UNIVERSITÀ DEGLI STUDI DI TRIESTE

Natural Language Processing

Motahhareh Nadimi

November 30, 2020

Abstract: In this work we study the usage of Topic modelling (LSA and LDA) and multi-class text classification with Logistic regression, Naive Bayes and deep learning methods. The dataset consists of texts of three different classical authors. The goal of text classification is to predict probabilities of a text belonging to each author. For text classification modelling, I use deep learning methods (LSTM, BI-LSTM and CNN). I also analyse the impact of glove word embeddings, dropout regularisation and kernel/bias regularisation on training.

Contents

1 Introduction

Spooky author identification was a competition at Kaggle. The data consists of texts from three authors. The following text is a relatively long sample from the dataset:

I wept for a long time until I saw him about to revive, when horror and misery again recurred, and the tide of my sensations rolled back to their former channel: with a terror I could not restrain I sprung up and fled, with winged speed, along the paths of the wood and across the fields until nearly dead I reached our house and just ordering the servants to seek my father at the spot I indicated, I shut myself up in my own room. Mr. Kirwin charged himself with every care of collecting witnesses and arranging my defence. - Mary Shelley (Mathilda, 1819).

In this work we try to perform topic modelling and classify the authors by using different methods of natural language processing.

The report is structured as follows. In Sections 2 and 3 we describe the data and pre-process it. In Section 4 we do Latent Semantic Analysis (LSA) and in Section 5 Latent Dirichlet Analysis (LDA). We study Multi-class text classification in Section 6, performing logistic regression, naive bayes classification and finally deep learning. We give the conclusions of this study in Section 7.

2 Dataset Description

The dataset provided in the Kaggle is divided into two parts - train and test set. I further divided the train set into train (80 %) and evaluation (20 %). We show in Tab. 2.1 the amount of the train data and evaluation data set (times 1 for the different authors). Fig. 2.1 shows the word cloud of the most frequent words, also shown as histogram of the most frequent Uni-gram in Fig. 2.2. For illustration, we also show in Fig. 2.3 the most frequent Bi-gram (two words). Throughout the report, EAP, HPL and MWS refer to Edgar Allan Poe, H.P. Lovecraft and Mary Shelly respectively.

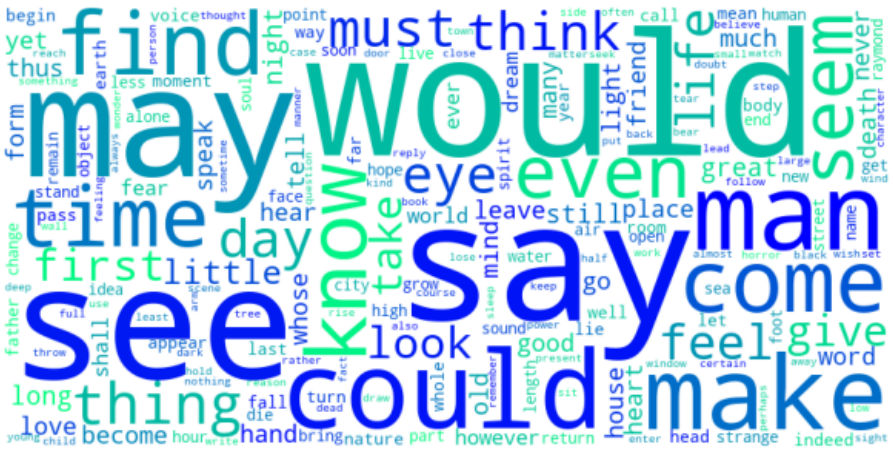

Figure 2.1: Word Cloud of Dataset.

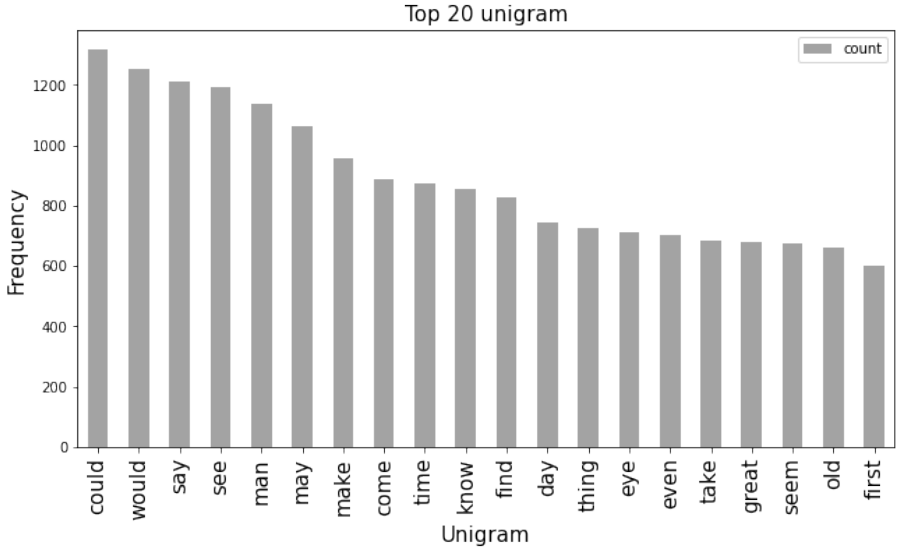

Figure 2.2: The most frequent unigram.

3 Data Pre-processing

To start, I perform some data cleaning and pre-processing as follows:

- Remove *stop words*
- Convert all text to lowercase
- Perform stemming
- Remove words with length less than 2
- Remove punctuation
- Clean text of symbols, etc.

Table 2.1: Dataset Description. The data is multiplied by the number of authors (3).

Train data	Test data
15663×1	3916×1

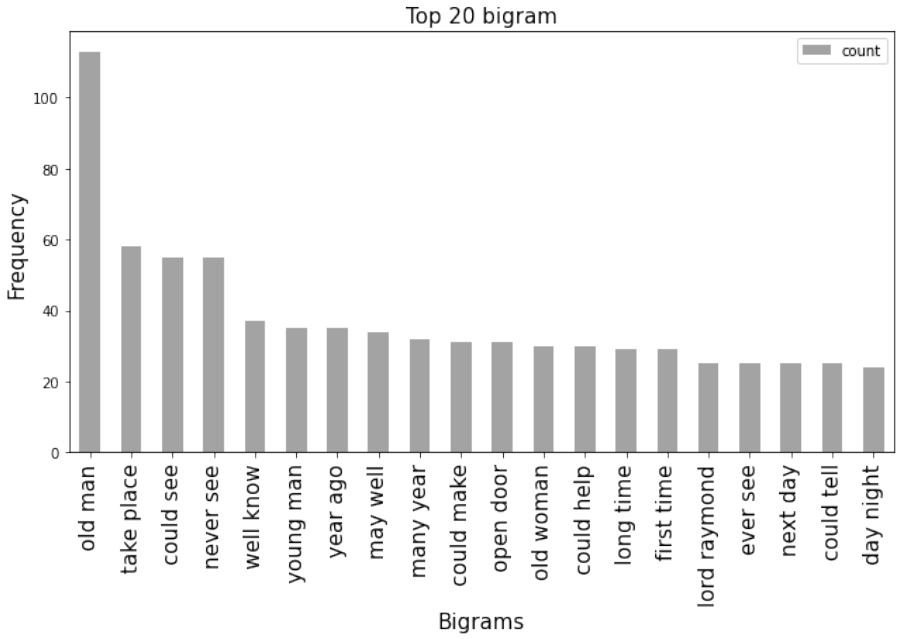

Figure 2.3: The most frequent BI-gram.

4 Latent Semantic Analysis

Topic modelling is an unsupervised text analytics algorithm which automatically discovers the hidden patterns of a text. LSA learns latent topics by performing a matrix decomposition on the document-term matrix using singular value decomposition. For identifying the optimum number of topics in the corpus I used topic coherence measure for identifying the number of topics.

I used TF-IDF to identify the important words in the text data and for improving the results I used χ^2 feature selection, then I applied SVD to find a reduced dimensional representation of the matrix that emphasises the strong relationships and removes the noise. The matrix is reconstructed with the least possible information, by deciding how many dimensions (concepts) to use.

With the TF-IDF, the shape of data is (19579, 2215) and af-

ter applying χ^2 feature selection, the shape of data has changed to (19579, 1500). After implementing SVD and setting the number of topics to 30, the list of topics are shown as below:

The topics for LSA are grouped, for example, as

Topic 0: *"flash" "frightful" "lead" "interval" "chasm"*
Topic 1: *"flash" "atmosphere" "date" "enable" "contemplate"*
Topic 2: *"frightful" "chasm" "flash" "drop" "justine"*
Topic 3: *"lead" "interval" "alive" "aperture" "paris"*
Topic 4: *"attack" "lead" "frightful" "interval" "kind"*

I also implemented the K-MEAN (10 group of data) to return the natural grouping of data points after χ^2 feature selection. I plot the clusters generated by the K-Means operation using PCA. PCA reduces the number of dimensions to number of components equal to 2. For simplicity, in Fig. 4.1 I plot the reduced sized of 1500 data points.

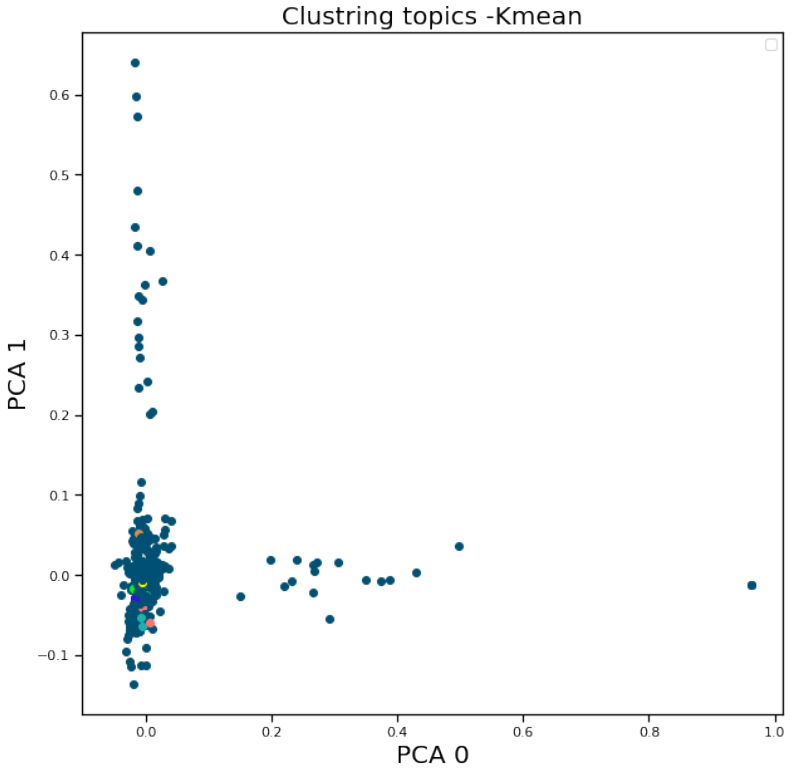

Figure 4.1: Data points of the TF-IDF vector.

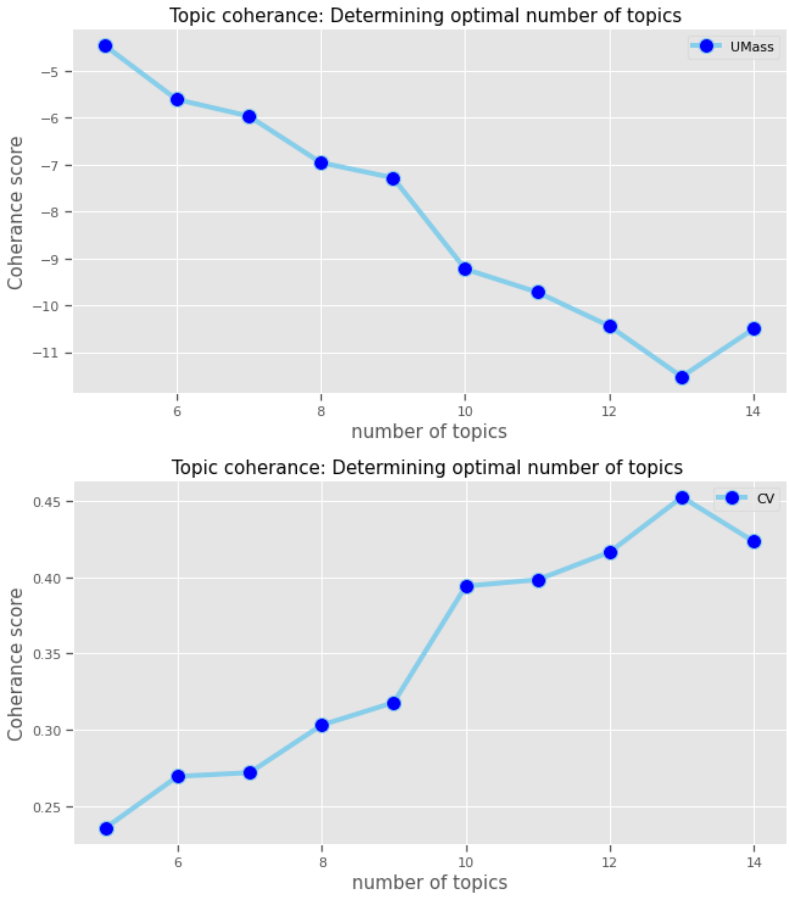

Figure 5.1: Top) UMass score. Bottom) CV score.

5 Latent Dirichlet Allocation

I evaluated the coherence scores both UMass and CV in LDA model, and based on the coherence score I set the number of topics equal to 13. We show in Fig. 5.2 the normalised number of topics per author.
The topics for LDA are shown as following:
Topic 0: *"fear", "word", "would", "die", "far"*
Topic 1: *"come", "look", "go", "much", "tell"*
Topic 2: *"man", "old", "long", "year", "death"*
Topic 3: *"make", "find", "dream", "could", "water"*
Topic 4: *"must", "hand", "many", "yet", "shall"*
Topic 5: *"see", "house", "pass", "turn", "way"*

Figure 5.2: Topics per author.

6 Multi-class text classification

6.1 Logistic regression classifier

The task is multi-class text classification. For this goal, I implement logistic regression, Naive Bayes and deep learning algorithms. I implemented the logistic regression and Naive Bayes classifier on the TF-IDF result and I transformed labels into numbers. I implemented k-fold cross validation to see the performance of both Logistic regression and Naive Bayes classifier.
The following results show the performance of Logistic regression classifier on the train set:

- *F1 with 2 folds for bag-of-words is 0.73*
  Training on 7831.5 instances/fold, testing on 7831.5
- *F1 with 3 folds for bag-of-words is 0.73*
  Training on 10442.0 instances/fold, testing on 5221.0
- *F1 with 5 folds for bag-of-words is 0.74*
  Training on 12530.4 instances/fold, testing on 3132.6
- *F1 with 10 folds for bag-of-words is 0.74*
  Training on 14096.6 instances/fold, testing on 1566.3
- *F1 with 15 folds for bag-of-words is 0.74*
  Training on 14618.8 instances/fold, testing on 1044.2
- *F1 with 20 folds for bag-of-words is 0.75*
  Training on 14879.85 instances/fold, testing on 783.15

For the test set, the total accuracy is 74%. The confu-

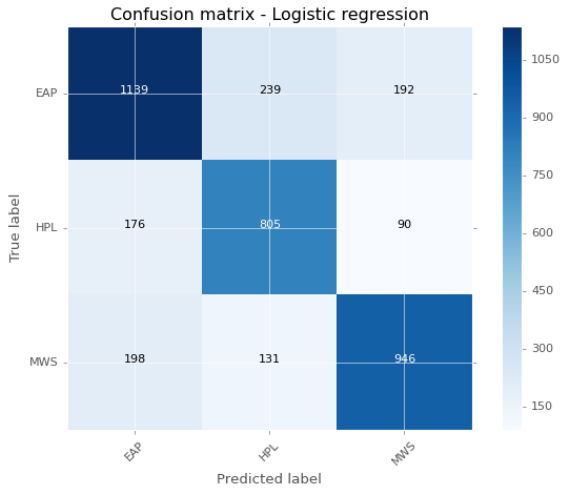

Figure 6.1: Confusion matrix of the Logistic Regression classifier.

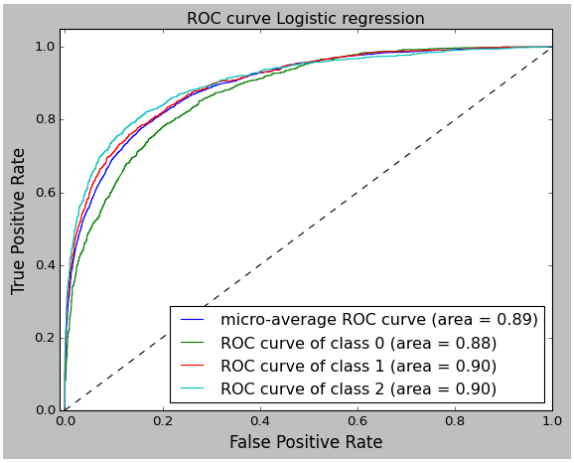

Figure 6.2: ROC curve of the Logistic Regression classifier.

sion matrix is shown in Fig. 6.1. The total number of misclassified samples is 1026. The area under roc curve is shown in the Fig. 6.2. I used micro average roc curve area, as we are dealing with multi class classification.

6.2 Naive Bayes classifier

The confusion matrix is shown in the Fig. 6.3. The total number of misclassified samples is 1003. The area under roc curve is shown in Fig. 6.4.
The following results show the performance of Naive Bayes classifier using K-fold cross validation:

- *F1 with 2 folds for bag-of-words is 0.75*

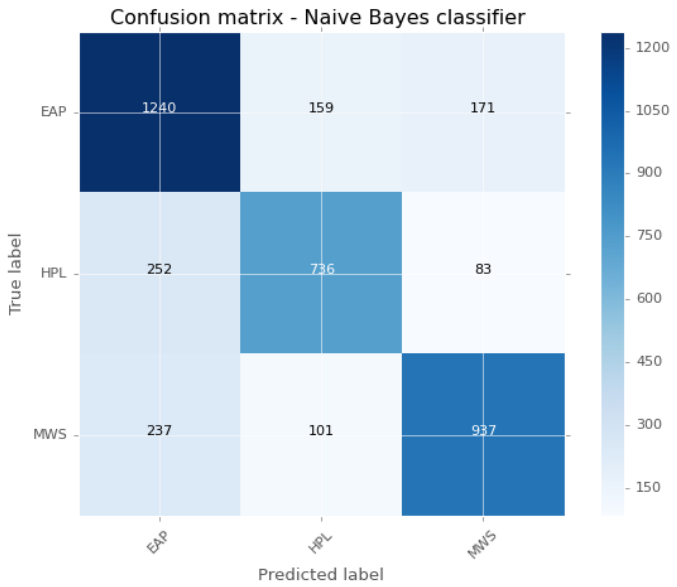

Figure 6.3: Confusion matrix of the Naive Bayes classifier.

[!htb]

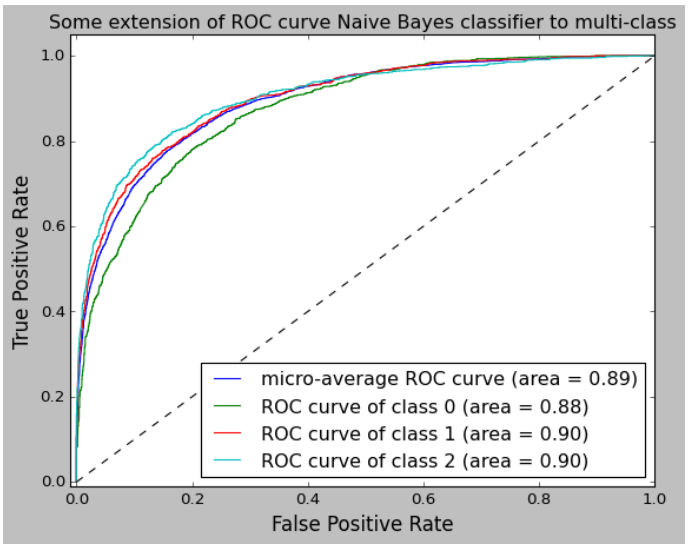

Figure 6.4: ROC curve of the Naive Bayes classifier.

Training on 7831.5 instances/fold, testing on 7831.5

- *F1 with 3 folds for bag-of-words is 0.75*
  Training on 10442.0 instances/fold, testing on 5221.0
- *F1 with 5 folds for bag-of-words is 0.75*
  Training on 12530.4 instances/fold, testing on 3132.6
- *F1 with 10 folds for bag-of-words is 0.75*
  Training on 14096.6 instances/fold, testing on 1566.3
- *F1 with 15 folds for bag-of-words is 0.75*
  Training on 14618.8 instances/fold, testing on 1044.2

- *F1 with 20 folds for bag-of-words is 0.75*
  Training on 14879.85 instances/fold, testing on 783.15

6.3 Deep Learning

Before the data can be fed into a model, it has to be transformed into a format the model can understand. To ensure that the model is not affected by the data order I shuffle the data. I split the samples after shuffling; 80 % for training and 20 % for validation. I apply label vectorisation and one-hot encoding to my label. For the model loss function I use categorical cross entropy. My sequence vectorization is as following:

- Tokenises the texts into words
- Creates a vocabulary using the top 20,000 tokens
- Converts the tokens into sequence vectors
- Pads the sequences to a fixed sequence length that is 409.

I apply the **LSTM**, **BILSTM** and **convolutional neural network** on data. I also tried a simple attention model but the results do not improve. Because I have 3 classes, I used softmax activation. I also applied kernel and bias regularisation to deal with over fitting.[1] We show in Tab. 6.1 the learning parameter of model.

Table 6.1: Learning Parameter

Learning parameter	Value
Metric	accuracy
Loss function multi class classification	categorical cross entropy
Optimizer	adam

6.3.1 LSTM model

The LSTM model is shown in Fig. 6.5. The result is shown in the Fig. 6.6. I reach 0.81% of accuracy in the test set.

[1] I also applied GolVe word embeddings but my model without GolVe word embeddings shows a better result. If I froze the weights of the pre-trained embeddings and trained just the rest of the network, the models did not perform well and there is under-fitting.

```
Model: "LSTM"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 409, 100)          1766600
_________________________________________________________________
spatial_dropout1d_3 (Spatial (None, 409, 100)          0
_________________________________________________________________
lstm_3 (LSTM)                (None, 64)                42240
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
Dense (Dense)                (None, 3)                 195
=================================================================
Total params: 1,809,035
Trainable params: 1,809,035
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 6.5: LSTM model

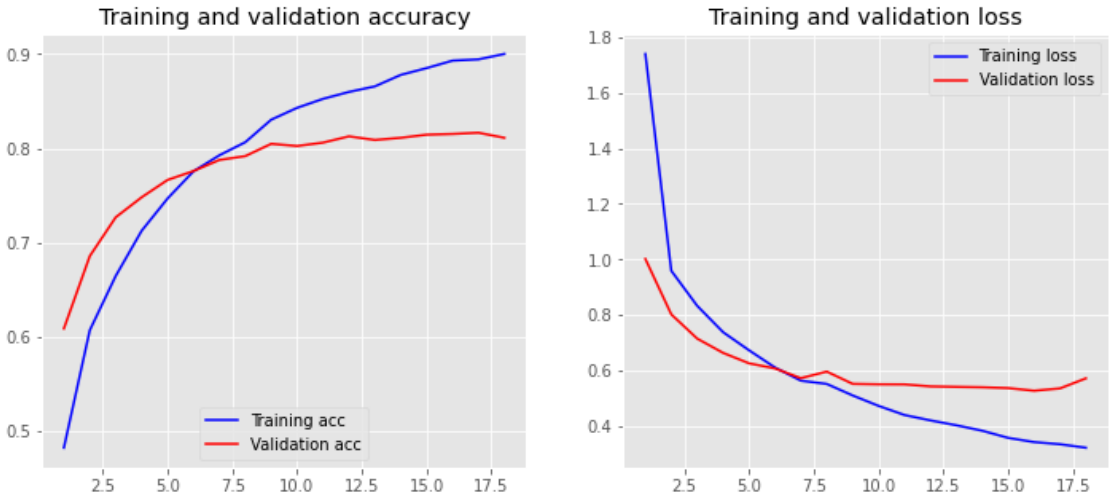

Figure 6.6: LSTM results

6.3.2 CNN

The Convolutional Neural Networks model is shown in the Fig. 6.7. I applied 3 stacked CONVD1 and The result is shown in the Fig. 6.8. I reach 78.35% of accuracy in the evaluation set. I applied also 2 layers of CONVD1 followed by MaxPlooling1D and I reached 1% less accuracy compared to 3 layers.

```
Model: "CNN"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 409)]             0
_________________________________________________________________
Embedding (Embedding)        (None, 409, 100)          1766600
_________________________________________________________________
CONV1 (Conv1D)               (None, 409, 32)           16032
_________________________________________________________________
MaxPool1D1 (MaxPooling1D)    (None, 310, 32)           0
_________________________________________________________________
FLATTEN (Flatten)            (None, 9920)              0
_________________________________________________________________
Dropout (Dropout)            (None, 9920)              0
_________________________________________________________________
Dense (Dense)                (None, 3)                 29763
=================================================================
Total params: 1,812,395
Trainable params: 1,812,395
Non-trainable params: 0
```

Figure 6.7: CNN model

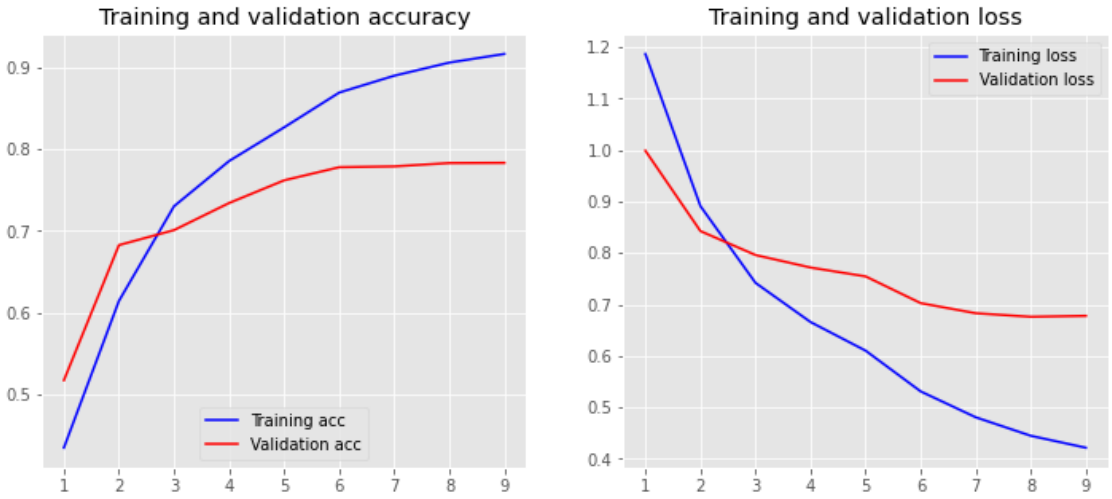

Figure 6.8: CNN results

6.3.3 BI LSTM+GloVe Embeddings

AS I mentioned above, the model does not perform good with GloVe Embeddings when I freeze the learning parameters.

6.3.4 BI LSTM

The LSTM model is shown in Fig. 6.9. The result is shown in the Fig. 6.10. I reach 0.81% of accuracy in the test set.

I also applied 2 stacked layers of BILSTM with 32 units each, but it did not produce a better result. The area under roc curve and the confusion matrix for BI LSTM model is shown in the Fig. 6.11. Sample results of the model is shown in the Tab. 6.2

```
Model: "BI-LSTM"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Embedding (Embedding)        (None, 409, 100)          1766600
_________________________________________________________________
bidirectional_5 (Bidirection (None, 64)                34048
_________________________________________________________________
Dropout (Dropout)            (None, 64)                0
_________________________________________________________________
Dense (Dense)                (None, 3)                 195
=================================================================
Total params: 1,800,843
Trainable params: 1,800,843
Non-trainable params: 0
```

Figure 6.9: BI LSTM model

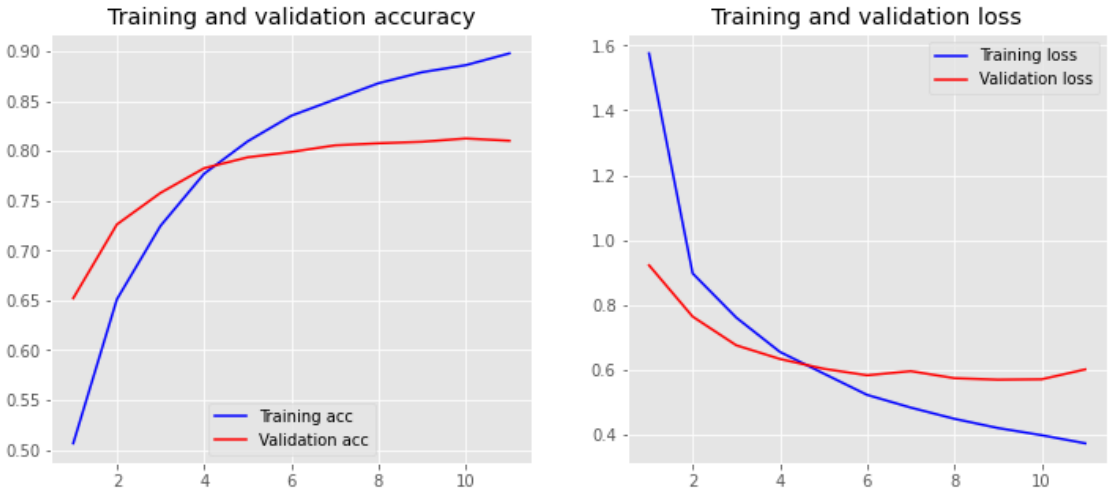

Figure 6.10: BI LSTM results

6.3.5 BI LSTM +Attention

I developed a BI LSTM model with a simple Attention model. The model is shown in Fig. 6.12. I implement a custom layer which compute attention on the positional/temporal dimension in Keras. The validation accuracy now reaches 80% accuracy after the addition of

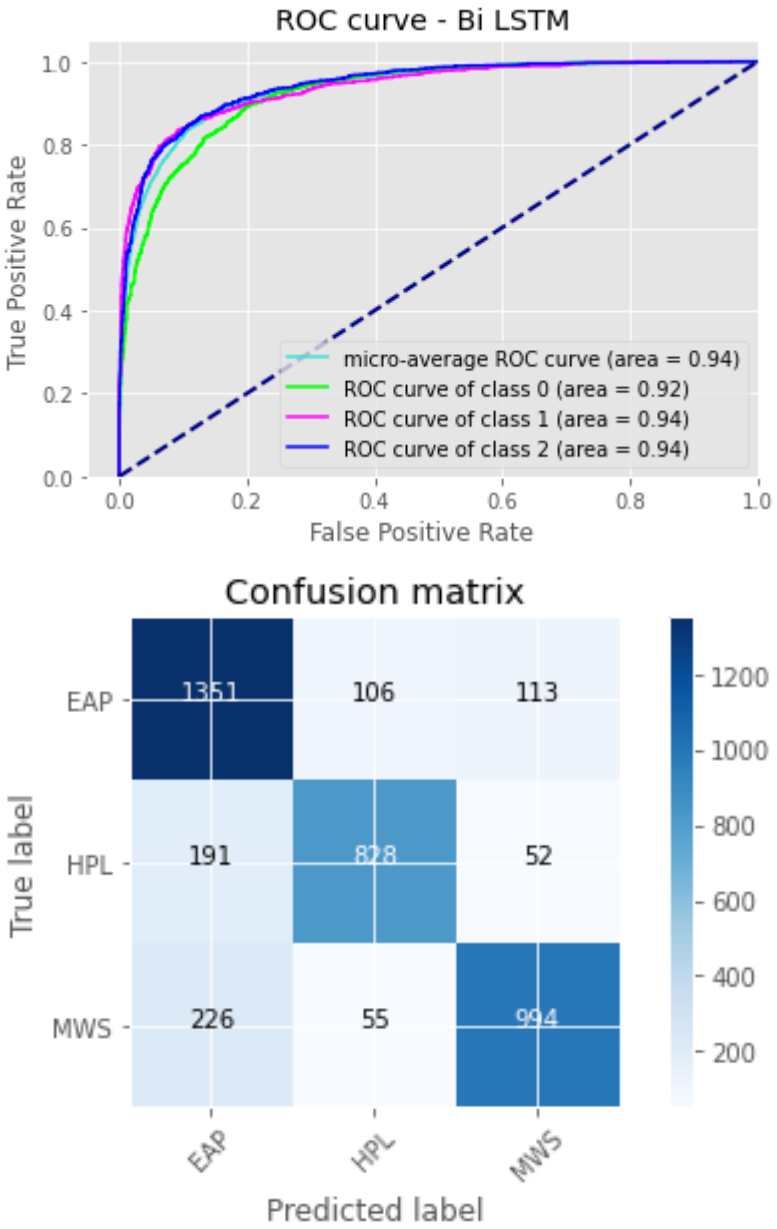

Figure 6.11: Top) ROC curve of the BILSTM model. Bottom) Confusion matrix of the BILSTM mode.

the custom Attention layer. [2]

[2] Bahdanau et al (2015) came up with a simple but elegant idea where they suggested that not only can all the input words be taken into account in the context vector, but relative importance should also be given to each one of them.

Table 6.2: Sample results of BILSTM model.

Row	Author	New prediction	Truth
6148	EAP	EAP	EAP
4881	MWS	MWS	MWS
16114	MWS	MWS	MWS
738	EAP	EAP	EAP

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_7 (Embedding)      (None, 409, 100)          1766600
_________________________________________________________________
bidirectional_6 (Bidirection (None, 409, 128)          84480
_________________________________________________________________
attention_3 (attention)      (None, 128)               537
_________________________________________________________________
dropout_7 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 387
=================================================================
Total params: 1,852,004
Trainable params: 1,852,004
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.12: BILSTM + Attention model

7 CONCLUSIONS

The result of the text classification with different algorithm is shown in the Fig. 7.1. As we can see from the chart, LSTM and BI LTM reach a better accuracy (81%).

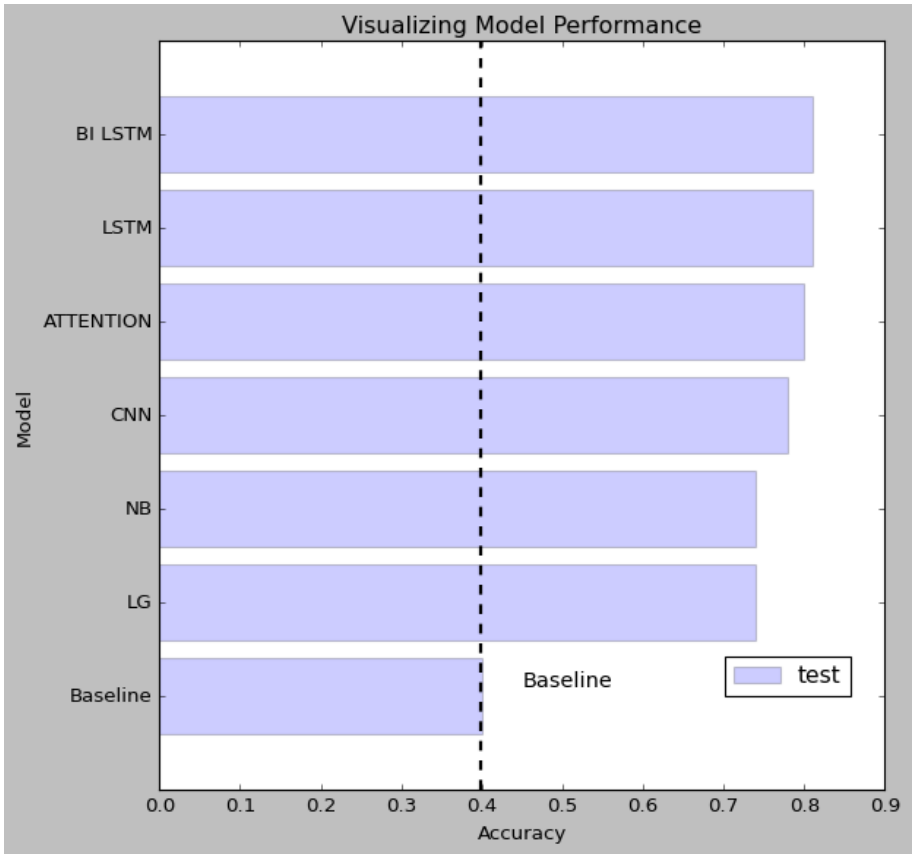

Figure 7.1: The result of the text classification with different algorithm

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. Jacobs University Bremen, Germany, Universite de Montre al, 2014.

[2] Comprehensive guide attention mechanism
https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/

[3] Keras examples.
https://keras.io/examples/nlp/

[4] Solving sequence problems with lstm in keras.
https://stackabuse.com/solving-sequence-problems-with-lstm-in-keras/