Alex Smith
MIDS W205

ASSIGNMENT #3

OVERVIEW

I wrote lexicalfind.py to examine the lexical diversity of tweet data collected from Twitter's REST API. I define lexical diversity as the ratio of unique words to all words in a given text corpus. The tweets contain two hashtags: #NBAFinals2015 and #Warriors. I store the data in Mongo databases to have quick access for my analyses.

ARCHITECTURE & DESIGN

The program can be run by simply typing "python lexicalfind.py" into the command line. The user can modify the search queries by simply changing the fields, "QUERY1" and "QUERY2."

The program requires importing a number of modules. We list the modules here and explain their purpose in comments in the code: pymongo, bson, sys, urllib, datetime, boto, json, nltk, tweepy, signal, threading, and time.

The architecture of this program is designed around the main program, lexicalfind.py. The program uses a slightly modified tweetfetcher and tweetserializer from Assignment #2 to download new tweets. It adds the data to a Mongo database called db_restT using the createmongo class. It then adds the tweets from Assignment #2 to another Mongo database called db_tweets. The code for downloading these tweets directly from S3 is included but commented out because I didn't correctly upload my data to S3 in Assignment #2.

We use MongoDB because it is a NoSQL database and handles semi-structured tweet data well. Instead of relational tables, MongoDB uses a document structure to store its data. I call the program to find the top retweets by using the findtop class. The findtop class uses Mongo's built-in querying functionality to sort all tweet data by the the number of retweets ("retweet_count") and limits this list to the top 30.

After finding the top retweets, the program calls the lexicaldiversity class to calculate the a lexical diversity score for each individual in the db_restT database. It searches Twitter using the user's screen name because each screen name corresponds to a unique user to collect a sample of 200 tweets. I choose to limit my sample because this sample size provides me a large sample while minimizing the number of times I exceed Twitter's rate limit. I then export the top 30 user's lexical diversity to a CSV file using a new class, mongotocsv.

Findfollowers identifies the followers of those users with the top 30 retweets. The class searches Twitter for each user's followers and stores the information in a new Mongo database called db_followers. The program, unfriended, calculates the followers of users in db_followers and identifies which followers have stopped following the top 30 retweeters. The code is commented out until the user wants to find the difference.

I back up my files using the mongos3 class. I choose to export my files in JSON format because the semistructured nature of the tweet data lends itself well to JSON files. I also choose to use JSON files as back up because it is relatively easy to iterate over a JSON file and add its content to a MongoDB database. I design my program to add a time stamp to the back up file because often users need multiple back ups of a database.

RESILIENCY

This code is designed to be resilient to exceptions from both software and the user. The code uses the threading module to ensure that if the user aborts the program, the program will still finish the tweet it is writing. The code handles Twitter's rate limits by using the Tweepy module and waiting every time the limit is reached. The code handles Twitter connection shut downs by putting the downloading of tweets in a try function and waiting for 1,000 seconds if Twitter closes the connection. Sometimes, a user's profile may be unavailable. In such cases, the program simply prints an error, waits 5 seconds, and then continues to the next user.

OUTPUTS

I store a back ups of my Mongo databases to S3. I include a method to download these files and reload them into MongoDB. My files are here:

https://s3-us-west-1.amazonaws.com/keivahn-w205-assignment3/46b9b24d9b463ade99f20f6eb0c5f1dc

https://s3-us-west-1.amazonaws.com/keivahn-w205-assignment3/7b32393919110fdedd2a5dac44089d7a