COM3110: Sentiment analysis Report

Salma Hassan

December 9, 2022

1 Implementation

To implement a complete workflow for sentiment analysis of movie reviews using Naive Bayes classification several stages had to be carried out.

Firstly all the data had to be pre-processed, this is done in the PreProcess.py file. for all data sets (training,dev,test) lower casing and punctuation removal were performed. An additional pre-processing step removing stop words (common words e.g. its, am and or) was also carried out on all the data sets.

The next stage was to extract the words that were the most representative of the phrase. Several methods of feature extraction were experimented with. The final implementation was to extract features based on word type. More information will be provided in the experimentation section.

In this stage the Naive Bayes model was trained using the train data, this is done in the NaiveBayesClassifier.py file. The aim of the Naive Bayes model is to assign a phrase the class with the highest posterior probability. The posterior probability for a phrase is the product of the posterior probability for each word. The posterior probability for a word is calculated using equation 1

$$p(s_i|T) = \frac{p(T|s_i) * p(s_i)}{p(T)} \tag{1}$$

Where T is the text, s_i is the class. In my implementation of The Naive Bayes Model, slight modifications where made to the equation. This will be explained further in section 2

In the Evaluation stage the performance of the model was evaluated, this was done in the Evaluate.py file. To evaluate the performance of the model the f1 metric was used and and the f1 across all classes was averaged to produce the macro f1 score. The equation 2 was used to calculate the f1 score for a class.

$$F1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{2}$$

2 Naive Bayes Modification

When using equation 1 to calculate the posterior probability for a word, the performance of the classification was quite poor. Upon further inspection of the confusion matrices it became quite apparent that the model was really bad at making predictions for the neutral class (class 1) when the number of classes was 3 and the negative, neutral and positive classes (classes 0,2,4) when the number of classes where five. The reason behind the models negative performance was to do with the distribution of classes within the data. The train data set is unbalanced because the number of phrases for each class vary greatly which will affect the calculation of the prior probability e.g. when the number of classes is 3 the word count for each class 0,1 and 2 are 0.38, 0.2, 0.42 respectively, this means the posterior probability for words in class 1 will be a lot smaller compared to other classes since prior probability is half as small. This causes the classifier to be biased. To combat this problem I've excluded the prior probability from the calculation. The effect is shown in figure 1

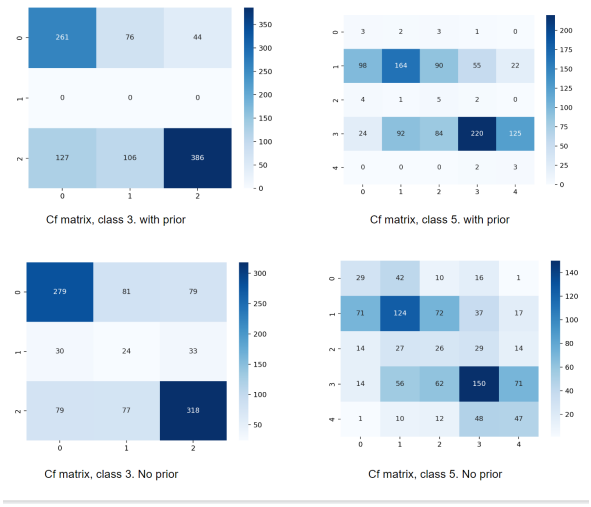

Figure 1: Confusion matrices with and without the prior probabilities

3 Experimentation

To find the model with the highest performance i experimented with different methods of feature extraction.

The first methods extracts words based on their type. For example if i specified nouns only nouns would be extracted from the train data and the classifier would be trained on the modified train data. The subset of word types that were most effective were adjectives,nouns,verbs and adverbs

The next two methods filtered words based on whether they had polarity or not and whether a word was subjective, has a subjectivity score that was not zero.

The last method got the ranked words based on their subjectivity score and created a set with the top 74% highest scoring words. The train data was filtered and only words present in the top scoring set were kept.

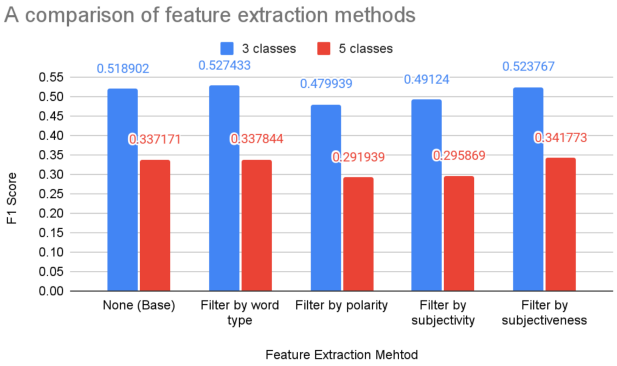

Figure 2: Comparison of feature extraction methods

4 Conclusion

Based on figure 2 the extraction method with the highest performance on 3 classes was filtering by word type and the method with the highest performance on 5 classes was filtering by suggestiveness. The best overall model based on the combination percentage change from the base results for both classes if filtering by word type