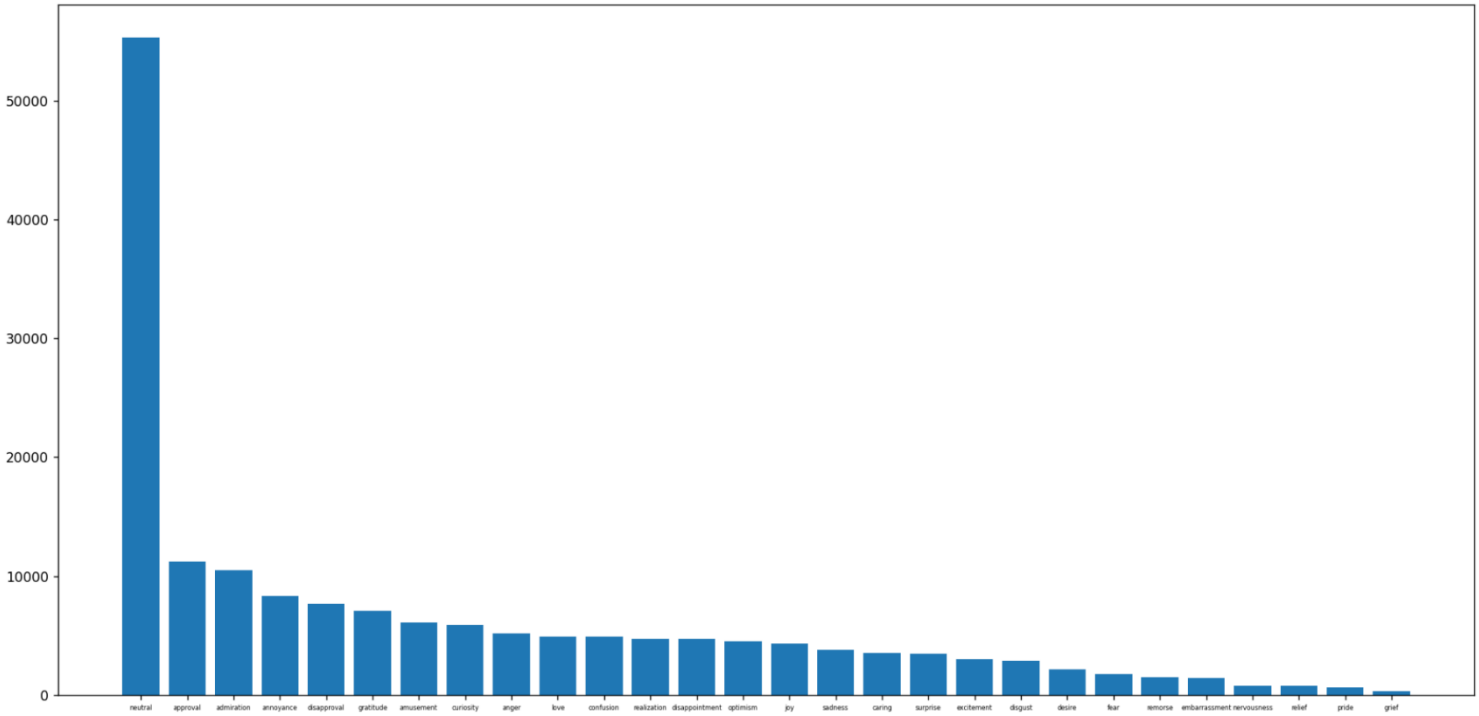
MP1

ANALYSIS

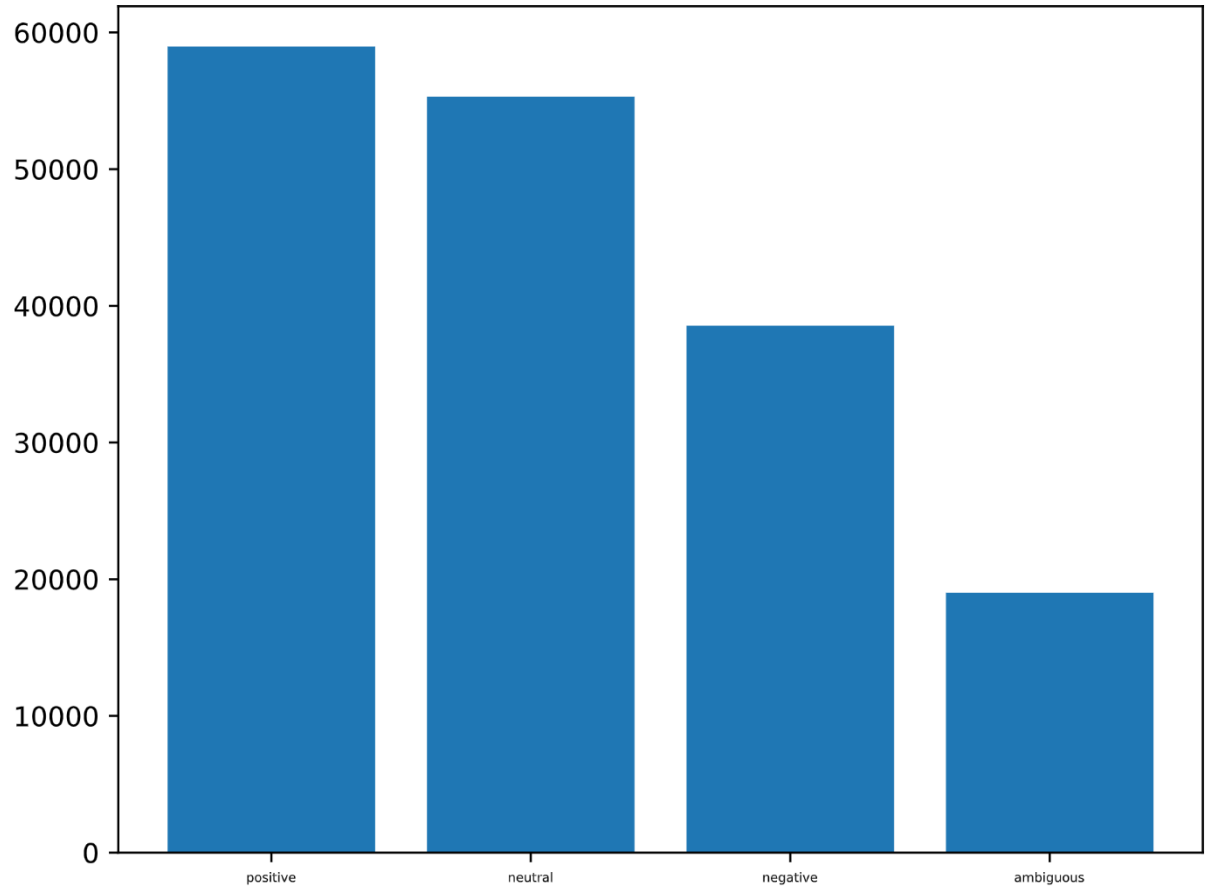
Task 1:

Task 1 asked us to download the “go emotion” dataset from Moodle and then load it into python. This dataset contains 58000 reddit comments which are distributed into 28 emotion categories (admiration, anger, amusement etc.), which are further classified into 4 sentiment categories (positive, negative, ambiguous, and neutral). When looking over the dataset, we noticed there was quite a bit of noisy input, meaning there were many duplicate reddit posts which had different emotion and sentiment labels. This of course played a part in our classifier model’s accuracies. We were then asked to extract the posts and then plot them against the 2 labels in a graph. We chose to plot a histogram and got the following results:

Emotion plot:



Sentiment Plot:



By looking at the 2 plots we can see that the emotion plot is skewed, with an overwhelming majority of posts being classified as neutral. Even though the sentiment plot was more evenly distributed, we felt it was best to use the emotion labels when training and testing our models as it offered a much wider variety of outputs, with 28 different outputs it could be while sentiment would only have 4, making it a much more ambiguous label to train our model with.

Task 2:

For task 2, we first extracted all the tokens(words) in the dataset and then displayed the total size of the vocabulary in the dataset. We then split the dataset up into two sections, 80 percent for the training set and 20 percent for the testing set. After this,

we created 6 different classifiers and trained and tested them with the dataset, getting the following results:

(For complete results for this section, please refer to *performance.txt*)

NOTE: Only one epoch was run for MLP as it was taking extremely long.

BASE-MNB:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.33	0.14	0.17
WEIGHTED AVG	0.37	0.39	0.31
ACCURACY	-	-	0.39

BASE-DT:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.29	0.28	0.28
WEIGHTED AVG	0.38	0.36	0.36
ACCURACY	-	-	0.36

BASE-MLP:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.30	0.29	0.29
WEIGHTED AVG	0.38	0.39	0.38
ACCURACY	-	-	0.39

TOP-MNB:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.30	0.20	0.22
WEIGHTED AVG	0.36	0.39	0.35
ACCURACY	-	-	0.39

TOP-DT:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.33	0.16	0.18
WEIGHTED AVG	0.36	0.41	0.31
ACCURACY	-	-	0.41

TOP-MLP:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.36	0.21	0.22
WEIGHTED AVG	0.42	0.44	0.35
ACCURACY	-	-	0.44

Recall and Precision were both of equal importance for each classifier, which gave us the F1 score. It is better to look at the F1 score as a measure of correctness rather than accuracy as there was a lot of noisy input in the dataset, skewing the class distribution.

When looking at all the classifiers, it is clear to see using GridSearchCV to find optimal hyper-parameters did help increasing the f1 score of the TOP-MNB classifier, with an increase from 0.31 to 0.35. However, the TOP-DT's was reduced from 0.36 to 0.31 and the TOP-MLP's was reduced from 0.38 to 0.35. This could be because in decision trees, it can change every time as the train and test data are changed with a different random seed, meaning tree-based algorithms are influenced by the dataset.

Therefore, the TOP-MLP classifier was the best performing, having the highest weighted accuracy score of 0.45 compared to TOP-DT, the next best performing classifier's, 0.41. Although its f1 score was lower than that of the BASE-MLP, it

was only a 0.03 deduction in the f1 score while there was an increase of 0.06 in its accuracy

We then changed the sizes of the training and testing sets, making the training set 60% of the total dataset and the testing set 40% of the dataset. After this, we re-ran the classifiers and got the following results:

(For complete results for this section, please refer to *performance1.txt*)

NOTE: Only one epoch was run for MLP as it was taking extremely long.

BASE-MNB:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.32	0.12	0.14
WEIGHTED AVG	0.37	0.38	0.29
ACCURACY	-	-	0.38

BASE-DT:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.28	0.27	0.27
WEIGHTED AVG	0.36	0.35	0.35
ACCURACY	-	-	0.35

BASE-MLP:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.34	0.26	0.29
WEIGHTED AVG	0.38	0.42	0.38
ACCURACY	-	-	0.42

TOP-MNB:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.32	0.17	0.20

WEIGHTED AVG	0.35	0.38	0.33
ACCURACY	-	-	0.38

TOP-DT:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.31	0.16	0.17
WEIGHTED AVG	0.35	0.41	0.30
ACCURACY	-	-	0.41

TOP-MLP:

	PRECISION	**RECALL**	**F1 SCORE**
MACRO AVG	0.35	0.19	0.20
WEIGHTED AVG	0.40	0.43	0.33
ACCURACY	-	-	0.43

We can see from the above results that reducing the size of the training set reduced the f1 scores for most classifiers and did not have an increase on any classifier's accuracy other than the BASE-MLP and even then, it was minimal. This was however expected as since the classifiers have less data to train with, it is obvious they would not be as efficient.

Here however, the BASE-MLP is the best performing classifier, with the highest f1 score of 0.38 and the second highest accuracy of 0.42. However, this could be since not only was the training set smaller, the MLP's were only run for one epoch, meaning over time the TOP-MLP could outperform the BASE-MLP.

Task 3:

Task 3 asked us to download the word2vec-gooogle-news-300 pre-trained embedding model using the genism API. We then used the NLTK package to tokenize all the words in the reddit posts from the dataset. We then split the dataset into a training set and test set, and then compared these words to those words in the embedding model, and if they were found, received vectors of length 300. We then used these vectors for the words to find the average, which gave us the embedding

value for each reddit post. We then calculated the hit rates of both the testing and training set (i.e. the percentage of words found in the embedding model). The hit rates we got were:

Training set: 77.3285%

Testing set: 77.2782%

Even though the 2 sets are of different sizes, both have nearly identical hit rates. This was expected as many words are repeated throughout the reddit posts, meaning that a majority would be found either way.

We then attempted to train a BASE-MLP and TOP-MLP using these embedding values in an embedding array. However, we ran into an error when trying to do a fit, as the array must be homogenous however every post had a different amount of words. This hindered us from being able to complete the training of the two models and being able to compare the difference between using embedding values and tokens directly as we did in task 2.

In conclusion, the TOP-MLP was by far the best performing model even though it was only run for one epoch. With a full run of several epochs, we believe the difference in performance would be highlighted even greater.

GROUP CONTRIBUTIONS:

Omar Mahmoud 40158127 – Tasks 1,2,3,4

Athanas Bakleh 40093110 – Tasks 1,2,3,4

Mohammad Aamir Parekh 40136289 – Tasks 1,2,3,4