Isaiah Erven and Puyan Gholizadeh
20 December 2019

Merriam Webster Yelp Dictionaries

Original Goals:
-Use Facebook API and Yelp API to give emotion scores to various Facebook posts and Yelp Reviews
-Rank various family members based on the 'emotion' of their posts

Revised Goals:
-Use the Yelp API and Merriam Webster API to aggregate Yelp data for various cities
-Use this data to compare restaurants on the West Coast to restaurants on the East Coast

Goals Achieved:
-Use the Yelp API and Merriam Webster API
-Give emotion score to Yelp reviews
-Compare East Coast and West Coast by comparing various college towns
-Developed command line interface w/ command line arguments
-Used data in database to create visualizations of data

Problems faced:
-Facebook reluctance to release any data
-Reading about Merriam Webster API, as it seems to be less popular
-Figuring out API keys
-Keeping track of large amounts of data
-Making sure there wasn't code that was going to make too many requests
-Maintaining code with good style
-Reliability of APIs
-Reliability of database software

Documentation:
Instructions for running the code:
-Easier to run if you navigate to the folder where the files are present ('cd' command)
-Run "python main.py --help" for instructions in the command line
-The initialize command must be run first
-Run "python main.py --init" to initialize data from Merriam Webster API
-Each new run of the program will gather data for an individual city
-East Coast cities are indexed 0-4 and West Coast cities are indexed 5-10

-Run the program with command line argument 0-10 in order
- “python main.py 0”, “python main.py 1”, “python main.py 2”... “python main.py 10”
-When running the program with index 0, a new table for the East Coast restaurants will be created
-When running the program with index 6, a new table for the West Coast restaurants will be created
-When running the program with index 10, a print statement will indicate that all data collection is complete

-Run visualizations and calculations file to acquire further information
- “python visualsANDcalculations.py”

Files/Functions in project:

main.py:
Driver file of program
# REQUIRES: command line arguments --help, --init, value 0-10
# MODIFIES: creates .db file, JSON files, modifies all of the created files
# EFFECTS: reads data from 2 APIs, and creates JSON files/.db files with processed data

thesInit.py:
Responsible for dealing with the Merriam Webster API and creating files with word data to be used in emotion calculations

Functions:

-getWordsMerriamAPI

# REQUIRES: Null (No parameters)
# MODIFIES: Modifies JSON files containing word data from Merriam Webster API
# EFFECTS: Makes requests to Merriam Webster API to obtain synonyms
# Appends list of positive/negative words to be used in emotion score calculation
# Writes word data to JSON files to be used in emotion calculation and db storage

yelpInit.py:
Responsible for dealing with the Yelp API and getting business/review data. Creates json files with data

Functions:

- getCityData
  - # REQUIRES: Name of city
  - # MODIFIES: JSON Files with review data
  - # EFFECTS: Makes request to Yelp API, and acquires review data for given city. Parses this data into various dictionaries and eventually a JSON file. Prints out steps of the process

emotion.py:
Responsible for calculating emotion score based on review text and positive/negative words

Functions:

- getEmotionScores
  - # REQUIRES: Null (No parameters)
  - # MODIFIES: JSON files with review data
  - # EFFECTS: Reads in review data and word data, calculates emotion score for each review,
  - # appends emotion score to review data and writes to new JSON file
- calcEmotionScore (contained w/in getEmotionScores)
  - # REQUIRES: restaurant review text
  - # MODIFIES: Null
  - # EFFECTS: Returns calculated emotion score for provided review text

wordsdb.py:
Responsible for creating the positive/negative words tables and writing data from json files to database

Functions:

- createWordsDBS
  - # REQUIRES: json_data of word data from Merriam Webster API, DB cursor, DB connection
  - # MODIFIES: .db file
  - # EFFECTS: Creates tables to store words and their synonyms, extracts 'root words' and their synonyms
  - # Inserts words into the database as pairs (root word, synonym)
  - # Makes DB insertions in bundles of 20
  - # Print statements for each step

Reviewdb.py:
Contains db initialization functions and responsible for creating the East coast/West coast tables and writing data from json files to database

Functions:

-readDataFromFile
  - # REQUIRES: filename string
  - # MODIFIES: Null (Read only)
  - # EFFECTS: Returns json data in provided filename

-setUpDatabase
  - # REQUIRES: name of database
  - # MODIFIES: connection to database
  - # EFFECTS: returns cursor and connection to database, location of the database

-insertIntoDB
  - # REQUIRES: json_data to be inserted, cursor/connection to database, city index, list of cities
  - # MODIFIES: database from connection
  - # EFFECTS: Writes provided review data to database

visualsANDcalculations.py:

Creates all of the visuals along with calculations which are written to a text file

Functions:

- getEastTotalAggregate
  - REQUIRES: cursor and connection to database
  - MODIFIES: .txt file
  - EFFECTS: Calculates the average aggregate rating for the east coast restaurants and returns it

- getWestTotalAggregate
  - REQUIRES: cursor and connection to database
  - MODIFIES: .txt file
  - EFFECTS: Calculates the average aggregate rating for the west coast restaurants and returns it

- getEastAggregatePerPrice
  - REQUIRES: cursor and connection to database
  - MODIFIES: .txt file
  - EFFECTS: Calculates the average aggregate rating for the east coast restaurants for each price category and returns them in a list

- getWestAggregatePerPrice
    - REQUIRES: cursor and connection to database
    - MODIFIES: .txt file
    - EFFECTS: Calculates the average aggregate rating for the west coast restaurants for each price category and returns them in a list

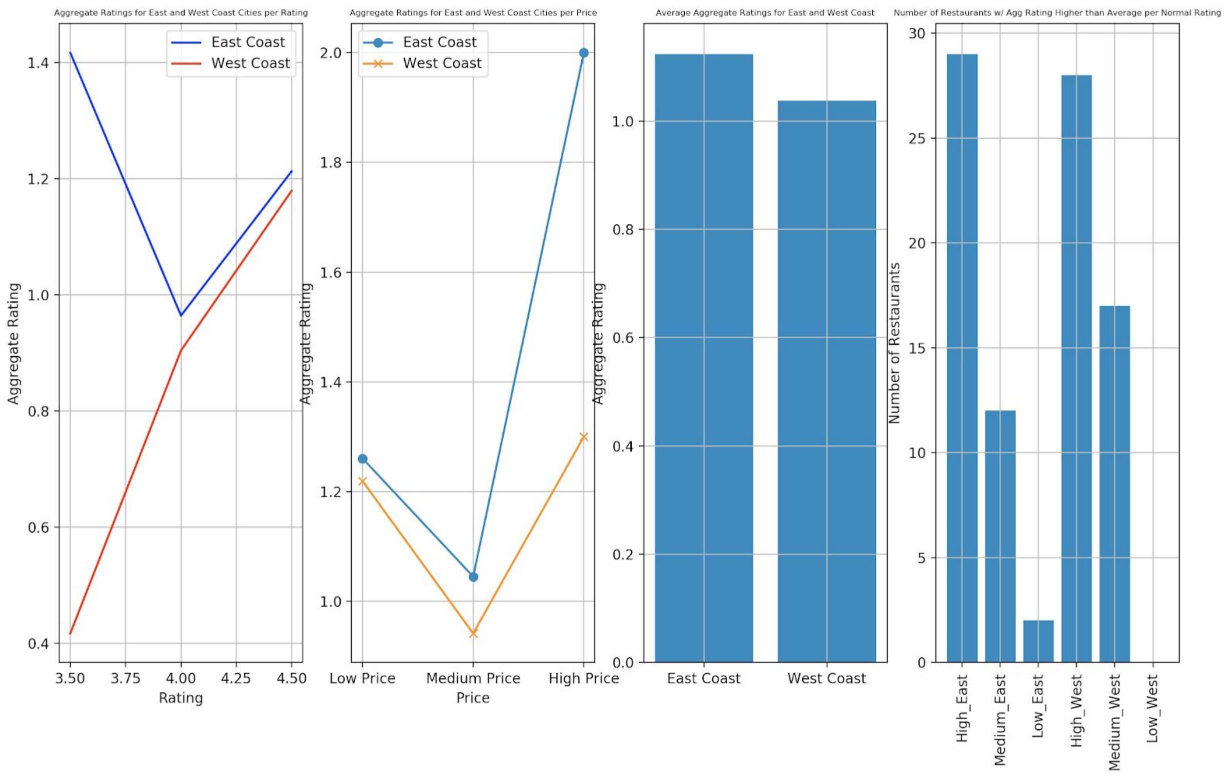
- getEastAggregatePerRating
    - REQUIRES: cursor and connection to database
    - MODIFIES: .txt file
    - EFFECTS: Calculates the average aggregate rating for the east coast restaurants for each rating and returns them in a list

- getWestAggregatePerRating
    - REQUIRES: cursor and connection to database
    - MODIFIES: .txt file
    - EFFECTS: Calculates the average aggregate rating for the west coast restaurants for each rating and returns them in a list

- joinEast
    - REQUIRES: average aggregate of east coast city restaurants, cursor, and connection to database
    - MODIFIES: .txt file
    - EFFECTS: Returns a list of values corresponding to how many restaurants per rating category had a higher aggregate rating that the average aggregate of east coast city restaurants

- joinWest
    - REQUIRES: average aggregate of west coast city restaurants, cursor, and connection to database
    - MODIFIES: .txt file
    - EFFECTS: Returns a list of values corresponding to how many restaurants per rating category had a higher aggregate rating that the average aggregate of west coast city restaurants

Visualizations:



Sources:

Date:	Description:	Location:	Result:
Dec 8th	Article for python command line args	https://stackabuse.com/command-line-arguments-in-python/	Success
Dec 8th	Yelp API Docs	https://www.yelp.com/developers/documentation/v3	Success
Dec 8th	Merriam Webster API Docs	https://dictionaryapi.com/products/json	Success
Dec 6th	Facebook API Docs	https://developers.facebook.com/docs/	Failure
Dec 18th	Appending to Text File	https://www.guru99.com/reading-and-writing-files-in-python.html	Success
Dec 18th	Adjusting Matplotlib Visualizations	http://ishxiao.com/blog/python/2017/07/23/how-to-change-the-font-size-on-a-matplotlib-plot.html	Success

Github link: https://github.com/ierven/206final