NLP Lab Session Week 8
Constructing Feature Sets for Sentiment Classification in the NLTK
Part 2: Adding Features from a Sentiment Lexicon

Continuing our session with the movie review sentences

Sentiment Lexicon: Subjectivity Count features

We will first read in the subjectivity words from the subjectivity lexicon file created by Janyce Wiebe and her group at the University of Pittsburgh in the MPQA project. Although these words are often used as features themselves or in conjunction with other information, we will create two features that involve counting the positive and negative subjectivity words present in each document.

Copy and paste the definition of the readSubjectivity function from the Subjectivity.txt file. We'll look at the function to see how it reads the file into a dictionary.

Create a path variable to where you stored the subjectivity lexicon file. Here is an example from my mac, making sure the path name goes on one line:
nancymacpath =
"/Users/njmccrac1/AAAdocs/research/subjectivitylexicon/hltemnlp05clues/subjcluesl
en1-HLTEMNLP05.tff"

Create your own path for your computer:
SLpath = <put the path here>

Now run the function that reads the file. It creates a Subjectivity Lexicon that is represented here as a dictionary, where each word is mapped to a list containing the strength, POStag, whether it is stemmed and the polarity. (See more details in the Subjectivity.py file.)
SL = readSubjectivity(SLpath)

Now the variable SL (for Subjectivity Lexicon) is a dictionary where you can look up words and find the strength, POS tag, whether it is stemmed and polarity. We can try out some words.
SL['absolute']
SL['shabby']
Or we can use the Python multiple assignment to get the 4 items:
strength, posTag, isStemmed, polarity = SL['absolute']

Now we create a feature extraction function that has all the word features as before, but also has two features 'positivecount' and 'negativecount'. These features contains counts of all the positive and negative subjectivity words, where each weakly subjective word is counted once and each strongly subjective word is counted twice. Note that this is only one of the ways in which people count up the presence of positive, negative and neutral words in a document.

```
def SL_features(document, SL):
        document_words = set(document)
```

```
    features = {}
    for word in word_features:
            features['contains(%s)' % word] = (word in document_words)
# count variables for the 4 classes of subjectivity
weakPos = 0
strongPos = 0
weakNeg = 0
strongNeg = 0
for word in document_words:
    if word in SL:
        strength, posTag, isStemmed, polarity = SL[word]
        if strength == 'weaksubj' and polarity == 'positive':
            weakPos += 1
        if strength == 'strongsubj' and polarity == 'positive':
            strongPos += 1
        if strength == 'weaksubj' and polarity == 'negative':
            weakNeg += 1
        if strength == 'strongsubj' and polarity == 'negative':
            strongNeg += 1
        features['positivecount'] = weakPos + (2 * strongPos)
        features['negativecount'] = weakNeg + (2 * strongNeg)
    return features
```

Now we create feature sets as before, but using this feature extraction function.

```
SL_featuresets = [(SL_features(d, SL), c) for (d,c) in documents]

# features in document 0
SL_featuresets[0][0]['positivecount']
7
SL_featuresets[0][0]['negativecount']
2
SL_featuresets[0][1]
'pos'
train_set, test_set = SL_featuresets[1000:], SL_featuresets[:1000]
classifier = nltk.NaiveBayesClassifier.train(train_set)
print nltk.classify.accuracy(classifier, test_set)
```

In my random training, test split, these particular sentiment features did improve the classification on this dataset. But also note that there are several different ways to represent features for a sentiment lexicon, e.g. instead of counting the sentiment words, we could get one overall score by subtracting the number of negative words from positive words, or other ways to score the sentiment words. Also note that there are many different sentiment lexicons to try.