COM3029 & COMM061 Individual Coursework Report

Moiz Saifuddin Nagaria mn01030@surrey.ac.uk

Abstract

The purpose of this report is to discuss the development of a multi-class classifier prototype for the GoEmotions dataset. This dataset contains 58,000 comments extracted from Reddit, which have been human-annotated into 27 emotion categories, including a neutral category. For this project, a subset of 13 emotion categories and the neutral category were selected as part of the experiment.

1. Introduction

Text classification is highly sought-after for solving various problems. It involves categorizing documents into topics, such as news articles on business, sports, politics, or other events. Another scenario is identifying spam in emails or determining the sentiment of messages on social media or product reviews on websites like Amazon. During inference, the input text may receive a single label (multi-class) or multiple labels (multi-label) depending on the application's requirements.

In this report, I present a detailed breakdown of my approach to achieve optimal performance in multi-class sentiment analysis. I have meticulously documented each step and provided justification for my decisions.

For this project, I opted to utilize TensorFlow as my framework of choice in order to further explore its available libraries and gain familiarity with the latest industry practices.

Link to Google Colab Notebook.

1.1. Labels

The Data-set has 27 + Neutral labels; As a part of the challenge the assignment demanded choosing only 13 + Neutral label. For which my group had chosen the following labels - 'love', 'gratitude', 'approval', 'disapproval', 'surprise', 'confusion', 'fear', 'sadness', 'realisation', 'joy', 'desire', 'optimism', 'pride' and 'neutral'.

For which I tried to map each emotion closely to of the P. Ekman's map of emotions and merged them into a superclass of emotion. 1

love	love	caring				
gratitude	gratitude					
approval	approval	admiration				
disapproval	disapproval	disgust	anger	annoyance		
surprise	surprise	curiosity				
confusion	confusion					
fear	fear	nervousness				
sadness	sadness	embarrassment	grief	remorse	disappointment	
realization	realization					
joy	joy	amusement	excitement	relief		
desire	desire					
optimism	optimism					
pride	pride					
neutral	neutral					

Figure 1. Mapping of emotion as closely as the Ekman's map of emotions [2]

admiration	amusement	approval	caring	anger	annoyance	disappointment	disapproval	confusion
great (42)	lol (66)	agree (24)	you (12)	fuck (24)	annoying (14)	disappointing (11)	not (16)	confused (18)
awesome (32)	haha (32)	not (13)	worry (11)	hate (18)	stupid (13)	disappointed (10)	don't (14)	why (11)
amazing (30)	funny (27)	don't (12)	careful (9)	fucking (18)	fucking (12)	bad (9)	disagree (9)	sure (10)
good (28)	lmao (21)	yes (12)	stay (9)	angry (11)	shit (10)	disappointment (7)	nope (8)	what (10)
beautiful (23)	hilarious (18)	agreed (11)	your (8)	dare (10)	dumb (9)	unfortunately (7)	doesn't (7)	understand (8)
desire	**excitement**	**gratitude**	**joy**	**disgust**	**embarrassment**	**fear**	**grief**	**curiosity**
wish (29)	excited (21)	thanks (75)	happy (32)	disgusting (22)	embarrassing (12)	scared (16)	died (6)	curious (22)
want (8)	happy (8)	thank (69)	glad (27)	awful (14)	shame (11)	afraid (16)	rip (4)	what (18)
wanted (6)	cake (8)	for (24)	enjoy (20)	worst (13)	awkward (10)	scary (15)		why (13)
could (6)	wow (8)	you (18)	enjoyed (12)	worse (12)	embarrassment (8)	terrible (12)		how (11)
ambitious (4)	interesting (7)	sharing (17)	fun (12)	weird (9)	embarrassed (7)	terrifying (11)		did (10)
love	**optimism**	**pride**	**relief**	**nervousness**	**remorse**	**sadness**	**realization**	**surprise**
love (76)	hope (45)	proud (14)	glad (5)	nervous (8)	sorry (39)	sad (31)	realize (14)	wow (23)
loved (21)	hopefully (19)	pride (4)	relieved (4)	worried (8)	regret (9)	sadly (16)	realized (12)	surprised (21)
favorite (13)	luck (18)	accomplishment	relieving (4)	anxiety (6)	apologies (7)	sorry (15)	realised (7)	wonder (15)
loves (12)	hoping (16)	(4)	relief (4)	anxious (4)	apologize (6)	painful (10)	realization (6)	shocked (12)
like (9)	will (8)			worrying (4)	guilt (5)	crying (9)	thought (6)	omg (11)

Figure 2. Top 5 words associated with each emotion [1]

2. Data Analysis & Visualisation

GoEmotions[1] is a corpus of 58k carefully curated comments extracted from Reddit, with human annotations to 27 emotion categories or Neutral. [1]

- Number of examples: 58,009.
- Number of labels: 27 + Neutral.
- Maximum sequence length in training and evaluation datasets: 30.

2.1. Analysis

The heatmap in Figure 3 shows that emotions that are related in intensity (e.g. annoyance and anger, joy and excitement, nervousness and fear) have a strong positive correlation. On the other hand, emotions that have the opposite sentiment are negatively correlated.

[1]Data and Code available at https://github.com/google-research/google-research/tree/master/goemotions

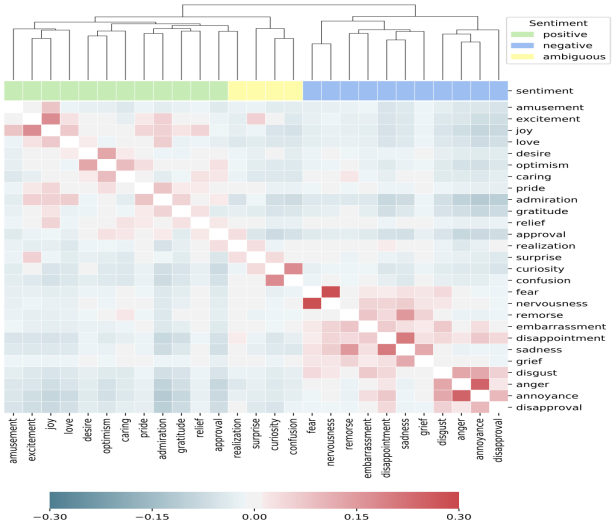


Figure 3. The heatmap shows the correlation between ratings for each emotion. The dendrogram represents the a hierarchical clustering of the ratings. The senti- ment labeling was done 'a priori' and it shows that the clusters closely map onto sentiment groups. [1]

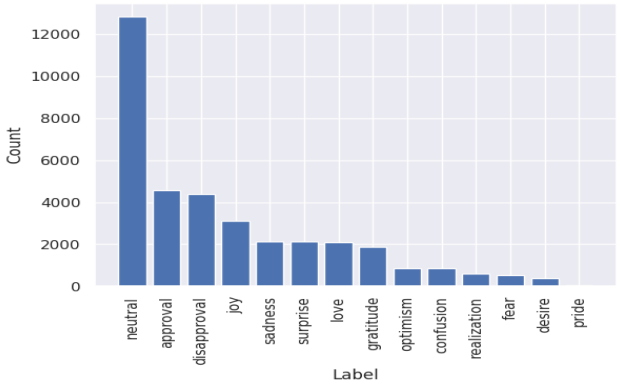


Figure 4. Histogram to represent basic distribution frequency of all the labels in the merged dataset for training data

From the histogram generated we can observe how unequal are the frequencies of the labels within the data in figure 4. We will further see the significance of this discrepancy in the F1-score 1 for each of the labels.

One interesting finding from our analysis in figure 5 is that each label or emotion appears to be associated with a distinct set of words. Specifically, when we examine the most common words for each label, we observe that these words tend to be unique to that label, rather than shared across multiple labels. This suggests that our dataset contains strong signals for distinguishing between different emotions based on the frequency of certain words. Overall, this analysis highlights the power of natural language

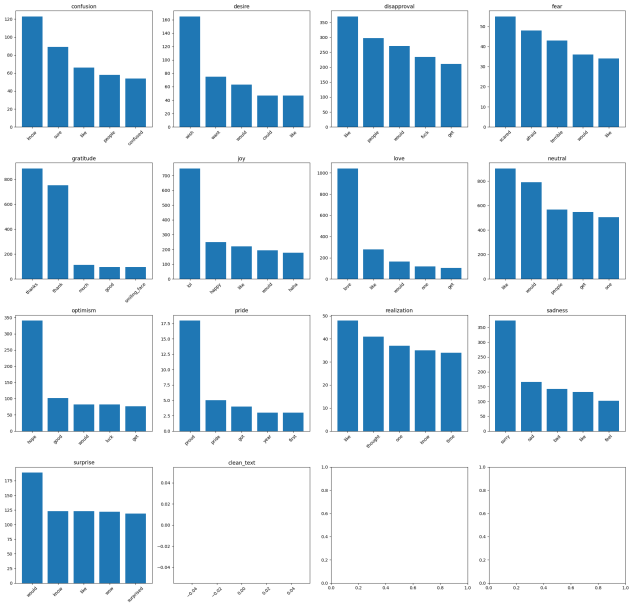


Figure 5. Word frequency for each label

processing techniques in identifying patterns and trends in large datasets, and underscores the importance of carefully selecting and pre-processing textual data in order to maximize the accuracy and relevance of our results.

Label	Number of Samples
neutral	12823
approval	4583
disapproval	4376
joy	3103
sadness	2121
surprise	2109
love	2076
gratitude	1857
optimism	861
confusion	858
realization	586
fear	515
desire	389
pride	51

Figure 6. Number of samples in the dataset for each label after being merged into columns according to our grouped taxonomy.

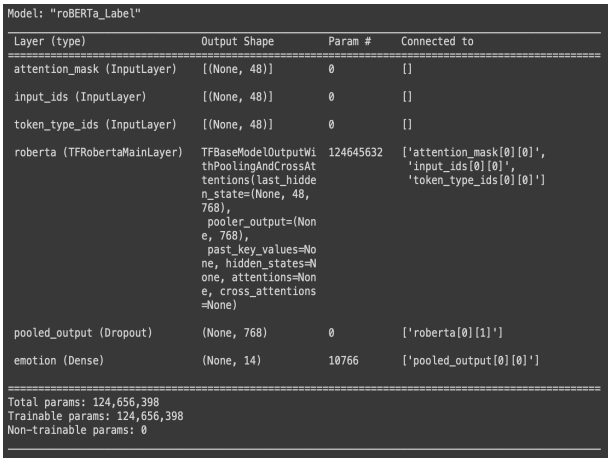
3. Experiments

3.1. Experiment I

For this experiment, I will provide a detailed explanation of the algorithm and techniques utilized in the experiments. As we progress, we will continue to fine-tune and optimize the model to achieve the highest possible score for our final model.

Regarding the dataset, we implemented one-hot-encoding for the entire dataset, and removed sentences with multiple labels. Subsequently, we merged the columns based on the selected label map.1

I have chosen to construct my model by finetuning RoBERTa [3] model since the GoEmotions Paper [1] used BERT model for their experimentation on the dataset.
In addition, a dense output layer was added on top of the pre-trained model for the purpose of fine-tuning. To support multi-label classification, a sigmoid cross entropy loss function was implemented.



```
Model: "roBERTa_Label"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
=================================================================================================
 attention_mask (InputLayer)    [(None, 48)]         0           []

 input_ids (InputLayer)         [(None, 48)]         0           []

 token_type_ids (InputLayer)    [(None, 48)]         0           []

 roberta (TFRobertaMainLayer)   TFBaseModelOutputWi  124645632   ['attention_mask[0][0]',
                                thPoolingAndCrossAt               'input_ids[0][0]',
                                tentions(last_hidde               'token_type_ids[0][0]']
                                n_state=(None, 48,
                                768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 pooled_output (Dropout)        (None, 768)          0           ['roberta[0][1]']

 emotion (Dense)                (None, 14)           10766       ['pooled_output[0][0]']

=================================================================================================
Total params: 124,656,398
Trainable params: 124,656,398
Non-trainable params: 0
_________________________________________________________________________________________________
```

Figure 7. Structure of the model

For evaluation purposes, we will utilize the F1 score and its macro-average score to compare all of our fine-tuned models, as well as the BERT model trained on the same dataset. However, it's worth noting that this comparison may not be entirely fair, as the original paper employed all the labels within the dataset. Additionally, as a student, I'm at a disadvantage compared to an MNC with an $18.9 billion budget and researchers from Stanford University. To put things in perspective, my project budget is only £15.50, of which £3.6 has already been spent on two cans of Red Bull, and £9.72 on a Google Colab Pro subscription, leaving me with just £2.18.

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (1)$$

The F1 score provides a balance between precision and recall, making it a useful metric when both false positives and false negatives are important considerations.

The following are the hyper-parameters chosen for the model according to standard practices.

The cost function of choice for our experiment is the BinaryCrossEntropy, a classic loss function commonly used in deep learning. However, during the training process, we encountered a class imbalance issue that resulted in the model producing mostly zero predictions for certain labels in the F1 score table. To address this issue, we conducted research and developed a custom modification of the BinaryCrossEntropy function. Our modified version calculates the mean of the product of the Background weights and the Signal weights, which is then used to derive the new BCE for more accurate results.[2]

Epochs	10
Optimizer	Adam with LR = 1.e-0.6
Batch Size	16
Data Preprocessing	Stop Words Removal, Stemming, Lemmatization

3.2. Experiment - II

	comment_text	clean_text
37673	Annnnnnd you get downvoted by people who don't...	annnnnnd get downvoted people like getting dow...
519	Why is this here?	?
25937	He said "Guess he didn't get the victory [NAME...	said guess get victory name frowning_face
27739	[NAME] has filled a speech bubble with savage ...	name filled speech bubble savage roasts froze ...
11246	I wonder if losing their majority in the 2019 ...	wonder losing majority locals would shift unio...

Figure 8. Comparison between raw text from the dataset and after applying preprocessing techniques

The objective of this experiment is to showcase the contrast between implementing data preprocessing techniques on the corpus and merely cleaning the data.
The techniques employed in this experiment include Stop Words Removal, Stemming, and Lemmatization. These are widely-used practices in natural language processing to minimize the size of the dataset by retaining only the relevant words within the sentences. By reducing the size of the sentences, these techniques expedite the training process.

3.2.1 Observations

These techniques offer a significant advantage by reducing the training time of the model by 20%, without compromising on the validation accuracy. However, it is crucial to ensure that the length of the longest sentence in the corpus is taken into account to facilitate the insertion of padding (pad) according to the sentence size.

3.2.2 Conclusion

Despite its success in reducing training time and achieving reasonable validation accuracy, the implementation of these techniques has not yielded satisfactory results, as evident

[2]Solution taken from https://stackoverflow.com/questions/48485870/multi-label-classification-with-class-weights-in-keras

Figure 9. F1-score for model using preprocessing techniques

from the low F1-score of 0.21 (see Figure 9). Figure 8 further illustrates the adverse impact of these techniques, as several sentences are reduced to a single or no word, rendering the model's performance ineffective. On the other hand, a significant improvement in performance is evident when training is performed on clean text, without any preprocessing techniques. The F1-score of 0.24 (see Figure 10) attests to this improvement.

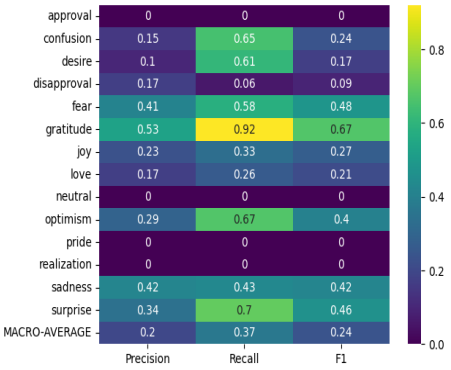


Figure 10. F1-score for model which does not use any preprocessing techniques

3.3. Experiment - III

The softmax function is commonly used in multi-class classification problems to map the output of a model to a probability distribution over the predicted classes. By using the softmax activation function in the output layer, we ensure that the sum of the predicted probabilities for all classes is equal to one. This helps us interpret the output of the model as a probability distribution over the classes. Using softmax activation in conjunction with binary cross-entropy loss allows us to train a multi-label classification model that outputs a probability distribution over all labels for each input. The binary cross-entropy loss function computes the difference between the predicted probabilities and the true labels, and is commonly used in multi-label classification tasks.

$$\text{softmax}(z)i = \frac{e^{z_i}}{\sum j = 1^K e^{z_j}} \text{ for } i = 1, 2, \ldots, K \quad (2)$$

where $z = (z_1, z_2, \ldots, z_K)$ are the input values and $\text{softmax}(z)_i$ represents the i-th output of the softmax function.

We will also change our weight initializer to 'Truncated Normal' The values generated are similar to values from a 'tf.keras.initializers.RandomNormal' initializer except that values more than two standard deviations from the mean are discarded and re-drawn.

3.3.1 Observation

The improved model has demonstrated a noteworthy enhancement in performance, as indicated by a considerable increase in the F1-score to 0.48 (see Figure 11). The precision of the model has also improved overall. Notably, the model makes more accurate judgments on sentences labeled as expressing gratitude. Additionally, the use of Softmax activation function has prevented the loss of certain tags during the BinaryCrossEntropy process.

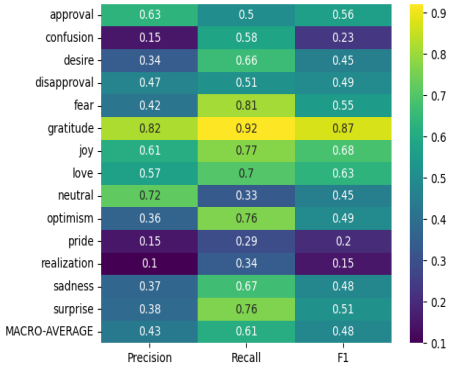


Figure 11. F1-score for model with 'Softmax' activation function and Truncated Normal Initializer

3.3.2 Conclusion

The use of the 'Softmax' activation function facilitates the application of the Binary Cross Entropy function to predicted values. This is due to the function's ability to fit and normalize data within the range of 0 to 1. Consequently, it represents the most suitable activation function for a multi-label model.

3.4. Experiment IV

In this experimental variation, I applied my observational skills to place a bet using the last remaining £2.18 in my possession. My prediction was based on the belief that, despite a slight reduction in the number of epochs and an increase in the batch size to 128, a high level of accuracy could still be attained.

```
Epoch 1/10
2270/2270 [==============================] - 462s 182ms/step - loss: 0.6844 - accuracy: 0.1944 - val_loss: 0.6583 - val_accuracy: 0.3672
Epoch 2/10
2270/2270 [==============================] - 401s 177ms/step - loss: 0.6283 - accuracy: 0.3522 - val_loss: 0.6219 - val_accuracy: 0.3522
Epoch 3/10
2270/2270 [==============================] - 393s 173ms/step - loss: 0.6049 - accuracy: 0.3427 - val_loss: 0.6040 - val_accuracy: 0.3058
Epoch 4/10
2270/2270 [==============================] - 377s 166ms/step - loss: 0.5878 - accuracy: 0.3412 - val_loss: 0.5883 - val_accuracy: 0.3507
Epoch 5/10
2270/2270 [==============================] - 388s 171ms/step - loss: 0.5723 - accuracy: 0.3703 - val_loss: 0.5754 - val_accuracy: 0.3819
Epoch 6/10
2270/2270 [==============================] - 392s 173ms/step - loss: 0.5576 - accuracy: 0.3867 - val_loss: 0.5635 - val_accuracy: 0.4061
Epoch 7/10
2270/2270 [==============================] - 382s 168ms/step - loss: 0.5449 - accuracy: 0.4066 - val_loss: 0.5558 - val_accuracy: 0.4131
Epoch 8/10
2270/2270 [==============================] - 385s 170ms/step - loss: 0.5326 - accuracy: 0.4297 - val_loss: 0.5454 - val_accuracy: 0.4270
Epoch 9/10
2270/2270 [==============================] - 373s 165ms/step - loss: 0.5213 - accuracy: 0.4416 - val_loss: 0.5368 - val_accuracy: 0.4376
Epoch 10/10
2270/2270 [==============================] - 377s 166ms/step - loss: 0.5113 - accuracy: 0.4536 - val_loss: 0.5295 - val_accuracy: 0.4510
```

Figure 12. Number of Epochs run on the training model

The training process was conducted over 10 epochs; however, it was observed that the increase in accuracy was not significant after the 7th epoch as shown in Figure 13. It is possible that overfitting was not occurring in this instance, and it may be feasible to achieve improved results through further training. However, considering the available resources, comparable accuracy can be achieved with less computation and time by concluding training after the 7th epoch.

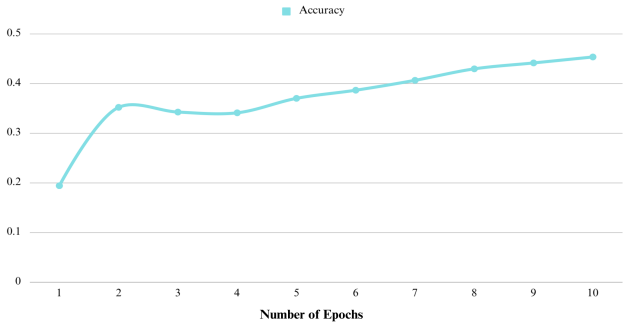


Figure 13. Line Graph of Number of Epochs v/s Accuracy

3.4.1 Observation

By concluding training after the 7th epoch, it is possible to achieve a comparable level of accuracy with significantly reduced training time, without compromising the quality of results. Theoretical calculations suggest that this approach should result in a 30% reduction in training time, as three epochs are not executed out of the total ten. However, it is also observed that as the loss approaches zero, the training time decreases.

3.4.2 Conclusion

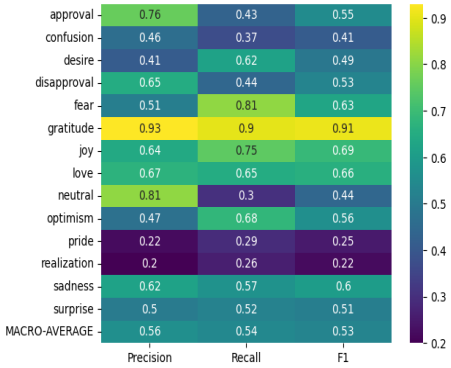


Figure 14. F1-score for training the model on 7 Epochs

A favorable F1-score of 0.53 was attained using only seven epochs, which is in proximity to the original score of 0.61 achieved by the BERT model for Ekman's Taxonomy.

```
Epoch 1/8
284/284 [==============================] - 186s 487ms/step - loss: 0.5799 - accuracy: 0.3499 - val_loss: 0.4829 - val_accuracy: 0.4982
Epoch 2/8
284/284 [==============================] - 99s 348ms/step - loss: 0.4504 - accuracy: 0.5054 - val_loss: 0.4212 - val_accuracy: 0.5365
Epoch 3/8
284/284 [==============================] - 100s 352ms/step - loss: 0.3956 - accuracy: 0.5314 - val_loss: 0.4074 - val_accuracy: 0.5123
Epoch 4/8
284/284 [==============================] - 88s 310ms/step - loss: 0.3548 - accuracy: 0.5516 - val_loss: 0.4049 - val_accuracy: 0.5361
Epoch 5/8
284/284 [==============================] - 90s 316ms/step - loss: 0.3211 - accuracy: 0.5633 - val_loss: 0.4278 - val_accuracy: 0.5532
Epoch 6/8
284/284 [==============================] - 91s 322ms/step - loss: 0.2867 - accuracy: 0.5900 - val_loss: 0.4448 - val_accuracy: 0.5446
Epoch 7/8
284/284 [==============================] - 89s 316ms/step - loss: 0.2614 - accuracy: 0.6093 - val_loss: 0.4526 - val_accuracy: 0.5339
Epoch 8/8
284/284 [==============================] - 93s 329ms/step - loss: 0.2327 - accuracy: 0.6314 - val_loss: 0.4722 - val_accuracy: 0.5413
```

Figure 15. Running 8 Epochs

4. Best Result

Following several model training attempts, the most optimal outcome was achieved with the parameters specified in the following table 1

5. Conclusion

Based on the results obtained from the experiment, it can be inferred that the model aligns precisely with the intended objective. The prototype demonstrates good accuracy, which is satisfactory for its intended purpose. Further improvements can be achieved by fine-tuning the hyperparameters and modifying the configuration file of the RoBERTa model. However, it should be noted that the prototype is a multi-class model, rather than a multi-label model.

HyperParameter	Value
Model	roberta-base
Activation Function	Softmax
Initializer	Truncated Normal
Epochs	8
Optimizer	AdamW
Learning Rate	5.e-05
Weight Decay	0.004
Data Preprocessing Techniques	None
Cost Function	Custom Binary Cross Entropy
Batch Size	128

Table 1. Final List of Hyperparameters which achieved the highest F1-score for my model

Reducing the learning rate and increasing the number of epochs are potential strategies that could enhance the model's accuracy. However, such measures come at a cost to the model's efficiency and speed, which may not be viable in certain resource-constrained scenarios. To achieve a suitable balance, a pragmatic approach was taken during the experimentation process.
Similarly, the incorporation of certain preprocessing techniques may enhance the model's efficiency. However, the potential trade-off is a decrease in accuracy. Therefore, a judicious selection of preprocessing techniques was adopted to balance the competing considerations of efficiency and accuracy for the available resources.

An ArgMax function was incorporated at the end of the output during deployment to generate the most appropriate prediction for the emotion conveyed in the input text. This function was not utilized during the training phase, as it does not offer a suitable estimate for the cost function.

The final F1-score for the model can be observed in Figure 1. By comparing this score to the label distribution depicted in Figure 4, it can be concluded that the emotion label 'neutral' is overfitting, whereas 'pride' and 'realization' are underfitting. On the other hand, the label 'gratitude' is exhibiting exceptional performance across all scenarios.

Less frequent emotions are often confused with more frequent emotions related in sentiment and intensity, while more frequent emotions are also susceptible to being misclassified as less common emotions with similar sentiment and intensity.

The model can potentially achieve a higher performance score if the text is thoroughly cleaned and tone tags are accurately replaced with their appropriate meanings, such as the use of '/s' to denote sarcasm. An improved accuracy of the F1-score greater than 7.0 is recommended, as it is imperative for the model to accurately predict emotions for various applications including hate speech detection and product review classification. Achieving this level of accuracy will help ensure that the model meets user expectations and delivers reliable results.

5.1. Error Analysis

During the development of the model, I encountered challenges with the F1-score label, which returned a value of 0. Despite spending a significant amount of time troubleshooting the issue, I was unable to identify the source of the error. In future projects, I intend to prioritize logging and debugging to facilitate error identification.

Furthermore, during the construction of a TensorFlow architecture, I discovered that the stock GPU provided by Google Colab was slow in its performance. Subsequently, I migrated the notebook to my system, where I encountered a further challenge in that TensorFlow lacked GPU library support for the Mac M2 chips. Despite this setback, I persisted with the development of the model, dedicating a substantial amount of time to its construction, despite its initial sluggish performance.

In hindsight, I could have addressed the issue of overfitting by down-sampling the 'neutral' tag and generating additional synthetic data for the under-performing labels. However, when analyzing the 'fear' and 'desire' labels, it is evident that they performed at a level close to the average. This suggests that the vocabulary associated with each label is distinctive and readily discernible by the model. As a result, it may not be necessary to undertake extensive measures to improve their performance.

References

[1] Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. Goemotions: A dataset of fine-grained emotions, 2020. 1, 2, 3

[2] Paul Ekman. Emotions revealed. *Bmj*, 328(Suppl S5), 2004. 1

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. 3