Assignment 4

This is an individual assignment. If you get help from others you must write their names down on your submission and explain how they helped you. If you use external resources you must mention them explicitly. You may use third party libraries but you need to cite them, too.

Date posted: April 7th, 2016

Date Due: April 14th, 2016

Goal: Classification. SVM classifier

In this assignment, you will gain hands-on experience on applying SVM classifier to solving a sentiment classification problem. You are not expected to implement the classifier by yourself, however, you should know how to use an open source software (LibSVM).

Task1: Download LIBSVM

Please download LIBSVM from https://www.csie.ntu.edu.tw/~cjlin/libsvm/, and download the .tar.gz version. Please unzip and compile it (simply by *Make*) in a UNIX system.

After unzipping, please very carefully read the README file (especially the svm-train and svm-prediction part).

Task 2: Process Data Collection

Please download the provided data collection *textcat.zip.* The data in textcat.zip are, when unzipped, organized in the following directory structure:

```
train/
   train/neg/
   train/pos/
dev/
  dev/neg/
  dev/pos/
test/
```

Each directory contains some .txt files, with one review per file. For the training and development data, the positive and negative reviews are known, and indicated by the

subdirectory names. For the test data, all reviews are mixed together in a single directory.

The text has been pre-tokenized and lower-cased. All you have to do to get the individual terms in a review is to split the tokens by whitespace, a sequence of spaces and/or newlines. You don't need to remove stop-words or do any further stemming.

Please ignore any terms that appear **fewer than 5 times** in the combined positive and negative training and development data (in the train folder and dev folder), and therefore you can get the vocabulary. Please assign each term in the vocabulary an integer termID starting from 1, and generate a *termID->term* mapping file.

Task 3: Extract features and represent documents

A. Please combine all the reviews in the train/neg and train/pos to generate one *train_data* file. In this *train_data* file, each review is represented as one row, as follows:

   <label> <index1>:<value1> <index2>:<value2> … <indexN>:<valueN>

   Where:

   <label> is either +1/-1, indicating whether this review is known as positive or negative

   <index> is the termID for each term in the vocabulary set. It should be of Integer type, and start from 1. We are using unigrams as features to represent each review.

   <value> is the feature value. In this experiment, we set the value to be the normalized term frequency of each term in this review, i.e. for termID=t1 in review d, the value for t1 can be computed as:

$$value(t_1, d) = \frac{tf(t_1, d)}{len(d)}$$

   Please sort index in an ascending order, and you only need to record on each row those terms with non-zero term frequency.

B. Please combine all the reviews in the dev/neg and dev/pos to generate one *test_data* file. In this *test_data* file, each review is represented as one row in exactly the same format as introduced in Task 3.A for *train_data*.

Task 4: Run LibSVM

A. Training

Run the following command:

./svm-train train_data model_file

You do not need to specify any optional parameters, just use their by-default values.

B. Testing

Run the following command:

./svm-predict test_data model_file result

The *model_file* is the file generated in the training process. "*result*" file will contain the predicted label for each review in the testing data set, and the system can report to you the **accuracy** automatically.

What to hand in:

1) Your temID->term mapping file
2) Your generated *train_data* file.
3) Your generated *test_data* file.
4) Your classification *result* file.
5) Report the accuracy value.