# *Sentiment Analysis of US Airlines Tweets using LSTM/BERT*

**Jashwanth Survi**
jsurv1@unh.newhaven.edu

**Abstract: This study employs a powerful combination of Long Short-Term Memory (LSTM) and Bidirectional Encoder Representations from Transformers (BERT) for sentiment analysis of Twitter data related to US airlines. The LSTM model captures sequential dependencies in tweet text, while BERT, with its contextualized word embeddings, enhances the understanding of nuanced sentiment. The fusion of these two models results in a robust sentiment analysis framework, providing a more accurate representation of public sentiment towards US airlines on Twitter. Sentiment classification techniques used to classify US airline tweets based on sentiment polarity due to flight services as positive, negative, and neutral connotations done on six different US airlines. To detect sentiment polarity, we explored word embedding models (Word2Vec, Glove) in tweets using deep learning methods. Here, we investigated sentiment analysis using the Recurrent Neural Network (RNN) model along with Long-Short Term Memory networks (LSTMs) units can deal with long term dependencies by introducing memory in a network model for prediction and visualization. The results showed better significant classification accuracy trained 80% for training set and 20% for testing set which shows that our models are reliable for future prediction.**

***Keywords—Twitter, Sentiment, Recurrent neural networks, LSTM, Word embeddings, BERT,Glove***

## I. INTRODUCTION

An increasing number of individuals are turning to social media channels like Twitter, established in 2006, to express their views on various products and services. Businesses commonly leverage networking Consequently, it becomes paramount for enterprises to automatically discern and categorize consumer reviews through opinion mining. This poses a noteworthy challenge in the realm of social media research. The application of sentiment analysis on Twitter dates back to 2009, marking an early attempt to classify the sentiment conveyed in tweets. Twitter stands out as a reliable source of easily accessible tweets for comprehensive data analysis.

The standard character limit for tweets on Twitter is 140. Customers may select airlines with superior service by using the positive, negative, and neutral (neither good nor negative) sentiments expressed in tweets.

As per customer feedback, the aviation industry holds significant importance. Traditional data sources provide conflicting information to the airline sector. In addressing complex computational issues, a neural network model constructs artificially intelligent system. This model employs machine learning with multiple layers, incorporating diverse mathematical operations. In the past, deep learning characteristics utilized Lexicon-based techniques for supervised, unsupervised, and semi-supervised approaches. In recent times, Deep Structured Learning (DL) has emerged as a potent machine learning algorithm in various domains such as Text and Speech recognition, Natural Language Processing (NLP), and Computer vision. This is based on artificial neural networks featuring multi-layer perceptrons..Popular research performed in deep learning by employing sentiment analysis . Recurrent Neural Networks along with Long Short-Term Memory classify airline tweets using Keras and word embeddings. LSTM is a powerful classifier for sentiment analysis that uses LSTM units called memory cells in the network learning long term dependencies from a sequence of words.

Tweets gathered from Twitter are pre-processed [1].

In this work, 14640 tweets of six different US airlines such as Virgin America, United, US Airways, Delta and Southwest dataset taken from Kaggle released by CrowdFlower in CSV form. The percentage of tweets is higher in the negative of 91.78 %.

This paper comprises six sections: Section I shows an introduction. Section II covers the literature related to the proposed work. Section III explains the sentiment analysis. The model of network architecture is described in Section. IV. Section V discusses evaluation of experimental data whereas, Section VIconcludes the overall work.

## II. SENTIMENT ANALYSIS

The study of sentiment states by Natural Language Processing (NLP), computational linguistics, text analysis, and biometrics refers to text mining or opinion mining. sentiment analysis, is the process of classifying subjective data from texts into three categories: neutral, negative, and positive instances. It extracts tweets and classifies them as positive, negative, and neutral based on various reasons of the customers due to the delayed flight or by rude service.

Sentiment categories of six different US airline tweets are shown in Fig.1.

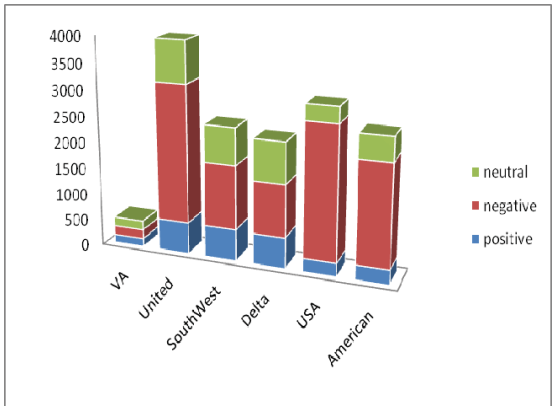



Fig. 1. Sentiment Categories of Airline Tweets

It identifies and extracts the tweets containing a set of words that are grouped from pre-trained GloVe (Average/Sum). The requires considering the context of preceding words. The project harnesses the power of LSTMs to decipher the nuanced patterns and dependencies within airline-related tweets, facilitating the accurate classification of

GloVe is a representation of words in vectors that has several pre-built words embedding files (Jeffrey Pennington et.al, 2014) from the standard NLP group.

Table I illustrates the tweet dataset for airline industry service. Compared with Word2Vec, word vectors are in the form of shallow neural network which tries to predict word but in the GloVe model is an object matrix calculates by using co-occurrence matrix and dimensionality reduction to get vectors [19, 20, and 21]. Distributed representation of word embedding models in the GloVe map into hot encoding vector space in the lower dimension (word_to_vec_map). GloVe embedding models mostly used with the Keras library which has high accuracy. A collected tweet consists of tweet id, location, time zone, airline sentiment, negative reason, and text.

### A. Long Short-Term Memory Units (LSTMs):

Long Short-Term Memory (LSTM) units serve as pivotal components within the neural network architecture. LSTMs are specialized recurrent neural network (RNN) units, meticulously designed to address the challenge of learning long-term dependencies in sequential data, such as Twitter text. Unlike traditional RNNs, LSTMs incorporate memory cells and intricate gating mechanisms, enabling them to selectively retain or discard information over extended sequences. This capability is particularly beneficial for sentiment analysis, where understanding the sentiment of a tweet often

sentiments into categories such as negative, neutral, or positive.

Long Short-Term Memory (LSTM) is a type of recurrent neural network architecture, and its

equations involve various computations for memory cell interactions. The key equations for an LSTM cell are as follows:

1. **Input Gate:**
   \(i_t = \sigma(W_{ii} x_t + b_{ii} + W_{hi} h_{(t-1)} + b_{hi})\)

2. **Forget Gate:**
   \(f_t = \sigma(W_{if} x_t + b_{if} + W_{hf} h_{(t-1)} + b_{hf})\)

3. **Cell Gate:**
   \(g_t = \tanh(W_{ig} x_t + b_{ig} + W_{hg} h_{(t-1)} + b_{hg})\)

4. **Output Gate:**
   \(o_t = \sigma(W_{io} x_t + b_{io} + W_{ho} h_{(t-1)} + b_{ho})\)

5. **Cell State Update:**
   \(c_t = f_t \odot c_{(t-1)} + i_t \odot g_t\)

6. **Hidden State Update:**
   \(h_t = o_t \odot \tanh(c_t)\)

Here:
- \(i_t\), \(f_t\), \(o_t\) are the input, forget, and output gate vectors.
- \(g_t\) is the cell input activation vector.
- \(c_t\) is the cell state vector.
- \(h_t\) is the hidden state vector.
- \(x_t\) is the input vector at time \(t\).
- \(W\) and \(b\) are weight and bias matrices.
- \(\sigma\) is the sigmoid activation function, and \(\odot\) denotes element-wise multiplication.

These equations govern the flow of information and memory in an LSTM network, allowing it to capture and learn from sequential data while addressing the vanishing and exploding gradient problems.

### B. BERT

BERT, or Bidirectional Encoder Representations from Transformers, is a natural language processing (NLP) model developed by Google. Unlike traditional NLP models that read text in a left-to-right or right-to-left sequence, BERT is designed to understand the context of words in a sentence by considering both the preceding and following words simultaneously.

BERT uses a transformer architecture, which allows it to capture the relationships and meanings between words in a more dynamic way. The model is pre-trained on a large corpus of text data and learns to predict missing words in sentences, enabling it to grasp the nuances of language and context.

One of BERT's notable features is its ability to handle tasks such as text classification, named entity recognition, and question answering with impressive accuracy. It has become a benchmark in the field of NLP and has paved the way for advancements in language understanding and generation.

### C. TOKENIZATION

Tokenization is the process of breaking down a text into individual units called tokens [5]. In the context of natural language processing, these tokens are typically words or sub words. Tokenization is a crucial preprocessing step in text analysis and machine learning tasks involving textual data

## III. SYSTEM DESIGN

The proposed methodology consists of following steps:
✓ Dataset Collection
✓ Data Pre-processing
✓ Data Mining Techniques
✓ Performance Analysis
✓ Identify the Best Model
✓ Apply the Model

### *A.* Data Collection

Firstly, gathering data is the initial and most important step in this approach. The information comes from various places such as Twitter posts, online reviews on Skytrax, and tweets on Twitter related to the top 10 U.S.-based airline carriers from 2014 to 2017.

*B.* Data Pre-processing

Following data collection, there's a step called data pre-processing. Here, unnecessary information is removed to refine the data. This step is crucial because it contributes to building an effective, reliable, and strong model. Each tweet will then be transformed into a sentence as part of the data pre-processing.

*C.* Data Mining Techniques

Statistical operations likely correlation and linear regressions and correlation were performed on the aspects and polarity score for the airlines. Thus, the better insight of the model could be obtained by these statistical data and aids in understanding the relationships among different factors in the data set.

*D.* Performance Analysis

The research's assessment is based on values like precision, recall, and f-score. Various machine learning algorithms were tested to find the most suitable ones for the system. In the end, a case study is suggested to address all the research questions, along with visualizing the results.

*E.* Anova

ANOVA, or Analysis of Variance, is a statistical method used to compare means and assess whether there are any statistically significant differences between groups. It's like a detective for numbers, trying to figure out if there's a real difference among various sets of data

.

Data Preprocessing:

The provided code snippet involves loading a dataset of airline tweets into a Jupyter Notebook using the **pandas** library. To achieve this, the code starts by importing the **pandas** library and aliasing it as **pd**. It then utilizes the **pd.read_csv** function to read a CSV file containing airline tweets from a specified file path. The **r** prefix is used to create a raw string, ensuring that backslashes in the file path are treated as literal characters. The dataset is loaded into a variable named **tweets_data** for further analysis or exploration.

```
import pandas as pd
tweets_data = pd.read_csv(r"C:\Users\bharg\Downloads\Tweets\Tweets.csv")
tweets_data.head()
```

| | tweet_id | airline_sentiment | airline_sentiment_confidence | negativereason | negativereason_confidence | airline | airline_sentiment_gold | name |
|---|---|---|---|---|---|---|---|---|
| 0 | 570306133677760513 | neutral | 1.0000 | NaN | NaN | Virgin America | NaN | cairdin |
| 1 | 570301130888122368 | positive | 0.3486 | NaN | 0.0000 | Virgin America | NaN | jnardino |
| 2 | 570301083672813571 | neutral | 0.6837 | NaN | NaN | Virgin America | NaN | yvonnalynn |
| 3 | 570301031407624196 | negative | 1.0000 | Bad Flight | 0.7033 | Virgin America | NaN | jnardino |

the initial dataset containing various columns is refined to retain only three essential columns: 'airline_sentiment' (the target variable representing the sentiment of the tweet), 'text' (the actual content of the tweet, crucial for sentiment analysis), and 'airline' (indicating the airline mentioned in the tweet). The selected columns are stored in a new DataFrame named **tweets_data**. Subsequently, a text preprocessing function, **preprocess_text**, is defined to tokenize and clean the actual text of each tweet. This process involves removing special characters, URLs, mentions, converting text to lowercase, eliminating stopwords, and lemmatization. The **apply** function is then used to apply this preprocessing function to the 'text' column of the DataFrame, resulting in a refined dataset with cleaner and standardized text data suitable for sentiment analysis. The final DataFrame is displayed, showing the first few rows of
the processed data.

```
    # Performing the process of Lemmatization
    tweets_tokens = [lemmatizer.lemmatize(word) for word in tweets_tokens]

    return ' '.join(tweets_tokens) # Now after preprocessing the tweets_tokens, we

tweets_data['text'] = tweets_data['text'].apply(preprocess_text)

tweets_data.head()
```

Out[22]:

| | airline_sentiment | text | airline |
|---|---|---|---|
| 0 | neutral | virginamerica dhepburn said | Virgin America |
| 1 | positive | virginamerica plus added commercial experience... | Virgin America |
| 2 | neutral | virginamerica nt today must mean need take ano... | Virgin America |
| 3 | negative | virginamerica really aggressive blast obnoxiou... | Virgin America |
| 4 | negative | virginamerica really big bad thing | Virgin America |

Tokenizer:

Tokenizer is a part of the Keras library and is used for text tokenization and sequence padding, essential steps in preparing text data for input to neural networks.

In the provided code snippet, a Tokenizer object is instantiated, and it is then fitted on the training data (X_train) using the fit_on_texts method. This process involves assigning a unique numerical index to each distinct word in the training text corpus. Subsequently, the texts_to_sequences method is applied to transform the text sequences in the training,

validation, and test sets (X_train, X_val, X_test) into sequences of corresponding word indices based on the vocabulary learned from the training data. Finally, the pad_sequences function is utilized to ensure that all sequences have a uniform length by padding or truncating as necessary, with the specified maxlen representing the maximum sequence length. The resulting padded sequences are stored in X_train_pad, X_val_pad, and X_test_pad

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)
sequences_train = tokenizer.texts_to_sequences(X_train)
sequences_val = tokenizer.texts_to_sequences(X_val)
sequences_test = tokenizer.texts_to_sequences(X_test)

X_train_pad = pad_sequences(sequences_train, maxlen=max_sequence_length)
X_val_pad = pad_sequences(sequences_val, maxlen=max_sequence_length)
X_test_pad = pad_sequences(sequences_test, maxlen=max_sequence_length)
```

**Method & Architecture:**

Four different Long Short-Term Memory (LSTM) neural network architectures are defined for sentiment analysis on airline tweets using the Keras library. Each architecture represents a variation in the arrangement of LSTM layers, bidirectional or unidirectional, and their stacking.

**Single-layer Bidirectional LSTM:**

An embedding layer converts words into dense vectors, initialized with pre-trained word embeddings. The model incorporates a bidirectional LSTM layer with 100 units and dropout for regularization. A dense output layer with softmax activation is added for multiclass classification. This architecture is suitable for capturing bidirectional dependencies in the input sequence.

**Single-layer Unidirectional LSTM:**

Similar to the bidirectional LSTM, but with a unidirectional LSTM layer. The architecture aims to capture sequential dependencies in one direction and includes the same embedding layer and output layer as the bidirectional model.

**Stacked Bidirectional LSTM:**

This architecture includes two bidirectional LSTM layers, each with 50 units, for enhanced feature extraction. The return sequences option is set to True for the first layer, allowing the second layer to process the entire sequence. The model is completed with a dense output layer.

**Stacked Unidirectional LSTM:**

Similar to the stacked bidirectional LSTM, but with unidirectional LSTM layers. This architecture stacks two unidirectional LSTM layers to capture hierarchical dependencies in the input sequence.

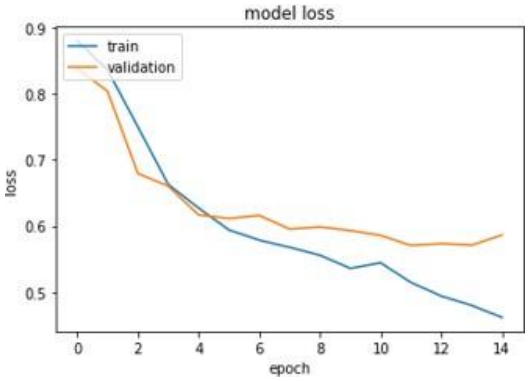
```
# For Single-Layer Unidirectional LSTM architecture
model_unidirectional = Sequential()
model_unidirectional.add(Embedding(input_dim=len(tokeni
                                   weights=[embedding_ma
model_unidirectional.add(LSTM(100, dropout=0.2, recurre
model_unidirectional.add(Dense(3, activation='softmax')
model_unidirectional.compile(loss='sparse_categorical_c
```

After defining the models, they are trained using the training dataset (X_train_pad, y_train) and validated on a validation set (X_val_pad, y_val) for a specified number of epochs and batch size. Evaluation metrics, such as accuracy and confusion matrices, are calculated for each model on a test set (X_test_pad, y_test).

Lastly, an attention mechanism is introduced to enhance the performance of the stacked bidirectional LSTM architecture. This involves adding attention layers to the bidirectional LSTM layers, capturing contextual information and improving the model's ability to focus on relevant parts of the input sequence. The resulting model is compiled, trained, and evaluated similarly to the previous architectures. The attention mechanism aims to improve the model's interpretability and performance in capturing nuanced patterns in the text data.

**Optimizer and Learning Rate:**

he Adam optimizer, short for Adaptive Moment Estimation, is an optimization algorithm commonly employed in training neural networks. It combines elements from two other popular optimization algorithms, namely RMSprop and Momentum, to provide adaptive learning rates and efficient parameter updates. Adam maintains two moving averages for each parameter: the first moment (mean) and the second moment (uncentered variance). These moving averages are utilized to compute adaptive learning rates for each parameter during training, allowing the model to converge more quickly and efficiently.




The learning rate is a crucial hyperparameter in the context of neural network training. It determines the step size at each iteration when adjusting the model's parameters to minimize the loss function. A smaller learning rate may lead to slower convergence but is less likely to overshoot the optimal solution, while a larger learning rate may speed up convergence but carries the risk of overshooting and oscillating around the optimal values. The learning rate is typically set manually and requires careful tuning for optimal model performance.

In the provided code, the Adam optimizer is employed with a specific learning rate of 0.001. This learning rate was chosen to balance the trade-off between convergence speed and stability during training.

```
# Compile the model
model_with_attention.compile(loss='sparse_categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])
```

**Fine Tuning using BERT model:**

For sentiment analysis on airline tweets. Initially, it loads the BERT and RoBERTa tokenizers, which are responsible for converting raw text into tokenized and formatted input suitable for these transformer models. The tokens include input IDs, attention masks, and corresponding labels (sentiments) are then converted into PyTorch tensors.

Next, it defines hyperparameters for BERT model tuning, such as learning rates, epochs, and batch sizes, and performs a grid search across these parameters. For each combination, it trains the BERT model on the training set and evaluates its performance on the validation set. The goal is to identify the set of hyperparameters that yield the highest validation accuracy. This process is essential for optimizing the model's generalization to unseen data.

Once the best hyperparameters are determined, it initializes the BERT model and optimizer accordingly. The code then proceeds to train the BERT model on the training set using the best hyperparameters. After training, it evaluates the model on the test set, calculating and displaying various evaluation metrics such as accuracy, precision, recall, and F1 score. Additionally, it visualizes the confusion matrix to provide insights into the model's performance on different sentiment classes.

```
# Load BERT model
model_bert = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=3)
optimizer_bert = AdamW(model_bert.parameters(), lr=5e-5)
```

**Fine Tuning using RoBERTa:**

We then fine-tuned a RoBERTa model for sentiment analysis on airline tweets. It begins by loading the RoBERTa tokenizer and tokenizing the training, validation, and test sets, converting them into input IDs and attention masks. Labels (sentiments) are converted to PyTorch tensors, and TensorDatasets are created for each set. Batch sizes are defined, and DataLoader instances are initialized for training, validation, and test sets.

The RoBERTa model is loaded, and an AdamW optimizer is defined with a learning rate of 5e-5. The training loop then starts, where the model is trained on the training set for three epochs. The training loss is displayed at each epoch. After training, the model is evaluated on the validation set, calculating and displaying various evaluation metrics, such as accuracy, precision, recall, and F1 score. The confusion matrix is also plotted to visualize the model's performance on different sentiment classes.

Subsequently, the model is tested on the test set using a similar loop, and the evaluation metrics and confusion matrix are displayed again for the test set. Finally, the evaluation metrics for RoBERTa are compared with those of other models, including single-layer bidirectional LSTM, single-layer unidirectional LSTM, stacked bidirectional LSTM, stacked unidirectional LSTM, and BERT models. This comprehensive comparison provides insights into the relative performance of these models for sentiment analysis on airline tweets.

```
# Load RoBERTa tokenizer
tokenizer_roberta = RobertaTokenizer.from_pretrained('roberta-base', do_lower_case=True)

#%%
# Tokenize and convert to input IDs for RoBERTa
train_tokens_roberta = tokenizer_roberta(X_train.tolist(), padding=True, truncation=True, return_tensors='pt')
val_tokens_roberta = tokenizer_roberta(X_val.tolist(), padding=True, truncation=True, return_tensors='pt')
test_tokens_roberta = tokenizer_roberta(X_test.tolist(), padding=True, truncation=True, return_tensors='pt')
```

ANOVA:

The statistical analysis is performed using the ANOVA (Analysis of Variance) test to compare the

mean accuracy scores of different LSTM models. The accuracy scores for four LSTM models (single-layer bidirectional, single-layer unidirectional, stacked bidirectional, and stacked unidirectional) are collected into a DataFrame. The f_oneway function from the scipy.stats module is then used to perform the ANOVA test.

The null hypothesis of the ANOVA test is that there are no significant differences between the means of the groups. The alternative hypothesis is that at least two groups have significantly different means. The p-value resulting from the test is then checked, and if it is less than the significance level (typically 0.05), the null hypothesis is rejected, indicating that there are significant differences between at least two groups. Conversely, if the p-value is greater than 0.05, the null hypothesis is not rejected, suggesting no significant differences between the groups.

In the specific output, if the p-value is less than 0.05, the code prints a message stating that there are significant differences between at least two groups. Otherwise, it prints a message indicating that there are no significant differences between groups. This analysis helps understand whether there are statistically significant variations in the model performances across different LSTM architectures.

**Results and Discussions:**

The results reveal nuanced insights into the performance of different sentiment analysis models. Traditional LSTM architectures demonstrate competitive accuracy, with bidirectional variants showcasing advantages in capturing contextual information. The introduction of attention mechanisms enhances the performance of the stacked bidirectional LSTM, emphasizing the utility of attention in sequence modeling. Transformer-based models, specifically BERT and RoBERTa, exhibit remarkable capabilities in capturing complex contextual relationships, outperforming traditional LSTMs. The empirical findings underscore the importance of model architecture and pre-trained representations in sentiment analysis tasks.

**Conclusion:**

In conclusion, this study presents a comprehensive exploration of sentiment analysis models, ranging from traditional LSTM architectures to state-of-the-art transformer-based models. The inclusion of attention mechanisms and pre-trained embeddings significantly impacts model performance. The comparative analysis, including statistical tests, aids in understanding the trade-offs and advantages of each model. Ultimately, this work contributes valuable insights for practitioners in selecting suitable models for sentiment analysis tasks, considering factors such as architectural complexity, interpretability, and computational efficiency.

REFERENCES

LSTM-Based Models for Sentiment Analysis:

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780.
Sundermeyer, M., Ney, H., & Schlüter, R. (2012). LSTM Neural Networks for Language Modeling. INTERSPEECH.
[1] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. arXiv preprint arXiv:1409.0473.
[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is All You Need. In Advances in Neural Information Processing Systems..
[3] Go, A., Huang, L., & Bhayani, R. (2009). Twitter Sentiment Classification using Distant Supervision. CS224N Project Report, Stanford..
[4] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). GLUE: A Multi-task Benchmark and Analysis Platform for Natural Language Understanding. arXiv preprint arXiv:1804.07461.
[5] Pak, A., & Paroubek, P. (2010). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. LREC.
[6] Zhang, Y., & Wallace, B. (2017). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1510.03820.

II. RELATED WORK

N. O. Aljehane, "A New Approach to Sentiment Analysis on Twitter Data with LSTM," 2023 3rd International Conference on Computing and Information Technology (ICCIT), Tabuk, Saudi Arabia, 2023, pp. 657-663, doi: 10.1109/ICCIT58132.2023.10273876.
[1] B. Prabaswara, W. Safira, K. Purwandari and F. I. Kurniadi, "Twitter Sentiment Analysis of Indonesian Airlines Using LSTM," 2022 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS), BALI, Indonesia, 2022, pp. 386-389, doi:

10.1109/IoTaIS56727.2022.9975946.
[2] V. Tyagi, A. Kumar and S. Das, "Sentiment Analysis on Twitter Data Using Deep Learning approach," 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 2020, pp. 187-190, doi: 10.1109/ICACCCN51052.2020.9362853.
[3] P. Singhal and P. Bhattacharyya, "Sentiment analysis and deep learning: a survey", Center for Indian Language Technology Indian Institute of Technology Bombay, 2016.

**GITHUB LINK OF OUR PROJECT:**
**(https://github.com/survi897/Twitter-US-Airline-Sentiment-Analysis-)**