ASSIGNMENT 1: SENTIMENT MINING

Motivation: The motivation of this assignment is to get you some practice with text categorization using Machine Learning algorithms. This assignment will proceed in two parts. Each part is worth equal points.

PART I

Problem Statement: The goal of the assignment is to build a sentiment categorization system for business reviews from annotated data. The input of the code will be a set of annotated business reviews from the website Yelp.

Training Data: We are sharing a training dataset of business reviews. Each data point has review text, and a rating. Ratings are floating point numbers between 1 and 5.

The Task: You need to write a classifier that given a review, predicts its sentiment polarity. In this Part, you must develop a non-neural classifier. Ideas include Naïve Bayes, logistic regression, nearest neighbor, rule/lexicon-based systems and so on.

For Part I, as a baseline algorithm, you could use all words as features and learn a classifier (naïve Bayes or logistic regression). Before you begin, try to carefully think about the problem a little. To improve your system performance over the baseline, a few ideas to try are listed below.

1. Try changing the classifiers. Try SVMs or random forests.
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 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.
7. Your idea here…

Methodology and Experiments:

You are being provided two sets of data – train.json and dev.json. You must do all your hyperparameter tuning on the dev set. Once you find the best hyperparameters, set them in your final training script to be submitted. We will be retraining your models on the training data and using the retrained model for evaluation, so make sure you make your code deterministic (or not). Try to take the assignment in the proper spirit and refrain from trying out funny businesses with the submissions (i.e., do not search for test set or use test set in any form while training).

Most importantly, as you work on improving your baseline system, document its performance. Perform (statistical) error analysis on a subset of data and think about why the model is making these mistakes and what additional knowledge could help the classifier the most. That will guide you in picking the next feature (or model component) to add.

Test Format:

Your final program will take as input a set of reviews in the same format as training, without the rating. Your program will output predictions (floating point numbers between 1 and 5) one per line – matching one prediction per review.

What to submit?

1. Different deadlines will be announced as the course evolves. The first deadline will be on 19th Feb 11:55 PM for the submission of Part I system.
   Submit your code in a .zip file named in the format **<EntryNo>.zip.** Make sure that when we run "unzip yourfile.zip" a new directory is created with your entry number (in all caps). In that directory, the following files should be present.
   compile.sh
   train.sh
   test.sh
   writeup.txt

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

Your code will be run as :

1. ./compile.sh
2. ./train.sh trainfile.json devfile.json model_file
3. ./test.sh model_file testfile.json outputfile.txt

The outputfile.txt should only contain a sequence of numbers (between 1 and 5), one per line, with total number of lines matching the number of reviews in testfile.json. If your code doesn't conform to the requirements you will lose 20% of the credit.

Your code should work on a single HPC node with 1 cpu and 1 gpu. You will be penalized for any submissions that do not conform to this requirement. The training script will be given maximum of 6 hrs.

2. The writeup.txt should have first line that mentions 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 this first line you are welcome to write something about your code, though this is not necessary.

Evaluation Criteria

(1) This part is worth 70 points.
(2) Bonus given to outstanding performers.

What is allowed? What is not?

1. The assignment is to be done individually.
2. **You may use only 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 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. Popular examples include 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 significant penalty if it is does not conform to given guidelines.