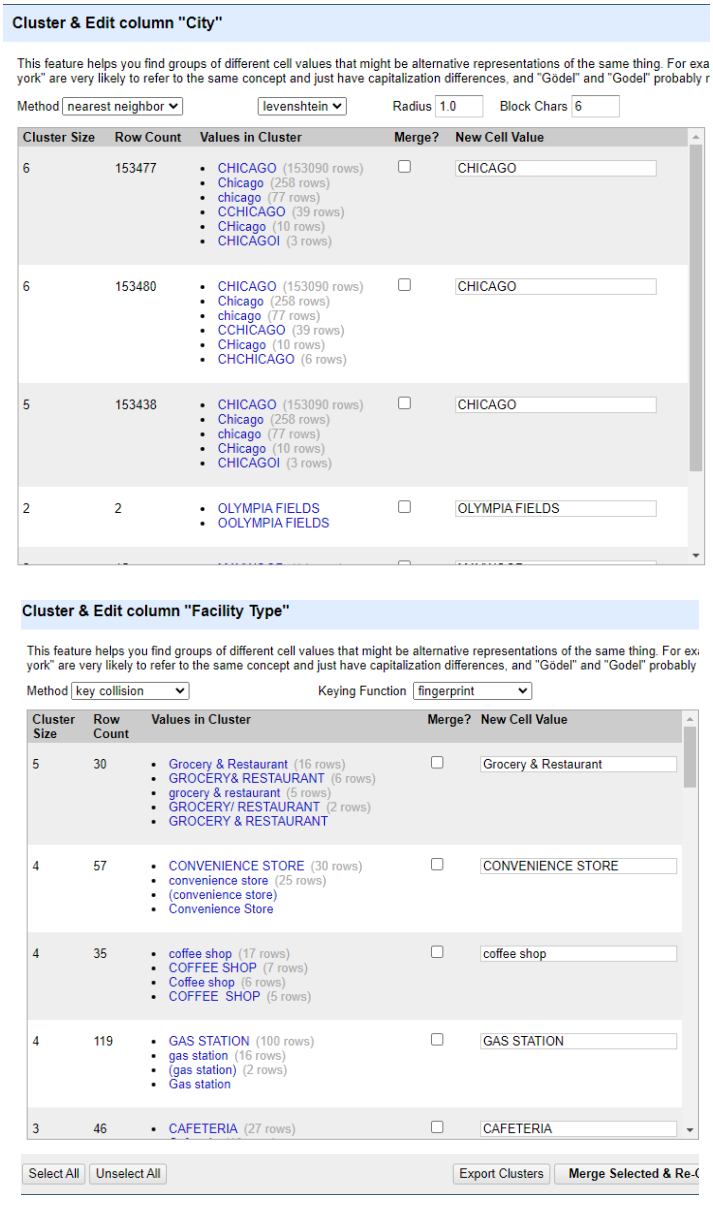


<u>The most obvious and important data quality issue</u> for our main use case is the “Violations” is an open text field and that we will need to clean and separate each value in order to show this to our users.

The violations have general structure being pipe “|” delimited, but having the addition of comments poses an extra challenge to cleaning and parsing the data into either of our purposed data structures. This is the most important element to clean as the information holds the most value to our use case.

An example of the open text held within the Violations(right) can be seen to have a structure of being pipe “|” delimited between consistently numbered violation types with descriptions. This is then usually followed by a comment by from the inspector, which could potentially also be delimited by the dash “-“ character.

Another obvious and expected quality issue with the data has to do with differences in the spelling between various entries. This is a common quality error and as can be seen in the screen shot from OpenRefine (right-above), there are a plethora of spellings for the city of Chicago although they are obviously meant to be the same.



Another example, shown in an OpenRefine screenshot (right-below) shows the Facility Type cluster containing many instances of this quality issue with multiple spellings detected for simple labels such as “Gas Station” and “Coffee Shop”.

Further cases of this quality issue can be found in almost every column that contains a form of label, and will thus need standardized to a common form during our cleaning.

Further examples of obvious issues that are less extensive are things such as:

- Removing dashes from “Location”
- Ensuring unique values in columns potentially to be use as DB keys such as
  - “License #”
  - “Inspection ID”
- Standardizing the date format in “Inspection Date”

5.) DEVISE AN INITIAL PLAN

S1: Description of dataset D and use case U1

Dataset D is a collection of 153811 entries from the Chicago Department of Health that contains fields describing Chicago businesses, their risk level with regard to health codes, and information on their health code inspections. Our U1 main use case will be for a fictional app called Helph that combines yelp like crowdsourcing review, and food inspection information. It will allow users to look up health code violations for nearby restaurants.

S2: Profiling of D to identify quality problems P

First, we will prune the data so that entries that are not relevant to our main use case our removed. This includes making sure that all inspections are for restaurants, are in Chicago and have codes that indicate that an inspection took place.

Second, we will address general issues in the data, including merging misspellings, dealing with null entries, and removing punctuation or white space that would inhabit data manipulation.

Third we will address problems in the open text field “Violations”, which contain the core of our use case. We will need to split out each individual violation, while finding a way to capture the inspector’s comments, which provide rich color.

S3: Describe what tools you plan to use to address problems P

OpenRefine will be used for the general cleaning and standardization of the data columns. This will include standardization of labeled data columns such as City, Facility Type, Results etc. OpenRefine will also be used for standardizing datatypes in columns such as “Inspection Date” for consistent date formatting.

Python with Pandas will then be used to remove unneeded data items as an initial cleaning step. This will include data items that are not needed for our proposed use case such as non-restaurant entries, restaurant entries that are indicated to be closed, or entries that are indicated to not be in the Chicago land area.

In addition, we plan to use Python to handle the issues presented in section 4 with regard to parsing and standardizing the “Violations” column. This will include delimiting the data as well as potentially creating new columns to distribute the Violation numbers and their descriptions in an indexable manner. Further, delimiting the inspection comments could provide additional value to our proposed use case.

If time allows, we also would like to implement a YesWorkflow schema in order to create a more robust and trackable relationship diagram for the different alterations and actions done on the data.

S4: Check that dataset D Prime is an improved version of D

We will know that we have cleaned and improved the dataset when each violation from an inspection is separated in a way that will be readable both for humans and machines. Additionally, we will perform randomized QA inspections on rows to make sure they do not

contain errors that would harm our proposed use case. These inspections will also make comparisons to the original dataset so we can see the improvement and can be certain that we haven't lost important information during the cleaning process.

S5: Document the types and amount of changes executed on D to get D Prime

Our current plan is to use a combination of the providence tracking capabilities in OpenRefine and potentially or2yw or YesWorkflow to create a scientific workflow visualization of the steps we undertook to clean the data. This will allow us to track both the alterations made during the initial cleaning utilizing OpenRefine as well as alterations and extensions made utilizing Python scripting. We will also use the contents of this document as a general outline for documenting restructuring.

Assignment of Tasks

S1 - Data Description:

Was completed by Tony Petrotte and Paul Bromen

S2 - Problem Identification:

Will be completed by Tony Petrotte and Paul Bromen

S3 - Data Cleaning:

- Prune Data in Open Refine
  - Paul Bromen
- Clean Data in OpenRefine
  - Paul Bromen
- Clean and Parse data with Python
  - Tony Petrotte

S4 - Quality Assurance:

Whole Team

S5 - Documentation:

Whole Team