Owen McCormack & William Thomas

Our method & code can be broken down into 7 main sections / functions:

1) Pre Processing *(Raw_Tweet_Preprocessor.py):*

This function reads in raw twitter data, selects only the English tweets for the target company and extracts the desired information for these tweets, namely:

- Unique Tweet ID
- Timestamp
- The tweet
- Number of followers.

This information is then saved to a .txt file in json format ready to be allocated a sentiment score.

Note: some blank fields are included in the output to make later manipulation easier.

To run this program:

1.) Firstly split the input into c.100MB files in order to enable easy file open/close. Use 'split -b 100m filename' in the command line. Files will be titled in ascending alphabetical order, starting at *'xaa'*.
2.) Uncomment appropriate key words options in the program depending on target company.
3.) Run Raw_Tweet_Preprocessor with the following arguments:
    1) 1: Address of directory containing split files.
    2) 2: Output file name / address
    3) 3: The last file in the directory + 1. i.e. if last file is called xbe, enter xbf.

2) Score with sentiment Lexicon and identifying key indicators *(scorer.py)*:

This function reads in a json format database structure, where each tweet text is an entry in each row. I also imports the dictionary (lexicon) of adjectives that is taken from the SentiWordNet dictionary (note this has been constructed to provide one score for each word i.e. it is not contextual). It iterates through the tweets and provides the relevant sentiment scores by:

- Splitting the tweet into individual words;
- Checking if the word exists in the SentiWordNet dictionary of adjectives;
- If the word is in the dictionary, sums up the positive and negative scores for each word of each tweet and adds to the json
- Checks if the tweet includes any of the symbols in the list 'PositiveTermsToCheck' and 'NegativeTermsToCheck' (the lists of the smileys) – if so, update the json accordingly.
- Output the data as a json.

3) Sentiment Summarising *(Sentiment_Summariser.py)*:

This Function consolidates the results from program 2 on a daily basis, providing outputs in '.csv' format that can then be used in matlab to attempt to predict the fluctuations in the stock market. Run the function with the input file as the first argument and the output file as the second.

The output contains the following columns:

- Adj_Days: *3-digit number representation of date. First digit = month (1:Jan, 2:Feb, 3:March). Second & third digit = date.*
- Count_of_scored: *Number of tweets given a sentiment score on each day.*
- total_pos: *Daily total positive sentiment score.*
- total_neg: *Daily total negative sentiment score.*
- total_pos_weighted: *Daily total of positive sentiment scores x number of followers on a tweet by tweet basis.*
- total_neg_weighted: *Daily total of negative sentiment scores x number of followers on a tweet by tweet basis.*
- total_pos_weighted_norm: *Daily total of positive sentiment scores x log(number of followers) on a tweet by tweet basis.*
- total_neg_weighted_norm: *Daily total of negative sentiment scores x log(number of followers) on a tweet by tweet basis.*
- mean_columns: *Above columns divided by count od scored for each day.*

4) Identifying training set *(Generate_Training_Set.py)*:

Function reads scored tweets from a json format, selecting tweets that have both an identifier and a sentiment lexicon score. The selected tweets are then saved in a dataframe format, ready to be used in matlab as a training set to train a better tweet classifier.

To use this program, simply run it using the input data .txt file as the first argument and the desired output.csv file as the second argument.

The information needed from the output file to train a classifier is the following:

- nos_identifier: *1 if identified as a positive tweet, o otherwise.*
- neg_identifier: *1 if identified as a negative tweet, o otherwise.*
- pos_score: *Original positive score from sentiment lexicon*
- neg_score: *Original negative score from sentiment lexicon*

5) Matlab sentiment classifier – method 3 *(TweetsLogReg.m)*:

This code accepts the training set list of tweets that have a recognised 'smiley' classifier and a pos/neg score from method 1, and constructs a classifier using logistic regression that maps the score from Method 1 to the 1/0 presence of either positive or negative score. It outputs the parameters of a logistic regression function, to be applied across the full dataset in the 'Re-Classifying' step.

6) Re-Classifying *(Re_Classifier.py)*:

This Function applies the linear classifier generated in program 5 using the identifier based training set to the broader dataset. The output is consolidated on a daily basis and saved to a csv file. Before using, ensure that the linear classifier coefficients are correct for the particular company you are studying. Then just run the function with the input file as the first argument and the output file as the second.

The new classification counts can be found in the output under the following columns:

- pos_Identifier

- neg_Identifier

7) Correlation & prediction *(matlabcode.m)*:

This code firstly sets some global parameters (e.g. proportion of data in the sample set), then iterates throught the 'k-fold' section 10 times. The data is randomised row-wise, split into training and testing and the columns are selected for the input variables. The Matlab corr() function provides the correlation analysis, and the fitnet() function the neural network fitting. The results for each fold is stored in the 5 matrices at the end of the script.