NLP : ASSIGNMENT - 3

README

3a : Sentiment Analysis in Twitter

FILES included

The folder “SENTIMENT_ANALYSIS” contains 7 files -

1. Twitter_Sentiment_Analysis.ipynb : The main jupyter-notebook for Q) 3a.
2. Twitter_Sentiment_Analysis.py : The python file for the above
3. Sentiment_train_5k.csv : Training dataset for 5k tweets
4. Sentiment_test_1k.csv : Testing dataset for 1k tweets
5. BingLiuPositiveLexicons.txt : Bing Liu +ve Lexicons extracted from BingLiuLexicons.txt
6. BingLiuNegativeLexicons.txt : Bing Liu -ve Lexicons extracted from BingLiuLexicons.txt
7. Contractions.json : File consisting of contraction words

Methodology, Preprocessing steps, and Assumptions

- The initial analysis has been done for 5k training tweets & 1k testing tweets, which is roughly 0.4% of the total dataset, but still the model managed to achieve a decent accuracy of 69%
- For preprocessing, the code performs a number of steps to reduce the redundant data, for eg - removing usernames, URLs, digits, single characters, stripping extra spaces, etc.
- As for the features, following 11 features have been implemented -
    1. check_elongation : Counts the freq of elongated words in a tweet
    2. check_hashtag : Counts the freq of hashtags used in a tweet
    3. check_CAPS : Counts the freq of CAPS words in a tweet
    4. check_negation : Counts the freq of negative words in a tweet
    5. check_p_emoji : Count the occurence of positiveemoji in a tweet

6. check_n_emoji : Count the occurence of negativeemoji in a tweet
7. check_p_lexicon : Count the freq of Positive Lexicons from BingLiu dataset
8. check_n_lexicon : Count the freq of Negative Lexicons from BingLiu dataset
9. Adding normalised score of a tweet
10. Adding the number of relevant punctuation like !!!??!
11. Performing count Vectorization, Unigram + Bigram

- All the DataFrames & models are saved (using Pickle) into a new folder called “assets”. The folder is created if not already & models are loaded if already saved
- Bag of Words has been implemented for Naive Bayes approach
- Rest of the classifiers have sklearn library implementation

---

3b : Emotion Intensity Prediction

FILES included

The folder EMOTION_INTENSITY_PREDICTION further contains 2 folder

FOLDER 1 : JOY

This folder “JOY” contains 7 files -

1. Emotion_Intensity_JOY.ipynb : The main jupyter-notebook for Q) 3b JOY part
2. Emotion_Intensity_JOY.py : The python file for the above
3. Joy_train.csv : Training dataset for JOY
4. Joy_test.csv : Testing dataset for JOY
5. Joy_emotion.csv : JOY words extracted from '8. NRC-word-emotion-lexicon.txt'
6. Joy_hashtag.csv : JOY hashtags extracted from '5.NRC-Hashtag-Emotion-Lexicon-v0.2.txt'
7. Emotion_expanded.csv : File from '6. NRC-10-expanded.csv' given

FOLDER 2 : ANGER

This folder “ANGER” contains 7 files -

1. Emotion_Intensity_ANGER.ipynb : The main jupyter-notebook for Q) 3b ANGER part
2. Emotion_Intensity_ANGER.py : The python file for the above
3. Anger_train.csv : Training dataset for ANGER
4. Anger_test.csv : Testing dataset for ANGER
5. Anger_emotion.csv : ANGER words extracted from '8. NRC-word-emotion-lexicon.txt'
6. Anger_hashtag.csv : ANGER hashtags extracted from '5.NRC-Hashtag-Emotion-Lexicon-v0.2.txt'
7. Emotion_expanded.csv : File from '6. NRC-10-expanded.csv' given

Methodology, Preprocessing steps, and Assumptions

- The code is fast executes in a relatively short amount of time as compared to 3a, so no need to save the models (takes at max 13 sec)
- The list of words containing JOY/ANGER as word or emotions has already been preprocessed & saved in files
- Following 11 features have been implemented -
    1. check_elongation : Counts the freq of elongated words in a tweet
    2. check_hashtag : Counts the freq of hashtags used in a tweet
    3. check_CAPS : Counts the freq of CAPS words in a tweet
    4. check_tag : Counts the freq of tagged people
    5. check_negation : Counts the freq of negative words in a tweet
    6. check_word_emotion : Count the freq of words found in word_emotion in a tweet
    7. check_joy_hashtag : Return the average score of words found in joy_hashtag in a tweet
    8. check_exp_emo : Return the average score of words found in emotion_expanded in a tweet
    9. VADER : Append the list of dict values {'pos', 'neu', 'neg', 'compound'} as given by polarity_score

10. Adding the number of relevant punctuation like !!!??!
11. Performing count Vectorization, Unigram + Bigram

- The following terms are printed for the results -
    1. Mean Absolute Error
    2. Mean Squared Error
    3. Root Mean Squared Error
    4. R2 - Score
    5. Pearson correlation, p-value
    6. Spearman Result

More details & screenshots can be found in Output.pdf