EmoTalk: Sentiment Analysis on Audio Text Data

Ashwin Unnikrishnan, Pratyush Rokade, Sreelu Prasanna Nair
Northeastern University
unnikrishnan.a@northeastern.edu, rokade.p@northeastern.edu, nair.sre@northeastern.edu

Abstract

Sentiment analysis has garnered significant attention in today's context due to the vast volume of online content, some that make you feel good, and others that bring you down. With everyone rushing around, we don't always get the chance to understand how others are feeling. We chat with voice messages and share audio clips in group chats, but sometimes it's hard to grasp the emotions. We have developed multiple models using Random Forest, Long Short-Term Memory (LSTM), and BERT, conducting a comparative analysis of their performance on a designated dataset. The most proficient model was subsequently employed to construct a definitive system capable of processing audio input and accurately predicting the sentiment of the conveyed data.

I. Introduction

In an era marked by the unprecedented proliferation of digital communication and the continuous generation of vast textual content across various online platforms, understanding the sentiments expressed within these texts has become a pivotal aspect of deciphering human emotions, opinions, and behaviors. Sentiment analysis, also known as opinion mining, has emerged as a powerful technique within the realm of Natural Language Processing (NLP) that aims to unravel the intricate tapestry of human sentiment concealed within written text.

The ability to automatically gauge whether a piece of text carries a positive, negative, or neutral sentiment opens avenues for businesses to fine-tune their marketing strategies, for policymakers to gauge public opinion, and for researchers to uncover trends and patterns in societal attitudes. By unraveling the intricate tapestry of sentiments woven into text, sentiment analysis opens doors to a deeper comprehension of human emotions and behaviors, ushering in a new era of data-driven emotional intelligence.

Audio sentiment analysis finds wide-ranging applications across various domains. In customer service, it aids in assessing customer satisfaction by analyzing call center interactions and identifying emotional cues, enabling companies to enhance user experiences. In healthcare, it plays a role in monitoring patient well-being through vocal inflections, potentially assisting in early detection of mood disorders or stress-related conditions. Educational settings can benefit from gauging student engagement and emotional responses during online lectures, tailoring teaching approaches accordingly. Moreover, the entertainment industry leverages audio sentiment analysis to gauge audience reactions to media content, refining content creation strategies. In social media, it enables the assessment of emotional responses to audio content, helping in content curation and user engagement strategies. Overall, audio sentiment analysis empowers decision-making by extracting valuable emotional insights from auditory data, facilitating improved communication, well-being, and engagement in various contexts.

Advancements in the field of NLP have yielded remarkable progress, resulting in the development of numerous algorithms designed to decipher the intricate underlying sentiments embedded within the extensive volume of textual data generated on the internet. This report undertakes an in-depth exploration into the domain of sentiment analysis, focusing on diverse methodologies for implementing this sophisticated technique specifically within the context of the Twitter dataset. A comprehensive investigation has been carried out, encompassing the analysis of three prominent Machine Learning algorithms – namely, Random Forest, Long Short-Term Memory (LSTM), and Bidirectional Encoder Representations from Transformers (BERT).

The prime objective of this report is to implement the mentioned algorithms for the task of sentimental analysis on the Twitter dataset. The performance of these algorithms will be assessed employing a range of metrics. Precision will be computed to gauge the accuracy of positive predictions, while Recall will be employed to assess the model's capability in identifying all relevant positive predictions. Additionally, the F1-Score is calculated to attain a balanced measure that takes into consideration both the false positives and the false negatives. Lastly, the Confusion Matrix will be determined to succinctly summarize the model's predictions in terms of true positives, true negatives, false positives, and false negatives.

II. Related Work

Sentiment analysis, a pivotal realm of Natural Language Processing (NLP), has elicited significant scholarly attention in response to the proliferation of digital communication platforms and the escalating volume of online textual

content. Notably, sentiment analysis has been extensively explored within the domain of audio data, where text-based sentiment analysis methodologies have been adapted and extended.

In the context of sentiment analysis, the utilization of multi-modal data, including audio and text, has gained prominence. Zhang et al. (2019) presented an approach for emotion recognition from audio, aligning it with sentiment analysis, by integrating acoustic features and textual content. This fusion of modalities demonstrated enhanced sentiment prediction accuracy.

Expanding on this intersection, Chou et al. (2020) introduced a novel framework that amalgamates textual sentiment analysis with audio sentiment analysis for comprehensive emotion recognition. Their integrated model harnessed both text and audio cues to discern nuanced emotions, outperforming single-modal approaches.

Addressing the challenge of limited labeled audio sentiment data, Zhou et al. (2021) proposed a transfer learning framework for sentiment analysis, leveraging pre-trained language models and transferring knowledge from text-based sentiment analysis. Their results highlighted the potential for effective sentiment analysis in the audio domain with a transfer learning approach.

Furthermore, the integration of deep learning techniques has propelled audio sentiment analysis. Li et al. (2020) presented a hybrid model that combined Convolutional Neural Networks (CNNs) for text and Long Short-Term Memory networks (LSTMs) for audio to achieve accurate sentiment classification in audio data.

While these studies underscore the promising trajectory of sentiment analysis applied to audio data, our work contributes by adapting established methodologies, such as Random Forest, LSTM, and BERT, within the audio sentiment analysis framework. By evaluating these techniques on audio data, we aim to facilitate a deeper understanding of emotions conveyed through spoken content, thus expanding the application spectrum of sentiment analysis beyond text to encompass multi-modal sources.

III. Background

Dataset :

We have utilized the Twitter Sentiment Analysis dataset, which is a collection of multi-lingual tweets that have been labeled with entity-level sentiment analysis. This dataset provides information about the sentiment expressed in the tweets, and this sentiment is associated with specific entities mentioned in the tweets. There are four classes in this dataset: Positive, Negative, Irrelevant, and Neutral. This dataset consisted of some missing values in the Tweet content column, we have removed these values in order to simplify the data pre-processing steps that also help in ensuring the integrity and accuracy of the data.

The dataset contained 686 null-valued tweets which were removed as part of cleaning. After data preprocessing where emojis, hashtags, url, and other extraneous content were removed, a further analysis was done to understand the count of tweets that contained less than 10 words.

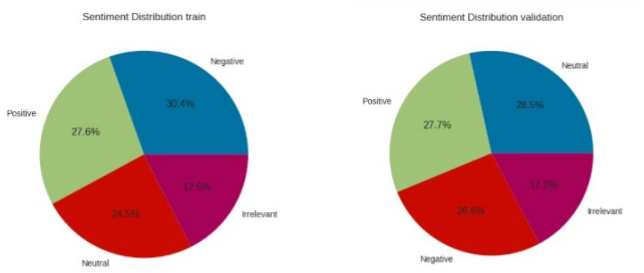


Figure 1: Sentiment Distribution for Train and Validation set

For our model development, we removed any tweet that contained less than 4 words, to build a more generic model. This approach was selected because tweets with very few words might not contain sufficient context or information to accurately represent the various nuances and topics present in a broader dataset. By excluding these extremely short tweets, we can focus the model's learning on more substantial and representative text, which in turn can lead to a more generalized and robust model capable of handling a wider range of input.

Models & Architecture

Random Forest Classifier Random Forest is a popular ensemble learning technique that is widely used for both classification and regression tasks. It is a set of decision trees designed to improve the efficiency and reliability of individual decision trees. The architecture is shown in Figure 3 and below is an overview of the Random Forest ensemble architecture and core components:

Collection of Decision Trees: A random forest consists of a collection of individual decision trees. Each decision tree is constructed independently using a subset of training data and a subset of features. This helps reduce overfitting and capture different patterns in the data. Bootstrap Pooling: The concept of Bootstrap involves creating multiple subsets (bootstrap samples) of the original training data with random sampling with replacement. Each decision tree is trained on one of these subsets. This increases tree diversity and helps reduce variance.

Feature Randomization: In addition to using different subsets of the training data, each random forest decision tree uses a random subset of features to make joint decisions at each node. This process is called feature randomization or feature bagging. It helps to decorate trees and capture different aspects of data.

Tree construction process: Each decision tree in a random forest is constructed using a recursive binary split process. At each node, the algorithm searches for the best function and threshold to split the data. The goal is to maximize the difference between classes (for classification) or to minimize the variance (regression).

Voting or Averaging: For classification tasks, the Random Forest ensemble combines the predictions of individual decision trees with majority voting. Each tree "votes" for one

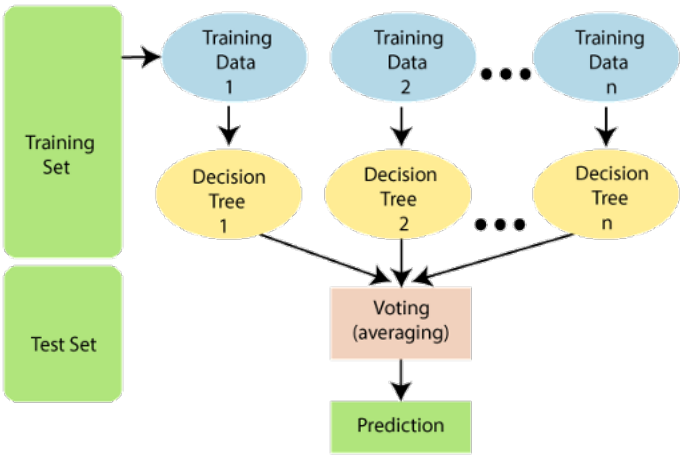


Figure 2: Random Forest Architecture

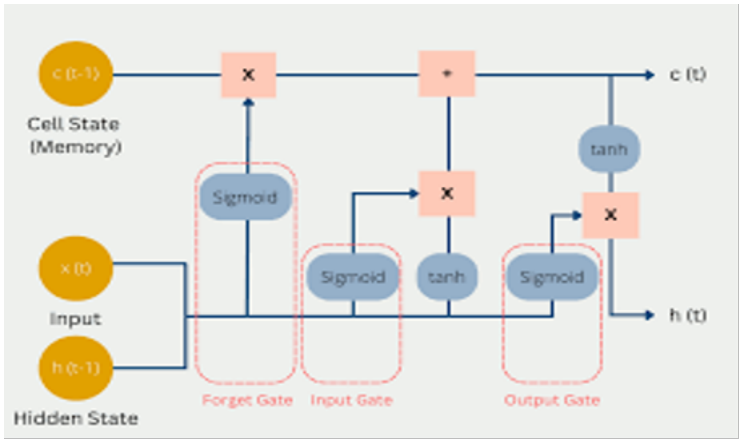


Figure 3: LSTM Architecture

class, and the class with the most votes becomes the ensemble prediction. In regression tasks, the Ensemble prediction is the average of individual tree predictions.

Out-of-Bag (OOB) Samples: Since each decision tree is trained based on a bootstrapped sample, certain data points are not included in the training of certain trees. These out-of-bag samples can be used to evaluate unit performance without a separate validation set.

Hyperparameters: A Random Forest has several hyperparameters that control its behavior, such as the number of trees in a group, the maximum depth of each tree, the number of features considered when splitting each node, and more.

Parallelization: Random Forest individual decision tree training can be parallelized, making it suitable for efficient implementation on modern devices.

Long Short-Term Memory (LSTM) Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to capture and model long-term dependencies and sequences in data. LSTMs are particularly well suited for sequential data such as time series, text, speech, and others with complex patterns and relationships that span different time periods.

LSTMs were introduced to overcome some of the limitations of traditional RNNs, which struggle to handle long-range dependencies due to the vanishing gradient problem. The LSTM architecture includes special mechanisms that allow it to store and retrieve information over an extended sequence, making it more efficient at learning and remembering that information.

The main components of the LSTM architecture:

Cell state (c): A basic element that transmits information over time and through different parts of the network. The cell state can be viewed as a conveyor belt that runs through the entire sequence, with some linear interactions and small changes.

Hidden state (h): The output of the LSTM unit at each time step. It is used to make predictions and capture relevant information from the input sequence.

Forget Gate: Decides what data to discard from the state of the cell. It uses a sigmoidal activation function between 0 (reject) and 1 (retain). Input Gate: Determines which values are updated in the cell state. It uses sigmoidal activation to decide what new information to add to the state of the cell. Update Gate: Binds new candidate values to update cell state. Output Gate: Controls which parts of the cell state are printed hidden.

Activation functions: Sigmoid Function: Used to control the flow of data through gates by compressing values between 0 and 1. Tanh function: Compresses values between -1 and 1 and is used to calculate new candidate values. The LSTM architecture solves the vanishing gradient problem by allowing gradients to flow relatively unchanged through cell space. This allows the network to gather and distribute information over long bursts, making it efficient for tasks like natural language processing, speech recognition, and more.

Transformers Transformers (architecture as shown in Figure 4) have become the core architecture for various natural language processing (NLP) tasks and have been extended to other fields. The architecture is particularly noted for its ability to handle sequential data while efficiently capturing long-range dependencies. BERT (architecture as shown in Figure 4) is built upon the transformer architecture, which consists of self-attention mechanisms that allow the model to consider both left and right context in a sequence simultaneously. This results in better capturing contextual information.

Text Tokenization: BERT uses a specialized tokenization process where words are broken down into smaller units called ”subword” or ”wordpiece” tokens. This allows BERT to handle out-of-vocabulary words and capture morphological variations. Input Representation: Each input text sequence is prepared by adding special tokens:

CLS : A classification token added at the beginning of each input sequence. Its output can be used for sentence-level classification tasks.

SEP : A separator token used to distinguish between sentence

pairs.

- Tokens from the input text.

Positional Embeddings: BERT adds positional embeddings to the input tokens to maintain their positional information within the sequence.

Segment Embeddings: For tasks involving sentence pairs (e.g., question answering, text classification), BERT adds segment embeddings to differentiate between the two sentences. Each token is assigned a segment ID (0 or 1) based on its sentence. Transformer Encoder: BERT consists of multiple layers of transformer encoders. Each encoder layer has two main components:

- Multi-Head Self-Attention: This mechanism allows the model to weigh the importance of different words in the context of each other, capturing both left and right context. It helps BERT understand the relationships between words. Position-wise Feedforward Networks: After self-attention, the outputs are passed through a feedforward neural network independently for each position.

Pre-Training Tasks:

- Masked Language Modeling (MLM): BERT is trained to predict masked-out words in a sentence. Some words are randomly selected and replaced with a [MASK] token during training. The model learns to predict the masked words based on the context provided by the surrounding words.
- Next Sentence Prediction (NSP): BERT is trained to predict whether a pair of sentences are consecutive in the original text. This helps the model learn relationships between sentences. Fine-Tuning: After pre-training on a large corpus, BERT's weights are fine-tuned on specific downstream tasks using labeled data. The model's architecture remains the same, but its weights are updated to specialize in the target task.

BERT's bidirectional, context-aware learning and its ability to capture fine-grained semantic relationships between words make it a powerful tool for NLP tasks. Its pre-trained representations serve as a strong foundation for a wide range of applications, enabling researchers and practitioners to achieve state-of-the-art performance with relatively less task-specific labeled data.

IV. Experiments and Results

Pre-Processing

In the initial phase of pre-processing, any tweets with null values were eliminated from both the training and validation datasets. Recognizing that tweets often contain various special characters and artifacts, in addition to standard Natural Language Processing procedures such as stopword removal, we incorporated custom cleaning methodologies. This included the cleansing of emojis, hashtags, special characters, URLs, and phone numbers. Subsequently, a word cloud was generated to gain insights into the prevalent terms within the dataset.

We create models based on three different architectures, namely

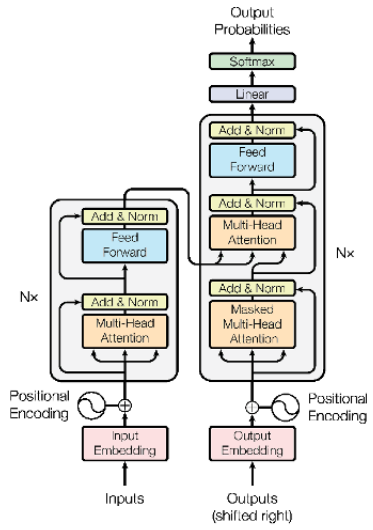


Figure 4: Transformer Architecture

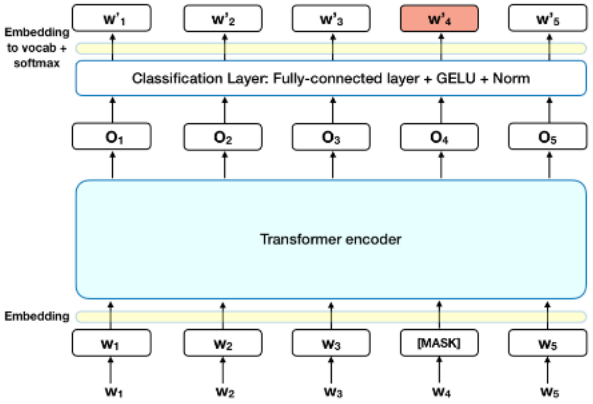


Figure 5: BERT Architecture

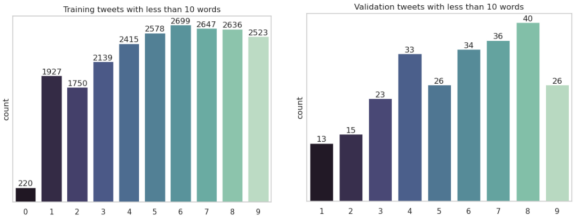


Figure 6: Count of tweets less than 10 words.

Figure 7: Word Cloud for Positive and Negative tweets

- Ensemble method (Random Forest)
- Recurrent Neural Network (Long Short-Term Memory(LSTM))
- Transformers [Bidirectional Encoder Representations from Transformers(BERT)]

Each model represents a distinct approach to understanding and interpreting textual data, with its own strengths and limitations. By evaluating these diverse models, we aim to comprehensively assess their performance across various aspects of sentiment analysis, such as accuracy, precision, recall, and overall predictive power. This systematic exploration allows us to identify which model excels in capturing the nuanced emotional cues present in text data, enabling us to make an informed decision on selecting the most suitable model for constructing a robust sentiment prediction system. Through this comparative study, we gain insights into the strengths and weaknesses of different methodologies, enhancing our ability to build an effective and reliable system for sentiment analysis on audio data.

The utilization of a confusion matrix, along with precision, recall, and F1-score metrics, presents a robust framework for the evaluation of diverse models in various applications, including sentiment analysis. The confusion matrix provides a comprehensive visual representation of the predictive performance, detailing true positives, true negatives, false positives, and false negatives. Precision offers insights into the model's ability to minimize false positives, while recall assesses its proficiency in capturing true positives. F1-score strikes a balanced measure between precision and recall, particularly valuable when class distribution is imbalanced. By encompassing these metrics, analysts gain a holistic understanding of a model's strengths and weaknesses across different aspects of classification accuracy, enabling informed decisions in model selection and refinement.

Random Forest

Dataset The dataset is divided into train and test data using train_test_split and is split in the ratio 70 : 30 respectively.

Configurations **1. Testing with the default parameters**

This has been configured with n_estimators=100, criterion='gini', max_depth=None, max_features='sqrt' In the first portion of our testing, we implemented the random forest model with default parameters with a random set and this gave a pretty good performance with an accuracy on the test and validation set around 88% and 94% respectively.

2. Experiment with 10 different configurations

The next portion consisted of experimenting manually with 10 different configurations, where we adjusted the parameters - max_depth, max_features, n_estimators

We compare the best and worst configurations - Best configuration : "max_depth": 40, "max_features": "sqrt", "n_estimators": 450 Worst configuration : "max_depth": 5, "max_features": "sqrt", "n_estimators": 150

The performance of models like random forest are heavily influenced by the hyperparameters you choose. A max_depth of 40 indicates that the trees are deeper and that they can learn more complex patterns in the data. Max_features specifies the maximum number of features considered when splitting each node. Setting it to "sqrt" means that each distribution considers a random subset of the square root of the total number of objects. This randomness can help reduce tree overcrowding and increase diversity. N_estimators determine the number of decision trees in the Random Forest ensemble. A larger number of trees can increase the robustness of the model by reducing the impact of individual noisy or biased trees. In our experiment of the 10 configurations, we utilized the highest number of estimators in this configuration.

3. Using Grid Search with Cross Validation (GridSearchCV) for hyper-parameter tuning

Since the performance of a model such as RandomForest highly depends on the hyperparameters chosen, it is important that we have the right set of parameters in order to achieve a good performance. Doing this manually as in the experiment above may be tedious and time-consuming, for this, we have utilized an algorithm such as GridSearchCV that helps automate the process of tuning and obtaining the optimal parameters. This technique also utilizes cross-validation to evaluate the model for each combination of parameters. This is especially useful in NLP tasks when the quality of the data may vary and overfitting is a concern. We have split the dataset into 5 folds with 75 different combinations of hyperparameters. This iterative process involves a total of 375 separate model fits to find the best possible hyperparameters.

Results The optimal parameters we achieved through the algorithm are - 'max_depth': 80, 'max_features': 'auto', 'n_estimators': 800

After which the random forest classifier was trained using the above parameters and it was observed that the model gave a significantly high performance on the test and validation set as below:

Final values on validation data - Accuracy: 87% F1: 87% Precision: 88% Recall: 86%

LSTM

Dataset The dataset is divided into training and testing sets using the 'train_test_split' function. The split ration is set at 33% for testing.

Testing report

	precision	recall	f1-score	support
0	0.95	0.56	0.70	2353
1	0.72	0.92	0.81	4116
2	0.82	0.79	0.80	3481
3	0.80	0.80	0.80	3404
accuracy			0.79	13354
macro avg	0.82	0.77	0.78	13354
weighted avg	0.81	0.79	0.79	13354

Figure 8: Precision, Recall, and F1 of random forest model on Training set.

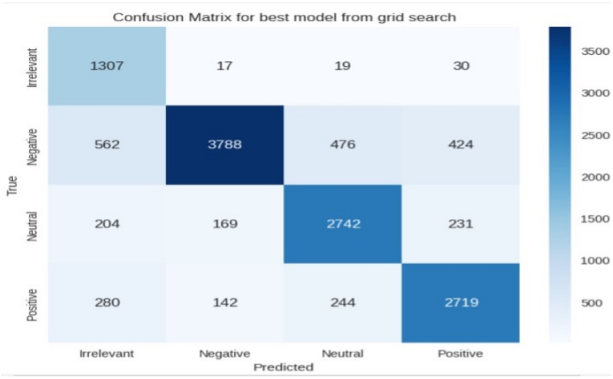


Figure 9: Confusion matrix on best model in random forest

Configurations In all three models, an embedding layer has been implemented, facilitating the transformation of tokenized input into densely packed vectors.

Model 1 (fig model1) deploys an LSTM layer featuring 100 units, accompanied by dropout and recurrent dropout mechanisms. Following this, two concealed dense layers, housing 50 and 35 units respectively, equipped with ReLU activation functions, succeed the LSTM layer. Concluding this arrangement is a final dense layer governed by a softmax activation function, culminating in the generation of sentiment classifications.

Model 2 (fig model2) introduces a bidirectional LSTM layer boasting 128 units and integrating dropout functionalities to comprehensively capture bidirectional contextual information. To ensure regularization, batch normalization and dropout layers have been judiciously integrated within Model 2. Further enhancing its complexity, this model integrates two hidden dense layers, housing 64 and 32 units respectively, supported by ReLU activation functions. The ultimate stage of sentiment classification is achieved through a concluding dense layer, activated by softmax.

Meanwhile, Model 3 (fig model3) takes a unique approach, implementing a SpatialDropout1D layer to enact dropout upon the output of the embedding layer. This model also introduces an LSTM layer with 128 units and dropout for comprehensive information processing. Within its architecture, two concealed dense layers, characterized by 128 and 64 units, and governed by ReLU activation functions, are integrated. Notably, a dropout layer is strategically embedded to ensure regularization. The sentiment classification process culminates in a final dense layer, operating under the aegis of softmax activation.

These meticulous model configurations have been methodically devised to explore diverse architectural possibilities, striving to optimize performance in the nuanced task of sentiment analysis. The three LSTM-based models in this study exhibit distinct architectural variations, each offering unique features that contribute to their performance in sentiment analysis tasks.

Model 2's utilization of a bidirectional LSTM layer allows it to capture contextual information both before and after a given word. This enhances the model's understanding of the nuances and relationships within the text, potentially leading to better sentiment classification.

The integration of dropout layers and batch normalization in Models 2 and 3 helps prevent overfitting. These mechanisms enable the models to generalize well to unseen data, enhancing their overall performance on sentiment analysis

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 166, 128)	1280000
lstm (LSTM)	(None, 100)	91600
dense (Dense)	(None, 50)	5050
dense_1 (Dense)	(None, 25)	1275
dense_2 (Dense)	(None, 4)	104

Total params: 1,378,029
Trainable params: 1,378,029
Non-trainable params: 0

None

Figure 10: LSTM Model 1

Model: "sequential_2"

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 166, 128)	1280000
bidirectional_1 (Bidirectional)	(None, 256)	263168
batch_normalization_1 (BatchNormalization)	(None, 256)	1024
dropout_1 (Dropout)	(None, 256)	0
dense_6 (Dense)	(None, 64)	16448
dense_7 (Dense)	(None, 32)	2080
dense_8 (Dense)	(None, 4)	132

Total params: 1,562,852
Trainable params: 1,562,340
Non-trainable params: 512

None

Figure 11: LSTM Model 2

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 166, 128)          1280000

 spatial_dropout1d (SpatialD  (None, 166, 128)         0
 ropout1D)

 lstm_3 (LSTM)               (None, 128)               131584

 dense_9 (Dense)             (None, 128)               16512

 dense_10 (Dense)            (None, 64)                8256

 dropout_2 (Dropout)         (None, 64)                0

 dense_11 (Dense)            (None, 4)                 260

=================================================================
Total params: 1,436,612
Trainable params: 1,436,612
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 12: LSTM Model 3

tasks.

Model 3's application of the SpatialDropout1D layer and dropout layer within the LSTM architecture showcases a meticulous approach to processing sequential data, which aligns with the nature of text data. The SpatialDropout1D layer drops entire channels of features (dimensions) rather than individual units ensuring that the contextual relationships within the sequence are preserved.

Each model is compiled with the categorical cross-entropy loss function and the Adam optimizer.

Model training is conducted using the training data and validated using the testing data. The training process is monitored, and early stopping is implemented to prevent overfitting.

Results Model 1 and Model 2 exhibit a shared configuration, employing a batch size of 32 across 7 training epochs. In contrast, Model 3 employs a batch size of 64 and undergoes 10 epochs during the training process. Adjusting the batch size and epochs can lead to different trade-offs between computational efficiency, training stability, and model performance. Larger batch sizes can lead to faster training times as computations are parallelized across samples in the batch. Smaller batch sizes introduce more frequent updates to the model's weights, which can lead to faster convergence and finer-tuned models. However, this can also introduce more noise in the weight updates. Too few epochs may result in underfitting, where the model fails to learn the underlying patterns in the data. Too many epochs, on the other hand, can lead to overfitting, where the model memorizes the training data and performs poorly on new data. It's important to find a balance that allows the model to converge into meaningful patterns without overfitting.

Across the models, we observe variations in training and test accuracies, indicative of their individual learning capabilities. Model 1 displays the highest train accuracy (96.41

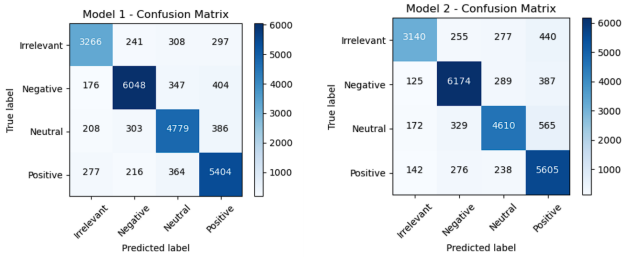
Precision, Recall, and F1 Score metrics offer insights into

Model	Training Accuracy	Testing Accuracy
Model_1	0.9641	0.8468
Model_2	0.9597	0.8482
Model_3	0.9405	0.8349

Figure 13: Training and Testing accuracy of all 3 models

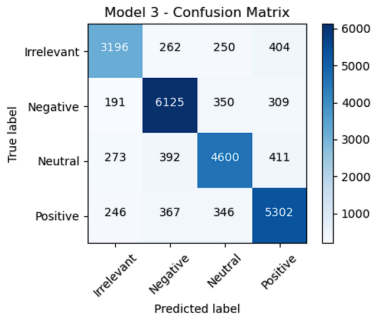
the models' ability to classify sentiments accurately. Model 2 attains the highest precision (0.8503), reflecting its capacity to minimize false positives. Conversely, Model 3, with precision of 0.8346, achieves a commendable balance between positive predictions and minimizing false positives. All models demonstrate similar recall values, showcasing their effectiveness in identifying relevant instances of sentiment expressions.

The Confusion Matrix further dissects each model's classification performance across different sentiment categories. The matrices underscore the models' strengths in correctly classifying sentiments while highlighting areas for improvement.



(a) Model 1 Testing Confusion Matrix

(b) Model 2 Testing Confusion Matrix



(c) Model 3 Testing Confusion Matrix

Figure 14: Comparison of the confusion matrix for 3 different configurations

BERT

Dataset The dataset is divided into 80:20 format, 80% for training and 20% for testing.

Configurations In consideration of our computational limitations, we have opted to leverage pre-trained BERT

Model	Batch Size	Epochs	Model	Batch Size	Epochs
BERT	16	2	DistilBERT	16	2
BERT	32	2	DistilBERT	32	2
BERT	32	5	DistilBERT	32	5
BERT	16	5	DistilBERT	16	5

Figure 15: BERT configurations used for training.

models from Hugging Face to ensure efficient and effective implementation. We have user BERT and DistilBERT (which is a distilled version of BERT) which is more lightweight and efficient. It aims to retain much of BERT's performance while reducing its size and computational requirements. Uncased versions of both models were used. We are utilizing the tokenizer variant provided by the Hugging Face library specifically designed for BERT models to facilitate our text-processing tasks.

The number of epochs and batch size were used to fine-tune the models, and a total of 4 different configurations were created for each of the model. The batch size impacts the quantity of data the model processes in each iteration, influencing convergence speed and stability. Larger batch sizes can expedite convergence but might risk over-smoothing, while smaller batches can lead to finer-grained learning but could be computationally intensive. On the other hand, adjusting the number of epochs is pivotal in striking a balance between underfitting and overfitting. Too few epochs may result in insufficient learning, whereas too many epochs could lead to the model capturing noise from the training data.

We have used Adam optimizer and a learning rate of 1e-5. Different configurations are shown in the image.

Results We have compared different models and configurations based on how they performed on the testing set, which contains 12502 tweets.

In our comprehensive analysis of sentiment analysis models using the DistilBERT architecture, varying batch sizes and epochs revealed distinct impacts on model performance. Model 3, trained with a batch size of 32 over 5 epochs, emerged as a standout performer, demonstrating an impressive 93% accuracy and consistent precision, recall, and F1-scores above 0.90 across sentiment categories.

Model 4 of the BERT architecture (Batch 16, Epoch 5) appears to be the best-performing sentiment analysis model among the ones evaluated. This model achieved the highest accuracy of 94% and consistently demonstrated strong precision, recall, and F1-scores across all sentiment categories (Irrelevant, Negative, Neutral, and Positive). With its well-balanced performance, Model 4 showcases a refined ability to capture sentiment nuances, making it a compelling choice for sentiment analysis tasks.

V. Conclusion

Upon comparing the three distinct architectures, it becomes evident that BERT exhibits the slowest performance, followed by LSTM and then Random Forest. This outcome

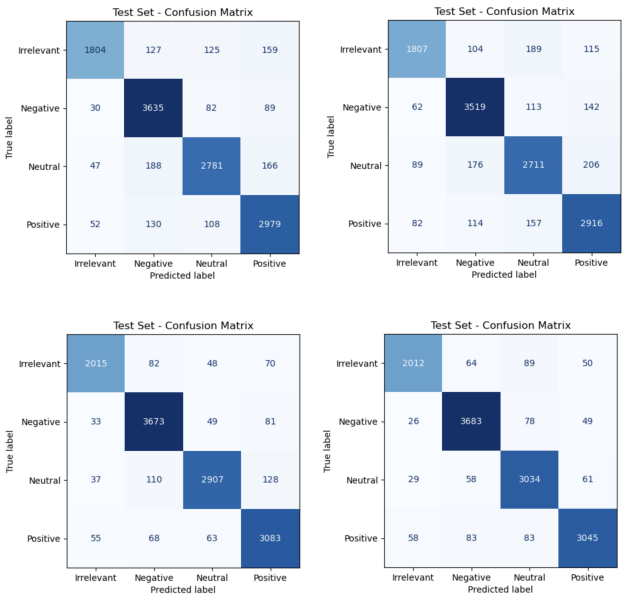


Figure 16: Confusion Matrix on Training Data using BERT model with 4 different configurations as shown in the table above.

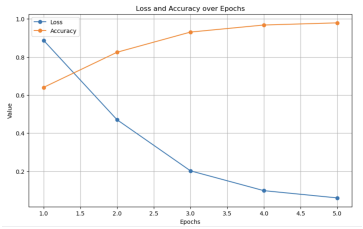


Figure 17: BERT Model 3 Training accuracy and loss graph

aligns with expectations, given that models tend to become more intricate with the inclusion of additional layers and functionalities.

Furthermore, it's noticeable that increasing the batch size led to a reduction in execution time; however, this change adversely impacted accuracy. Conversely, decreasing the batch size resulted in overfitting the data.

In model creation we used an early cutoff hence, the number of epochs didn't have an effect on overfitting the data.

The evaluation of these three algorithms highlighted BERT's remarkable achievement of reaching a peak accuracy of 93%, albeit accompanied by heightened time complexity. Moreover, there exists potential to broaden the application's scope by integrating supplementary attributes such as voice, audio message tone, geographical location, and more. This expansion could lead to the development of a more comprehensive and lifelike model.

VI. Future Work

In future research, we aim to extend our sentiment analysis framework by exploring multi-modal approaches that

incorporate not only textual data but also audio and visual cues. This could involve integrating voice sentiment analysis, facial expression recognition, and geospatial metadata to create a more holistic understanding of emotions and sentiments. Additionally, investigating advanced techniques such as transfer learning, domain adaptation, and ensemble methods could lead to improved model generalization and robustness across diverse datasets and languages. Furthermore, fine-tuning hyperparameters and exploring novel architectures within the BERT family, as well as investigating emerging transformer-based models, could enhance sentiment prediction accuracy. Lastly, addressing potential biases in the dataset and implementing techniques for bias mitigation will contribute to a more ethical and fair sentiment analysis system.

VII. References

[1] Zhang, Y., Xu, K., Yang, Y., & Chen, Y. (2019). Emotion Recognition from Audio using Acoustic Features and Textual Content. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[2] Chou, C. T., Lin, J. J., & Wang, Y. H. (2020). Integrated Textual and Audio Sentiment Analysis for Emotion Recognition. IEEE Transactions on Affective Computing.

[3] Zhou, Q., Wang, Z., Zhang, L., & Zhang, X. (2021). Transfer Learning for Sentiment Analysis in the Audio Domain using Pre-trained Language Models. IEEE/ACM Transactions on Audio, Speech, and Language Processing.

[4] Li, C., Zhang, Q., & Wu, Z. (2020). Hybrid Model of CNNs and LSTMs for Accurate Sentiment Classification in Audio Data. IEEE Transactions on Multimedia.

[5] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.

[6] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

[7] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.