CS 513 Theory & Practice of Data Cleaning

Final Project Report: What's On The Menu?

Artsiom Strok  
NetID: astrok2   
[astrok2@illinois.edu](mailto:astrok2@illinois.edu)  
Ramin Melikov  
NetID: melikov2  
[melikov2@illinois.edu](mailto:melikov2@illinois.edu)Jonah Willis  
NetID: jonahww2  
[jonahww2@illinois.edu](mailto:jonahww2@illinois.edu)

*Abstract*—This electronic document is a “live” template and already defines the components of your paper [title, text, heads, etc.] in its style sheet. *\*CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract*.

Keywords—NYPL, menus, data cleansing, OpenRefine, YesWorkflow, provenance.

# Overview and initial assessment of the dataset

The New York Public Library is digitizing and transcribing its collection of historical menus. The collection includes about 45,000 menus from the 1840s to the present, and the goal of the digitization project is to transcribe each page of each menu, creating an enormous database of dishes, prices, locations, and so on. As of early July, 2020, the transcribed database contains 1,334,431 dishes from 17,545 menus.

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 their individual file descriptions below.

## 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.

* id – The unique identifier of the menu
* name – The name of the restaurant
* sponsor – Who sponsored the meal (organizations, people, name of restaurant)
* event – The category (e.g. lunch, annual dinner)
* venue – The type of place (e.g. commercial, social, professional)
* place – Where the meal took place (often a geographic location)
* physical\_description – The dimension and material description of the menu
* occasion – The occasion of the meal (holidays, anniversaries, daily)
* notes – The notes by librarians about the original material
* call\_number – The call number of the menu
* keywords – The keywords of the menu
* language – The language of the menu
* date – The date of the menu
* location – The organization or business who produced the menu
* location\_type – The type of the location
* currency – The system of money the menu uses (dollars, etc.)
* currency\_symbol – The symbol for the currency ($, etc.)
* status – The completeness of the menu transcription (transcribed, under review, etc.)
* page\_count – How many pages the menu has
* dish\_count – How many dishes the menu has

Each menu is associated with some number of MenuPage values.

The first inspection of the data shows us that this file has 17545 entries and 20 columns. Three columns ‘*keywords*’, ‘*language’*, ‘*location*\_type’ do not have any values and can be deleted. The ‘id’ column has all unique numeric values. Thus we can assume no issues with this column. The ‘*name*’ column has only 3197 non-empty values. There also placeholders for missing value, e.g., *‘[Restaurant name and/or location not given]*’ or *‘[Not given]*’. There are a lot of names that exactly the same but due to extra spaces, punctuations, different order of words, typos, diacritical mars they don’t match exactly. The ‘*sponsor*’ column has 15984 non-empty values, and these values have similar issues as ‘*name*’ column. Also, some of the values are just question marks. The ‘*event*’ column has 8154 non-empty values. The values for this column can be grouped into different buckets such as ‘*breakfast*’, ‘*lunch*’, ‘*dinner*’ etc. Also, some of these values are written in different language e.g., French or German, and depends on the use case can be grouped together. The values such as ‘*107th, 108th ... anniversary dinner*’ can be grouped together as just ‘*anniversary dinner’*. Each value can have multiple categories e.g., ‘*lunch and dinner’*, which also can be post-processed based on the use case. The ‘*venue*’ column has 8119 non-empty values. The values in this column have the most of common issues, including question marks, extra punctuations, etc., and new unique issues with abbreviations e.g., ‘*SOC*’ and ‘*SOCIAL*’, ‘*COM*’ and ‘*COMMERCIAL*’. In addition, this column can also have multiple categories within one value. The ‘*place*’ column has 8123 non-empty values. And again, besides common issues, this column has an issue with partial values. The value can represent just the name of the place or place and city or address line, city and state, etc. The ‘*physical\_description*’ column 14763 non-empty values. There as some *‘#N/A*’ values. Each value in this column has multiple sub-values such as type of menu e.g. ‘*booklet*’, ‘*card*’, ‘*folder*’ and physical dimensions of the menu e.g. ‘*5.75 X 7.25*’, ‘*5 X 8*’ and some unique features of the menu e.g. with or without illustration, regular or column layout, folded or open. And this column can have multiple variations of such properties within one value. The ‘*occasion*’ column has 3791 non-empty values. The values of this column also can be grouped into multiple buckets. The ‘*notes*’ column has 10613 non-empty values. The values in this column mostly represented by paragraphs of free text, mostly unstructured. Depends on the use case, additional features can be derived for this column. The ‘*call\_number*’ column has 15983 non-empty values. The majority of values in this column are numeric with some OCR-like issue e.g. we see ‘*o*’ instead of ‘*0*’, or ‘*l*’ instead ‘*1*’. Some of them have postfixes such as ‘*item*’, ‘*\_wotm*’, ‘*copy*’. And some of them starting from the word and continuing with a number, e.g. ‘*Zander 645*’, ‘*Soete 162*’, ‘*Baratta 35*’. The ‘*date*’ column has 16959 non-empty values. And only three values where there are some issues with the year and can be easily detected using timeline facet from OpenRefine. The ‘location’ column does not have empty values. However, there are values such as question mark. The issues are similar to the issues with ‘*name*’ or ‘sponsor’ columns. The columns ‘*currency*’ and ‘*currency\_symbol*’ both have 6456 non-empty values, and they look good. Some preprocessing can be done for cents because it can be cents of different currency. The ‘*status*’ column has all values available and does not have any issues. The ‘*page\_count*’ and ‘*dish\_count*’ columns also have all values available. There are some extreme values that need to be analyzed.

## 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.

* id – The unique identifier of the menu page
* menu\_id – The unique identifier of the menu, corresponds to Menu id
* page\_number – The number representing sequence of page in the menu
* image\_id – The unique identifier of the page image
* full\_height – The height of the page image in pixels
* full\_width – The width of the page image in pixels
* uuid – The universally unique identifier for the highest resolution version of the image

Each MenuPage is associated with some number of MenuItem values.

## MenuItem

Each MenuItem refers to both the MenuPage it is found on -- via the menupageid 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.

* id – The unique identifier of the menu item
* menu\_page\_id – The unique identifier of the page the menu item is on, corresponds to MenuPage id
* price – The first price of menu item
* high\_price – If the item has more than on price on a single menu, the highest price. If there are more than two values for price, the web application instructs volunteers to enter the lowest and highest prices rather than all values.
* dish\_id – The unique identifier of the dish, corresponds to Dish id
* created\_at – The date/time of the first transcription
* updated\_at – The date/time of the last edit to the value
* xpos – The horizontal coordinate on the page for the upper left point where menu item is on the page
* ypos – The vertical coordinate on the page for the upper left point where the menu item is on the page

## 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.

* id – The unique identifier of the dish
* name – The name of dish
* description – The description of the dish
* menus\_appeared – The total count of menus on which dish with this id appears
* times\_appeared – The total count of appearances of the dish with this id across all menus
* first\_appeared – The earliest year of a menu on which a dish with this id appears
* last\_appeared – The latest year of a menu on which a dish with this id appears
* lowest\_price – The lowest price associated with a dish with a given id
* highest\_price – The highest price associated with a dish with a given id

<http://menus.nypl.org/about>

<https://www.kaggle.com/nypl/whats-on-the-menu>

You should describe the structure and content of the dataset and quality issues that are apparent from an initial inspection. You should also describe a (hypothetical or real) use case of the dataset and derive from it some data cleaning goals that can achieve the desired fitness for use. In addition, you should answer the following questions: Are there use cases for which the dataset is already clean enough? Are there use cases for which the dataset will not be clean enough? You can speculate a bit here – but the rest of the project should focus on a “middle of the road” use case that requires a practically feasible amount of data cleaning.

# Data cleaning with OpenRefine

In this first hands-on part of the project, you should use OpenRefine to clean the chosen dataset—either (a) or (b) or your own (c)—as much as needed for the use case. Document the process and result of this phase, both in narrative form along with supplementary information (e.g., which columns were cleaned and what changes were made?). Can you quantify the results of your efforts? Also, provide provenance information from OpenRefine. Pay close attention to what OpenRefine includes and does not include in its operation history! If important information is missing in the latter, provide that information in narrative form.

# Data cleaning with other tools

If you find that certain data cleaning steps are not well suited for OpenRefine (e.g. due to scalability or other issues), consider using an alternative, more suitable solution, e.g., Python, R, or other tools such as Trifacta Data Wrangler, Tableau, etc. Document your choice and answer the same questions as in Step 2.

# Developing a relational schema

Develop a relational schema for your dataset. What logical integrity constraints (ICs) can you identify? Load the data into a SQLite database with your target schema. Use SQL queries to profile the dataset and to check the ICs that you have identified! You can also use other query languages such as Datalog to profile the dataset and check the ICs, but you should not use a procedural language such as Python, R, etc.

A screenshot of a computer

Description automatically generated

# Creating a workflow model

Create a workflow model of your overall data cleaning workflow: What are the key inputs and outputs of your workflow? What are the dependencies? Note: Here you may want to model the various steps you have executed with OpenRefine as parts of the workflow. This way, the workflow model more clearly describes what actually happened to what parts of the data. Create a visual representation of your overall workflow using YesWorkflow or other diagramming tools. Supplementary material to help with YesWorkflow will be posted on Piazza. Also create a visual representation of your OpenRefine workflow using OR2YWTool (https://pypi.org/project/or2ywtool) or other appropriate tools. The OR2YWTool provides an auto-parsing method from Openrefine Operation History JSON file to YesWorkflow model (developed by Lan Li and Nikolaus Nova Parulian). Please include both overall workflow and OpenRefine workflow in your project report.

# Developing provenance

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

# Contribution

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