**ETL Project Report- by Domingo Rodriguez & Eric Salkin**

**Project Proposal**

Does the ethnic breakdown of a Chicago Northside neighborhoods’ population have any correlation with the presence and popularity of the ethnic restaurants in that neighborhood?

**Finding Data \* \*\*E\*\*xtract**

Data Sources used:

1. Yelp
2. OpenTable
3. Chicago Tribune metromix
4. CMAP
5. See Chicago Real Estate

Finding our original data sources was both easy and challenging, like the experience we had before with our first Group Project. We didn’t run into issues in terms of identifying potential sources but securing usable, free data typically can be time consuming.

We first tackled finding data on ethnic restaurants, knowing from our activities in class that this would be the easier part. We used API requests from Yelp, metromix. Com, and OpenTable to get a complete view on Chicago’s ethnic restaurants by neighborhood.

Next finding a data source for Chicago neighborhood zip codes was straightforward although many sites simply offered maps. We found a list of zip codes at chicago-zip-codes-by-neighborhood.php and then formatted the codes into a CSV file.

# Finding free, current data on Chicago’s neighborhood ethnicity proved to be the most challenging task in terms of data mining. First thinking the U.S. Census Bureau could be a good source, we only found city-wide, aggregate data available. Next we went to the City of Chicago data portal but couldn’t find a suitable data source by neighborhood. Finally, after much searching we found datahub.cmap.illinois.gov that had CSV files entitled Community Data Snapshots Raw Data.

# After reviewing the data, we discovered the postal zip codes and neighborhood breakdown were not completely aligned. However, with limited time and no budget, we decided this was still our best and only neighborhood source.

# Data Cleanup & Analysis\*\*T\*\*ransform

# Extensive data cleaning was required to remove NaN columns, null values, and extraneous columns. In addition, we iterated over rows of text to convert our Data Frame.

# We did a join of the zip code and the neighborhood population data with our ethnic restaurant data. We also did a join of the neighborhood data with the zip code data. As stated earlier given the differences between the two files in terms of zip code classification and neighborhood breakout we could only get match data on ethnic populations for only 13 of the 77 Northside neighborhoods. We knew of the limitations of our data before doing the join but still we were surprised by this low number.

With Yelp, it took a long time to get our API setup because the authentication requests involved creating an app and clearly this was not our purpose. In the very limited authentication request documentation, there was no guidance to requests as a user or a client. Also, the definition of a client in the documentation was unclear (e.g. is it an individual, or an actual client, or a program/application?).

Once we were able to get our API, the documentation to do authentication was also sparse. What was provided was their own partnered API interface IDE, "Postman." This required downloading in order to use, build and develop the API request to Yelp and various API endpoints. However, there was quite a learning curve to using Postman's interface and once again with very limited documentation on how to use it. This required a lengthy time of trial and error to distinguish how the authentication header files had to be formatted. We had to find an online tutorial to create the API key python.gotrained.com/yelp-fusion-api-tutorial and found out we had to create a header file in python. More searching ensued involving how to create a dictionary containing an URL string as the value, a header file with the API key as a value, and then parameters as a list of strings where the format is category=string arguments that pass to JSON.get function, method, and argument.

While formatting for OpenTable was less challenging, the way they concatenated the search string was in a very different format than Yelp. This was like our past experiences with APIs but documentation to find the format string was again non-existent, requiring a search for format examples.

With Chicago Tribune metromix, we could not find any documentation at all. In order to use this data source, we would need to build a web scraping strategy to interface with their website first. Then we would need to have it search for the 13 neighborhoods we identified in our data extraction. Given our time limitations and other available data from Yelp and Open Table, this strategy was never initiated.

Once the documentation and formatting was resolved in building the API request, we first had to build a list of zip codes cast as string objects. Having them in a data frame, they were basically a list of lists but with some neighborhoods having multiple zip codes. We had to extract them out of the lists the lists. This was necessary to create individual string objects and to also drop duplicate zip codes.

Using Yelp first, we did a search for all restaurants based on the zip code to see restaurant ethnic distributions. We implemented the API request asking for all restaurant in the zip code and returning a JSON response object. With just 13 zip codes, we still had thousands of restaurants. The response was a list of 13 elements but each one was a dictionary containing each individual restaurant with sub dictionaries and sub lists. The next hurdle would be parsing each JSON response by row to focus on the single zip code. OpenTable JSON responses were similar to Yelp and the amount of work required would be similarly time consuming. This proved to be impossible with our time constraints on the project.

# Data \*\*L\*\*oad

As previously stated, parsing restaurant data by individual zip code proved to be challenging and time consuming. Extensive unwanted information was outputted (e.g. longitude and latitude). If in a perfect world with more time for this project, then once our data was in a more manageable format, we would import it into the SQL database for "storing " data. Then we could use Pandas and other modules to get necessary data to do analysis to verify hypothesis .

If we had time to parse the massive amount of data form our responses, we would choose to do our tables for zip codes and neighborhoods’ ethnic breakout in a SQL database using Postgre SQL. A relational database would be used because they require few assumptions about how data is related or how it will be extracted from the database. We wanted to view the database in many ways, as an important feature of relational systems is that a single database can be spread across several tables.

# We would choose this flexible method as we want to organize and analyze our data using multiple criteria (e.g. majority ethnic group, largest presence of cuisine type, etc.) to potentially identify any unexpected cuisine trends and/or correlations between the neighborhood restaurants and its ethnic population.

# Lessons Learned

# With this assignment, our “training wheels” were taken off dealing with real world data and API key challenges. Even by limiting our sample size to only 77 of Chicago’s over 150 neighborhoods, we would need to complete the lengthy process to slice up the JSON to get a more workable response. If we originally chose all of Chicago’s neighborhoods, we would have exponentially increased our difficulties parsing out the data.

# This project taught us the valuable lesson of when things go sideways. You must be able to document and communicate the challenges that inevitably will occur and outline the strategies you might use to solve the issues.