Assignment 3

Name: Vishwanath Dattatreya Doddamani
Student Id: *A0286188L*
NUSNET Id: *E1237250*
Kaggle User Id: 'Vishwanath Dattatreya Doddamani'

***Abstract*—This report aims to classify five emotions – anger, love, amusement, disapproval, gratitude using text mining techniques and presents the steps involved in arriving at the final model.**

Keywords – Emotion Classification, Text Mining, BERT

I. Motivation

Comprehending underlying intentions expressed through text can uncover a vast number of valuable insights. Emotion classification has hence garnered significant attention as it has many applications such as social media analysis, customer feedback analysis, human computer interaction. This report aims to present one such model which was used for the task.

II. Model Training and Evaluation

A. Data Description

The data consists of 7500 entries with text along with its assigned one of five labels (anger, love, amusement, disapproval, gratitude). Figure 1 represents pie chart with distribution of data according to labels.

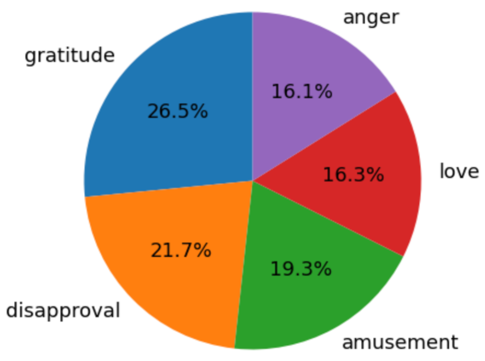


Figure 1 depicting distribution of data according to labels

B. Data Cleaning

The text case is lowered, and all the punctuation are removed in data cleaning step. The labels are converted to numerical form.

C. Data Pre-Processing

BERT (Bi-directional Encoder Representations from Transformers) model's pre-trained tokenizer available in the Hugging Face Transformers library is used for tokenization. Specifically, 'bert-base-uncased' pre-trained tokenizer model is used which is developed by Google on large corpus of texts. Tokenizer splits the texts into words and then converts it to a numerical representation suitable to use for neural networks (BERT models to be more specific).

D. Training and Model Selection

Pytorch is used for training. 'BertForSequenceClassification' pre-trained model which is fine tuned for sequence classification tasks is used for model initialization. 'Adam' optimizer is used for training. Then, to fine tune the model according to the data, data loaders are created for training and validation sets. Data loaders help to iterate data with batches while training. The training and test (validation is also same as test in this case) set sizes are 80% and 20% respectively. In training loop, the training dataset is iterated over batches, loss is computed (cross entropy loss), then with backpropagation, model parameters are updated. Along with the above steps, metrics such as training loss, training accuracy, validation loss and validation accuracy are computed. This helps in monitoring model's performance over time and detect over or under fitting. Early stopping is implemented to prevent overfitting. Learning rate is set to '0.000001' and batch size is 64. Early stopping patience is set to 3, i.e., if validation loss keeps increasing for 3 epochs, then early stopping is triggered, and training is stopped. Figure 2, represent training and validation loss against epochs. Figure 3 represents training and validation accuracy against epochs.

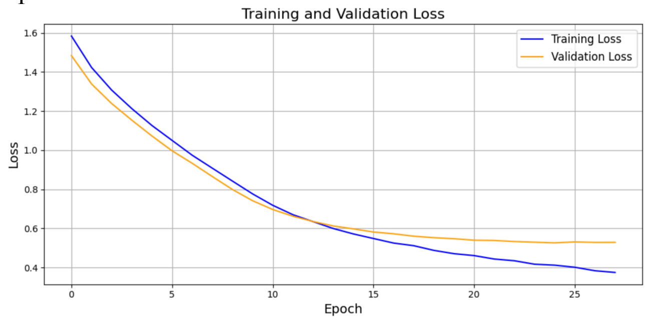


Figure 2 depicting 'Training and Validation Loss vs Number of Epochs'

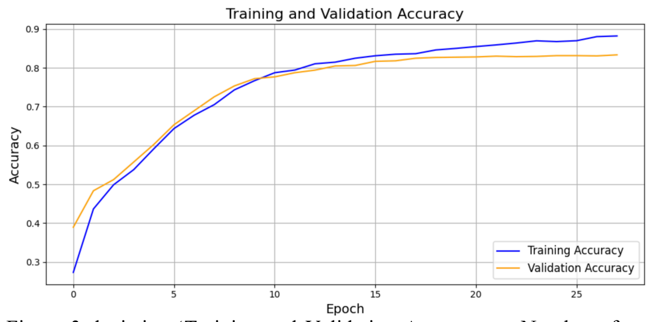
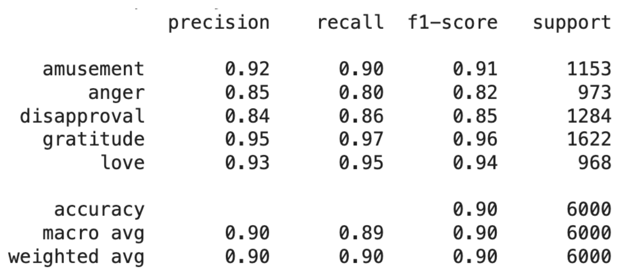


Figure 3 depicting 'Training and Validation Accuracy vs Number of Epochs'

The model with the least validation loss is selected to be the best model.

E. *Model Evaluation*

The best validation accuracy achieved with the above training process is 83.33 and training accuracy of 88.2. Figures 4 and 5 represent classification report for the training (6000 data points) and validation set (1500 data points) respectively. As it can be seen from both the classification reports, 'Anger' and 'Disapproval' have lower f1-scores. However, 'Amusement' has a higher f1-score in training set, but it is unable to replicate the same in validation set. 'Gratitude' and 'Love' have high f1-scores in both sets of data. Same is the case for precision and recall metrics as well.

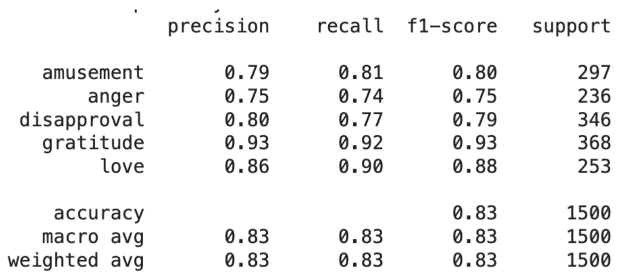


```
              precision    recall  f1-score   support

   amusement       0.92      0.90      0.91      1153
       anger       0.85      0.80      0.82       973
 disapproval       0.84      0.86      0.85      1284
   gratitude       0.95      0.97      0.96      1622
        love       0.93      0.95      0.94       968

    accuracy                           0.90      6000
   macro avg       0.90      0.89      0.90      6000
weighted avg       0.90      0.90      0.90      6000
```

Figure 4 depicting classification report of training set.



```
              precision    recall  f1-score   support

   amusement       0.79      0.81      0.80       297
       anger       0.75      0.74      0.75       236
 disapproval       0.80      0.77      0.79       346
   gratitude       0.93      0.92      0.93       368
        love       0.86      0.90      0.88       253

    accuracy                           0.83      1500
   macro avg       0.83      0.83      0.83      1500
weighted avg       0.83      0.83      0.83      1500
```

Figure 5 depicting classification report of validation set.

F. *Fine Tuning*

The learning rate and batch size are varied to find the best model hyperparameters. The learning rate and batch size mentioned in the model training is selected based on this process. Learning rate is outer loop (1e-3, 5e-3, 1e-4, 5e-4, 1e-5, 5e-5, 1e-6, 5e-6) and batch sizes are varied in inner loop (4, 8, 16, 32, 64, 128). The best combination was found to be learning rate 1e-6 and batch size of 64. The best combination was found using lowest validation loss along with monitoring model's performance metrics. The hyperparameters that resulted in low validation loss and the model that was not overfitting was selected. Based on the results, higher class weights were assigned to 'Anger' and 'Disapproval' while training, however that did not result in significant improvement and hence was not included in the final model. The data was also oversampled for 'Anger' and 'Disapproval' and tested for improvement. However, this technique also did not yield significant improvement and hence was not included in the final model. Custom Tokenizers using n-gram range was also tested, but that too did not yield much improvement in results.

III. Discussion

Many pre-trained models were tested for this task and the best one was found to be 'bert-base-uncased' as used in this assignment. Other models which were tried and tested are 'distilbert-base-uncased', 'distilbert-base-uncased-finetuned-sst-2-english', 'roberta-base' and word embedding models such as 'British National Corpus', 'Word2Vec' were also tested. Other than pre-trained models, using tf-idf and count vectorizers as pre-processing steps with varying n-grams, the model was also tested using logistic regression, KNN, MLP, Naïve Bayes using sklearn libraries. Logistic Regression performed best while knn performed worst. However, none of the above techniques gave more than 80% validation accuracy. Table 1 depicts validation accuracy for KNN, Naïve Bayes and MLP for tf-idf and count vectorizer methods.

	Tf-idf Vectorizer	**Count Vectorizer**
KNN (n=15)	0.368	0.332
Naïve Bayes	0.623	0.609
Logistic Regression	0.798	0.762
MLP	0.765	0.771

Table 1 depicting validation accuracy for different algorithms.

Table 2 represents validation accuracy for pre-trained models.

	Validation Accuracy
bert-base-uncased	0.833
distilbert-base-uncased	0.825
distilbert-base-uncased-finetuned-sst-2-english	0.812
roberta-base	0.824

Table 2 depicting Validation accuracy for different pre-trained models

IV. Conclusion

As seen in the training process during model building and evaluation, the best model was found to be 'bert-base-uncased' for this task. However, there is only about 3-4% increment in validation accuracy compared to the lighter models (logistic regression). As seen in the results, the model was able to best identify data corresponding to 'Gratitude' and 'Love'. Multiple methods were tried as mentioned in fine tuning section in an attempt to increase the f1-score of 'Anger' and 'Disapproval'. Additional techniques need to be implemented and tested to accurately classify the data corresponding to these labels.

V. References

[1] http://vectors.nlpl.eu/repository/
[2] https://huggingface.co/docs/transformers/en/tasks/sequence_classification
[3] https://huggingface.co/docs/transformers/model_doc/bert