# CSCI 544 Applied NLP: Homework 2

## Due Date: September 30, 2015 (10:59 AM PST)

## Bitbucket project name = csci544-hw2

## Updates

- 9/20/15: See new discussion of smoothing, and common and rare and unknown words below. Also, because SVM light requires feature names to 1 or greater, I've updated the assignment to include this as a preprocessing step.

## Overview

The goal of this assignment is to get some experience with text classification and two popular machine learning toolkits. You will be working with two datasets (emails and IMDB reviews) and performing binary classification: SPAM or HAM (not spam), and POSITIVE or NEGATIVE. You will be comparing three machine learning techniques for making these classifications: naive Bayes classification, maximum entropy modeling, and support vector machines. In the case of naive Bayes classification, you will be implementing the technique yourself as it is good practice and the algorithm is relatively simple. In the other two cases you will be using implementations of these machine learning techniques written by someone else (MegaM and SVM-Light). In all cases, you will be using tokens as your features. You will write a short report about what you did and your findings.

We are providing two sets of data. One is labeled, and one is specifically for testing and we are not providing the labels. You will use the labeled data for three tasks. You will discuss and document your efforts in a file called report.txt. The report should include any instructions necessary for running your code.

1. You will split the labeled data using 75% for training and 25% for development to measure the performance of the three machine learning techniques (i.e., train on the training data and measure performance on the development data). Report the precision, recall and F1 score for SPAM, HAM, POSITIVE and NEGATIVE for the three machine learning techniques. Answer the following questions: In each case, which technique performs best? Based on class discussions, why do think this is? How does performance differ between SPAM detection and sentiment analysis (POSITIVE v. NEGATIVE)?
2. You will again split the labeled data into training and development but this time only use 25% of the data for training and use 75% for development. The goal is to measure the effect of a smaller training size. Report the precision, recall and F1 score for SPAM, HAM, POSITIVE and NEGATIVE for the three machine learning techniques. Answer the following questions: How much did performance drop for each of the machine learning techniques? Were some machine learning techniques more robust given a smaller training set? Is

there a difference between SPAM detection and sentiment analysis?
3. You will use all the labeled data to create models with the three machine learning techniques. You will then run these models on the unlabeled testing data and save the output. You will include the output in your bitbucket repository. We will measure the accuracy of your models and compare it to reference models developed by the TAs.

# Project Data

In this work, we are using tokens as features. However, rather than use the text of the token as the feature name, we will assign each unique token an integer identifier. For the IMDB movie reviews, the file imdb.vocab is a list of all the unique tokens in the corpus, one per line (all lower case). The token on line 1 of the file should be given the identifier 1; the token on line 2 has the identifier 2 and so on. For the email corpus, we will give you a file, enron.vocab in the same format as imdb.vocab.

**Smoothing, and common, rare and unknown tokens:** For the machine learning toolkits, you don't need to worry about smoothing or tokens that appear in testing but not training. For your Naive Bayes classifier, you will need to worry about these issues. The reference solution written by the TAs will be using add-one smoothing on the labeled data. For tokens unique to the unlabeled (testing) data, the reference solution will simply ignore these tokens (i.e., pretend they did not occur). There are more sophisticated methods of smoothing and handling unknown tokens that we will discuss in class and on piazza but the TAs will be using the above approach.

The reference solution will not be taking into account the frequency of tokens. All tokens in the labeled data will be considered. However, you have the freedom to remove high frequency and low frequency tokens from consideration if you think this will improve your performance.

For your naive Bayes classifier, you will use the following representation. For training and development data, the first item will be a label for the example. For spam detection, it will be either SPAM or HAM where HAM refers to not spam. For sentiment detection, it will be either POSITIVE or NEGATIVE. Following the label will be a list of features and values for that example. The features correspond to the tokens present in the example which will be represented by their unique identifiers and listed in increasing order. The value of a feature will be the count of how many times that token appears in the example. The feature and value should be separated by ":".

So for a movie review, each feature would correspond to a token followed by ":" followed by the count of that token in the movie review. If "1" corresponded to "the", then "1:38" means that "the" appeared 38 times. If "2" corresponded to "a", then "2:10" means that "a" appeared 10 times.

The next line in the data file will correspond to a different example. It may have a different number of features since it may contain different tokens. For test data, the format will be the same except that there will be no LABEL.

```
FORMAT OF TRAINING/DEVELOPMENT DATA:

LABEL FEATURE_NAME1:FEATURE_VALUE1 FEATURE_NAME2:FEATURE_VALUE_2 ... FEATURE_NAME_Q:FEATURE_VALUE_Q
```

### Spam Data

On Blackboard, we will post two sets of data: labeled and unlabeled (test) as well as enron.vocab. All data has already been cleaned up (leaving only the text parts of the subjects and bodies) and tokenized with tokens separated by spaces.

All email data will be stored as gzipped tar files. You'll need to **gunzip** the files and then untar them **tar xvf**. The archives contain a large number of individual files (one per email) divided into subfolders "ham" and "spam". You will need to write a Python script to convert these large number of individual files into a single file in the Project Data format.

### Sentiment Analysis Data

On Blackboard, you will find labeled and unlabeld data as well as a vocabulary list. Use gunzip to uncompress the data. The original data are IMDB reviews associated with ratings either >= 7 (positive) or <= 4 (negative). The vocabulary list defines the 89527 unique tokens in the corpus (one per line of the file). The data is almost in the Project Data format but the identifiers start at 0. So you'll have to go through the file and add one to each.

We are not providing the reviews themselves. But you can write a Python script to tranform the features in the labeled data from numbers into the text of the associated tokens for debugging purposes.

## Tools you need to build / install

### Naive Bayes classifier in Python

- You must not use any additional packages, libraries or modules. Just the standard installation.
- You will create two scripts. nblearn.py will learn a model from training data. nbclassify.py will use a model to classify new text.

```
TRAININGFILE and TESTFILE will be in the Project Data Format.
We should be able to run your scripts in the following manner:

python3 nblearn.py TRAININGFILE MODELFILE

python3 nbclassify.py MODELFILE TESTFILE
```

The format for MODELFILE is up to you but should contain sufficient information for nbclassify.py to process a file in the Project Data Format and for each line (example) print to STDOUT the name of the more probable class (one per line).

### SVM-Light toolkit

You'll need to go to the SVM-Light website to get the toolkit and find documentation. Under the section "Source Code and Binaries", grab the binary for 64-bit Linux. Because you are grabbing the binary, you do not need to follow the "Installation" instructions; these are for users who have downloaded the source code.

In this case, the data must be formatted as follows. TARGET is either +1 or -1. The SVM will classify examples as either positive (+1) or negative (-1).

```
TARGET FEATURE1:VALUE1 FEATURE2:VALUE2 ... FEATUREQ:VALUEQ
```

For the sentiment analysis, the only change needed to the training data is converting POSITIVE to +1 and NEGATIVE to -1. For the test data, you will need to assign a target to each example since the software requires it (it reports accuracy) but you can put either +1 or -1, ignore the reported accuracy and just focus on the predictions which the model outputs.

For spam detection, you'll have to replace the target SPAM with +1 and the target HAM with -1. For the test data, you'll have to add targets (+1 or -1) to satisfy the data format.

Note, you'll need to post-process the SVM-Light output file so that +1 is transformed back to SPAM or POSITIVE, and -1 is transformed back to HAM or NEGATIVE.

### MegaM toolkit

You'll need to go to the MegaM website to get the toolkit and find documentation. Under the section "Code and Executables", you'll find an entry for Linux. Grab the optimized version.

In the documentation, under "File Formats", you will find that for a binary classification problem with token count features, you want "non-bernoulli implicit" format where CLASS is 0 or 1.

```
CLASS FEATURE1 VALUE1 FEATURE2 VALUE2 ... FEATUREQ VALUEQ
```

So for training and test data you'll need to:

- transform your "feature:value" formats to the space delimited format above.
- transform SPAM and POSITIVE labels to 1.
- transform HAM and NEGATIVE labels to 0.
- invent labels for the test data to satisfy the file format.

You'll then need to post process the output back to the original class names (i.e., SPAM, HAM, POSITIVE, NEGATIVE).

## What to turn in:

Remember, we are downloading your entire Bitbucket repository. DO NOT put any of the input data in this repository. Instead, this is what should be in your repository. You need to use exactly the same names as found in the list below. Also, put everything in the top level directory. Do not create subdirectories.

- report.txt
- Python code: nblearn.py, nbclassify.py, and any supporting code such as code to convert different data formats. For the supporting code, please be clear about what each file does. Include this information in report.txt.
- Model files generated from all the labeled data:
    - spam.nb.model, sentiment.nb.model
    - spam.svm.model, sentiment.svm.model
    - spam.megam.model, sentiment.megam.model
- Output files generated by running these models on the test data
    - spam.nb.out, sentiment.nb.out
    - spam.svm.out, sentiment.svm.out
    - spam.megam.out, sentiment.megam.out

# Grading

The grade will be calculated as described below. Performance is in comparison to a reference solution created by the teaching assistants following the instructions of the assignment. The comparison is numeric (i.e., difference in F1 score between your work and the reference solution).

- 30% of the grade is based on the performance of your Naive Bayes classifier on the SPAM/HAM test set.
- 30% of the grade is based on the performance of your Naive Bayes classifier on the POSITIVE/NEGATIVE test set.
- 15% of the grade is based on the performance of your SVM-light/MegaM model on the SPAM/HAM test set.
- 15% of the grade is based on the performance of your SVM-light/MegaM model on the POSITIVE/NEGATIVE test set.
- 10% of the grade is based on the report document.

# Late Policy

We will immediately attempt to download assignments after the deadline. We will grade the files we find. Any submissions after this deadline and before October 5, 2015 (10:59 AM PST) will be graded with a 30% penalty (i.e., best score will be 70/100). After this second deadline, you will receive 0 points for this assignment.

# Other Rules

- DO NOT look for any kind of help on the web outside of the Python documentation.
- DO discuss issues on Piazza
- This is an individual assignment. DO NOT work in teams or collaborate with others. You must be the sole author of 100% of the code and writing that you turn in.
- DO NOT use code you find online or anywhere else.
- DO NOT turn in material you did not create.
- Failing to follow these rules will result in a grade of zero. All cases of cheating or academic dishonesty will be dealt with according to University policy.