**Data Wrangling Assignment**

Part B

Myles Neilson

[mylesneilson@gmail.com](mailto:mylesneilson@gmail.com)

Matric Number: 98100378

**The model is based on Pandas and NLTK packages**

**Pre-processing**

For the model I imported the data using Pandas. After this I created a function to extract the word features. As rows are separated by newlines it was fairly simple to split the data into sentences, put to lower case, then tokenize the words in the sentences using NLTK word tokenise.

I created another function with further pre-processing, this included using NLTK stop word remover and NLTK porter stemmer. Adding this pre-processing did improve the accuracy of the model it is therefore retained in the final model.

The function then returned a simple dictionary which maps the words in the sentence to ‘True’ if the word exists in the data, these are the features for the model. I also renamed the columns of the data frame to make the inputs for the model more obvious.

**Model selection**

I decided to use Naïve Bayes as my classifier. It is relatively simple to implement, NB can also train quickly, has no tuneable hyper-parameters and is also widely used in text classification.

Applying the model

The sentiment column of the data frame contained 0 for negative sentiment and 1 for positive sentiment these were split into two lists labelled ‘neg’ and ‘pos’.

Then the words from each sentence were extracted as negative (neg reviews) and positive (pos reviews) features. We now have two dictionaries of negative and positive feature-label pairs.

These reviews have to be combined into one dataset to train the model. The reviews are then put into random order so that training can be more effective. The dataset was then split into training and test data. I used a 80%/20% training/test split, this seemed to be a common split in the literature for model training. This was done by dividing the dataset at the record that applied to 80% of the data, in this case 5534.

The Naïve Bayes Classifier was then applied to the training data, and this was assigned to a classifier variable.

**Model evaluation**

I initially used NLTK, nltk.classify.util.accuracy(classifier, test) for accuracy evaluation.

This is a simple metric to evaluate a classifier. Accuracy measures the percentage of inputs in the test set that the classifier correctly labelled.

I also investigated the metrics of Precision, Recall and F-measure. These measures make more inquiry into how relevant the results of the classifier are.

To implement these metrics I had to create two sets; reference sets and test sets. Reference sets will contain the actual values for the testing data (which we know because the data is pre-labelled) and test Sets will contain the predicted output.

For each one of the testFeatures (the reviews that need testing), a function was created to iterate through three things: an arbitrary ‘i’, so be used as an identifier, and then the features (or words) in the review, and the actual label (‘pos’ or ‘neg’).

This creates a list of identifiers in referenceSets[‘pos’], which are the reviews known to be positive (and the same for the negative reviews). It also gives a list of identifiers in testSets[‘pos’], which are the reviews predicted to be positive (and similarly for predicted negatives). What this allows comparison of these lists and to see how well the predictor did.

**Results**

The results from all metrics were extremely positive.

|  |  |
| --- | --- |
| Accuracy | 0.9768618944323934 |
| pos precision | 0.9744245524296675 |
| pos recall | 0.9844961240310077 |
| pos F-measure | 0.9794344473007711 |
| neg precision | 0.9800332778702163 |
| neg recall | 0.9671592775041051 |
| neg F-measure | 0.9735537190082646 |

The high precision value (also called positive predictive value), means there was a high percentage of relevant instances among the retrieved instances. The high recall value (also known as sensitivity) means there was a high percentage of relevant instances that have been retrieved over the total amount of relevant instances. F-measure combines precision and recall is the harmonic mean of precision and recall.

**Suggestions for improvement**

One possible area to investigate improvements would be in pre-processing. The literature on Sentiment Analysis and improving prediction often uses forms of negation handling.

For instance I could have applied ‘NOT\_’ to every word occurring after the negation words (Agarwal et al 2013) and see if there were improvements on classification. Another popular element of pre-processing is to extract not only unigram features but also bi-grams and tri-grams to improve performance (Parmar et al. 2014)

Other area to look to would be word frequency. I could have tried using a bag of words representation of the data and looked at feature weighting. Frequency-based schemes, such as term frequency (TF), term-frequency-inverse document frequency (TF-IDF.) are often referred to in the literature but often binary weighting (Pang et al. 2002) is regarded as better for sentiment analysis.

Finally I could have used cross fold evaluation of the model. Often in the literature a K fold cross validation method is used, looping through all the data to get a more representative value of overall accuracy it is claimed.