Assignment 2

1604827
NLP CS918

December 9, 2019

Preprocessing

I start by making a preprocessing function that removes URLs first. It then separates any hyphenated words into the separate words. As the usernames of people on twitter shouldn't affect the accuracy I remove all '@. . . ' which correspond to usernames. I also remove any hash tags leaving the hash tagged word as they do carry useful information about the sentiment. I then remove all non-alphanumeric characters and any number i.e.1281 and any single letter word. I then remove all stopwords as these don't contain useful information for sentiment analysis. I then lemmatize each word and return a list of dictionaries each containing the id, sentiment and the processed text.

Naive Bays classifier

In building Naive Bays model this model will be trained and tested on unigrams. I start by creating the a set of unique words in the training data. I then count the appearance of each word in the training data given its sentiment i.e. if the word 'great' is mentioned in a positive tweet its added to the count of the word 'great' in the positive counter. Assign each word in each vector a probability, this is done according to the formula:

$$\text{Probability(word)} = \frac{\text{occurrence of word} + 1}{\text{sentiment count} + \text{number of unique words}} \tag{1}$$

Where sentiment count is the number of words in a given sentiment class, in other words the total number of words in all the positive/negative/neutral tweets. The plus one is used to assign a probability to words that haven't been seen in a certain sentiment class.

The model has now been trained and can now be used to classify the testing data. For each tweet in the testing data the probability of it being a positive, negative or neutral tweet given the words it contains. For each word the probability of it appearing in a sentiment class is looked up in the dictionary of words that have been developed in the training model. If a word hasn't been seen in the test data its probability is given to be the minimum value of a word in each sentiment class.

Max entropy classifier

For the maximum entropy classifier I have made use of the sklearn library and in particular the logistic regression algorithm. I start by changing the sentiment of each of the training and test data into a numeric value. I then transform the tweets into vectors based on the occurrence of words using the count vectorise function, I then transform this data using term frequencyinverse document frequency. This is used to reduce the importance the model gives to words that appear a lot in the tweets such as for words like 'and'. The model then can be formed using logistic regression on multiple classes, I increase the number of iterations to make sure the model converges. The model can then be tested using the 'predict' package in sklearn, this returns a vector containing the predicted class as a number. I then simply convert the category number back to a sentiment and create a dictionary of each of the test tweet ids and the predicted sentiment.

LSTM classifier

To start I begin by processing the sentiment of the training, testing and developing data into a list of lists. The list contains lists of length 3 each having consisting of zeros in each place except one where it is a one this represents the sentiment, i.e. if the vector is [0,1,0] this may represent a neutral sentiment, I do this using the onehot-encode package.

I then go onto building a tokenized list of 5000 words based on the training data, and tern each of the data sets (test, training, developing) into a sequence based on the tokenization.

Next I build the embedding matrix which is a matrix that assigns each word a vector of length 100, based on the pre-trained GloVe file. The matrix is of 5001 by 100, that means there are 5000 words each being represented by a vector of length 100.

I then build the neural network model, I first add an embedding layer consisting of weights given by the embedding matrix and set its trainable variable to false as to not update the values given by the GloVe file. I then add a 'LSTM' layer and finally a dense layer. The dense layer takes in three categories and uses softmax as its activation. I then compile the model using 'rmsprop' optimizer and catagorical-crossentropy loss metric.

To train the model I use the tokenized training data with the onehot-encoded vectors. I make use of the development data in the model by using it as validation data to hopefully prevent over fitting. This model can be tested again by using the model predict attribute that produces a list of the predicted sentiment, this can then be converted back into a list of the sentiment and formed into a dictionary with the testing data ids.

Evaluation

Naive Bays classifier

When testing on the three data sets I obtained accuracies of 57.8%, 60.2%, 56.9% for test sets 1,2,3 respectively.

F1 scores are 0.549, 0.539, 0.528 for test sets 1,2,3 respectively.

The confusion matrices for test sets 1,2,3 are shown below, in order.

	positive	negative	neutral
positive	0.587	0.078	0.335
negative	0.177	0.492	0.330
neutral	0.254	0.150	0.595

	positive	negative	neutral
positive	0.645	0.066	0.289
negative	0.232	0.424	0.344
neutral	0.333	0.120	0.548

	positive	negative	neutral
positive	0.615	0.079	0.306
negative	0.255	0.402	0.343
neutral	0.281	0.136	0.583

The f1 scores for all three test sets are very similar suggesting that there it is unlikely that testing on new similar test sets will cause a huge change in accuracy. Looking at the confusion matrix for all three test sets it can be seen that, positive sentiments are classed correctly as positive or neutral most of the time with only a small number of cases a positive tweet is classed as negative. When looking at the other classes it seems that the positive class is being over predicted and that the negative class is being under predicted. Again in all examples the prediction of the negative class is predicted correctly less often than the other classes.

Maximum entropy classifier

When testing on the three data sets I obtained accuracies of 65.0%, 65.7%, 62.4% for test sets 1,2,3 respectively.

F1 scores are 0.552, 0.585, 0.542 for test sets 1,2,3 respectively.

The confusion matrices for test sets 1,2,3 are shown below, in order.

	positive	negative	neutral
positive	0.726	0.058	0.216
negative	0.147	0.751	0.102
neutral	0.254	0.154	0.592

	positive	negative	neutral
positive	0.755	0.048	0.197
negative	0.103	0.821	0.077
neutral	0.347	0.107	0.546

	positive	negative	neutral
positive	0.722	0.063	0.214
negative	0.137	0.691	0.171
neutral	0.305	0.137	0.558

The F1 scores for all three test are again very similar to each other and are slightly higher than than those of the previous classifier. Looking at the confusion matrices for all three tests they show that this models predicts the sentiment better than the previous method. Again the positive class appears to be over predicted, although the negative sentiment is a lot better predicted than in the previous classifier. The true negatives are predicted the best and substantially better than in the previous classifier.

LSTM classifier

When testing on the three data sets I obtained accuracies of 64.6%, 65.8%, 62.3% for test sets 1,2,3 respectively.

F1 scores are 0.554, 0.570, 0.527 for test sets 1,2,3 respectively.

The confusion matrices for test sets 1,2,3 are shown below, in order.

	positive	negative	neutral
positive	0.738	0.035	0.226
negative	0.146	0.701	0.153
neutral	0.266	0.160	0.574

	positive	negative	neutral
positive	0.765	0.031	0.204
negative	0.152	0.687	0.162
neutral	0.371	0.115	0.514

	positive	negative	neutral
positive	0.756	0.055	0.189
negative	0.176	0.581	0.242
neutral	0.322	0.132	0.546

The F1 score for all the test are about the same as each other, they are also about equal to the last tests F1 score. Again the model appears to be favouring the positive sentiment class for tweets in the other classes. This classifier predicts very similarly to the previous classifier the max entropy classifier. Although this classifier predicts the positive tweets the best followed by negative then neutral. As would be expected the neutral class is predicted the worst, this is because there are not as many words that associated with the positive or negative class so will be more difficult to classify.