Natural Language Processing Assignment-2

U229111

March 2023

1 Overview

In this coursework, three classifiers have been created: the support vector classifier, multinomial naive bayes, and long short-term memory (LSTM), which is a type of recurrent neural network (RNN). Each classifier will undergo training, testing, and evaluation at the end.

2 Data

In general, the first column of the data is the unique ID for each tweet, and the second column is the actual sentiment, whether the tweet is positive, neutral, or negative. And the last column is the text of the tweet.

2.1 Text Pre-Processing

After the data has been loaded, each file will undergo pre-processing before being mapped into the dictionaries.

Text cleaning : Twitter data often contains noise in the form of hashtags, mentions, URLs, punctuation, and emoticons. All URLs and punctuation will be removed. To handle emoticons, a dictionary of general emoticons has been created to convert them to text. Also, emojis are a common feature of Twitter data, and can contain important sentiment information.

Stop word removal : Stop words are commonly used words that do not carry much meaning, such as "the", "a", and "an". Removing stop words can help reduce the dimensionality of the data and improve the accuracy of the analysis.

Spell checking and correction : Twitter data often contains misspellings and abbreviations. External libraries such as textblob will be used to convert into a proper sentence.

Tokenization : Twitter data often contains text that is not properly divided into words. Therefore, each text will be split into a list of words with the NLTK library.

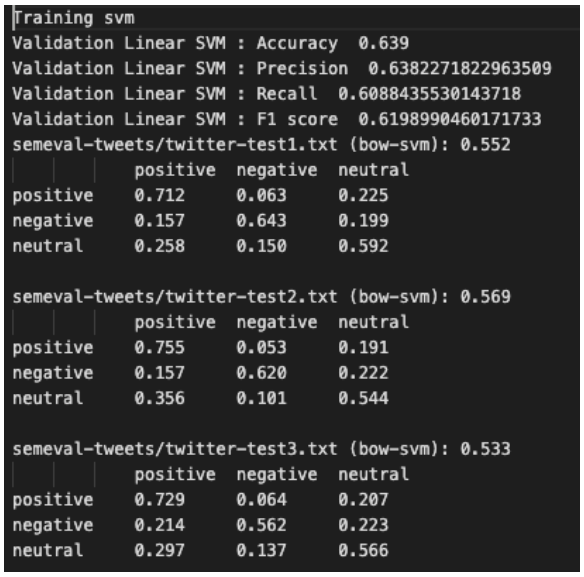
Normalization : Twitter data often contains text that is written in different forms, such as uppercase, lowercase, or mixed case. Normalizing the text to a standard form can help to improve the accuracy of the analysis.

3 Machine Learning Algorithms

3.1 Support Vector Machine

3.1.1 TF-IDF (Term Frequency-Inverse Document Frequency)

- TF-IDF is a more advanced technique that takes into account the importance of each word in a document and the corpus as a whole.
- TF-IDF assigns a weight to each word in a document based on its frequency in the document and its rarity in the corpus.
- The more frequently a word appears in a document, the higher its weight, but the more frequently it appears in the corpus as a whole, the lower its weight.
- This helps to identify the most important words in a document, which can be useful for tasks such as text classification or information retrieval.



```
Training svm
Validation Linear SVM : Accuracy  0.639
Validation Linear SVM : Precision  0.6382271822963509
Validation Linear SVM : Recall  0.6088435530143718
Validation Linear SVM : F1 score  0.6198990460171733
semeval-tweets/twitter-test1.txt (bow-svm): 0.552
            positive  negative  neutral
positive    0.712     0.063     0.225
negative    0.157     0.643     0.199
neutral     0.258     0.150     0.592

semeval-tweets/twitter-test2.txt (bow-svm): 0.569
            positive  negative  neutral
positive    0.755     0.053     0.191
negative    0.157     0.620     0.222
neutral     0.356     0.101     0.544

semeval-tweets/twitter-test3.txt (bow-svm): 0.533
            positive  negative  neutral
positive    0.729     0.064     0.207
negative    0.214     0.562     0.223
neutral     0.297     0.137     0.566
```

Figure 1: SVM classification

The linear SVC function has been used to predict the sentiment of tweets. The TfidfVectorizer function with a value of 5000 has been selected as it produced the highest F1-score during evaluation. From the figure above, the values of the f1-score are consistently above 0.5 for three different evaluation datasets.

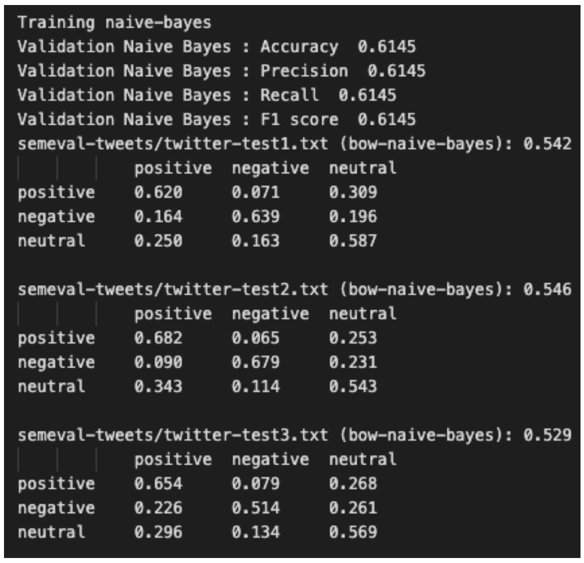
3.2 Multinomial Naive Bayes

3.2.1 Count Vectorization

- In count vectorization, a document is represented as a vector of word counts.

- Each element in the vector represents the number of times a particular word appears in the document.
- Count vectorization does not take into account the importance of each word in the document or the corpus as a whole.
- Count vectorization is a simple and efficient technique for representing text data, but it can be limited in its ability to capture the meaning of the text.

One of the classical classifiers is naive Bayes, which is a commonly used algorithm for text classification, especially for problems such as sentiment analysis. Other bag-of-word features such as count vectorization are able to convert a collection of text documents into a matrix of token counts, which can then be used as input to a machine learning algorithm. The feature represents each document as a vector of numerical values.



```
Training naive-bayes
Validation Naive Bayes : Accuracy  0.6145
Validation Naive Bayes : Precision  0.6145
Validation Naive Bayes : Recall  0.6145
Validation Naive Bayes : F1 score  0.6145
semeval-tweets/twitter-test1.txt (bow-naive-bayes): 0.542
            positive  negative  neutral
positive    0.620     0.071     0.309
negative    0.164     0.639     0.196
neutral     0.250     0.163     0.587

semeval-tweets/twitter-test2.txt (bow-naive-bayes): 0.546
            positive  negative  neutral
positive    0.682     0.065     0.253
negative    0.090     0.679     0.231
neutral     0.343     0.114     0.543

semeval-tweets/twitter-test3.txt (bow-naive-bayes): 0.529
            positive  negative  neutral
positive    0.654     0.079     0.268
negative    0.226     0.514     0.261
neutral     0.296     0.134     0.569
```

Figure 2: Multinomial Naive Bayes Result

Figure 1 above shows the highest f1-scores obtained on three different datasets. All scores are above 0.5, but on average, the f1-score from the SVM classifier is higher than the naive Bayes model.

3.3 Deep Learning LSTM model

LSTM (Long Short-Term Memory) is a type of Recurrent Neural Network (RNN) that is commonly used in natural language processing and other sequence prediction tasks.

RNNs are a type of neural network that are designed to handle sequential data, where the order of the inputs is important. They are able to maintain a "memory" of previous inputs and use this information to make predictions about future inputs. However, traditional RNNs suffer from the problem of vanishing gradients, which makes it difficult for them to remember information over long periods of time. Below, are the process to construct LSTM model.

3.4 Tokenization

In this process, the word will be converted into numerical values based on a training dataset of 5000 frequent words. The process started with tokenization and lemmatization processes with the NLTK library.

3.4.1 Word Embedding

The first 5000 most frequent words from the corpus will be selected as a reference for words in the pre-trained vectors from the GloVe dictionary. All selected words will be assigned a unique ID for reference. As a result, a list of words to index in the dictionary has been created along with embedding matrices. The created dictionary will be used to convert tweet data into numerical tags; unknown words will be tagged as index 1.

3.5 Padding the data

Data will be truncated and padded since the length of tweets varies. Therefore, based on average, only the first 15 words will be taken. This is to avoid bias during prediction, as longer padding will create more biased predictions on the neutral tweet.

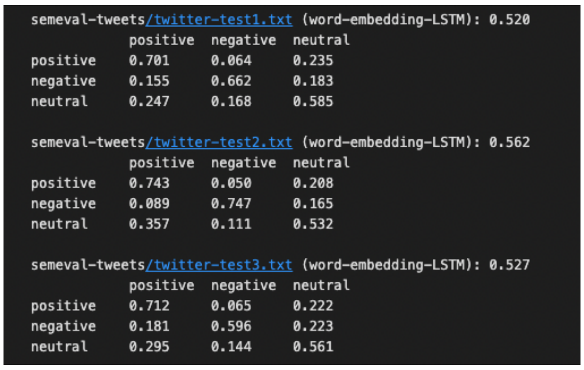
3.6 Architecture

As a standard LSTM model, there is an embedding layer with a dimension of (5000 x 100). The created embedding matrices from the previous process will be inserted here along with the false gradient flag. Since, from traditional classifiers, SVM has the highest performance than Naive Bayes. So the target is to have a model with a higher f1-score than traditional classifiers.

4 Error Analysis

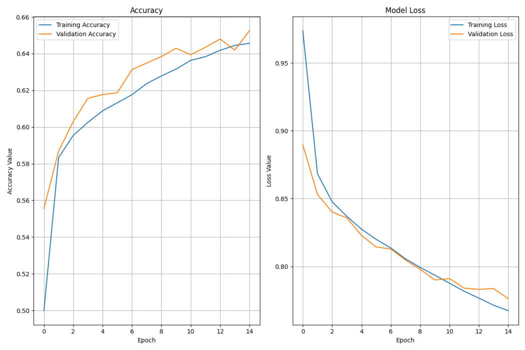
4.1 Standard 1-layer LSTM modelling

Figure 3 shows the f1-scores of three datasets and a line chart of accuracy and epoch losses. Due to only having 1 layer and a learning rate of 0.0001, the model is overfitting with training and development test data, as both loss values decrease as the epoch number increases. Even though both average accuracy increases, the prediction of unseen data is bad. Improvement of LSTM is needed.



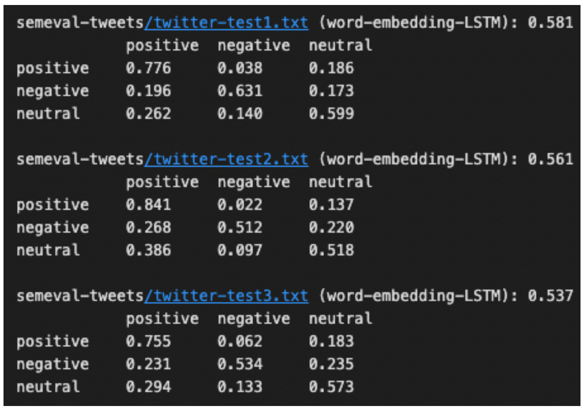
```
semeval-tweets/twitter-test1.txt (word-embedding-LSTM): 0.520
            positive  negative  neutral
positive    0.701     0.064     0.235
negative    0.155     0.662     0.183
neutral     0.247     0.168     0.585

semeval-tweets/twitter-test2.txt (word-embedding-LSTM): 0.562
            positive  negative  neutral
positive    0.743     0.050     0.208
negative    0.089     0.747     0.165
neutral     0.357     0.111     0.532

semeval-tweets/twitter-test3.txt (word-embedding-LSTM): 0.527
            positive  negative  neutral
positive    0.712     0.065     0.222
negative    0.181     0.596     0.223
neutral     0.295     0.144     0.561
```

1-LSTM Result



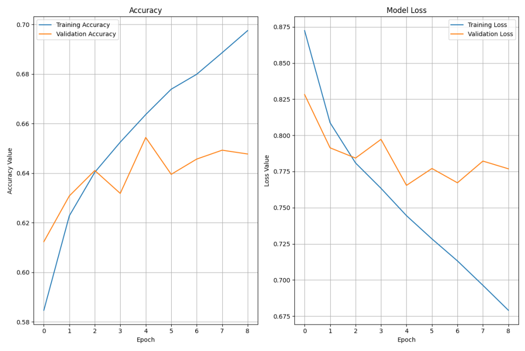
Accuracy and Epoch Loss

Figure 3: Initial LSTM Model Measurement



```
semeval-tweets/twitter-test1.txt (word-embedding-LSTM): 0.581
            positive  negative  neutral
positive    0.776     0.038     0.186
negative    0.196     0.631     0.173
neutral     0.262     0.140     0.599

semeval-tweets/twitter-test2.txt (word-embedding-LSTM): 0.561
            positive  negative  neutral
positive    0.841     0.022     0.137
negative    0.268     0.512     0.220
neutral     0.386     0.097     0.518

semeval-tweets/twitter-test3.txt (word-embedding-LSTM): 0.537
            positive  negative  neutral
positive    0.755     0.062     0.183
negative    0.231     0.534     0.235
neutral     0.294     0.133     0.573
```

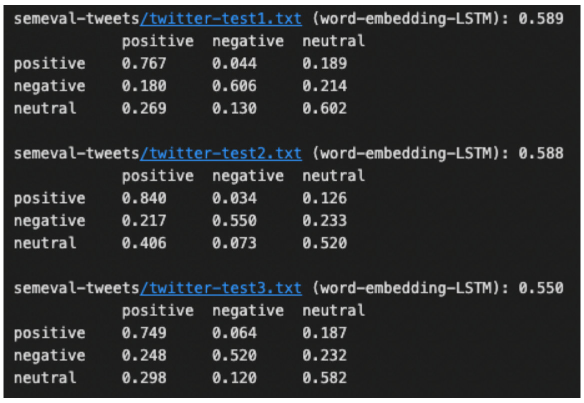
Attention-LSTM Result



Accuracy and Epoch Loss

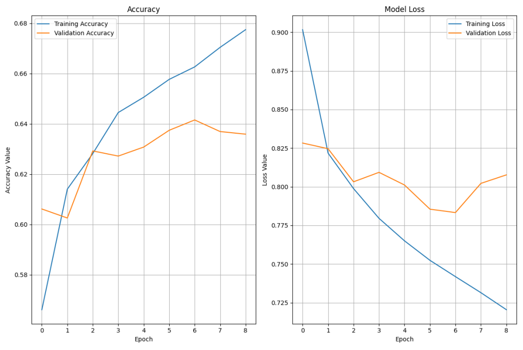
Figure 4: Attention LSTM Model Measurement

In Figure 4, the attention layer model measurements are shown, and the f1-scores indicate only a slight improvement over the SVM classifier. However, the validation chart starts to deviate from the training line, suggesting the need to prevent overfitting.



```
semeval-tweets/twitter-test1.txt (word-embedding-LSTM): 0.589
            positive  negative  neutral
positive    0.767     0.044     0.189
negative    0.180     0.606     0.214
neutral     0.269     0.130     0.602

semeval-tweets/twitter-test2.txt (word-embedding-LSTM): 0.588
            positive  negative  neutral
positive    0.840     0.034     0.126
negative    0.217     0.550     0.233
neutral     0.406     0.073     0.520

semeval-tweets/twitter-test3.txt (word-embedding-LSTM): 0.550
            positive  negative  neutral
positive    0.749     0.064     0.187
negative    0.248     0.520     0.232
neutral     0.298     0.120     0.582
```

Batch-Normalization LSTM Result



Accuracy and Epoch Loss

Figure 5: Batch-Normalization LSTM Model Measurement

4.2 Final Enhancement

The best architecture for improving the model is achieved by tuning hyperparameters and adding a regularization layer, such as batch normalization. With these modifications, the resulting f1-scores are greater than those obtained with a SVM classifier. Even though the architecture seems simple, it produces higher F1-scores than complex models.

Figure 5 displays the confusion matrix and f1-scores for each test set, demonstrating that the algorithm learns from the data and enhances prediction accuracy with each epoch. As the number of epochs increases, the training accuracy rises, and the batch-loss values decrease. However, to prevent overfitting the validation set, the epoch will be terminated after the ninth iteration.

In theory a complex architecture can provide the model with more capacity to learn complex relationships and features from the data, potentially leading to better performance. But, in this case it can also lead to overfitting and slow training, which can limit the model's ability to generalize to new, unseen data.

In practice, it is often the case that a simpler model can achieve comparable or even better performance than a more complex one. This is because a simpler model can be less prone to overfitting and can generalize better to new data.

4.3 Future Improvement

To further improve the model in the future, a proper technique can be applied to address the class imbalance in the training dataset. Other techniques, such as oversampling, undersampling, hybrid approaches, and augmentation, can be used to improve the precision of the model.

Also, building a more complex architecture where it can further find the pattern in the data, such as by increasing the number of LSTM layers, adjusting the hyperparameters, implementing attention mechanisms, and trying different architectures, Also, adding more training datasets can further improve the performance.