COM6513 Lab 2: Text Classification
Registration Number: 160122440

1. INTRODUCTION

The aim of this lab session is to implement one of the oldest and most successful classification learning algorithms, the perceptron.
The algorithm was implemented with three functions: a function to create the vector representation of the documents (bag of words), a function for training the perceptron, and finally a function for testing.
Two-thousand (2000) documents were provided, 1000 for positive reviews and another 1000 for negative reviews. The first 800 documents of each were used as the training set (total 1600) and the last 200 were used for testing (total 400). A matrix was created for both the training and testing datasets. An additional vector was added to the last column of the matrices to manually classify documents as positive and negative (positive = +1, negative = -1). These signs were used to compare the predicted classification with the actual classification of the documents, during the training procedure. If prediction was wrong and the original document classification was positive (1), then the positive weights were benefited and the negative weights were penalised. If prediction was wrong and the original document classification was negative (-1), then the negative weights were benefited and the positive weights were penalised.
The algorithm was first created without using averaging, multiple passes or shuffling. Therefore, the clasification accuracy provided by the perceptron was only 0.5.

2. IMPROVEMENTS

Three improvements were implemented: multiple passes, randomising the order of the training instances.

Multiple passes was implemented by inserting the whole training function in a *for* loop, which caused the function to repeat the procedure for 100 iterations. In order to visualise the effect of this improvement, the training accuracy was calculated and plotted in a graph. The benefit for this improvement is that as the algorithm repeats the training procedure, the accuracy is improved in every iteration. The graph can be seen below:

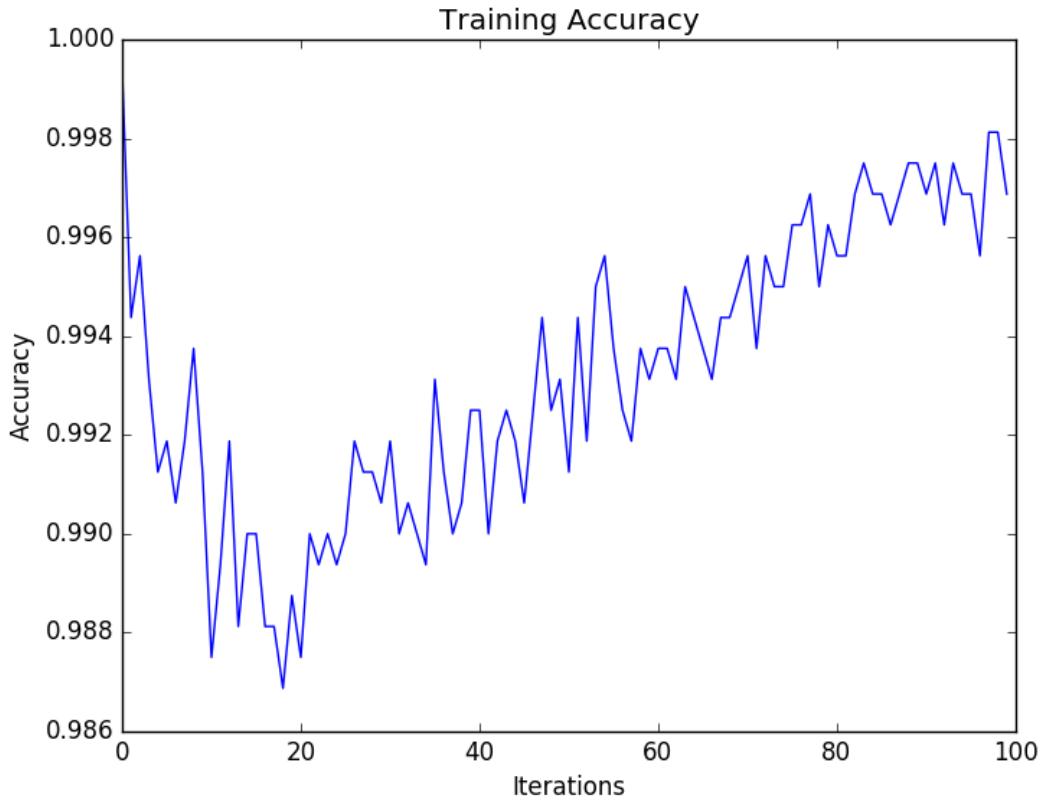

Figure 1: Learning progress - Training accuracy

In order to randomise the order of the training instances, the *random* function was imported and applied to the training matrix as: $random.shuffle(matrix)$. The benefit of random sampling is that it is 'safer' to trust the results, since the data partitions created came from different sources (in our case we had positive and negative sources). Therefore, training without shuffling is more likely to provide worst results, since the algorithm will be trained with firstly 800 consecutive positive reviews and then 800 consecutive negative reviews, instead of being trained with shuffled positive and negative reviews.

The final improvement was averaging, where the average was found by calculating the sum of the weights and dividing with the total number of iterations (counter). Averaging provides a better overall representation of the the weights.

3. EVALUATION

After the improvements the final accuracy was improved to 0.7925, which is a significant improvement from 0.5. Therefore, all three improvements help to increase the accuracy.

The most positively-weighted features for each class were then found, in order to visualise whether or not they seem sensible; which in fact they do. Some positive words such as 'great' and 'fun' appear for the positive class, and some negative words like 'boring' and 'worst' appear for the negative class. These can be shown in the following tables:

features:	american	comic	job	sometimes	seen	bond	human	great	quite	fun
weight:	36.6	37.6	37.6	37.6	38.6	40.6	40.6	49.5	49.5	51.5

Table 1: Most positively-weighed features for the positive class.

features:	mess	stupid	nothing	given	script	boring	supposed	worst	unfortunately	bad
weight:	45.5	46.5	47.5	49.5	51.5	57.4	57.4	64.4	67.3	73.3

Table 2: Most positively-weighed features for the negative class.

4. FEATURES

If the classifier was applied to a different domain such as laptop or restaurant reviews, then it will not probably generalise well. This is because most of the features for either movies or restaurant reviews will be different than the ones for the movies. As an example, 'comic' or 'script' features cannot be used to describe a laptop or a restaurant. On the other hand, some features such as 'great and 'worst' can be used for these cases, but still the algorithm will not generalise well.
Some better features for laptop reviews would be 'fast', 'slow', 'heavy', 'size', 'weight'.
Finally, a few better features for restaurants would be 'delicious', 'clean', 'variety', 'expensive', 'cheap'.