ASSIGNMENT 2: SENTIMENT MINING

Motivation: The motivation of this assignment is to get practice with text categorization using 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 even multilayer perceptrons.
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 use do tweet normalization on words where letters are repeated for intensity, e.g., haaaaaaate and looooooooove. You can even do more normalization using internet slang dictionary here: http://www.noslang.com/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 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…

Methodology and Experiments:

Option 1: Remove 10% of data randomly as test set and 10% as development set. Train on 80% of the data. If needed, do parameter fitting on the devset and finally test on the test set.

Option 2: Do a 10-fold cross validation. Train on 8 folds, use 9^{th} as devset and 10^{th} as test set. Repeat this 10 times with different folds as test set. This is a more robust option.

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.

After you find the best system explore a small research question. Do the experiments for that research question and write your results. Possible research questions include:

1. Impact on performance on the size of the training data. Use log-scale on the x-axis (size of training data). See how much an exponentially more training data helps.
2. Impact of different kinds of features (unigram/bigram/twitter specific/etc). For this experiment you could perform *Ablation Study,* in which you remove one feature set and see the impact in performance and do this for several feature sets. This will help you understand the incremental value of each feature set.
3. Compare machine learning based system, knowledge-based system and hybrid system. The knowledge-based system will be a classifier that only uses background sentiment resources. A hybrid will intelligently use this knowledge in the ML system.
4. If you are a core machine learning person, you could test various classifiers on this dataset and try to find a reason for why a certain classifier is performing better than the others.
5. Your idea here.

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. Here is a sample test file and the sample output.

What to submit?

1. Submit your best code (best if trained on all training data and not just on a subset) by Tuesday, 16th September 2014, 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.

   Submit your code is 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.sh
   run.sh
   Writeup.pdf (and not writeup.pdf, Writeup.doc, etc)

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

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

   Your code should run on Baadal machines with 2 GB RAM.

2. Your writeup (at most 2 pages, 10 pt font) should describe how you created your best sentiment classifier (about 1 page). Explain any interesting ideas that you used. Describe the research question you considered and what you found (about 1 page). Also mention the names of people you discussed the assignment with. The writeup will judged on clarity and innovation as well as experimental results.

Evaluation Criteria

(1) 20 points are for description of the system. Anything innovative may yield bonus points.
(2) 30 points for the experiments and research. Anything innovative may yield bonus points.
(3) 60 points for performance of your code.
(4) 10 points bonus given to outstanding write-ups and best performing systems.

What is allowed? What is not?

1. The assignment is to be done individually.
2. You may use Java, or 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. Cite the references in your writeup. 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. Popular examples include Weka (http://www.weka.net.nz/), 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. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.
7. Your code will be automatically evaluated. You get a zero if it is does not conform to output guidelines. Make sure it satisfies the format checker before you submit.