ASSIGNMENT 2: SENTIMENT MINING

Motivation: The motivation of this assignment is to get practice with text categorization using classical Machine Learning algorithms.

Problem Statement: The goal of the assignment is to build a sentiment categorization system for tweets. The input of the code will be a set of tweets and the output will be a prediction for each tweet – positive or negative.

Training Data: We are sharing a training dataset of positive and negative tweets. The format of each line in the training dataset is <“label”, “tweet”>. It has a total of 1.6 million tweets. A label of 0 means negative sentiment and a label of 4 means positive sentiment.

This dataset has been labeled not manually but automatically. In this automatic labeling process tweets with positive smileys were searched and labeled positive and tweets with negative smileys were labeled negative. The smileys have been stripped off from the tweets to make the classification task real.

The Task: You need to write a classifier that predicts for a given new tweet (which does have a positive or a negative sentiment) its sentiment polarity. To accomplish this, as a baseline algorithm, you could use all words as features and learn a classifier (naïve Bayes or logistic regression). To improve your system performance over the baseline I list a few ideas below.

1. Try changing the classifiers. Try SVMs, random forests or other classifiers.
2. If you use logistic regression try playing with regularizers – try L1 instead of L2. You could also implement a different feature selection procedure.
3. Try to work with the features. You could lemmatize. You could get rid of stop words and highly infrequent words. You could use tfidf-based weighting.
4. Try to use existing sentiment resources discussed in class. Examples: SentiWordnet or General Inquirer.
5. You could work with bigrams (in addition to unigrams).
6. You could do tweet normalization on words where letters are repeated for intensity, e.g., haaaaaaate and looooooove. You can even do more normalization using some internet slang dictionary.
7. You could define new features, like you could pos-tag each word and use the tagged word as a feature instead of the original word. You can use presence of capitalization or all caps as

features. You could use this code for POS tagging of tweets: https://github.com/aritter/twitter_nlp

8. You could look at data quality – if you find that data is not consistently good, you could label some tweets by hand, and use semi-supervised techniques to correct the data errors (or even remove parts of the data).
9. Your idea here… however, you are NOT allowed to use word vectors or neural networks. If in doubt, ask on Piazza.

Methodology and Experiments:

We split the data into train-dev-test. 10% of data randomly is test set, 10% is development set. Train on training data, i.e., 80% of the data. If needed, do parameter fitting on the devset and finally test on the test set.

As you work on improving your baseline system document its performance. Perform error analysis on a subset of data and think about what additional knowledge could help the classifier the most. That will guide you in picking the next feature to add.

Test Format:

Your final program will take input a set of tweets. Each line will have one tweet (without double quotes). Your program will output predictions (0 or 4) one per line – matching one prediction per tweet. Sample text file and sample output file are in your assignment package.

What to submit?

1. Submit your code (trained on all 100% data and not just on a subset) by Thursday, 30th September 2021, 11:55 PM. The code should **not** need to train again. You should submit only the testing code, after the models have been trained. That is, you should not need to access the training data anymore.
2. You must also submit training code with the best hyperparameters. If needed, we will train your code on HPC to verify the performance.

   Submit your code in a .zip file named in the format **<EntryNo>.zip.** Make sure that when we run "unzip yourfile.zip" the following files are produced in the present working directory:
   compile-test.sh
   compile-train.sh
   run-test.sh
   run-train.sh

You will be penalized if your submission does not conform to this requirement.

Your code will be run as ./run-test.sh inputfile.txt outputfile.txt. The outputfile.txt should only contain a sequence of numbers (0 or 4), one per line, with total number of lines matching inputfile.txt. We are providing a format checker. Make sure your code passes format checker before final submission.

Your code should run on HPC.

3. The writeup.txt should have two lines as follows
   First line should mention names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone else say None.
   After these first two lines you are welcome to write something about your code, though this is not necessary.

What is allowed? What is not?

1. The assignment is to be done individually.
2. You must use Python for this assignment.
3. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team.** Please read academic integrity guidelines on the course home page and follow them carefully.
4. Feel free to search the Web for papers or other websites describing how to build a tweet sentiment classifier. However, you should not use (or read) other people's sentiment mining code.
5. You can use any pre-existing ML softwares for your code. A popular example includes Python Scikit (http://scikit-learn.org/stable/). However, if you include the other code, use a different directory and don't mix your code with pre-existing code.
6. You are not allowed to use any component that has been developed based on neural models – this includes use of word vectors, deep learning models, pretrained language models, or even lists or other components that used neural models in their work. For all practical purposes assume you are in 2012. When in doubt, ask on Piazza!
7. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.
8. Your code will be automatically evaluated. You get significant penalty if it is does not conform to output guidelines. Make sure it satisfies the format checker before you submit.