Name: _____
CS159: Natural Language Processing
Sentiment Analysis Lab
Apr 27, 2015

Get Ready

Today, you'll get a chance to try your hand at sentiment analysis of tweets. We'll be using labeled data from the 2015 SemEval shared task (`http://alt.qcri.org/semeval2015/task10/`).

Before you start, read the first three sections of Chapter 6 of the NLTK book, which talks about how to do classification in NLTK: `http://www.nltk.org/book/ch06.html`.

Next, copy the starter files into a working directory on knuth:

```
cd /my/working/directory
cp -r /www/courses/current/cs159/resources/semeval .
cd semeval/
```

Data Analysis

Before you do any classification, take a minute to look at the format of the data files. In particular, look at the files `training_tweets.tsv` and `dev_tweets.tsv`. Make sure that you understand the type of data that they contain.

To know how well our algorithms work, it can be helpful to compare to a baseline algorithm.

For classification, one baseline algorithm is the *random chance* baseline. For this baseline, we assume that each label is equally likely. Based on the `training_tweets.tsv` file, what would the random chance baseline do on this task? What would you expect its accuracy to be on `dev_tweets.tsv`? Why?

■

Another baseline is the *majority class* baseline. For this baseline, we label every instance with the most frequent label from our training data. Based on the `training_tweets.tsv` file, what would the majority class baseline do on this task? What would its accuracy be on `test_tweets.tsv`? Why? Would the most common class be any different if we trained on `dev_tweets.tsv`? Does this surprise you? Why or why not?

■

Classification

Now we're ready to start classifying the tweets. The starter file `sentiment_classifier.py` tokenizes tweets using the `twokenize` package, and classifies each using a Naive Bayes classifier.

The starter code uses very simple (and not very informative) bag-of-words features. Each tweet is represented as a set of binary features, one per word in our vocabulary. The vocabulary is defined to be the 10 most common tokens

in the training set. You can most certainly come up with features that will result in better classification, but this should give you the structure to work in.

Your task is to improve the classifier, so that it trains on the training data and gets better accuracy on the test data. The first thing you'll probably want to do is to improve the `sentiment_features` function to return more informative features. You can think about how to better-define the bag-of-words features (the discussion in the NLTK chapter may give you some ideas), and/or apply your own world knowledge to identify features you think will be useful in distinguishing between positive, neutral, and negative tweets.
Once you have features that you think are more informative, you might also try out one of the other classifiers included in NLTK. In addition to NaiveBayes, there are implementations of Decision Trees, a Maximum Entropy classifier, and possibly others.
Report here on the features that you used, and on the best accuracy you were able to achieve on the development set.

■