Assignment Two: Sentiment classification for social media

CS918: 2020-21

Student ID: 2026619
University of Warwick
December 14, 2020

1 Problem Specification

The assignment is to build at least three sentiment classifiers based on the Task 4 of Semeval Competition 2017. In this report, all the classifiers will be presented along with the pre-processing used. I will also discuss about the model scores as well.

2 The Classifiers

For this assignment, I have chosen Naive Bayes, Support Vector Machine (SVM) and Maximum Entropy (MaxEnt)/Logistic Regression among the traditional based learning methods. For the recurrent neural network model, I used LSTM with a dense layer and glove embeddings.

3 Traditional Classifiers

In this section, I will discuss about the traditional classifiers that I have implemented in my project.

3.1 Preprocessing Twitter Data

The dataset is given in the format [id, emotion, tweet] where id is the tweet id, and emotion is the corresponding tweet emotion and tweet is the tweet sentence. So first, the tweet sentence needs to be processed. The tweet is processed using the following steps:

1. Transforming the tweet into lowercase

2. Using Regular Expression to remove/change phrases (Removing URLs, Username starting with '@', removing hashtags, removing ('s) after words, changing ('ve) to 'have', changing (n't) to 'not' and so on, removing everything except alphanumeric characters, removing words with only digits and removing whitespaces)
3. Tokenizing the sentence into words
4. Removing stopwords
5. Removing punctuations
6. Lemmatizing

Since these classifiers accepts string labels for classes, the labels do not need to be processed for these classifiers.

3.2 Bag-of-words and TF-IDF

After pre-processing the twitter data, I made the bag-of-words model of the train data using CountVectorizer from sklearn library. I used different maximum features to see which worked best. Finally chose 5000 features for naive bayes and kept the default for svm and logistic regression, as these values were giving good results. Also chose 30 as the minimum occurrence of any word in naive bayes classifier and 2 in svm and logistic regression. After that used sklearn's TfidfTransformer to extract features. It transforms a count matrix to a normalized tf or tf-idf representation. After these steps, I applied the following three classical algorithms for classification.

3.3 Naive Bayes Classifier

For naive bayes classifier, I chose sklearn's Multinomial Naive Bayes classifier which is suitable for text classification tasks such as this. The results of the classifier is shown in Table 2. The best f1 score among the testsets is 0.528. The confusion matrix for this testset is shown in Table 1. From the confusion matrix, we can see that the positive sentences were classified as positive 65% of times, negative sentences were classified as negative 55% of times and neutral sentences were classified neutral 57% of times. We can notice that the classifier has assigned neutral to positive and negative sentences for around 26% of times. This is because in our training data, the number of neutral sentences are much higher than the number of positive or negative sentences. This is why the classifier assigns majority of the values to neutral class.

twitter-test3.txt			
Confusion Matrix	Positive	Negative	Neutral
Positive	0.653	0.083	0.264
Negative	0.185	0.550	0.265
Neutral	0.282	0.142	0.576
Macroaveraged F1	0.528		

Table 1: Confusion Matrix of Naive Bayes Classifier

Overall Results			
Macroaveraged F1	Naive Bayes	SVM	MaxEnt
twitter-dev-data.txt	0.602	0.642	0.647
twitter-test1.txt	0.513	0.596	0.580
twitter-test2.txt	0.493	0.601	0.582
twitter-test3.txt	0.528	0.581	0.592

Table 2: Results of the Classifiers

3.4 Support Vector Machine

For the svm classifier, I chose sklearn's LinearSVC, or Linear Support Vector Classification. The results of the classifier is shown in Table 2. The best f1 score among the testsets is 0.601. The confusion matrix for this testset is shown in Table 3. From the result we can clearly see that SVM performed much better than Naive Bayes in classifying sentiments. Although it performed very well in the development set, in the test set, the accuracy reduced.

twitter-test2.txt			
Confusion Matrix	Positive	Negative	Neutral
Positive	0.636	0.064	0.301
Negative	0.221	0.552	0.227
Neutral	0.301	0.086	0.613
Macroaveraged F1	0.601		

Table 3: Confusion Matrix of SVM Classifier

3.5 MaxEnt/Logistic Regression

For the logistic regression, I chose sklearn's LogisticRegression model which is the same as Logit or Maximum Entropy Classifier. The results of the classifier is shown in Table 2. The best f1 score among the testsets is 0.591. The confusion matrix for this testset is shown in Table 4. Although it did not do better than the SVM classifier, but we can see that the result among the testsets were more consistent than other classifiers.

twitter-test3.txt			
Confusion Matrix	Positive	Negative	Neutral
Positive	0.605	0.086	0.309
Negative	0.136	0.563	0.301
Neutral	0.213	0.122	0.665
Macroaveraged F1	0.601		

Table 4: Confusion Matrix of MaxEnt Classifier

4 Recurrent Neural Network Classifier

For the neural network classifier, I used different preprocessing methods than the traditional classifiers. All the steps are given in the next subsections.

4.1 Preprocessing

For preprocessing and cleaning the data, I used the following steps for the neutral network model.

1. Removed hashtags in front of text and processed the hashtags if it was possible (if it was in a form where every first letter of a word was capitalize, for example: #BlackLivesMatter, #Football etc).
2. Used regular expressions for removing urls, '@', etc, same as before in classical algorithms
3. Removed stop words and words containing only 1 character

Here, we can see that I did not use lemmatizer because it did not help to improve the accuracy of the model.

4.2 Balancing Dataset (Oversampling)

The training dataset, as mentioned earlier, is not balanced. Meaning the number of sentences that are there for each label are not equal. In fact, neutral data count is a lot higher than positive and negative data counts which is why the classifiers tend to predict neutral more than the other two labels. So this is why I decided to oversample the dataset so that the weight of each label is distributed equally. I used a user-defined function called *extend_list* which turned every classes of labels equal by oversampling the smaller classes. This significantly improved the model predictions.

4.3 Preparing Training Set

To prepare the train sentences, I used Tokenizer from Tensorflow library, to fit the sentences and produce the vectors. Maximum number of vocabulary was set to 5001. Neural networks only accepts numerical values so for train labels, I used LabelEncoder (to convert the list of labels to integers) and OneHotEncoder (to convert the integers to binary encoded matrix).

4.4 Embedding Layer

For embedding layer, I used the glove embedding with dimension 100. I used the top most 5000 common words from the training set to convert them into weight vectors for the embedding layer. The values that were not in the glove embedding were assigned to vectors of zeros.

4.5 RNN Model

For the model, I used LSTM with 15 neurons. I tried different values of neurons and 15 seemed to give the best results. Also added a dense layer with softmax activation and 3 outputs since we have three class labels. For the loss function, used categorical_crossentropy and used RMSProp as the optimizer. Also tried other activation functions such as Sigmoid and optimizers such as Adam, but those did not perform well.

4.6 RNN Results

The results of the classifier is shown in Table 5. We can see from the table that RNN has performed a lot better than the other classifiers. A confusion matrix is shown for the best accuracy testset in 6. From the confusion matrix, we can see that the neural network performs a lot better in terms of finding out the positive and negative classes. It predicts the positive classes correctly 70% of times and predicts the negative classes correctly 55% of times. Even though the negative classes prediction is still not very good, overall, the neural network performed much better in sentiment classification of the twitter datasets.

Macroaveraged F1	Recurrent Neural Network
twitter-dev-data.txt	0.630
twitter-test1.txt	0.608
twitter-test2.txt	0.624
twitter-test3.txt	0.576

Table 5: Results of the RNN Classifiers

twitter-test2.txt			
Confusion Matrix	Positive	Negative	Neutral
Positive	0.707	0.050	0.244
Negative	0.250	0.553	0.197
Neutral	0.342	0.072	0.585
Macroaveraged F1	0.624		

Table 6: Confusion Matrix of RNN Classifier

5 Conclusion

For this assignment, we had to implement at least 3 classifiers for sentiment analysis in twitter data. I chose to do 4 different classifiers. From the results, it can be seen that naive bayes had a poor performance compared to other models. SVM and MaxEnt/Logistic Regression performed similarly on most of the test sets. For class labels, negative classes were not as accurate as the positive and neutral classes across all the classifiers. Even with oversampling, the negative classes performance was still not up to the standard. Regardless, the recurrent neural network using LSTM layer with a dense layer and with the glove embedding gave the best results among all the classifiers.