28.01.2025

FINAL PROJECT REPORT

TASK:

For this project, I chose **sentiment analysis of COVID-19 tweets in English language from Kaggle**. The goal was to finetune two chosen pre-trained models - **DistilBERT** (distilbert/distilbert-base-uncased) and **BERTweet** (vinai/bertweet-base), for multiclass classification of the tweets into three categories - negative, neutral positive. The motivation for this task is to understand social media sentiment during critical periods of the COVID-19 pandemic (specifically in various time periods from April 2020 to June 2021). Sentiment analysis is one of many important NLP tasks and a social medium like Twitter offers a continuous stream of user input, making this NLP task an invaluable tool for understanding the mood of the masses, which can be important for various reasons.

DATASET:

A dataset chosen for this task was found on the platform Kaggle. It is a COVID-19 Twitter dataset which consists of three subsets which contain from 320 000 up to 489 000 samples. The dataset creator gathered and labelled samples of COVID-19 tweets from three different time periods: April-June 2020, August-September 2020, and April-June 2021. This makes it perfect for a comparison of the sentiment of masses during different COVID pandemic waves.

For this task, I chose the sub-dataset of the time period **August-September 2020** which contains in total **320 316 tweets**. This contains 17 columns with various data which could be explored further. In order to save memory space and reduce computational power, only two columns were selected for this task, namely „**clean_tweet**“ (later renamed to „**text**“) and „**sentiment**“ (lately renamed to „**labels**“). The creator of the dataset used sentiment labels in the string format: **„neg“, „neu“,** and **„pos**“, which were converted into numerical sentiment labels respectively: **0, 1,** and **2**. The dataset of **2 000 samples** was divided proportionally as follows: **80% (1 600 samples) for training set**, **10% (200 samples) for validation set**, and **10% (200 samples) for test set**. Further preprocessing of this dataset included converting its original structure, which was a Pandas *DataFrame* as it originated from the Kaggle website, into a Hugging Face *DatasetDict*, a format required for subsequent processing steps. The dataset was finally tokenized. This was done separately for both models using their respective tokenizers.

HYPERPARAMETERS:

The following hyperparameters were used for the most successful model (used for both models):

- batch_size=20
- logging_steps=10
- num_train_epochs=5
- eval_strategy="epoch“
- save_strategy="epoch“
- learning_rate=2e-5

- weight_decay=0.01
- seed=224

The process of finding the ideal parameters was personally one of the most challenging steps in this project. I started simply thinking about possible options and combinations. After trying few hyperparameters out, I increased the number of samples for training, validation and testing to see whether it makes a difference. For finding even better combination of hyperparameters to refine the model for optimal performance, I experimented with *Optuna algorithm*. I defined an objective function where I specified a range of values for each hyperparameter I wanted to tune. This was the first time for me using such algorithm, and I believe I was not very successful. Although the algorithm found possible combinations of hyperparameters to improve the accuracy of the model, the validation loss became too high. I tried to run this algorithm with the maximum of 10 trials. It took some time and exhausted my resources, so I aborted the process and continue with the hyperparameters I found the best (see above).

SETUP:

The setup was very similar to the one we used during our classes. I used a **Google Colab Jupyter Notebook** for loading and editing the dataset, and for training both models. I found it quite useful for adding textual fields with explanations of the process. The notebook run on **GPU**, but I switched to **CPU** when needed (most of the times because of the Cuda errors). I decided to pay for the **Google Colab Pro** version to experiment in peace and without fear about exhausting the resources that fast. This, of course, sped up also the whole training and fine-tuning process immensely, and it took **max. 15 minutes to train a model**. I was very satisfied with this option, though, I believe that this is not an optimal permanent solution. I hope there will be some solution for more computational power for students soon.

QUANTITATIVE RESULTS:

The evaluation of the two models, DistilBERT and BERTweet, was conducted on the test set consisting of 200 tweets, which were not seen during training or validation. The performance of the models was compared using **accuracy, precision, recall**, and **macro F1-score**, as the evaluation metrics.

```
# Evaluate the model on the test set
results_distilbert_finetuned = trainer_test_distilbert.evaluate()

# Print the results
print("Evaluation Metrics:\n", results)
```

```
[10/10 00:00]

Classification Report:
              precision    recall  f1-score   support

    Negative       0.85      0.97      0.91        40
     Neutral       0.95      0.93      0.94        87
    Positive       0.96      0.90      0.93        73

    accuracy                           0.93       200
   macro avg       0.92      0.94      0.93       200
weighted avg       0.93      0.93      0.93       200
```

Figure 1: Classification report with used metrics on the test set of the fine-tuned DistilBERT model

```
[44] # Evaluate the model on the test set
     results_bertweet_finetuned = trainer_test_bertweet.evaluate()

     # Print the results
     print("Evaluation Metrics:\n", results)
```

```
[10/10 00:00]

Classification Report:
              precision    recall  f1-score   support

    Negative       0.91      0.97      0.94        40
     Neutral       0.88      0.95      0.92        87
    Positive       0.97      0.84      0.90        73

    accuracy                           0.92       200
   macro avg       0.92      0.92      0.92       200
weighted avg       0.92      0.92      0.91       200
```

Figure 2: Classification report with used metrics on the test set of the fine-tuned BERTweet model

The evaluation metrics for DistilBERT:

```
Evaluation Metrics:
'eval_loss': 0.25121039152145386, 'eval_model_preparation_time':
0.0025, 'eval_accuracy': 0.93, 'eval_f1': 0.9261382246970192,
'eval_mcc': 0.8921665559029573, 'eval_runtime': 0.2989,
'eval_samples_per_second': 669.068, 'eval_steps_per_second': 33.453}
```

The evaluation metrics for BERTweet:

```
Evaluation Metrics:
'eval_loss': 0.25121039152145386, 'eval_model_preparation_time':
0.0025, 'eval_accuracy': 0.93, 'eval_f1': 0.9261382246970192,
'eval_mcc': 0.8921665559029573, 'eval_runtime': 0.2989,
'eval_samples_per_second': 669.068, 'eval_steps_per_second': 33.453}
```

The results show that both models scored a high accuracy on the test set (DistilBERT: 93%, BERTweet: 92%), indicating that the models correctly classified 93% and 92% of the test samples. The calculated F1-score (DistilBERT: 93%, BERTweet: 92%) suggests that the models achieved balanced performance across all three sentiment classes. The models scored very similar scores, and both demonstrated robust and consistent performance.

In order to see where the model confused the predictions, confusion matrices were created for both models:

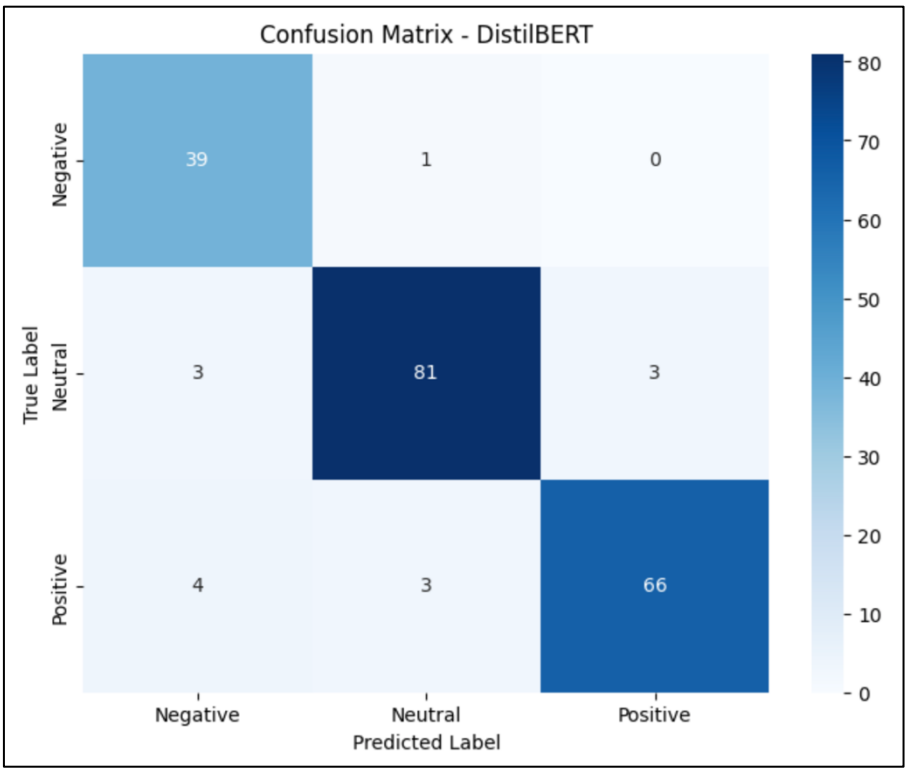

Figure 3: Confusion matrix for the fine-tuned DistilBERT model

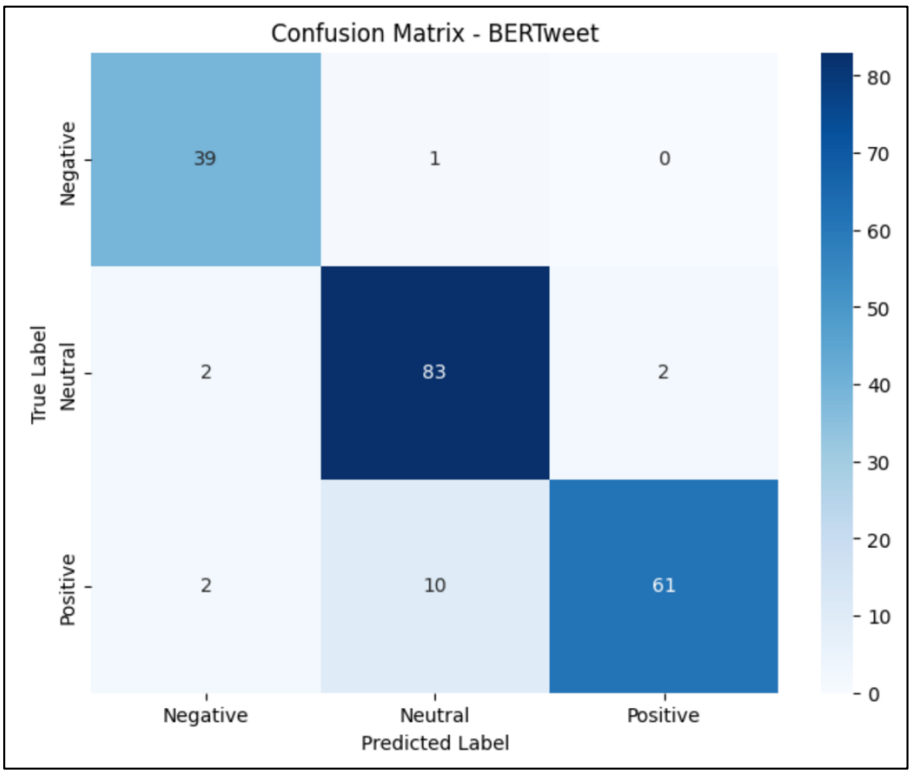

Figure 4: Confusion matrix for the fine-tuned BERTweet model

The diagonal elements of these confusion matrices represent the correctly classified instances of each class, which are very strong in both these matrices. Again, this only confirms very similar good and similar performance of both models on the same given task. DistilBERT appears to have slightly fewer misclassifications than BERTweet, which means that it generalized slightly better.

The comparison was taken one step further and error distribution plots were created for both models. This serves as an extension of the confusion matrix to better understanding and interpret the misclassifications of the models.

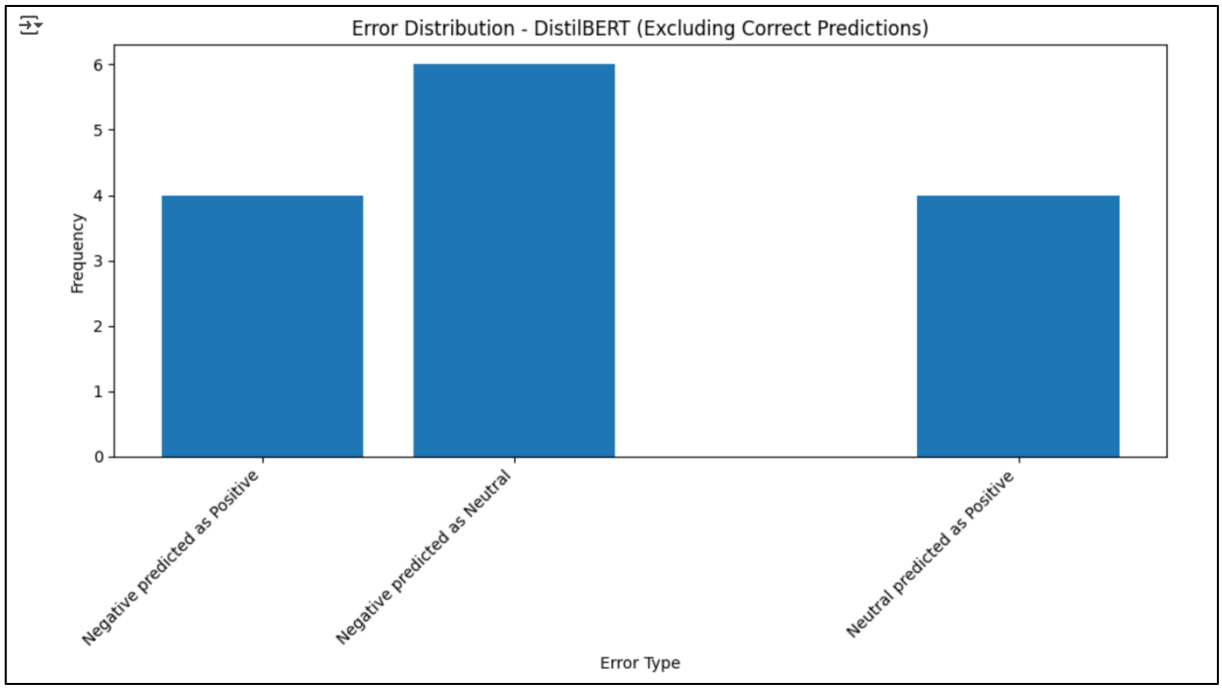

Figure 5: Error distribution for the fine-tuned DistilBERT model

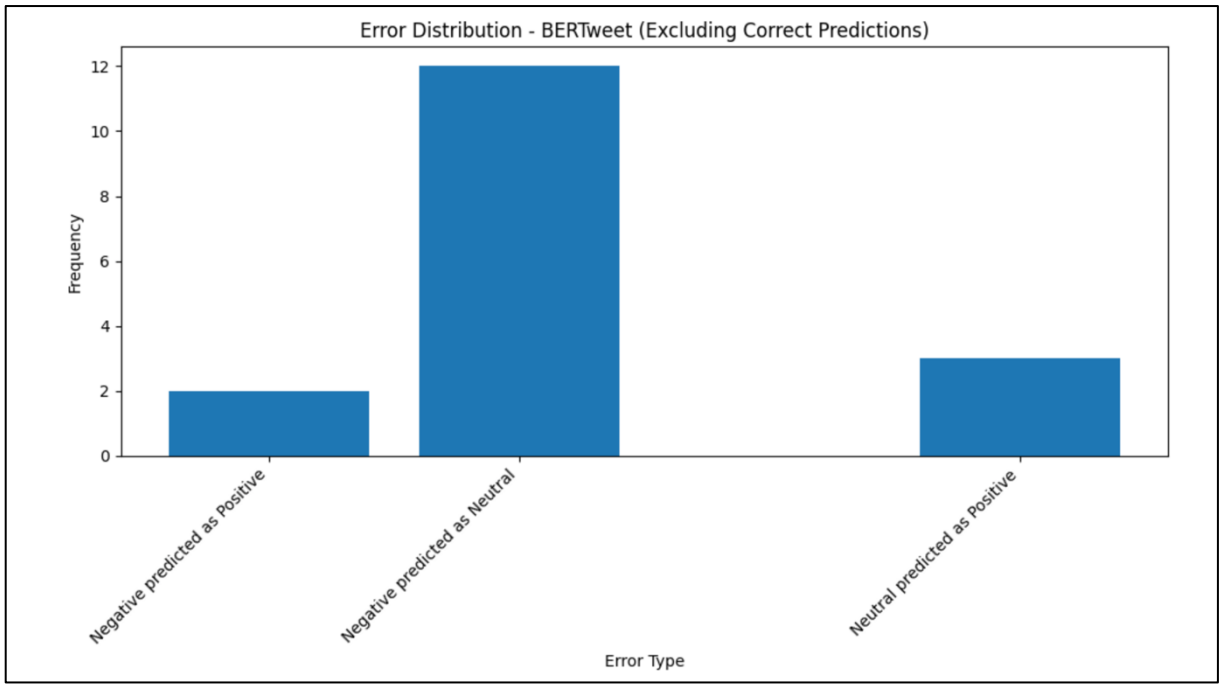

Figure 6: Error distribution for the fine-tuned BERTweet model

These two plots show the frequency of the occurrence of each misclassification. The fine-tuned BERTweet model struggles with distinguishing negative sentiment from the neutral sentiment more than the fine-tuned DistilBERT model. This could be caused due to mixed sentiments

within a tweet, or some tweet might have been more factual than emotionally expressive which was misleading for the model. In my opinion, classification of a positive tweet as a negative one might be more alarming. At this point, the fine-tuned BERTweet model outperformed the fine-tuned DistilBERT model.

To sum it up, DistilBERT outperforms BERTweet in overall accuracy and positive sentiment classification. It appears that BERTweet struggles with distinguishing positivity. Both models did very well with minimal misclassification. Further improvements could be done in handling neutral sentiment by adding more training data or by improving the hyperparameters.

QUALITATIVE RESULTS:

The misclassified samples from the test dataset included:

- positive sentiment misclassified as neutral
- neutral sentiment misclassified as negative
- positive sentiment misclassified as negative

After the closer look, it can be established that there are more reasons for the wrong classifications of the samples. Firstly, a tweet may contain implicit sentiment which makes it harder for the model to be properly classified. Also tweets which contain mixed sentiment (such as both positive and negative sentiments) may confuse the model and it will focus on one part of the tweet without considering the other. Many neutral or positive tweets were misclassified, because they did not contain clear sentiment markers, such as “angry”, “frustrated”, etc. Another reason may be the length of a tweet. Short tweets are problematic for the model as they lack enough context for an accurate sentiment detection. Finally, some COVID-19 related word (such as “lockdown”, “pandemic”, “fight”, etc.) may have already been contained in the pre-trained model. The model may than overemphasize these rather than considering the full tweet content of the dataset it is being fine-tuned on. At this point, attention visualization may be used as future improvements in order to better understand which words are influencing the model‘s predictions.

VISUAL ANALYSIS:

The following visualizations were done on the following website: https://projector.tensorflow.org. Each point represents a **tweet embedding** in a high-dimensional embedding space (projected into 3D), which was reduced with **PCA method** (directly on the website). **Three colours** correspond to the three sentiment classes – negative, neutral, positive. The **distribution and clustering of points** show how well the model differentiates sentiment as we move through deeper layers.

Fine-tuned DistilBERT model visualizations:

Layer 1 is a shallow representation, and the embeddings appear very unstructured without any clear clusters of the particular sentiment categories. The model has not yet processed much semantic meaning, it mainly focuses on the tokenization of the individual words, and not on their semantic meaning or context. Similar words, such as “lockdown” and

"pandemic" may have their embeddings near to each other, even though that their sentiment differs.

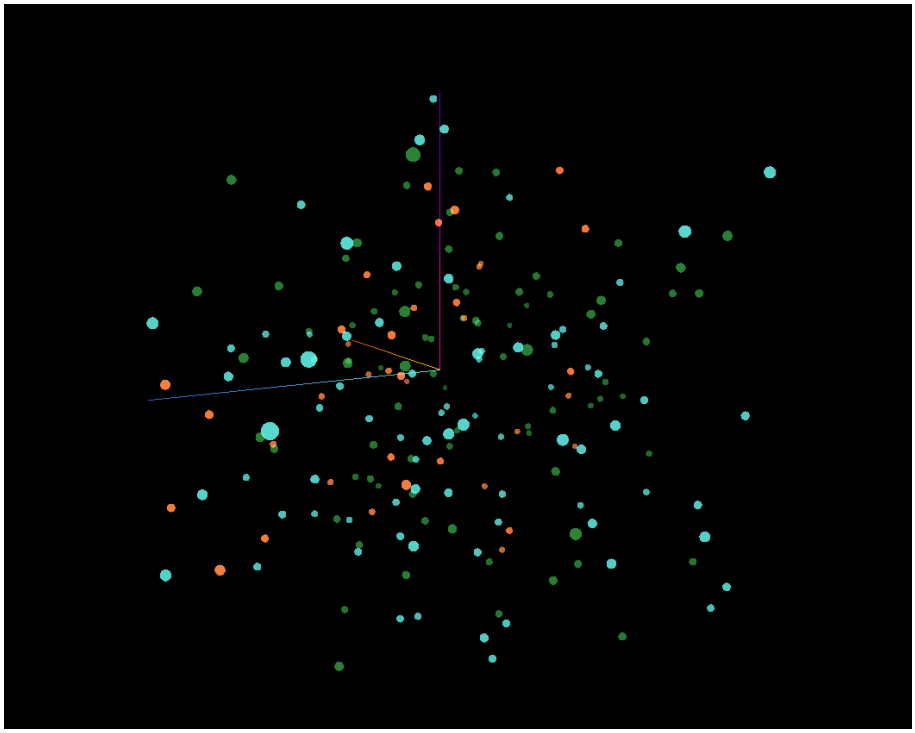

Figure 7: Layer 1 of the Epoch 3 of the fine-tuned DistilBERT model. Total variance described: 26.1%.

Layer 3 is already an intermediate representation as DistilBERT have six hidden layers by default. Compared to Layer 1, there is an emerging structure, some small clusters are forming, which indicates that the model is starting to capture semantic and contextual relationships. Although the model combines token meanings into phrase-level and sentence-level representations, some words are still treated in isolation and there is still visible amount of confusion between sentiments.

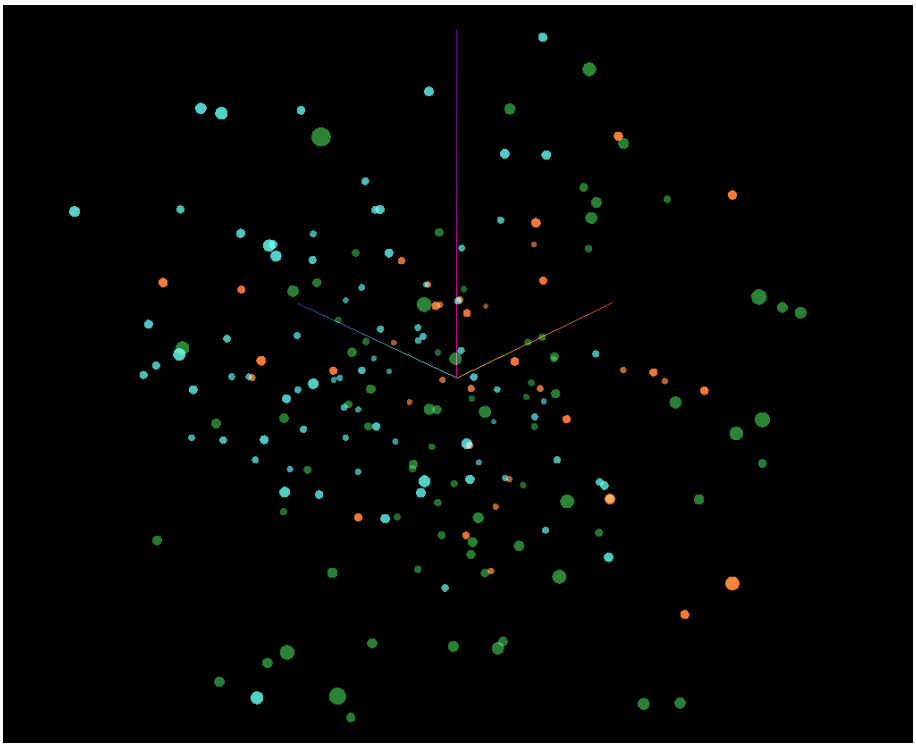

Figure 8: Layer 3 of the Epoch 3 of the fine-tuned DistilBERT model. Total variance described: 26.4%.

Layer 6 is the final layer which achieves clearer sentiment separation, improving classification accuracy. In this layer, the model encodes high-level meaning and captures the full context of a sentence. This could be seen on much more structured clusters of colours. Tweets with similar sentiment are grouped closely together, showing that the model has successfully learned to differentiate between negative, neutral, and positive sentiment.

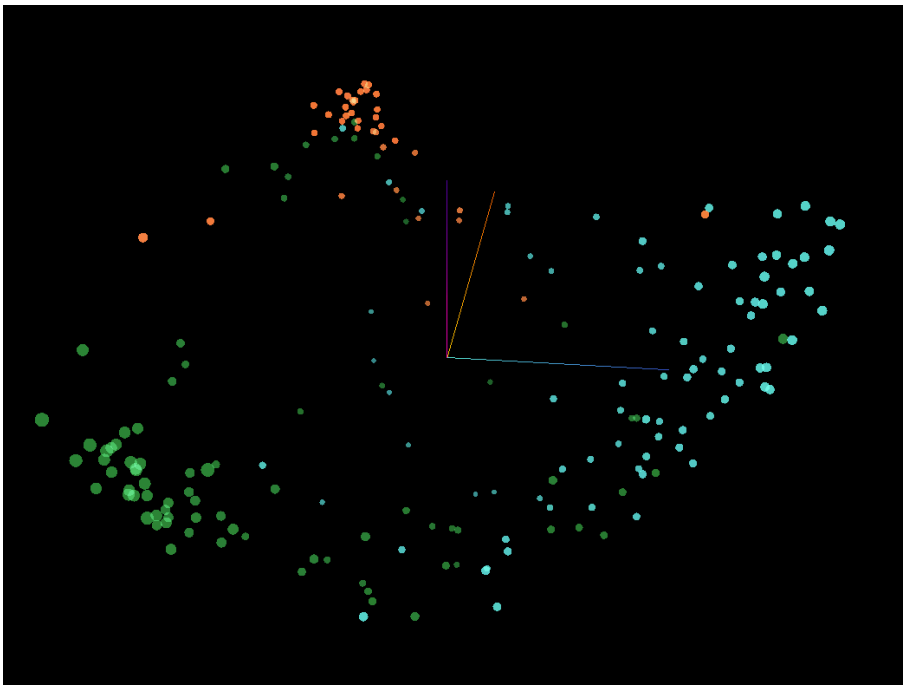

Figure 9: Layer 6 of the Epoch 3 of the fine-tuned DistilBERT model. Total variance described: 72.0%.

Fine-tuned BERTweet model visualizations:

Layer 1 is again a shallow representation of the basic token embeddings which appear randomly in the embedding space. At this stage, the BERTweet has not yet learned to structure sentiment. This representation is very similar to the representation of the Layer 1 of DistilBERT.

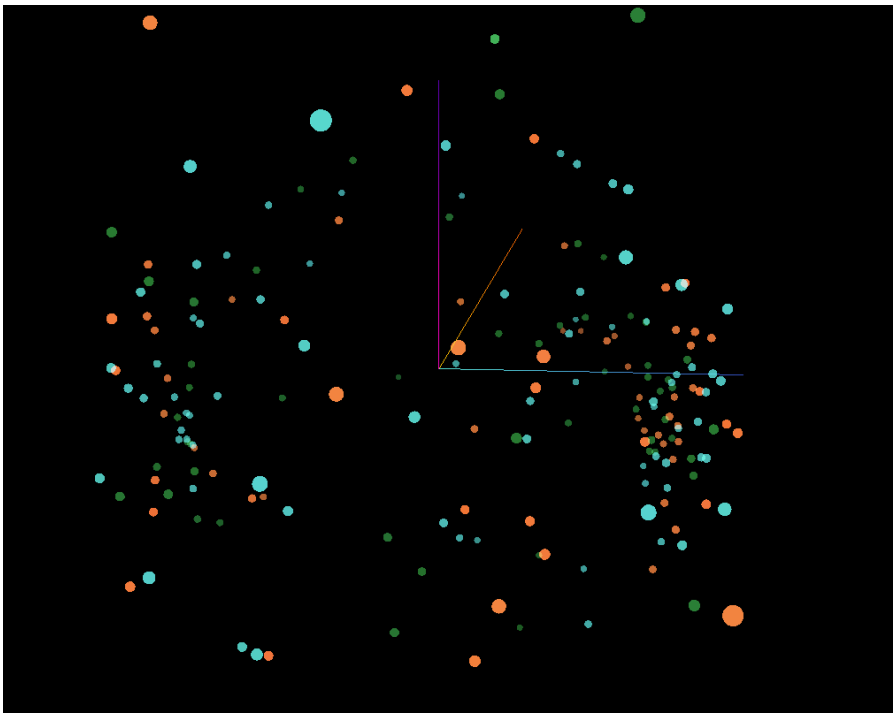

Figure 10: Layer 1 of the Epoch 3 of the fine-tuned BERTweet model. Total variance described: 61.9%.

Layer 6 is an intermediate representation, but this differs quite a lot from the intermediate representation of the DistilBERT model. This is because the BERTweet model has twelve hidden layers by default, while DistilBERT only six. There are clearer clusters, although there is still noticeable overlap between different sentiment categories. At this stage, the model is combining the token embeddings into more meaningful contextual representations.

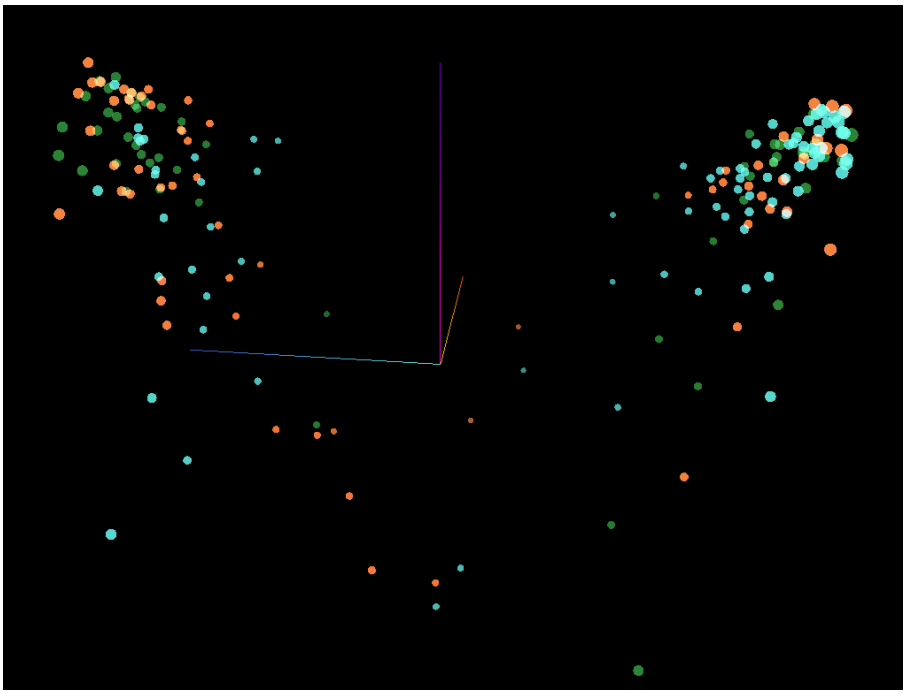

Figure 11: Layer 6 of the Epoch 3 of the fine-tuned BERTweet model. Total variance described: 92.1%.

Layer 12 is this model's final layer, which shows well-defined clusters and better classification into the three sentiments. The remaining overlap suggests that some expressions remain ambiguous. Compared to the final layer of DistilBERT model, the latter shows more distinct sentiment clusters, suggesting it learned sentiment classification slightly better. BERTweet model should have performed better on social media text, but the results suggest it did not learn sentiment separation as effectively.

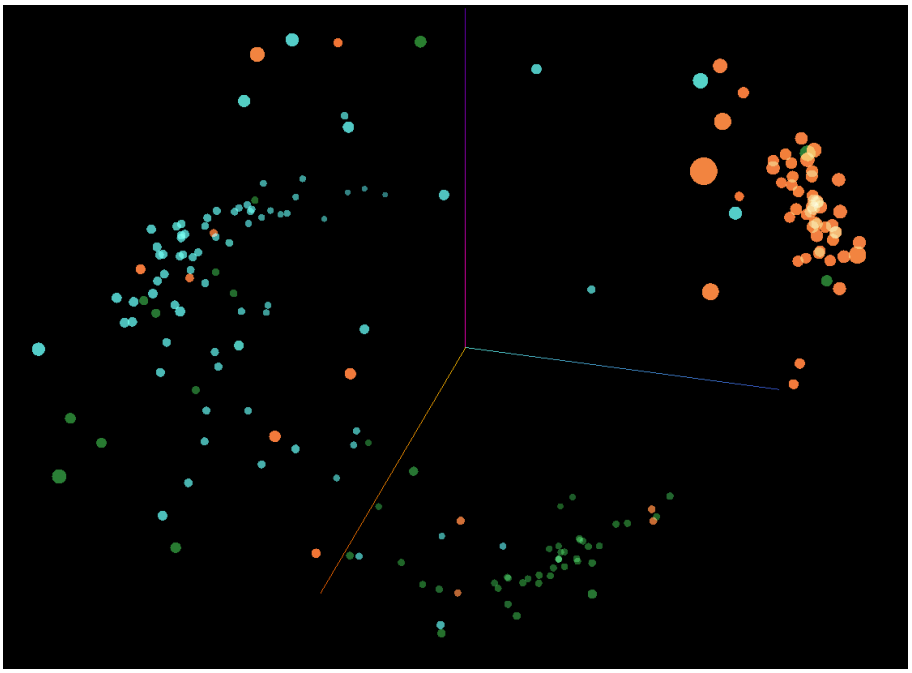

Figure 12: Layer 12 of the Epoch 3 of the fine-tuned BERTweet model. Total variance described: 81.5%.

28.01.2025

FINAL SUMMARY:

The goal of this project was to perform sentiment analysis on COVID-19-related tweets in English language, classifying them into three possible categories – negative, neutral, positive. Two pre-trained models from Hugging Face – DistilBERT and BERTweet, were fine-tuned on the named dataset and their performance was visualized and compared with particular metrics.

The overall performance of both models was very high. DistilBERT scored 93% accuracy, showing its strong generalization on the test set. BERTweet scored 92% accuracy, performing nearly as well as the first model. The latter, which is a pre-trained on Twitter data, was expected to outperform DistilBERT model, but the performance was almost the same. This could be corrected with more thorough hyperparameters engineering, in order to achieve even higher performance.

This project successfully demonstrated the architecture of transformers in general and in particular of the two chosen models, the importance of the "right" dataset for the "right" task, the important process steps of training and fine-tuning a language model, and the importance of visualizing embeddings to diagnose model strengths and weaknesses.