**README**

This Folder contains all our code for the Yelp Dataset Challenge 2018. Our work is titled ‘Recommendations for Restaurant Improvement using Random Walk Based Opinion Mining’. We have described each of our code files.

**Preprocessing:**

FilterRestaurants.java - Code that extracts businesses that are only restaurants from the businesses dataset

jsontoCSV.py - Code to convert json to CSV

Review Processor.java - Combines businesses and tips for all business ids in ‘ids.txt’

InitialFiltering.scala : Order the data by cities with most restaurants.

CategoriesGrouping.scala - Groups the Categories for each city and displays it in descending order to get the top categories.

Neighborhood.scala : Extracted the most popular neighborhood for each category from the output of CategoriesGrouping.scala and created a csv file for {cuisine, neighborhood}

TopCategories.scala - Extracts the restaurant that is in a particular neighborhood for the selected category and stores it in separate CSV for further processing in JaccardSimilarity.scala

JaccardSimilarity.scala - Split compound attributes into multiple columns and filtered restaurants with more than 50 reviews. Extracted the worst performing restaurant and found its Jaccard Similarity with other restaurants. The output is the business id of the worst performing restaurant and a similar performing restaurant.

Remove\_stopwords.py - removes stopwords and converts all reviews to lowercase.

**Random Walk:**

Sentiment\_graph.py - takes input from remove\_stopwords.py and builds positive and negative graphs for Rmin and Rsim. The random walk is also performed in this step to find the sentiment orientation scores.

Find\_TFIDF\_Dropped.scala - Finds the TF-IDF scores of words in each review and tip and creates a graph to be used in NetworkX to get the PageRank of each word. Dropped the TF-IDF scores due to bad results.

**Result:**

Combine.py : takes the output of sentiment\_graph.py and find the intersection of features for Rmin and Rsim. Uses word\_list.csv to filter out unnecessary features and words.

Evaluation.scala - Output of combined.py was combined with the rating associated with the reviews that contain the word. If the majority was ratings was positive (>=3), the word was given a positive value else negative. If the final count was positive was the good restaurant and negative for the worst restaurant, then it was considered accurate evaluation.

**Mapping**:

We have included the files used for accuracy calculation under Results/Files. Each feature has two numbers associated with it if the first number represents positive/negative impact from Rsim and

1. Montreal-Ville Marie-Pizza:

Rmin: NJOtdjuLfRfg-Ddi7Au7Vw

Rsim: s4fNTcuW6DGfiAagPMlQgw

1. Cleveland-Goodrich Kirtland-Chinese

Rmin: USRXU4ASBAFSKhlk-KVrCw

Rsim: Tm2dKN\_-DCdZfF2xMWrX7w

1. Missisuaga-East Credit-Bars

Rmin: 4sAbjATsbj5XfIDIAeHtXA

Rsim: iwFoA98-OgcdXmNS0LxNOA

1. Pittsburg-Downtown-Italian

Rmin: dHgbL5EAEawIcqk6aXe2Ow

Rsim: 4tJiL2mHKO-erM6xoZji9Q