**TEXT CLASSIFICATION FOR SENTIMENT ANALYSIS – NAIVE BAYES CLASSIFIER**

Sentiment analysis is becoming a popular area of research and social media analysis, especially around user reviews and tweets. It is a special case of text mining generally focused on identifying opinion polarity, and while it’s often not very accurate, it can still be useful. For simplicity (and because the training data is easily accessible) I’ll focus on 2 possible sentiment classifications: positive and negative.

NLTK Naive Bayes Classification

NLTK comes with all the pieces you need to get started on sentiment analysis: a movie reviews corpus with reviews categorized into pos and neg categories, and a number of trainable classifiers. We’ll start with a simple Naïve Bayes Classifier as a baseline, using boolean word feature extraction.

Bag of Words Feature Extraction

All of the NLTK classifiers work with featstructs, which can be simple dictionaries mapping a feature name to a feature value. For text, we’ll use a simplified bag of words model where every word is feature name with a value of True.

**Training Set vs Test Set and Accuracy**

The movie reviews corpus has 1000 positive files and 1000 negative files. We’ll use 3/4 of them as the training set, and the rest as the test set. This gives us 1500 training instances and 500 test instances. The classifier training method expects to be given a list of tokens in the form of [(feats, label)] where feats is a feature dictionary and label is the classification label. In our case, feats will be of the form {word: True} and label will be one of ‘pos’ or ‘neg’. For accuracy evaluation, we can use accuracy with the test set as the gold standard.

**####Training and Testing the Naive Bayes Classifier ###example1**

As you can see, the 10 most informative features are, for the most part, highly descriptive adjectives. The only 2 words that seem a bit odd are “vulnerable” and “avoids”. Perhaps these words refer to important plot points or character development that signify a good movie. Whatever the case, with simple assumptions and very little code we’re able to get almost 73% accuracy. This is somewhat near human accuracy, as apparently people agree on sentiment only [around 80](http://intelligent-enterprise.informationweek.com/channels/business_intelligence/showArticle.jhtml;jsessionid=CDFC5V5I1WXU5QE1GHPCKH4ATMY32JVN?articleID=224200667)% of the time. Future articles in this series will cover [precision & recall metrics](https://streamhacker.com/2010/05/17/text-classification-sentiment-analysis-precision-recall/), alternative classifiers, and techniques for improving accuracy.

**TEXT CLASSIFICATION FOR SENTIMENT ANALYSIS – PRECISION AND RECALL**

Accuracy is not the only metric for evaluating the effectiveness of a classifier. Two other useful metrics are precision and recall. These two metrics can provide much greater insight into the performance characteristics of a binary classifier.

Classifier Precision

Precision measures the exactness of a classifier. A higher precision means less false positives, while a lower precision means more false positives. This is often at odds with recall, as an easy way to improve precision is to decrease recall.

Classifier Recall

Recall measures the completeness, or sensitivity, of a classifier. Higher recall means less false negatives, while lower recall means more false negatives. Improving recall can often decrease precision because it gets increasingly harder to be precise as the sample space increases.

F-measure Metric

Precision and recall can be combined to produce a single metric known as F-measure, which is the weighted harmonic mean of precision and recall. I find F-measure to be about as useful as accuracy. Or in other words, compared to precision & recall, F-measure is mostly useless, as you’ll see below.

Measuring Precision and Recall of a Naive Bayes Classifier

The NLTK metrics module provides functions for calculating all three metrics mentioned above. But to do so, you need to build 2 sets for each classification label: a reference set of correct values, and a test set of observed values. Below is a modified version of the code from the previous article, where we trained a Naive Bayes Classifier. This time, instead of measuring accuracy, we’ll collect reference values and observed values for each label (pos or neg), then use those sets to calculate the precision, recall, and F-measure of the naive bayes classifier. The actual values collected are simply the index of each feature set using enumerate.

**###example2**

Nearly every file that is pos is correctly identified as such, with 98% recall. This means very few false negatives in the pos class.

But, a file given a pos classification is only 65% likely to be correct. Not so good precision leads to 35% false positives for the pos label.

Any file that is identified as neg is 96% likely to be correct (high precision). This means very few false positives for the neg class.

But many files that are neg are incorrectly classified. Low recall causes 52% false negatives for the neg label.

F-measure provides no useful information. There’s no insight to be gained from having it, and we wouldn’t lose any knowledge if it was taken away.

**Improving Results with Better Feature Selection**

One possible explanation for the above results is that people use normally positives words in negative reviews, but the word is preceded by “not” (or some other negative word), such as “not great”. And since the classifier uses the bag of words model, which assumes every word is independent, it cannot learn that “not great” is a negative. If this is the case, then these metrics should improve if we also train on multiple words, a topic I’ll explore in a future article.

Another possibility is the abundance of naturally neutral words, the kind of words that are devoid of sentiment. But the classifier treats all words the same, and has to assign each word to either pos or neg. So maybe otherwise neutral or meaningless words are being placed in the pos class because the classifier doesn’t know what else to do. If this is the case, then the metrics should improve if we eliminate the neutral or meaningless words from the feature sets, and only classify using sentiment rich words. This is usually done using the concept of information gain, aka mutual information, to improve feature selection.