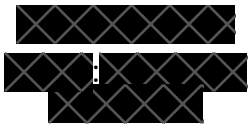
Natural Language Processing (COM3029) Individual Couresework



25th April 2023

Contents

1 Group Declaration

Label Selection: We have decided to merge certain sentiments into a single class, as demonstrated below.

1. anger = [anger, annoyance, disapproval]
2. disgust
3. fear = [fear, nervousness]
4. joy = [joy, amusement, excitement, relief,approval]
5. love = [gratitude, caring,admiration]
6. pride
7. sadness = [sadness, disappointment, grief, remorse]
8. surprise = [surprise, realization,]
9. optimism
10. confusion
11. embarrassment
12. desire
13. curiosity
14. neutral

Experiment Variations	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Data preprocessing	✓	✓	-	✓	-
NLP Algorithms	✓	✓	-	-	-
Text featurization/tokenization	✓	-	-	-	-
Train/Test/Validate dataset splitting	-	-	✓	✓	✓
Loss functions and Optimizers	-	✓	✓	✓	✓
Hyperparameters optimisation	✓	✓	✓	-	✓
Fine Tuning vs Full Training	-	-	✓	-	-
Different Fine Tuned models	-	-	-	✓	✓

Table 1: Summary of experiment variations for each team member

2 Introduction

2.1 Data Analysis

GoEmotions Demszky et al. (2020) is a dataset comprising 58K Reddit comments classified as Neutral, one or more of 27 emotions. The dataset uses data curation techniques to address Reddit's racial bias and toxic language. A few of these procedures are hiding proper names and religious phrases and eliminating insensitive and non-English comments.

For the following experiments I have decided to use the raw dataset. This dataset contains 211225 texts belonging to 28 sentiments 2. I noticed that some texts did not belong to any of the sentiments e.g. "other" 2 ; therefore, they were removed from the final processed data.

The dataset is first analysed by printing basic information, such as the features and number of rows in the training set. The dataset is then preprocessed and converted into a pandas DataFrame for further analysis. The 28 original sentiments are regrouped into 14 broader categories using a predefined mapping called sentiment groups to simplify the classification task. Each text sample is assigned a new label based on this mapping.

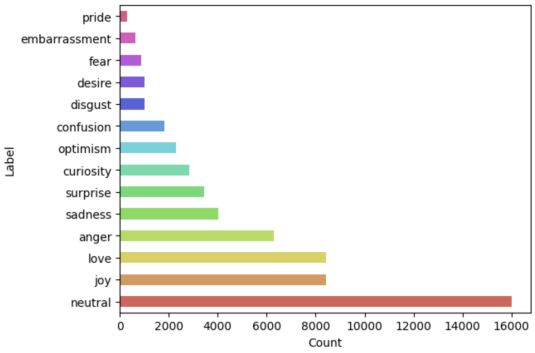


Figure 1: Preprocessed Data Distribution.

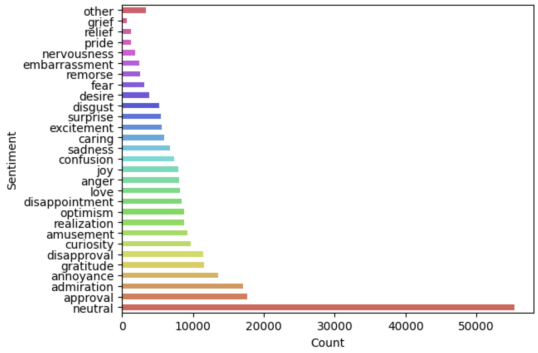


Figure 2: Raw Data Distribution.

A heatmap Figure 3 has been generated to visualise the correlations between different sentiments. The most notable correlations are "fear" and "nervousness" with a positive correlation value of 0.2, and "excitement" and "joy" with a positive correlation value of 0.1. This heatmap supports the decision to combine these sentiment classes, as their correlations indicate shared features.

While the majority of the combined sentiment classes display positive correlation values, there are a few exceptions, such as "anger" and "disapproval" with a negative correlation value of -0.018 and "joy" and "approval" with a -0.024 negative correlation value. Even after grouping the sentiments, we still face an imbalanced dataset, which will affect the model performance.

Upon analysing the dataset using the pandas DataFrame describe() function, it becomes clear that the "neutral" sentiment is the most predominant, with the highest mean value of 0.261797. Figure 1 further confirms the presence of an imbalanced dataset.

Sentiment	`One-Hot Vector`
Surprise	`[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]`

Table 2: Example of one-hot encoding.

Unwanted columns, such as author, subreddit, and other metadata, are removed from the DataFrame, leaving only the essential information like text, id and labels. The original sentiment columns are removed as they are no longer necessary. The sparse binary sentiment labels have been replaced with integer encoding, assigning each class a unique integer label ranging from 0 to 13 Tabels 3 2.

Sentiment	**Integer Label**
Surprise	8

Table 3: Example of integer encoding.

Finally, the preprocessed data is saved to a CSV file called 'emotions.csv,' which can be used in later experiments without the need to run through the preprocess cells again.

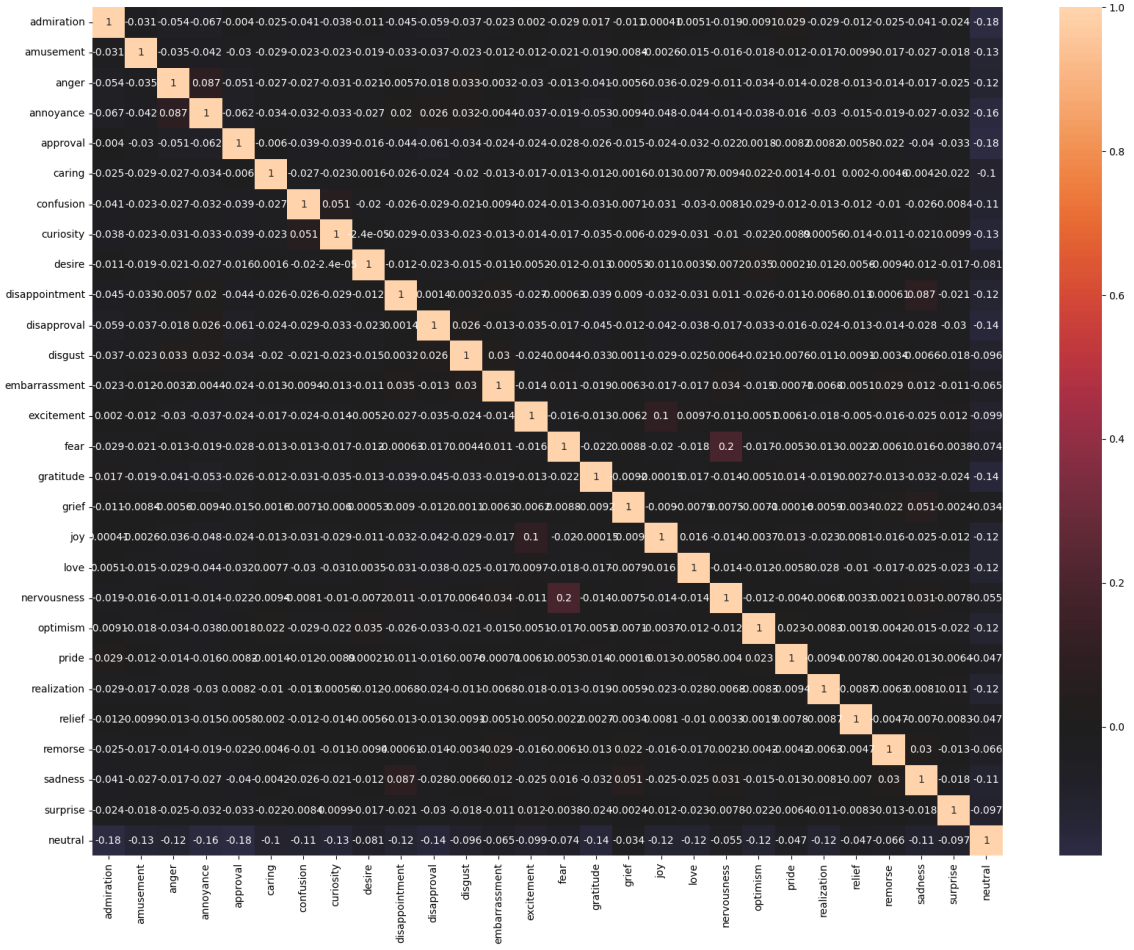


Figure 3: Heat Map

2.2 Models

Two models have been selected for the experiments: the pre-trained RoBERTa model (Liu et al. 2019) and a custom-built Bidirectional LSTM (BiLSTM) model. These experiments aim to demonstrate the effects of various processes on the performance of these models.

roBERTa (Robustly Optimised BERT Pretraining Approach)

The first model employs the pre-trained RoBERTa (ibid.) model with 14 different sentiment labels for sequence classification. RoBERTa (ibid.) is a variant of BERT (Devlin et al. 2019) that has been improved for performance by removing the next sentence prediction objective and employing dynamic masking. The Trainer (Trainer 2023) and TrainingArguments classes from the Hugging Face Transformers library is used to train and fine-tune this model.

BiLSTM (Bi-directional Long Short-Term Memory)

The second model is a Bidirectional LSTM (BiLSTM) model explicitly developed for sentiment analysis. To capture context from both directions, the BiLSTM model employs two LSTMs in parallel, processing input sequences in both forward and backward directions. An embedding layer, a bidirectional LSTM layer with dropout, and a final fully connected layer output the predicted sentiment class. Custom functions that handle text processing, padding, and batching are used to manage the training and evaluation processes. This model provides a more customised approach to sentiment analysis, with the

architecture and training parameters adjustable. This model and methods have been mainly inspired by the code found in the NLP-2023 lab06-1b by *Agarwal* and adapted from binary to multi class classification and can be used with or without pre-trained word embeddings.

3 Experimental variation 1

Data pre-processing

This first experiment presents results for the two models, keeping hyperparameters constant and just the data changes. In the first run, the models are fed the unprocessed text, whereas in the second run, the following data processing techniques are applied:

- Lowercasing: Converting all characters in the text to lowercase to ensure uniformity and reduce the vocabulary size.
- URL Removal: Eliminating any URLs present in the text using regular expressions.
- Emoji Removal: Removing emojis from the text by converting them to their corresponding descriptions using the emoji library and then removing them using regular expressions.
- Punctuation Removal: Stripping punctuation marks from the text to reduce noise and concentrate on the core content of the text.
- Stopword Removal: Eliminating commonly used words (stopwords) such as "and," "the," and "is" from the text.
- Lemmatization: Converting words to their base forms (lemmas) using the spacy library to reduce the number of unique words in the dataset.

3.1 Model Architecture and Hyperparameters

roBERTa Model

The RoBERTa For Sequence Classification from the Hugging Face Transformers library, with a base configuration (roberta-base) is fine-tuned for this classification task. The model's architecture and hyperparameters are described below:

- Tokenizer: Roberta Tokenizer Fast
- Max Sequence Length: 128 tokens, with padding and truncation
- Scheduler: linear schedule with warmup
- Evaluation Metrics: accuracy, F1-score, precision, and recall.
- Data Split: Training: 70%(40169), Validation: 18%(10329), Testing: 12%(6887)
- Number of parameters: 124,656,398

BiLSTM Model

Spacy library is used for tokenization and TorchText for creating a vocabulary. A custom collate function is defined for padding sequences in a batch. DataLoader objects are used for training, validation, and test datasets.

The following hyperparameters are used in the BiLSTM model:

- Vocab size: Length of the vocabulary
- Embedding dim: 100
- Hidden dim: 256
- Number of LSTM layers: 2
- Bidirectional: True
- Scheduler: linear schedule with warmup.
- Evaluation Metrics: accuracy, F1-score, precision, and recall.
- Dropout: 0.5
- Data Split: Training: 70%(40169), Validation: 18%(10329), Testing: 12%(6887)
- Number of parameters: 6,340,626

3.2 Results

Table 4: Performance/training parameters of RoBERTa/BiLSTM models

	Loss	Accuracy	F1 Score	Precision	Recall
Performance Metrics before data processing					
RoBERTa	2.10	42.75	36.12	36.13	36.39
BiLSTM	6.33	32.88	22.88	63.39	32.88
Performance Metrics after data processing					
RoBERTa	1.91	40.88	32.16	36.11	30.73
BiLSTM	6.48	32.87	20.07	76.05	32.87
Training Parameters					
Optimizer			AdamW		
Loss			Cross-Entropy		
Epochs			5		
Weight Decay			0.01		
Batch Size			32		
Learning Rate			5e-5		

3.3 Model Evaluation

Evaluation before data preprocessing

Looking at the confusion matrix Figure 4 for RoBERTa prior to applying data preprocessing, we can observe the following:

- The model seems to perform relatively well in predicting "anger" (0), "joy" (3), "love" (4), "sadness" (6), "surprise" (7), "optimism" (8) and "neutral" (13) emotions, while struggling to accurately predict the less frequent emotions as "disgust" (1), "pride" (5), "embarrassment" (10), and "desire" (11).
- It is also worth noting that there is considerable confusion between some pairs of emotions, as well as a bias towards the "neutral" (13) sentiment, possibly due to the large amount of "neutral" text in the dataset.
- In conclusion, the model demonstrates decent performance for some emotions. However, there is room for improvement, particularly for less frequent emotions.

For figure 5 the highest AUC values of 0.89 were obtained by "fear", "curiosity" and "love" demonstrating good model performance in identifying these emotions. Robust findings were also seen for "disgust", "sadness", "confusion", and "desire", with AUC values of 0.85 or higher. With AUC values around 0.80, emotions like "anger", "pride", and "embarrassment" performed moderately. However, with AUC values of 0.78, 0.75, and 0.71, respectively, the model had the worst time identifying "joy", "surprise", and "neutral" emotions.

The difference between the two plottings is that the confusion matrix provides a more detailed view of the classification results, indicating how many instances for each class were correctly or incorrectly classified. It aids in determining which classes the model excels at and which classes it struggles with. On the other hand, ROC curves depict the trade-off between the actual positive rate (sensitivity) and the false positive rate (1 - specificity) at various classification thresholds.

The confusion matrix 6 for the BiLSTM model highlights that the model strongly tends to predict the "neutral" class, with many instances of other emotions being misclassified as "neutral". This is particularly evident for emotions like "anger", "disgust", "fear", "pride", "surprise", "optimism", "confusion", "embarrassment", "desire", and "curiosity", where most instances are predicted as "neutral". However, the model performs relatively better in classifying emotions like "joy" and "love", correctly identifying 157 and 417 instances, respectively. Despite this, many instances are still misclassified as "neutral" for these emotions.

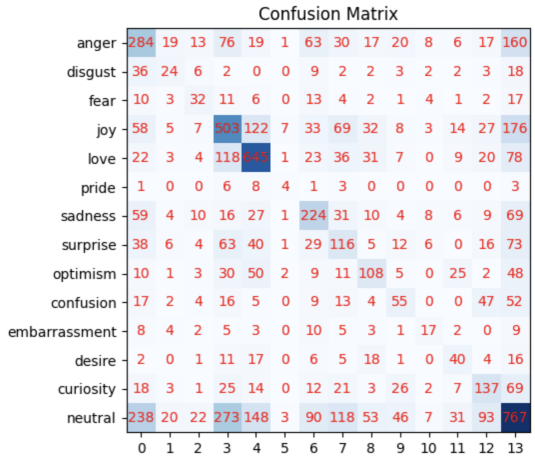


Figure 4: Confusion Matrix Roberta pre text preprocessing.

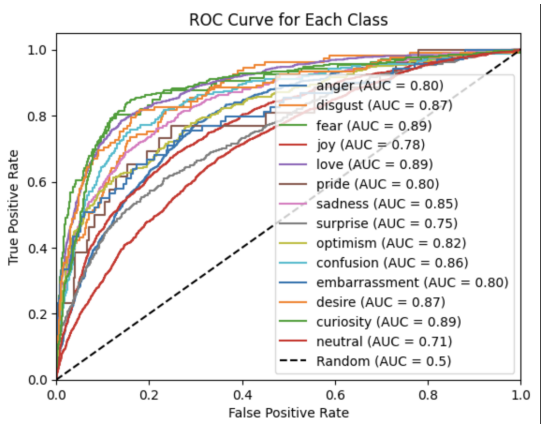


Figure 5: ROC roBERTa pre text preprocessing.

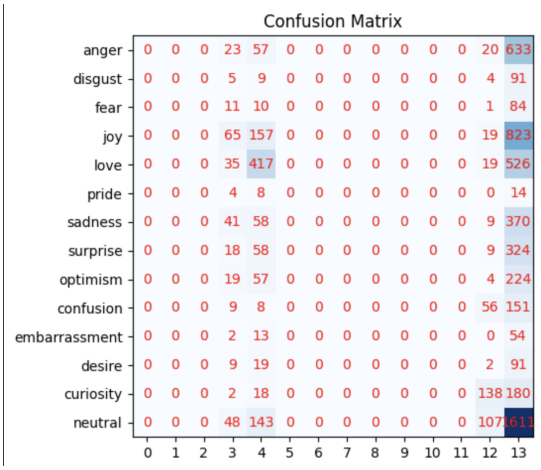


Figure 6: Confusion Matrix BiLSTM pre text preprocessing.

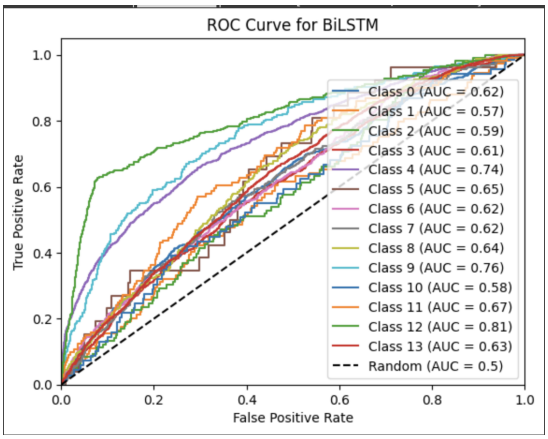


Figure 7: ROC BiLSTM pre text preprocessing.

Evaluation after data preprocessing

When the results of the roBERTa model before 4 and after 8 data preprocessing are compared, we can conclude that the model performs similarly in predicting emotions such as "anger" (0), "joy" (3), "love" (4), "sadness" (6), "surprise" (7), "optimism" (8), and "neutral" (13) and still struggles to predict the rest of the emotions with a bias towards "neutral."

The same applies to The BiLSTM model, with only slight differences in values before 6 and after 10 the data being procesed.

Overlall, as shown in Table 4, the RoBERTa model outperforms the BiLSTM model in terms of all the evaluation metrics including before and after data preprocessing. Also is important to note that pre-processing the data resulted in negligibly worse results for both models.

Although the BiLSTM produced slightly worse metrics for the pre-processed data, we can see that the gap between the training and validation loss results Apendix 35 36 is smaller for the pre-processed data, indicating that the model fits this data better. If the number of epochs increases, the model may produce better long-term results.

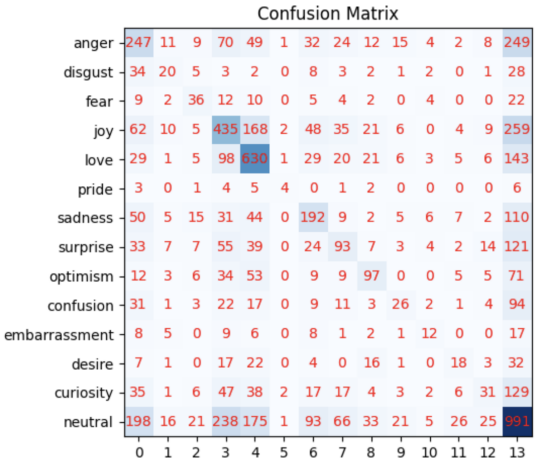


Figure 8: Confusion Matrix roBERTa after text preprocessing.

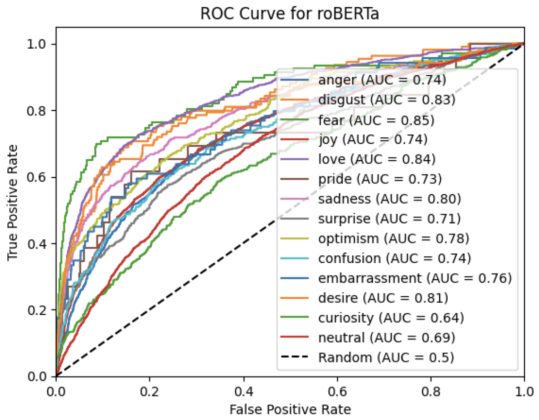


Figure 9: ROC roBERTa after text preprocessing.

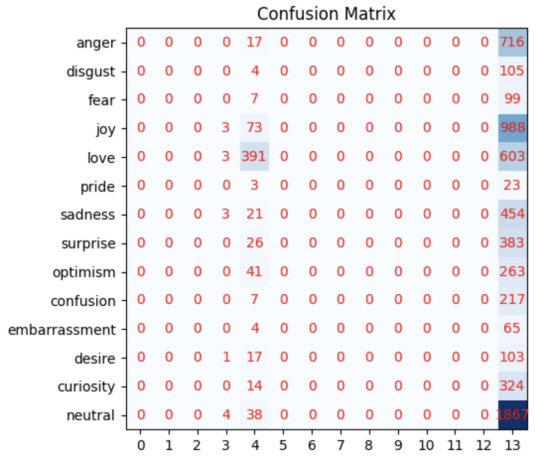


Figure 10: Confusion Matrix BiLSTM after text preprocessing.

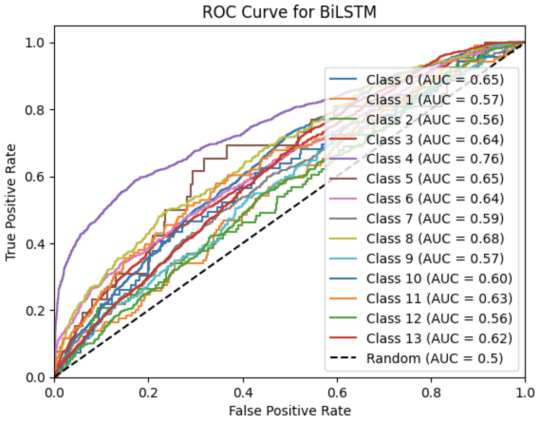


Figure 11: ROC BiLSTM after text preprocessing.

4 Experimental variation 2

Train, test and validate dataset splitting

In this second variation, I will try two diffrent approaches.

Firstly I will investigate the impact of different data proportions on the performance of the two models. Due to the complexity of the RoBERTa model and the constraints of using Google Colab resources, I will keep the number of epochs at 5. This model requires a significant amount of time to train, approximately 1 hour and 12 minutes. The BiLSTM model, on the other hand, is simpler. Because I am not using any pre-trained weights, I will increase the number of epochs to 30 for a more fair comparison.

Secondly, to address the issue of the imbalanced dataset, I will remove some texts from the over-represented classes to create a more balanced distribution. However, I will still maintain a more prominent presence of the neutral sentiment, as it was in the original dataset.

4.1 Model Architecture and Hyperparameters

RoBERTa Model

For this test, the model architecture will remain the same, and only the following hyperparameters will change:

- Data Split: Training: 90%(51646), Validation: 5%(2869), Testing: 5%(2870)
- Number of parameters: 124,656,398

BiLSTM Model

For this test, the model architecture will remain the same, and only the following hyperparameters will change:

- Data Split: Training: 90%(51646), Validation: 5%(2869), Testing: 5%(2870)
- Number of parameters: 6,340,626

4.2 Results

Table 5: Performance/training parameters of RoBERTa/BiLSTM models

	Loss	Accuracy	F1 Score	Precision	Recall
Metrics for train/test/validate split					
RoBERTa	1.62	46.95	37.82	38.93	37.73
BiLSTM	5.92	38.81	31.73	60.93	38.81
Metrics for balanced dataset					
RoBERTa	2.26	41.81	41.25	42.04	41.00
BiLSTM	9.22	23.91	19.74	56.35	23.91
Training Parameters					
Optimizer			AdamW		
Loss			Cross-Entropy		
Epochs RoBERTa			5		
Epochs BiLSTM			30		
Weight Decay			0.01		
Batch Size			32		
Learning Rate			5e-5		

4.3 Model Evaluation

4.3.1 Train, test and validate dataset splitting

As expected, with the increase in training data from the table 5, we can observe better results for both models. The RoBERTa model outperforms the BiLSTM model regarding all metrics, loss, accuracy,

F1 score, precision and recall. This comparison highlights the effectiveness of RoBERTa's performance, which leverages pre-trained representations, in achieving better overall performance than the simpler BiLSTM model without pre-trained weights.

Looking at the confusion matrix 12 for the RoBERTa model in this experiment, we can conclude that the model does well in recognising the sentiments "anger" (0), "joy" (3), "love" (4), "sadness" (6), "surprise" (7), "optimism" (8), and "neutral" (13) while also showing improved results for the "curiosity" (12) class. However, there is still a clear bias towards "neutral" sentiment.

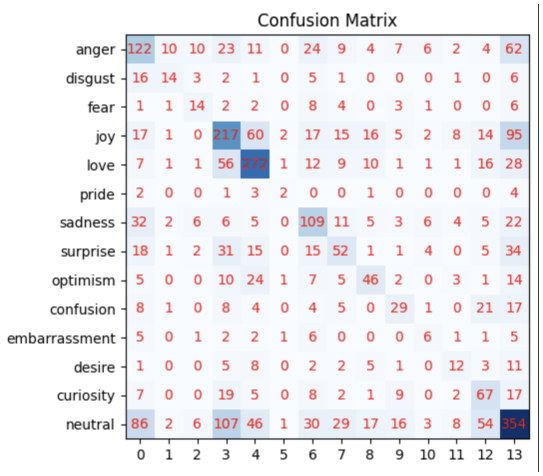


Figure 12: Confusion Matrix Roberta.

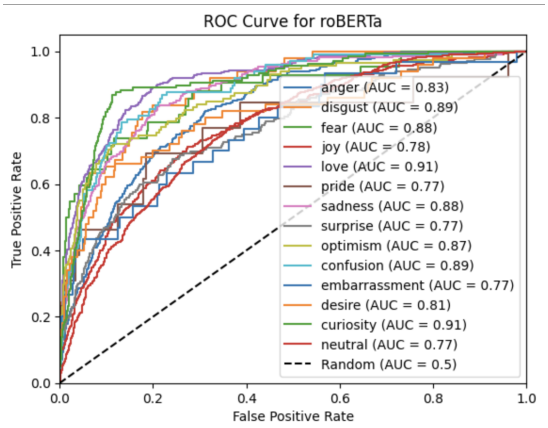


Figure 13: ROC roBERTa.

The detection of "joy," "love," "neutral," and "curiosity" sentiments has improved for the BiLSTM model 14. However, a significant bias towards "neutral" sentiment indicates a potential area for further refinement.

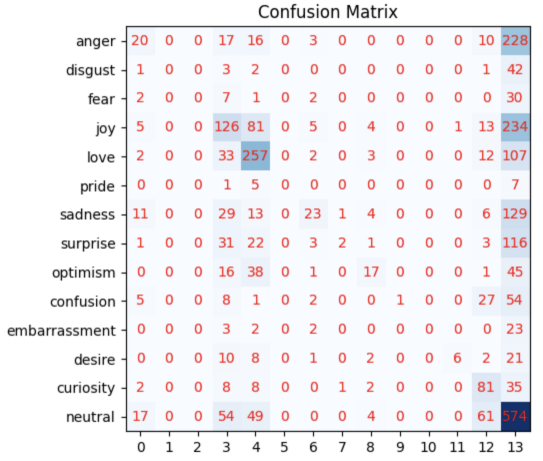


Figure 14: Confusion Matrix BiLSTM.

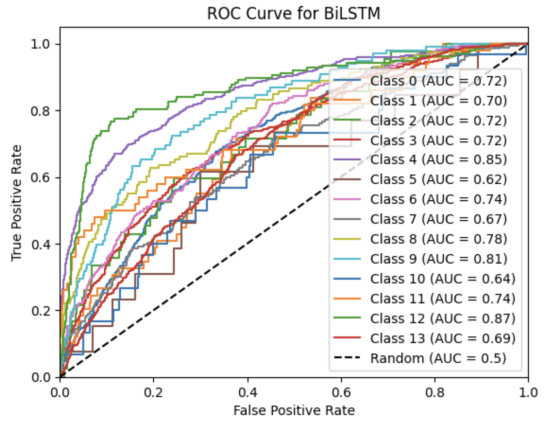


Figure 15: ROC BiLSTM.

4.3.2 Balanced dataset

A certain percentage of rows from each class label has been removed to balance the dataset, primarily by visualising the current distribution. The dataset was initially divided into separate DataFrames based on their class labels. The number of rows to be removed from each class was calculated using the specified removal percentages. These removal percentages were tailored to each class to ensure a more balanced data distribution across all classes while maintaining the prominence of neutral sentiment as in the original dataset, as seen in Figure 16.

Better Balanced data:

- Data Split: Training: 90%(12752), Validation: 5%(708), Testing: 5%(709)

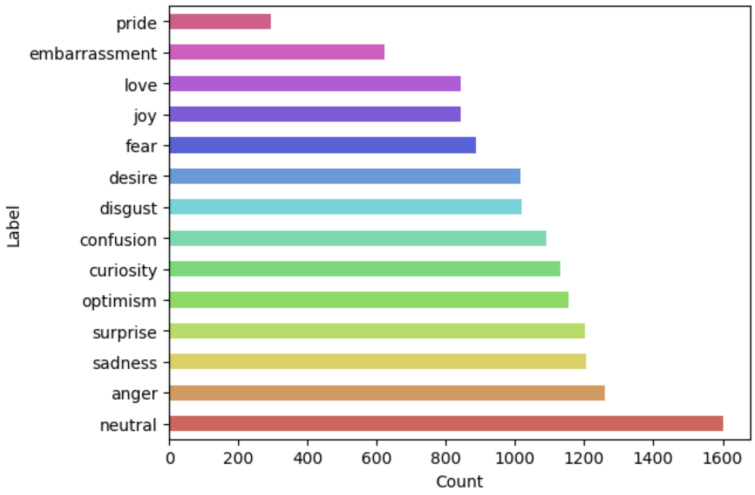


Figure 16: Balanced Dataset.

Although there is not a significant difference in the results of the two models when comparing the unbalanced dataset to the newly balanced data, the confusion matrices reveal improved performance on the balanced dataset. For the RoBERTa model, misclassifications still occur, such as "curiosity" (12) being misidentified as "confusion" (9). However, there is a notable reduction in other classes being misclassified as "neutral" (13) . The BiLSTM model continues to show a high rate of "neutral" (13) sentiment misclassification but demonstrates better results for sentiments like "sadness" (6), "desire" (11), and "curiosity" (12).

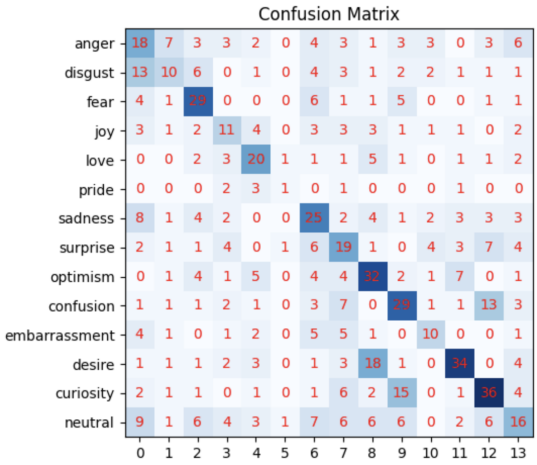


Figure 17: Confusion Matrix RoBERTa.

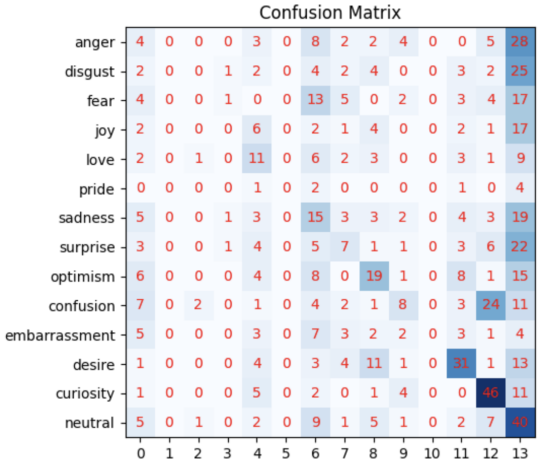


Figure 18: Confusion Matrix BiLSTM.

An additional important observation is that both models tend to overfit the data. As illustrated in Figures 37 and 38 in the Appendix, the validation loss consistently remains significantly higher than the training loss throughout the model training process.

5 Experimental variation 3

Loss Functions and Optimizers

This experiment focuses on the BiLSTM model. Evaluating different loss functions and optimizers using more sophisticated models, such as RoBERTa, will significantly increase the computational time required for each experiment, making exploring various combinations impractical to run on Colab resources. Considering the multiclass nature of the problem, two suitable loss functions will be explored: CrossEntropyLoss and KLDivLoss.

- CrossEntropyLoss: Cross-entropy loss is commonly used for classification tasks because it measures the difference between the predicted and true probability distributions Wikipedia (2023). Since it handles multiple output classes by default, this loss function is well-suited for our multiclass problem.
- KLDivLoss: The difference between two probability distributions is measured by the Kullback-Leibler (KL) divergence loss, also known as relative entropy. In this case, it quantifies the difference between the predicted and true class probabilities Brownlee (2019). It is worth noting that KLDivLoss requires one-hot encoding of the ground truth labels and log probabilities for the predicted probabilities.

In previous experiments, the AdamW optimizer was mainly employed. However, to explore the impact of different optimizers on the performance of the BiLSTM model, I will now conduct experiments using three different optimizers: Adam, Stochastic Gradient Descent (SGD), and RMSprop.

5.1 Model Architecture and Hyperparameters

BiLSTM Model

As before, the model's architecture will be kept simple to ensure computational efficiency during multiple experiments. The only modification required when using KLDivLoss is to adjust the model's output to produce log probabilities. To achieve this, we add a LogSoftmax layer at the end of our BiLSTM model. Additionally, the ground truth labels must be converted to one-hot encoding, which is necessary when calculating the loss using KLDivLoss.

5.2 Results

Table 6: Performance/training parameters of BiLSTM model

	Loss	Accuracy	F1 Score	Precision	Recall
BiLSTM with CrossEntropyLoss					
Adam	1.46	43.12	41.14	45.72	43.12
SGD	1.69	29.42	15.26	70.75	29.42
RMS	1.42	44.46	42.15	47.71	44.61
BiLSTM with KLDivLoss					
Adam	1.48	43.64	41.84	46.44	43.64
SGD	1.69	30.48	17.01	71.41	30.48
RMS	1.43	44.47	41.75	48.11	44.47
Training Parameters					
Epochs			30		
Batch Size			128		
Learning Rate			0.001		
Number of parameters			12,648,466		

The results presented in Table 6 show the performance of the BiLSTM model with different loss functions and optimizers. The BiLSTM model with KLDiv loss and the RMSProp optimizer achieved the best performance in terms of accuracy (44.47%), F1 score (41.75%), and precision (48.11%). The next best performance was obtained by the BiLSTM model with KLDiv loss and the Adam optimizer, which had

slightly lower metrics values. The model with the lowest performance utilized CrossEntropy loss and the SGD optimizer. Overall, SGD consistently underperformed as an optimizer for both loss functions. At the same time, RMSprop demonstrated the best performance among the optimizers tested.

5.3 Model Evaluation

As evidenced by the confusion matrices presented in Figures 21 to 22, it is clear that the SGD optimizer underperformed compared to Adam and RMSprop. Both Adam and RMSprop were successful at identifying sentiments such as "anger" (0), "joy" (3), "love" (4), "sadness" (6), "surprise" (9), "optimism", and "neutral". However, the bias towards "neutral" classifications persisted, primarily due to the imbalanced data distribution. Contrarily, the SGD optimizer struggled to classify sentiments beyond "neutral" (13) and "love" (4).

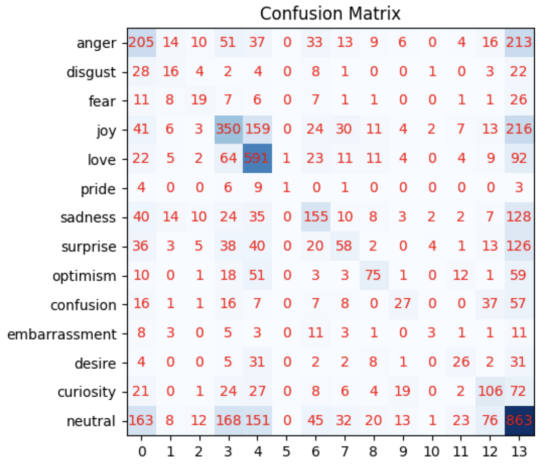


Figure 19: Adam CrossEntropyLoss.

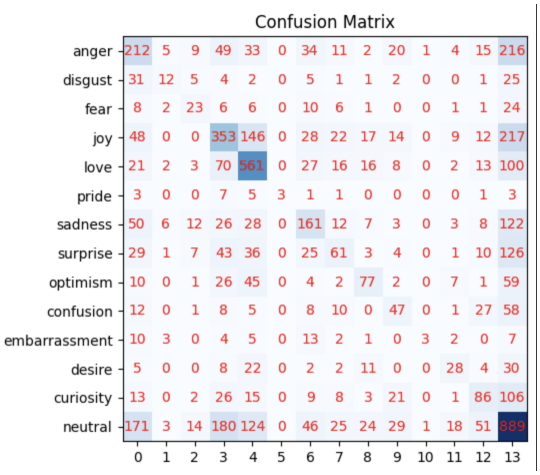


Figure 20: Adam KLDivLoss.

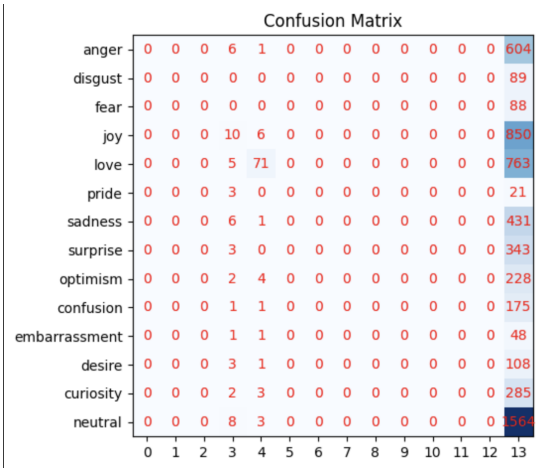


Figure 21: SGD CrossEntropyLoss.

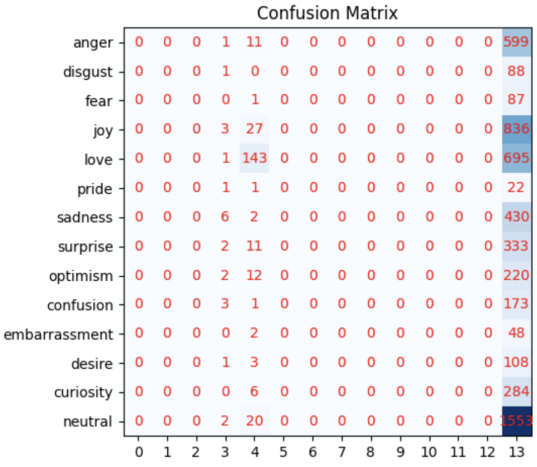


Figure 22: SGD KLDivLoss.

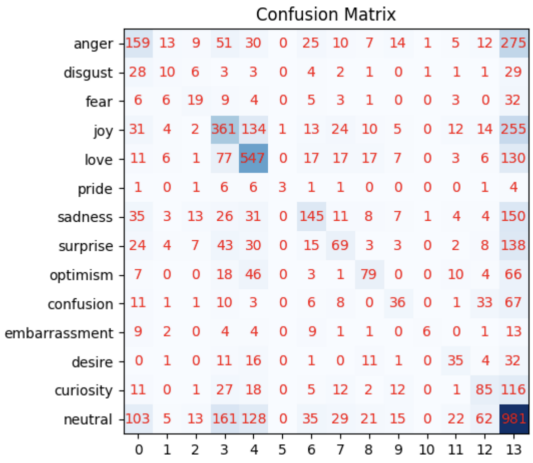


Figure 23: RMSprop CrossEntropyLoss.

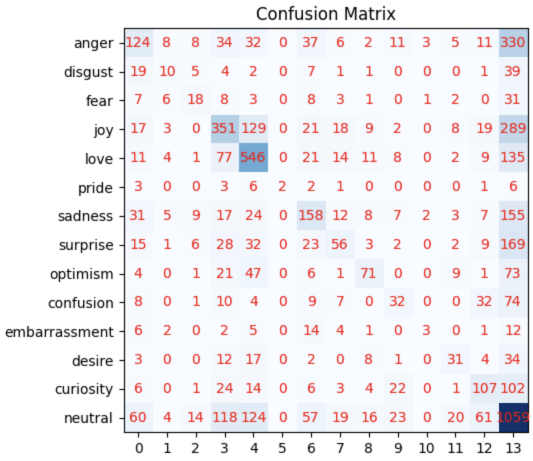


Figure 24: RMSprop KLDivLoss.

6 Experimental variation 4

Different fine-tuned models

In this experiment, the aim is to analyse the impact of fine-tuning on pre-trained models performance. To achieve this, I will employ a BiLSTM model with pre-trained vectors and increased numbers of layers, similar to the model in lab 06-1b, alongside RoBERTA, specifically utilising the "roberta-large" variant.

6.1 Model Architecture and Hyperparameters

RoBERTa Model

The RoBERTa model, utilising its large configuration (roberta-large), has been explicitly fine-tuned for this classification task. Compared to the initial experiment employing RoBERTa's base configuration, there is a substantial increase in the number of trainable parameters. This results in the parameter count rising from 123 million to 355 million, enhancing the model's capacity and potentially improving its performance on the task. In this final experiment, the increased demand for computational resources necessitated using a more powerful GPU. Consequently, an NVIDIA A100 40GB was employed for this task to reduce training time. The model's architecture and hyperparameters are described below:

- Tokenizer: RobertaTokenizerFast
- Max Sequence Length: 128 tokens, with padding and truncation
- Evaluation Metrics: accuracy, F1-score, precision, and recall.
- Data Split: Training: 90%(51646), Validation: 5%(2869), Testing: 5%(2870)
- Number of parameters: 355,374,094

BiLSTM Model

The model's architecture and hyperparameters are described below:

- GloVe"6B" pre-trained vectors
- Vocab size: 40233
- Hidden dim: 256
- Number of LSTM layers: 6
- Bidirectional: True
- Scheduler: linear schedule with warmup
- Evaluation Metrics: accuracy, F1-score, precision, and recall.
- Dropout: 0.5
- Data Split: Training: 90%(51646), Validation: 5%(2869), Testing: 5%(2870)
- Number of parameters: 12,648,466

6.2 Results

After experimentation with various hyperparameters for both models, I have determined that the following results represent the best configurations for each case. There is a slight noticeable performance improvement when comparing the RoBERTa base model from experiment 2 Table 5 with the current model Table 7. Reducing the batch size had the most significant impact on this model's performance. In contrast, for the BiLSTM with pre-trained vectors, the previous, simpler model surprisingly outperformed the current one. Although further optimization may lead to better results, computational constraints did not permit additional experimentation.

Table 7: Performance/training parameters of BiLSTM and RoBERTa models

	Loss	Accuracy	F1 Score	Precision	Recall
RoBERTa Large					
Best RoBERTa	1.54	48.50	40.41	44.76	39.73
Training Parameters					
Epochs			3		
Batch Size			16		
Learning Rate			3e-5		
Optimizer			AdamW		
Loss			Cross-Entropy		
BiLSTM					
Best BiLSTM	1.43	40.78	35.49	50.19	40.78
Training Parameters					
Epochs			100		
Batch Size			128		
Learning Rate			0.001		
Optimizer			Adam		
Weight Decay			0.0001		
Scheduler			Linear-Schedule		
Loss			Cross-Entropy		

6.3 Model Evaluation

Upon examining the confusion matrix for the BiLSTM model with the Adam optimizer and CrossEntropy loss in Figures 19 (experiment 3) and 26, we observe only minimal differences compared to the previous configuration. The model continues to display a bias towards the "neutral" sentiment while performing best for sentiments with the highest number of texts in the dataset, such as "anger," "joy," "love," and "sadness." A similar pattern is also evident in the RoBERTa model's performance Figures 4 25.

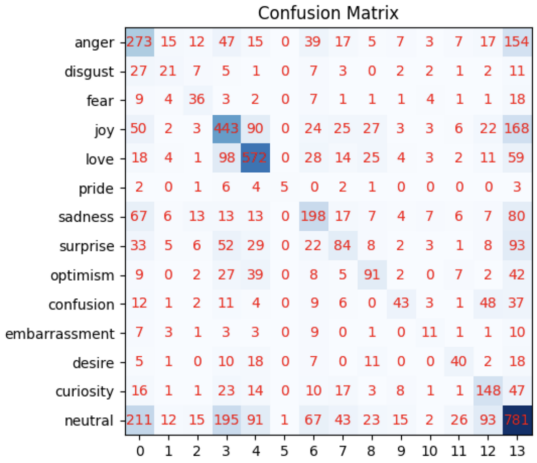


Figure 25: RoBERTa large

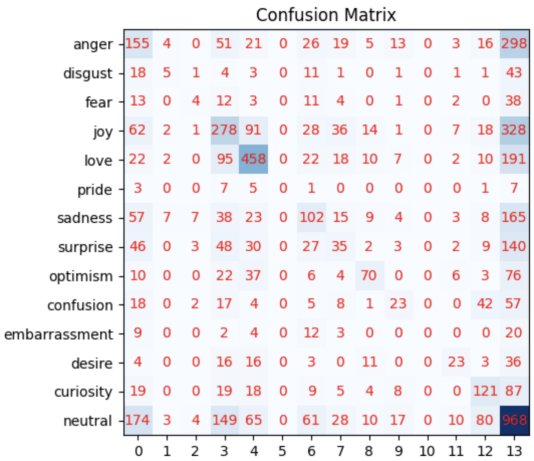


Figure 26: BiLSTM with GloVe vectors

7 Best results

Table 8: Best results

	Loss	Accuracy	F1 Score	Precision	Recall
Experimental Variation 1					
RoBERTa base	2.10	42.75	36.12	36.13	36.39
Training Parameters					
Optimizer			AdamW		
Loss			Cross-Entropy		
Epochs			5		
Weight Decay			0.01		
Batch Size			32		
Learning Rate			5e-5		
Data Split			70/18/12		
Experimental Variation 2					
RoBERTa base	1.62	46.95	37.82	38.93	37.73
Training Parameters					
Optimizer			AdamW		
Loss			Cross-Entropy		
Epochs			5		
Weight Decay			0.01		
Batch Size			32		
Learning Rate			5e-5		
Data Split			90/5/5		
Experimental Variation 3					
BiLSTM	1.43	44.47	41.75	48.11	44.47
Training Parameters					
Optimizer			RMSProp		
Loss			KLDivLoss		
Epochs			30		
Batch Size			128		
Learning Rate			0.001		
Data Split			90/5/5		
Experimental Variation 4					
RoBERTa large	1.54	48.50	40.41	44.76	39.73
Training Parameters					
Optimizer			AdamW		
Loss			Cross-Entropy		
Epochs			3		
Batch Size			16		
Learning Rate			3e-5		
Data Split			90/5/5		

In terms of accuracy, we can observe from Table 6 that Experiment 4 (RoBERTa-large) is the best-performing model with an accuracy of 48.50%. However, regarding the other metrics, Experiment 3 has achieved the best F1 score (41.75%), Precision 48.11% and Recall 44.47%.

Experimental Variation 1

As anticipated, RoBERTa-base has produced superior results in this experiment, mainly due to its pre-trained weights. As illustrated in Figure 27, it is evident that the model begins to overfit the data from the very first epoch. To handle this issue and improve model accuracy, I would fine-tune the learning rate and weight decay to prevent overfitting. Additionally, adjusting the batch size could potentially

contribute to the improvement of the model's performance.

[6280/6280 1:08:33, Epoch 5/5]

Step	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1256	1.044800	1.815306	45.096331	38.473481	42.675039	37.008191
2512	1.255600	1.647397	46.780908	39.455418	40.855881	39.231456
3768	1.095600	1.739980	45.551360	38.083230	41.764086	37.458619
5024	0.858900	1.951732	43.450479	36.788048	37.617448	36.808909
6280	0.677300	2.101851	42.753413	36.123762	36.132038	36.385757

Figure 27: Experiment 1 Metrics.

Experimental Variation 2

Similar to the previous experiment, adjusting the data split led to RoBERTa outperforming the BiLSTM model while also demonstrating enhanced performance compared to the same model in the previous experiment. As seen in figures 29, overfitting begins after approximately 3k steps, corresponding to around three epochs. Like the previous experiment, fine-tuning the learning rate and batch size would likely improve performance and prevent overfitting in this scenario.

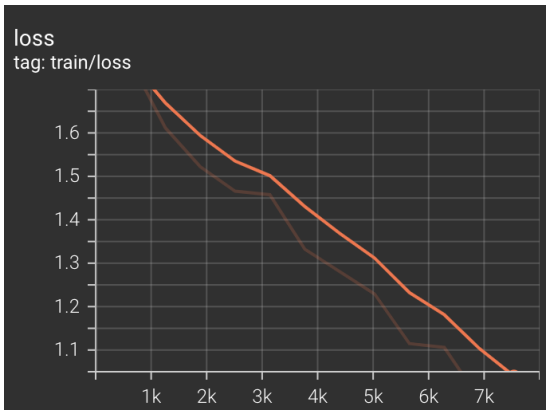


Figure 28: Train Loss

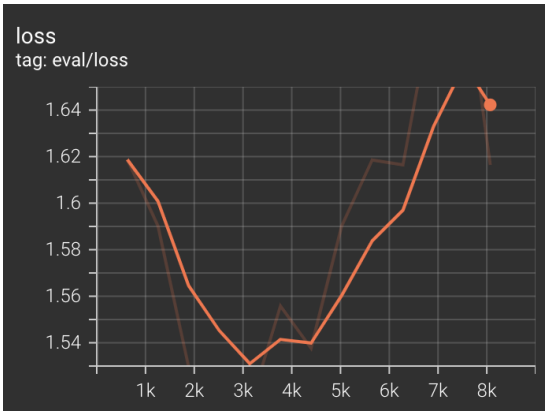


Figure 29: Validation Loss

Another thing to note is that the best performance for both experiments one 27 and two 30 is achieved during the first epoch. Therefore limiting the number of epochs for this model could be beneficial in terms of performance and computational efficiency.

[8070/8070 1:26:30, Epoch 5/5]

Step	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
628	1.764900	1.618737	46.253050	32.205737	55.824094	31.294059
1256	1.611800	1.590106	46.392471	32.925918	43.698113	32.685286
1884	1.522200	1.529788	48.867201	37.347276	44.621029	35.903722
2512	1.465800	1.522576	48.658069	37.085739	43.354472	35.903895
3140	1.457700	1.512423	48.623214	37.709055	41.957161	36.856019
3768	1.332200	1.555826	47.716975	37.349127	42.376458	36.499855
4396	1.280700	1.537686	48.135239	37.820777	40.006051	37.392317
5024	1.229100	1.590195	49.076333	38.856450	41.198691	38.189396
5652	1.114800	1.618568	47.472987	37.648363	38.770505	37.428259
6280	1.106100	1.616464	46.950157	37.817141	38.927020	37.728628
6908	0.988400	1.686820	47.194144	38.148539	39.529260	37.715401
7536	0.944600	1.698969	46.531893	38.708818	40.322355	38.527350

Figure 30: Experiment 2 Metrics.

Experimental Variation 3

In this experiment, various loss functions and optimizers were evaluated using the simple BiLSTM model. A slightly increased batch size and the number of epochs enabled a more ample comparison. The best performance was achieved using the RMSProp optimizer in conjunction with KLDivLoss. While there was a notable performance improvement compared to the same BiLSTM model in experiments 1 and 2, overfitting remained an issue, as evidenced in Figure 31. Despite its relatively small architecture, this model still has the potential for further enhancement through fine-tuning hyperparameters such as learning rate, batch size, layers number, and epoch count. The overfitting problem is addressed in the subsequent experiment.

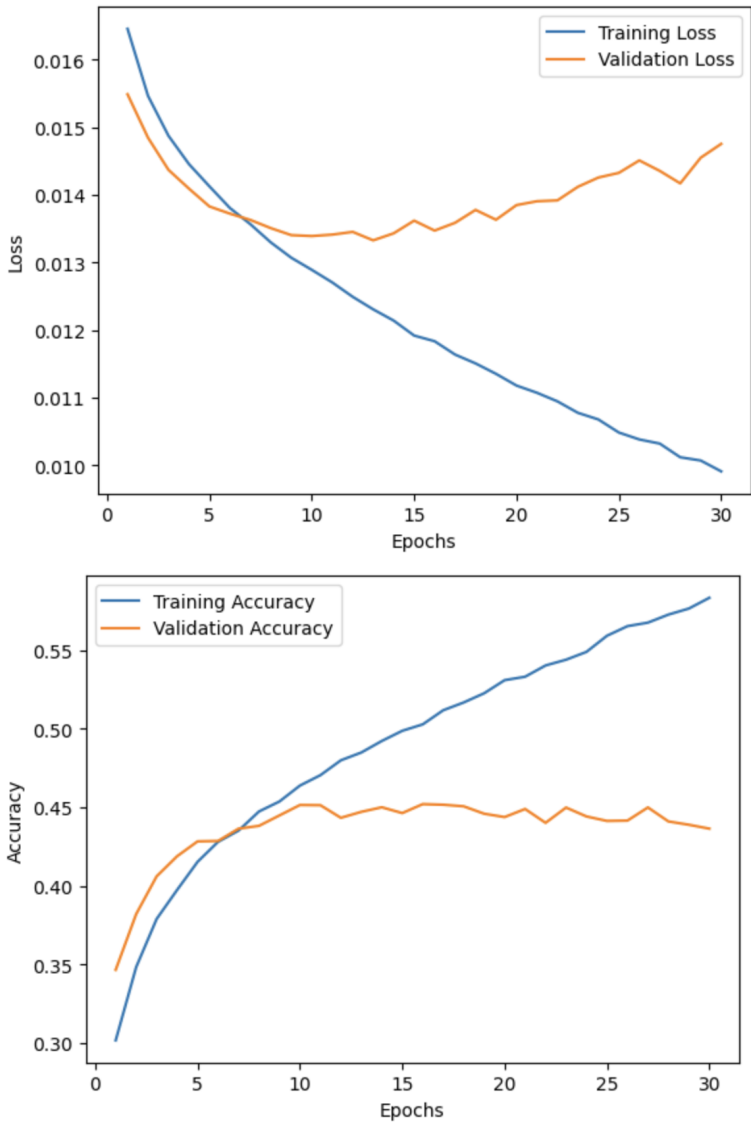


Figure 31: Train and Validation Loss/Accuracy.

Experimental Variation 4

In this experiment, both RoBERTa-large and BiLSTM with pre-trained vectors were fine-tuned to achieve optimal results. RoBERTa outperformed with the highest accuracy of 48.50%, while BiLSTM demonstrated better precision and recall. Compared to previous experiments, the RoBERTa model was trained with more parameters and a smaller batch size, learning rate, and number of epochs. Figures 32 33 shows that although overfitting persists, it is less pronounced than in earlier experiments. This model's performance could be further enhanced by fine-tuning hyperparameters or employing data augmentation techniques; however, such improvements come with high computational costs.

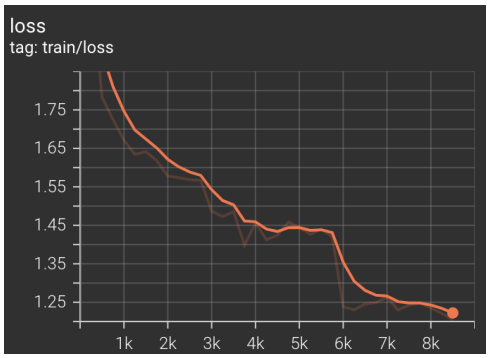


Figure 32: Train Loss

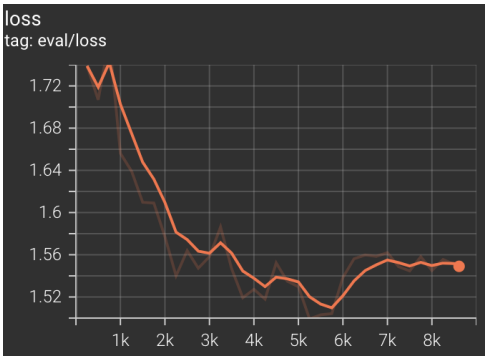


Figure 33: Validation Loss

In the fourth experiment, please see Table 7, while the BiLSTM model may not have achieved the highest accuracy, Figure 34 reveals that the model demonstrates a stronger ability to generalise to the validation set and effectively addresses the overfitting issue. This improvement was achieved by fine-tuning the learning rate, utilising a linear schedule, and incorporating regularization within the Adam optimiser.

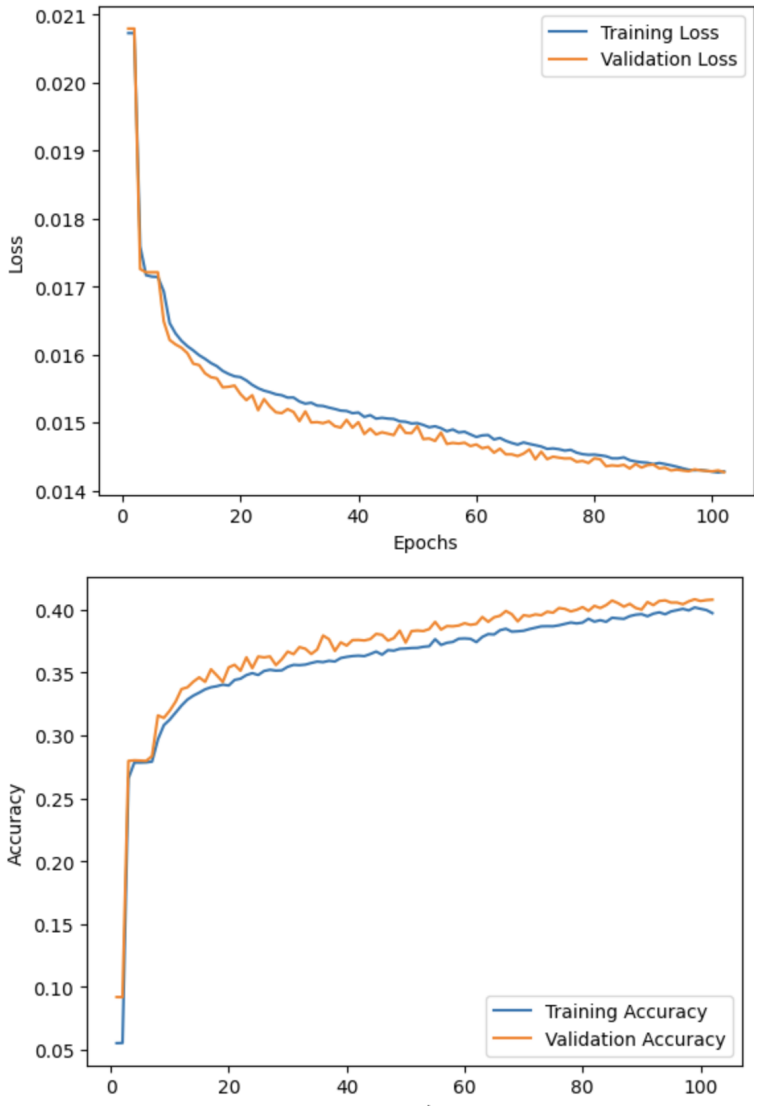


Figure 34: Train and Validation Loss/Accuracy.

8 Overall attempt and outcome

Can the models you built fulfil their purpose?

After training each model, as detailed in Appendix 1 and 2, they were tested on a challenging text phrase to evaluate their performance on new data. The phrase, *"It is kinda sad that this made me lol"* could be classified as either "sadness" or "amusement". The predictions for the best-performing models can be seen in the table. Notably, the only models that predicted a different "neutral" category were the BiLSTM models in 5 for the balanced dataset and 6 with SGD optimizer for both Cross Entropy and KLDiv loss functions.

Model	Experimental variation	Predicted Sentiment
RoBERTa base	1	sadness
RoBERTa base	2	sadness
BiLSTM	3	sadness
RoBERTa large	4	sadness

Table 9: Model predictions

As demonstrated by the test phrase above, human emotions are complex, and accurately classifying them based solely on text can be challenging even for humans. While the models serve their purpose as proof of concept, they may not be suitable for real-world applications in their current state. Additional work and refinements should be made before deploying these models. Adopting a multi-label approach might be more effective for addressing such nuanced problems.

What is good enough accuracy?

The importance of accuracy in sentiment analysis depends on the specific task. For instance, in the financial sector, where the analysis of public sentiment surrounding stocks or company shares is crucial, a high accuracy of at least 90% is desirable. Misinterpreting the prevailing sentiment may lead to unwanted consequences. On the other hand, in scenarios such as movie reviews, where sentiment analysis aims to measure the general audience's reaction to a film, minor inaccuracies are more tolerable. In this case, at least 70% accuracy would be ideal.

If any of the models did not perform well, what is needed to improve?

To my disappointment, none of the models performed exceptionally well, and there is significant room for improvement. First, I am disappointed with the RoBERTa results, as I expected higher accuracy from a pre-trained model. Second, although the BiLSTM's overfitting issues have been addressed, the training and validation losses remain high, potentially indicating underfitting. Lastly, the primary factor contributing to these limitations was the raw dataset and the unbalanced class distribution.

I used the raw data, hoping a more extensive dataset would improve overall performance. However, after removing duplicates, the amount of available data was roughly equivalent to what would have been provided by the simplified version. Employing the simplified version and balancing the dataset by augmenting underrepresented classes with additional data from other sources could significantly enhance the results, even with the current architectures of my best-performing models.

If any of the models performed really well, could/would you make it more efficient and sacrifice some quality?

The only model that demonstrated relatively strong performance was RoBERTa-large. However, the efficiency gain was a mere 2% compared to RoBERTa-base. Therefore, it may be worthwhile to sacrifice some additional efficiency for a lighter-weight pre-trained model, such as RoBERTa-base, DistilBERT, ALBERT, or even a well-fine-tuned BiLSTM model with a less complex architecture. Ultimately, the choice of model depends on the specific task and the balance between performance and efficiency requirements.

A Additional Figures

Listing 1: roBBERTa prediction pre text preprocessing

```
text = "It is kinda sad that this made me lol."
print(predict_emotion(model,tokenizer,text))

#sadness
```

Listing 2: BiLSTM prediction pre text preprocessing

```
def predict_sentiment(model, sentence, device):
    model.eval()
    with torch.no_grad():
        tokenized_sentence = text_pipeline(sentence)
        processed_sentence = torch.tensor(tokenized_sentence, dtype=torch.int64
            ).unsqueeze(0).to(device)
        prediction = model(processed_sentence)
        probabilities = torch.softmax(prediction, dim=1)
        predicted_class = torch.argmax(probabilities, dim=1).item()
    return predicted_class, probabilities

text = "It is kinda sad that this made me lol."
predicted_class, probabilities = predict_sentiment(model, text, device)
print(f"Predicted class: {class_labels[predicted_class]}, \n Probabilities: {
    probabilities}")

Predicted class: #neutral,
Probabilities: tensor([[0.1280, 0.0289, 0.0275, 0.1263, 0.0907, 0.0123, 0.0919,
    0.0731, 0.0554,
         0.0439, 0.0227, 0.0246, 0.0451, 0.2296]], device='cuda:0')
```

Listing 3: BiLSTM prediction after text preprocessing

```
Predicted class: #neutral,
Probabilities: tensor([[0.1406, 0.0186, 0.0163, 0.1350, 0.0673, 0.0054, 0.0902,
    0.0617, 0.0360,
         0.0373, 0.0120, 0.0147, 0.0466, 0.3184]], device='cuda:0')
```

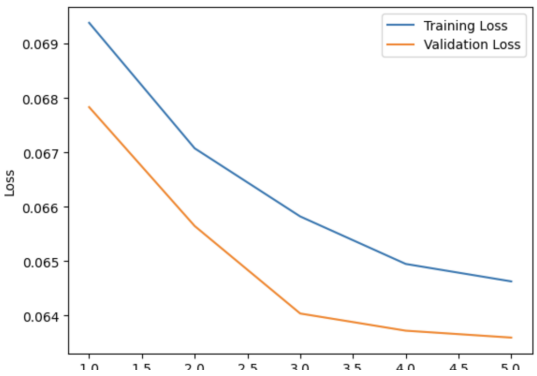


Figure 35: Losses BiLSTM before text preprocessing.

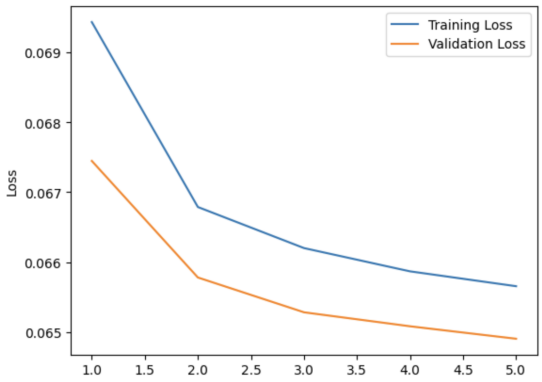


Figure 36: Losses BiLSTM after text preprocessing.

Step	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
217	1.295400	1.928016	42.090395	42.068016	45.751010	41.458140
434	1.239500	2.041315	42.937853	42.105946	45.720398	41.713013
651	0.949300	2.097490	41.384181	41.380576	43.889149	40.640408
868	0.852800	2.166163	42.372881	41.578669	42.508053	41.533278
1085	0.654300	2.259953	41.807910	41.255296	42.036253	41.001823

Figure 37: Metrics RoBERTa.

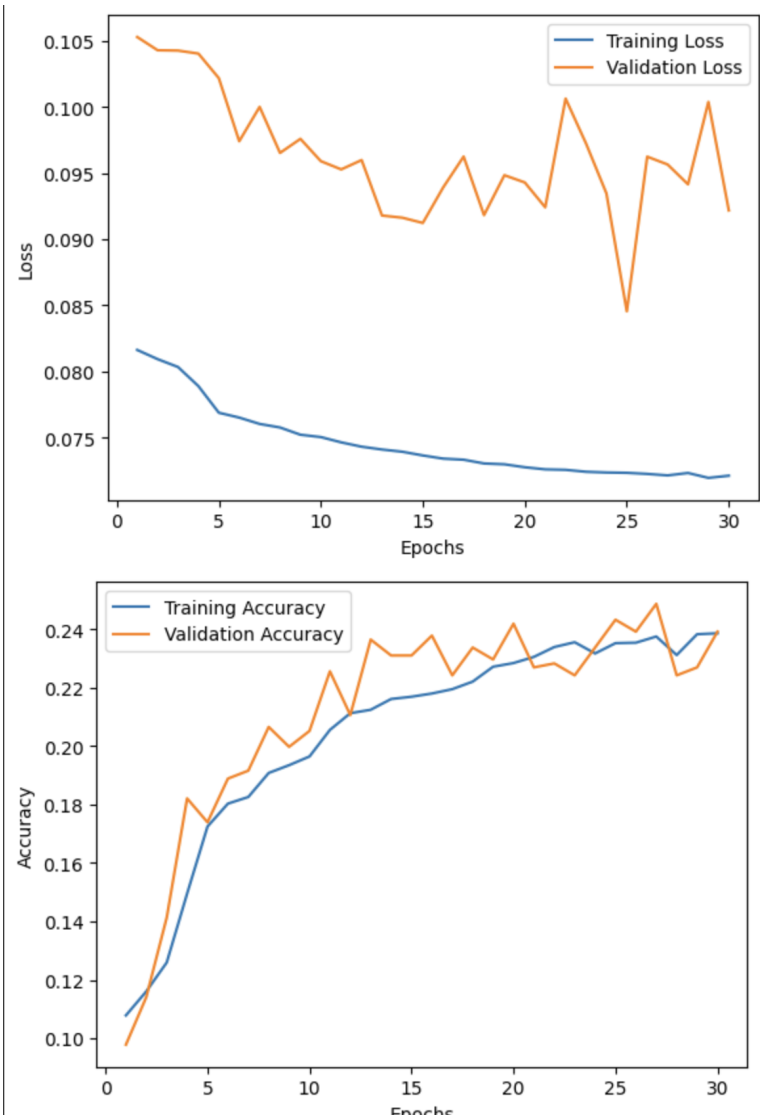


Figure 38: Loss/Accuracy BiLSTM.

References

Brownlee, Jason (2019). *How to Calculate the KL Divergence for Machine Learning*. Accessed: 2023-04-12. URL: https://machinelearningmastery.com/divergence-between-probability-distributions/.

Demszky, Dorottya et al. (2020). *GoEmotions: A Dataset of Fine-Grained Emotions*. arXiv: 2005.00547 [cs.CL].

Devlin, Jacob et al. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv: 1810.04805 [cs.CL].

Liu, Yinhan et al. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv: 1907.11692 [cs.CL].

Trainer (2023). *Trainer*. Accessed: 2023-04-05. URL: https://huggingface.co/docs/transformers/main_classes/trainer#transformers.Trainer.

Wikipedia (2023). *Cross entropy*. Accessed: 2023-04-05. URL: https://en.wikipedia.org/wiki/Cross_entropy.