NLP - Assignment 1

u1925912

March 2023

1 The aim of the project

The aim of the project is to create a sentiment analysis model to classify a particular tweet as positive, neutral, or negative.

2 Preprocessing

Instead of passing raw inputs from tweets as the input data, all the tweets from the training, validation, and testing datasets were first prepossessed. The preprocessing was done in the following way:

1. The files containing the tweets were read and lowercase, with each line containing a different tweet. The tweet was in string format.
2. URL addresses were removed by removing the words starting with "http","ftp" or "www".
3. the user mentions (@) and hashtags (#) were removed. For example, for "#BlackFriday", the whole "#BlackFriday" phrase would be removed.
4. the string containing the tweet was then split into words, the result of which for every tweet, there would be a list containing every token used in the tweet.
5. Some words were in the contracted format (for example: "shouldn't", "would've"). They had to be converted to the standard forms ("should not","would not"). This was done using the module "contractions" and every word containing a contraction was replaced by its regular form.
6. each word was lemmatized using the Part-Of-Speech tagging. That means that every word was mapped to a particular part of speech (noun, verb, adjective, or adverb). This allowed to correctly lemmatize the words.
7. The lists were then converted back to a string.
8. All words starting with a digit were removed.

9. All one-letter characters were removed.

It was considered that the non-alphanumeric characters should also be removed but not removing them only improved the performance of the model. For that reason they were kept.

3 Naive Bayes, SVM, Logistic Regression

3.1 Bag of words

The first feature used as the model input is the bag of words. The bag of words takes the input string and transforms it into a vector. The vector consists of numbers representing the number of occurrences each token has appeared in the document. In Python, this is done by Scikit-Learn's CountVectorizer() function. The vectorizer was applied to all datasets.
As an additional feature, the term frequency–inverse document frequency was also implemented.

3.2 Naive Bayes

The first model that was implemented was Multinomial Naive Bayes. It is important to point out that the classification problem has 3 classes and thus cannot be solved by determining a single boundary between the classes. That is why Multinomial Naive Bayes had to be used. The evaluation of the performance and the generation of the confusion matrices were done using the functions provided in the skeleton file.
The following results were obtained:

```
twitter-test1.txt (NB): 0.512
twitter-test2.txt (NB): 0.475
twitter-test3.txt (NB): 0.476
```

with the corresponding confusion matrices:

	positive	negative	neutral
positive	0.610	0.081	0.309
negative	0.114	0.692	0.194
neutral	0.264	0.166	0.570

	positive	negative	neutral
positive	0.675	0.074	0.251
negative	0.217	0.478	0.304
neutral	0.338	0.127	0.536

	positive	negative	neutral
positive	0.631	0.088	0.281
negative	0.223	0.466	0.311
neutral	0.296	0.157	0.547

The average evaluation score for the three tests was 0.488. Additionally, the model was evaluated on the tf-idf features and the following results were obtained:

```
twitter-test1.txt (NB_tfidf): 0.380
twitter-test2.txt (NB_tfidf): 0.402
twitter-test3.txt (NB_tfidf): 0.369
```

with the corresponding confusion matrices:

	positive	negative	neutral
positive	0.615	0.088	0.297
negative	0.025	0.825	0.150
neutral	0.249	0.205	0.546

	positive	negative	neutral
positive	0.681	0.076	0.243
negative	0.125	0.625	0.250
neutral	0.325	0.145	0.531

	positive	negative	neutral
positive	0.632	0.107	0.261
negative	0.319	0.511	0.170
neutral	0.290	0.174	0.536

The average evaluation score for the tf-idf NB model was 0.384, which is significantly lower than the regular bag of words model. As a result, it was decided that only a regular bag of words model will be used for Support Vector Machine and Logistic Regression classifiers.

3.3 Support Vector Machine

The next model that was implemented was the Support Vector Machine. Because this is a multitask classification problem, the classifier used for it was OneVsRestClassifier was implemented. In this classifier one class is fitted against all the other classes. Effectively, this becomes multiple binary classification problems with the predicted class being the class with the most confidence in the predictions.
The following evaluation scores were obtained, as well as the corresponding confusion matrices:

```
twitter-test1.txt (svc): 0.543
twitter-test2.txt (svc): 0.555
twitter-test3.txt (svc): 0.476
```

	positive	negative	neutral
positive	0.785	0.054	0.161
negative	0.112	0.807	0.081
neutral	0.267	0.151	0.583

	positive	negative	neutral
positive	0.810	0.056	0.134
negative	0.104	0.806	0.090
neutral	0.349	0.103	0.548

	positive	negative	neutral
positive	0.763	0.079	0.157
negative	0.211	0.648	0.141
neutral	0.306	0.144	0.549

The model works better than the Multinomial NB one, with the average evaluation score being 0.525.

3.4 Logistic Regression

The third classifier used was Logistic regression. The following evaluation scores were obtained, as well as the corresponding confusion matrices:

```
twitter-test1.txt (log_reg): 0.559
twitter-test2.txt (log_reg): 0.571
twitter-test3.txt (log_reg): 0.536
            positive  negative  neutral
positive    0.691     0.060     0.249
negative    0.158     0.653     0.189
neutral     0.276     0.142     0.583

            positive  negative  neutral
positive    0.736     0.049     0.215
negative    0.178     0.645     0.178
neutral     0.364     0.104     0.532

            positive  negative  neutral
positive    0.711     0.066     0.223
negative    0.205     0.545     0.250
neutral     0.301     0.134     0.565
```

The average evaluation score for Logistic Regression was found to be higher than Support Vector Machine, with an evaluation score of 0.555.

4 Embedding matrix and LSTM model

4.1 Embedding matrix

The embedding matrix indicates how each word is similar to other words. That similarity is conveyed by assigning each word with an n-dimensional vector. The closer the distances between two data points (words), the more similar they are to each other.
In order to build the embedding matrix, the training dataset needed to be tokenized. 5000 most common tokens were extracted. The 100-dimensional glove dataset was loaded and each of the 5000 extracted words was assigned a 100-dimensional vector. The words not present in the 5000 most common words were assigned with a vector of zeros.

4.2 LSTM model

The embedding matrix serves as the input of the LSTM model. The batch size of the training and testing datasets was set to 20. The size of the hidden layer was set to 128. The training occurred over 8 epochs. In order to ensure the optimal performance on the testing dataset, the model with the lowest validation loss was chosen (the model easily overfits and thus training the model for too

long results in poor performance on the testing dataset).
The following evaluation scores of the LSTM model were obtained:

```
twitter-test1.txt (lstm): 0.623
twitter-test2.txt (lstm): 0.621
twitter-test3.txt (lstm): 0.587
```

As well as the following confusion matrix:

	positive	negative	neutral
positive	0.719	0.040	0.241
negative	0.152	0.714	0.134
neutral	0.215	0.151	0.634

	positive	negative	neutral
positive	0.751	0.040	0.209
negative	0.108	0.735	0.157
neutral	0.306	0.114	0.580

	positive	negative	neutral
positive	0.759	0.053	0.189
negative	0.148	0.579	0.273
neutral	0.282	0.127	0.591

The evaluation scores for the LSTM model are the highest among all the classifiers, with 0.610 being the average evaluation score among the test sets.

5 Summary

This section provides a summary of the F1 scores for each classifier:

- LSTM: F1 score: 0.610
- Logistic regression: 0.555
- SVM: 0.525
- Naive Bayes: 0.488
- Naive Bayes with tf-idf as the input: 0.384