Deceptive Hotel Review Detection Assignment 3

Rahat Ibn Rafiq

Universty of Colorado Boulder

1 Introduction

The assignment provided two datasets for both true and false reviews, each with 214 instances. The task was to design a classifier to detect whether a review is true or false, in other words, deceptive.

1.1 Cross Validation

Because there was no test set, I applied 5-fold cross validation on the dataset. Which means for each turn, 42 true and 42 false reviews were set as test data and the other reviews were fed to the designed classifier for training purposes. By keeping track of the test dataset and train dataset, it was made sure that a review appears in the test dataset only once.

1.2 Classifier version 1

At the very first step, the classifier that was used for assignment two was applied to this dataset. Only preprocessing that was applied was removing non-alpha numeric characters. It gave an accuracy of around 65 percent. By applying bare word stems it gave an accuracy of around 55 percent. Thinking Naive bayes may not be an appropriate solution to this , I moved on to other classifiers and other features. Here it also should be mentioned that i applied several values for the smoothing, starting from 0.0001 to 1.

1.3 Other classifer performances

Here I applied Random Forest, Perceptron and kNN classifiers to the dataset. The features that were involved were bare unigrams with only special characters removed, parts of speech tags of the words, bare word stems, top words by counting the tfidf of the reviews and bigrams. As the table shows, these whole bunch of work failed to yield any promising results, much to my dismay and frustration. A lot of combinations were tried and only a few is shown in the table. The idf feature was applied in this way: the idf value for a word determines how common a word is across a list of documents. So I calculated the idf value for each words appearing in to true and false review datasets and taken only those words as features whose idf values differ by a reasonable amount across the true and false dataset. I also used the top words for the documents as computed by tfidf value but it failed to show any improvement.

For the nContaining features, it was computed like this: ncontaining is the value that returns the number of documents that contains a particular word. So in this feature, i took the words for which the difference of the number of true documents that contains that word and the number of false documents that contain that word is at least a reasonable amount. For that reasonable amount, i tried integers from 1 to 10. It was seen that when the difference is set to 1, it gave the best result and that is shown in the table.

Features	Random Forest	Perceptron	kNN
bare unigrams	53	52	52
bigrams and bare word stems	56	51	52
unigrams and bigrams	56	51	51
unigrams and bigrams and word stems	56	55	51
bigrams	52	50	48
idf words	55	53	46
nContaining words	65	67	51
POS tags and word stems	54	55	49
unigrams and word stems	51	52	40

1.4 Back to the Naive Bayes

So, with a losing heart, i went back to my assignment two classifer and applied the bag of words but this time took the bare stems of those words combined with the nContaining feature explianed in the previous section. This slightly improved the system's performance, giving around 60 percent accuracy. Still not good enough. The best performance was for Perceptron classifier with 67 percent accuracy with nContaining words feature.

1.5 packages used

python scikit, nltk, textblob, sklearn. Other classifers like Adaboost, Extra-TreeClassifer, RidgeClassifer were also used but they didnt give any promising results.