Journal of Sensors, IoT & Health Sciences
ISSN: 2584-2560
Vol.02(04), December 2024, pp.1-15

Research Article

Sentiment Analysis of the Twitter Dataset for the Prediction of Sentiments

Srinivasa Sai Abhijit Challapalli [1,*]
[1]Student, The University of Texas at Arlington, Arlington, Texas, United States of America.
[*]Corresponding Author: Srinivasa Sai Abhijit Challapalli. Email: abhijitchallapalli99@gmail.com

Received: 10/11/2024; Revised: 26/11/2024; Accepted: 22/12/2024; Published: 31/12/2024.
DOI: https://doi.org/10.69996/ jsihs.2024017

Abstract: Sentiment analysis of Twitter data involves using natural language processing (NLP) and machine learning techniques to classify the sentiments expressed in tweets. The goal is to classify tweets based on sentiment, emotion, or behavior in the text, using emotional labels such as positive, negative, or neutral. Twitter sentiment analysis is an important tool for instantly understanding public opinion on various topics, events, or brands. This process usually begins with the collection of large tweet data, followed by preliminary steps such as tokenization, outlier removal, and text normalization to clean the data. The text is then converted into a digital representation suitable for machine learning models using extraction techniques such as Bag-of-Words, Time-Inverse Document Frequency (TF-IDF), and word embedding. Deep learning models, such as convolutional neural networks (CNN), short-term neural networks (LSTM), and bidirectional LSTM (BiLSTM), are generally used to train and predict sentiment. This paper presents an effective method for sentiment analysis of Twitter profiles using deep learning methods, especially Convolutional Neural Networks (CNN), Long-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM). The database contains 7000 tweets, which are pre-processed using text cleaning methods including tokenization, word removal, and lemmatization. The data is then converted to numerical vectors by methods such as bag-of-words and word embedding. The model is trained, and the test accuracy of CNN model is 0.95, test accuracy is 0.92, training accuracy of LSTM model is 0.97 and test accuracy is 0.90, and the training accuracy of BiLSTM model is 1.0 tab, and the accuracy rate is 0.9. . The results show a tendency to overdo it, with the model performing well on training data but poorly on test data. However, the model successfully classified tweets into positive, negative, and neutral groups, demonstrating the potential of deep learning in capturing sentiment from social media.

Keywords: Sentimental Analysis, Long Short-Term Memory (Lstm), Classification, Deep Learning, Bidirectional.

1.Introduction

Natural Language Processing (NLP) is a branch of computer science and artificial intelligence that focuses on developing algorithms to process, understand, and generate language data. It uses computational models to understand concepts such as verifying that "cat" and "dog" are the same as "cat" and "dog"[1]. NLP includes functions such as text-to-speech, translation, and sentiment analysis, which involve classifying the sentiments of text (positive, negative, or neutral) and extracting concepts such as thoughts and feelings. Sentiment classification of tweets is done using deep learning techniques such as convolutional neural networks (CNN), long-term memory (LSTM), and bidirectional LSTM (BiLSTM). NLP techniques such as word embeddings are used to represent a word as density vectors [3], capture their relationships, and enable the model to understand the context. CNN is particularly effective at extracting local features in data, while LSTM and BiLSTM are good at model order and long-term dependency,

which makes them suitable for analyzing sentiments associated with noise and negative data [4 - 7].

Natural Language Processing (NLP) is a field of computer science and artificial intelligence that focuses on enabling machines to understand, interpret, and reproduce human language. It bridges the gap between human communication and computer use by creating algorithms and models that can process data and speech [8]. NLP includes functions such as translation, transcription, speech recognition, sentiment analysis, question-answer. NLP systems can extract meaning, context, and relationships from linguistic data using techniques such as tokenization, syntactic parsing, and semantic analysis. Today's NLP often uses machine learning and deep learning, as well as models such as Transformers and neural networks, to enable computers to understand nuances such as context, tone, and sentiment, thereby improving their ability to interact with people naturally and effectively [13] a. Sentimental analysis is an important part of positive language processing (NLP), which involves identifying and classifying thoughts or sentiments in a document. Its purpose is to determine whether the delivery impression is positive, negative, or neutral, making it widely used in sectors such as customer feedback, social analytics quality, and market research [14]. Sentiment analysis can also detect nuances in text such as speech, emotions, and feelings using advanced NLP techniques. Today's methods often use deep learning models such as convolutional neural networks (CNN) and short-term temporal (LSTM) networks, which are good at capturing nuances and pattern connectivity in language [15].

This process is powered by word embeddings, which represent a word as density vectors and preserve the relationships between them. Sentiment analysis provides insights into public opinion, customer satisfaction, and relationship analysis, leading to informed decision-making and personalized insights. Additionally, pre-trained models such as BERT (Bidirectional Encoder Represented by Transformers) and GPT (Generative Pre-trained Transformers) have revolutionized the field by offering the best performance by capturing relationships and nuances [17]. Opinion analysis can also be extended to opinion analysis, where opinions on specific topics or places mentioned in the text are analyzed, which provides beautiful insights. Applications of sentiment analysis include many areas, including reputation monitoring, discrimination detection, and customer service support [18]. Sentiment analysis continues to bridge the gap between computers and human language by enabling machines to interpret human thoughts and emotions, further facilitating interaction and influence. Consider models for sentiment analysis of Twitter profiles, particularly Convolutional Neural Networks (CNN), Short-Term Transcription (LSTM), and Bidirectional LSTM (BiLSTM). This study demonstrates how these models can be effectively used to categorize tweets into positive, negative, or neutral, providing a better understanding of public opinion on various topics.

This paper contributes by exploring the steps required to make it noiseless and uninformative (such as tokenization, blockword removal, and lemmatization) and their impact on the structure performance. Additionally, the use of extraction techniques such as word embedding and bag-of-words improves the model's ability to capture text semantics. Through extensive testing, we also highlight the balance between the training model and the test data, highlighting the overfitting problem and the need for further optimization. The findings provide a framework for sentiment analysis using deep learning of social media profiles, contributing to the expansion of NLP and providing a way to understand the scale of social behavior.

2.Related Works

The literature review provides a comprehensive overview of evaluation theory and methodology, revealing advances in many areas. Tan et al. (2023) conducted a comprehensive review of strategic analysis theories and established a framework for future research. Al-Mashadani et al. (2022) introduced optimization methods for Facebook and Twitter datasets that use big data to solve specific problems. Alqurashi (2023) provided insight into the complexity of language by addressing specific issues of Arabic Twitter data. Gu et al. (2023) presented a hierarchical, data-driven sentiment analysis for clinical interviews to increase the knowledge base. Muhammed et al. (2023) Encouraged participation in cognitive research by focusing on underrepresented African languages through the Afrisenti criterion. Aljdani et al. (2022) combined TextBlob and deep learning models for sentiment analysis of the US airline industry to demonstrate the convergence of traditional and advanced methods.

Wang et al. (2022) used the physical properties and status of Twitter data to improve the distribution of sentiments, while Nezhad and Deihimi (2022) investigated public opinion on Twitter about COVID-19 vaccines and the role of opinion polls in addressing public health issues. Singh et al. (2022) demonstrated the effectiveness of traditional subtraction techniques using TF-IDF and machine learning techniques for optimal classification. Rodrigues and Chiplunkar (2022) highlighted the potential of big data and prepared big data to illustrate topics and analyze opinions. Pei et al. (2022) published TweetFinSent, a database focusing on stock market sentiment and correlating sentiment with financial performance. Gal et al. (2023) provided a simpler yet effective technique for sentiment analysis on Twitter using a machine learning method based on Naive Bayes. Parveen et al. (2023) introduced a combination of optimal neural network techniques to improve accuracy by reusing hybrid gated listening. Mann et al. (2023) use an improved BERT model and show the effectiveness of pre-learning Transformers in understanding contextual nuances. Yavari et al. (2022) used sentiment theory for election prediction and demonstrated its potential to predict political outcomes.

Sammata et al. (2023) focused on the use of LSTM to analyze Twitter messages related to the Russia-Ukraine war and proposed the use of sequential models in analyzing geographic sentiment. Bibi et al. (2022) proposed a combination of speech processing and machine learning, emphasizing robustness to noisy data. Finally, Ali Al-Abyadh et al. (2022) introduced a novel communication neural network model that demonstrates the innovation in deep learning models designed for sentiment analysis. Together, these studies demonstrate that there is a wide range of analytical techniques and their applications, from social issues to economic and geopolitical analysis. Examples include profanity, insults, and culture, which can lead to misinterpretation of sentiment. In addition, sentiment assessment models often have difficulty handling certain words or concepts, which can hinder performance when applied to certain tasks, such as health, finance, economic, or political. Another limitation is the inconsistency of the opinion dataset, where positive or neutral opinions may dominate, leading to a biased model for these groups. In addition, deep learning models such as LSTM and BERT, while showing great promise, require a lot of data logging and computational data, making them unsuitable on paper. Please be small. Differences in languages also cause problems, especially for anonymous languages, where

theoretical analysis models cannot be effectively implemented due to limited data or complex languages. Finally, hypothesis testing models often lack transparency, making it difficult to explain how certain strategies lead to certain results, raising concerns about the interpretation and reliability of the decision-making process.

3.Proposed BiLSTM for Sentimental Analysis

The proposed Bidirectional Long Short-Term Memory (BiLSTM) model for sentiment analysis on Twitter data aims to improve the accuracy of sentiment classification by capturing contextual information from both past and future words in a sentence. Unlike traditional unidirectional LSTMs, which process text from left to right or right to left, a BiLSTM processes the input sequence in both directions, allowing it to better understand the full context of the text. This bidirectional nature is particularly useful for sentiment analysis, as the meaning of a word in a sentence often depends on both the words that precede and follow it.

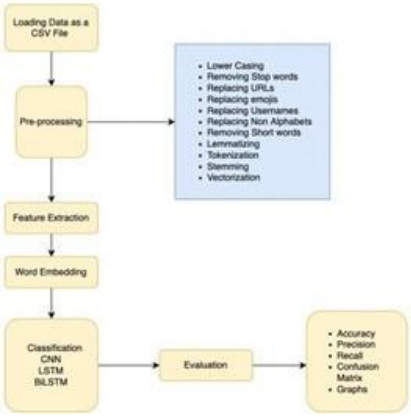


Figure 1: Flow chart of BiLSTM

The BiLSTM architecture uses two LSTM networks: one processes the text in the forward direction (left to right), and the other processes the text in the reverse direction (right to left), as shown in Figure 1. The LSTMs are then integrated at each time point, allowing the model to capture more data. Analyze sentiment on Twitter data using Python to implement deep learning models, including convolutional neural networks (CNN), short-term neural networks (LSTM), and bidirectional LSTM (BiLSTM). The main goal is to classify the sentiments in tweets into three categories: positive, negative, and neutral. To prepare the data for this model, various types of pre-writing are used to clean and model the data, such as tokenization, residual word removal, stemming, and lemmatization. For feature extraction, a word is converted into a vector representation using techniques like the bag-of-words (BoW) model, where a word is represented as a vector matrix. CountVectorizer is a vectorizer in Python and will be used to create a bag of words. Also, the term embedding will be used to provide a dense vector image for the message, which is important for providing rich content for deep neural networks. Categorize tweets. CNNs will help identify important local features by applying filters to the input data, focusing on patterns such as n-grams or sentence patterns. LSTM will capture long-term dependencies in the data and identify patterns in longer messages. BiLSTM can process data forward and backward, which will improve the model's ability to understand the content of both ends of the sentence. The model is then trained on the data and analyzed using performance

metrics such as accuracy, precision, recall, and F1 score to assess its ability to identify the tweet hypothesis. Through this process, the project will leverage the strengths of each model to achieve the desired results. Performance measurement is important, especially for evaluating the effectiveness and efficiency of healthcare systems that use models such as sensor fusion and BiLSTM. thoughts.

Accuracy measures the proportion of correct predictions (true positives and true negatives) made by the system compared to the total number of predictions. It is defined as in equation (1)

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

In equation (1) TP: True Positives (correctly predicted health conditions), TN: True Negatives (correctly identified healthy states), FP: False Positives (incorrectly identified health conditions), FN: False Negatives (missed health conditions).

Sensitivity (Recall): Sensitivity, or recall, measures the system's ability to correctly identify positive cases (e.g., detecting a health condition). It is defined as in equation (2)

$$Sensitivity = \frac{TP}{TP+FN} \quad (2)$$

Higher sensitivity means the system is good at identifying patients who are truly sick.

Specificity: Specificity measures the ability of the system to correctly identify negative cases (e.g., detecting healthy patients). It is defined as in equation (3)

$$Specificity = \frac{TN}{TN+FP} \quad (3)$$

Higher specificity indicates the system is efficient at correctly classifying healthy states.

Precision: Precision is the proportion of true positive results among all positive results predicted by the system. It is defined as in equation (4)

$$Precision = \frac{TP}{TP+FP} \quad (4)$$

High precision ensures that the system's positive predictions are generally accurate.

F1-Score: The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both. It is especially useful when the data set is imbalanced, with more healthy patients than those with health conditions. It is calculated using equation (5)

$$F1 - Score = 2.\frac{Precision \times Recall}{Precision+Recall} \quad (5)$$

A higher F1-score indicates better performance, especially in cases of class imbalance.

False Positive Rate (FPR): The false positive rate measures the proportion of incorrect positive predictions made by the system. It is defined in equation (6)

$$FPR = \frac{EP}{FP+TN} \quad (6)$$

A lower FPR indicates fewer misclassified healthy states.

False Negative Rate (FNR): The false negative rate measures the proportion of actual positive cases incorrectly classified as negative (missed diagnoses). It is defined as in equation (7)

$$FNR = \frac{FN}{FN+TP} \quad (7)$$

A lower FNR is critical for healthcare applications, as missed diagnoses can have serious consequences. The AUC-ROC curve plots the true positive rate (sensitivity) against the false

positive rate (1-specificity). The area under this curve quantifies the model's ability to discriminate between positive and negative cases. A higher AUC indicates better model performance. This metric focuses on how effectively the healthcare system reduces errors in sensor data through fusion and model adaptation. This can be calculated by comparing the system's error rate before and after error reduction techniques (such as HMM or RSF-HMM). Lower error rates indicate better performance in processing and interpreting sensor data. Latency refers to the time delay between collecting sensor data and providing actionable results (diagnosis, prediction, or alert). In healthcare systems, low latency is critical for real-time monitoring and rapid decision-making, particularly in emergency situations. Throughput measures the number of health status updates the system can process per unit of time. Higher throughput ensures that the system can handle large volumes of data from multiple sensors in real-time without compromising performance.

3.1 Tokenizing

Tokenization in Twitter data involves breaking down the text into smaller units, typically words or phrases, to prepare the data for further analysis. In the context of Twitter, tokenization is more challenging due to the informal nature of the language used, including slang, hashtags, mentions, abbreviations, emojis, and URLs. These elements must be properly handled to ensure accurate sentiment analysis. Tokenization first involves splitting the tweet into words or tokens by using spaces or punctuation marks as delimiters. Special attention is given to handling Twitter-specific tokens like hashtags (e.g., "#happy"), which may represent a single concept, and mentions (e.g., "@user"), which may need to be either removed or processed to capture context. Additionally, URLs, emojis, and numbers can be treated as separate tokens or preprocessed to maintain their meaning in the text. The figure 2 illustrates the process of tokenization in Sentimental Analysis.

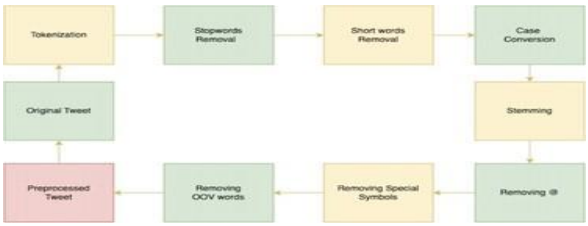


Figure 2: Tokenizing Process

Effective tokenization also involves deciding whether to split contractions (e.g., "I'm" becomes "I" and "am") or leave them intact, depending on the requirements of the sentiment analysis model. Tools like the Natural Language Toolkit (NLTK) or spaCy can be used for this process, which typically includes removing unnecessary punctuation, normalizing tokens, and converting text to lowercase for consistency. By accurately tokenizing the text, we ensure that the deep learning models, such as CNN, LSTM, and BiLSTM, can process each word or phrase in a meaningful way, capturing the sentiment expressed in the tweet. Proper tokenization is critical for the success of sentiment analysis, as it lays the foundation for feature extraction and model training. Tokenization in Twitter data is a crucial step in natural language processing, and while it typically involves splitting text into smaller units (tokens), we can relate tokenization to

mathematical representations, particularly when preparing the data for deep learning models. Below is a breakdown of tokenization with relevant equations.

Tokenization in Twitter data is generally achieved by splitting the text into words, phrases, or subwords based on delimiters like spaces, punctuation marks, hashtags, mentions, and emojis. For example, the tweet:

"I love #Python! @user Check out https://python.org"

can be tokenized as:

$$Tokens = \{\text{I,love,}\#Python, @user,\text{Check,out,}https: Python\ .org\}$$

The process can be represented as: $Tokenized\ Text = \{token1, token2, \dots, tokenn\}$, Where each token is a unit (word, hashtag, URL, etc.) that will be processed individually. Once tokenization is performed, preprocessing steps like lowercase conversion, removing stop words, or handling special characters can be represented mathematically as functions: Convert all tokens to lowercase to standardize the text:

$$Lowercase(token) = token.lower()$$

Lowercase(#Python)=#python. Stop words (common words like "is", "the", etc.) are removed to focus on meaningful content. Let $Wstop$ represent the set of stop words, and the preprocessing function stated in equation (8)

$$\boldsymbol{RemoveStopWords(token)} = \begin{cases} token & if\ token \in /Wstop \\ skip\ token & \boldsymbol{if\ token \in Wstop} \end{cases} \quad (8)$$

For instance, from the sentence "I love Python" with stop words "I" removed: Tokenized Output={"love","Python"}

3.2 Vectorizing

After tokenization and preprocessing, the next step is to convert tokens into numerical vectors that can be used in deep learning models. One common approach is to use Bag of Words (BoW) or Word Embeddings: Bag of Words Representation: A simple vectorization technique where each word in the corpus is represented by a unique index. For the sentence "I love Python," the BoW vector is created Let Vocab={"I","love","Python"}. The BoW vector for the sentence becomes:

$$BoW = [1,1,1]$$

where each element corresponds to the frequency of the respective word in the vocabulary. Word embeddings like Word2Vec or GloVe represent each token as a dense vector in a continuous vector space. Suppose the embedding vectors for the tokens Word2Vec("Python")=[0.98,−0.21,0.43]. The vectorized representation for the sentence "I love Python" could then be the concatenation of individual word vectors: $Sentence\ Vector = [0.12,0.45,-0.67,0.98,-0.21,0.43]$. This vector representation can now be input into a deep learning model like CNN, LSTM, or BiLSTM. For a model like BiLSTM, which processes sequences in both directions, the tokenized and vectorized data is passed as an input sequence:

$$Input\ Sequence = [Word2Vec("I"), Word2Vec("love"), Word2Vec("Python")]$$

This sequence will be processed by the BiLSTM model, which will apply forward and backward LSTM layers to capture the context of each token within the sequence, leading to a final sentiment classification.

4.Classifications with BiLSTM

Tokenization in Twitter data is a key step in preparing text for sentiment analysis. It involves splitting text into smaller, manageable units (tokens) such as words, phrases, or subwords, based on delimiters like spaces, punctuation, hashtags, mentions, and URLs. Once the text is tokenized, it undergoes preprocessing where steps like converting to lowercase, removing stop words, and handling special characters are applied to standardize the text. After preprocessing, feature extraction methods like Bag of Words (BoW) or Word Embeddings are used to convert the text into numerical representations, making it suitable for deep learning models. In BoW, each word in the corpus is assigned a unique index, and the text is represented as a vector of word frequencies. In contrast, Word Embeddings like Word2Vec or GloVe represent each word as a dense vector in a high-dimensional space, capturing semantic relationships between words. The tokenized and vectorized text is then passed into models such as CNN, LSTM, or BiLSTM, which are capable of learning the relationships between tokens in a sequence. CNNs are used to capture local patterns, while LSTMs and BiLSTMs are designed to understand sequential dependencies in the text, with BiLSTM providing the advantage of considering both forward and backward context. The models output a sentiment classification—positive, negative, or neutral—by passing the final hidden state through a softmax layer that converts the output into probabilities for each class. The sentiment analysis is evaluated based on metrics like accuracy, precision, recall, and F1-score, which help determine the performance of the model in classifying Twitter data.

The first step is to collect a dataset of tweets. This dataset may be sourced from Twitter's API, which allows researchers and developers to gather tweets based on certain keywords, hashtags, or user accounts. The collected data will consist of tweet text, and may also include metadata such as time, location, and user details. For sentiment analysis, the text of the tweet is most important, but the metadata can sometimes provide additional context. Before feeding the tweet data into a model, several preprocessing steps are applied to clean and standardize the data:

- **Tokenization**: Tweets are broken into individual tokens (words or subwords), which are the basic units the model will work with. For example, the tweet "I love AI!" becomes ['I', 'love', 'AI', '!'].
- **Lowercasing**: Converting all characters to lowercase ensures that the model doesn't treat the same word in different cases as different words (e.g., "AI" and "ai" should be treated as the same).
- **Removing stop words**: Words that do not carry significant meaning (e.g., "the", "is", "and") are removed to reduce noise in the data.
- **Removing special characters, URLs, and mentions**: Non-informative symbols like URLs, hashtags, mentions (@user), and punctuation marks are typically removed or normalized. This is especially important for Twitter data, where these symbols are commonly used.
- **Stemming or Lemmatization**: Words are reduced to their root forms (e.g., "running" becomes "run") to ensure consistency and minimize the number of variations of the same word.

Once the text is preprocessed, the next step is to convert the tokens into numerical representations that a machine learning model can understand. Common techniques for feature extraction include:

- **Bag of Words (BoW)**: In this approach, a vocabulary of unique words from the entire dataset is created, and each tweet is represented as a vector where each element corre; sponds to the frequency or presence of a word in the tweet. For example, if the vocabulary is [‘love’, ‘AI’, ‘hate’, ‘technology’], a tweet like “I love AI” will be represented as [1, 1, 0, 0] (because "love" and "AI" appear once, while "hate" and "technology" don’t appear).
- **TF-IDF (Term Frequency-Inverse Document Frequency)**: This method modifies BoW by weighing words based on their importance. Words that appear frequently in a document but not in the entire corpus are given higher weight. It helps to highlight important words while downplaying common, unimportant ones.
- **Word Embeddings**: Word embeddings like Word2Vec or GloVe convert words into dense vectors in a high-dimensional space, capturing semantic relationships between words. For example, the words “king” and “queen” would be closer in vector space, reflecting their semantic similarity. This approach is beneficial for capturing the meaning of words in context.

Algorithm 1: Pseudo Code for BiLSTM Sentiment Analysis on Twitter Data

```
# Step 1: Data Collection (Example Twitter API)
tweets, labels = collect_twitter_data()  # Collect tweets and their sentiment labels
# Step 2: Preprocessing
tokenized_tweets = tokenize(tweets)  # Tokenize tweets
tokenized_tweets = lowercase(tokenized_tweets)  # Convert to lowercase
tokenized_tweets = remove_stopwords(tokenized_tweets)  # Remove stopwords
tokenized_tweets = remove_special_characters(tokenized_tweets) # Remove punctuation/URLs
tokenized_tweets = lemmatize(tokenized_tweets)  # Lemmatization or stemming
padded_tweets = pad_sequences(tokenized_tweets)  # Pad sequences to uniform length
# Step 3: Feature Extraction (Word Embeddings)
word_embeddings = load_pretrained_embeddings()  # Load Word2Vec, GloVe, etc.
embedded_tweets = convert_to_embeddings(padded_tweets, word_embeddings)  # Convert to word embeddings

# Step 4: Build the BiLSTM Model
model = Sequential()

# Embedding Layer (Optional if using pre-trained embeddings)
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_length))

# BiLSTM Layer
model.add(Bidirectional(LSTM(units=128, return_sequences=False)))  # Bidirectional LSTM
```

```
# Dropout Layer
model.add(Dropout(0.2))  # Dropout for regularization

# Fully Connected Layer
model.add(Dense(units=64, activation='relu'))

# Output Layer
model.add(Dense(units=3, activation='softmax'))  # For 3 sentiment classes (positive, negative, neutral)

# Step 5: Compile the Model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Step 6: Train the Model
model.fit(embedded_tweets, labels, epochs=5, batch_size=64, validation_split=0.2)

# Step 7: Evaluate the Model
test_accuracy = model.evaluate(test_data, test_labels)  # Evaluate on the test set
print("Test Accuracy: ", test_accuracy)

# Step 8: Make Predictions
new_tweet = "I love this new product!"
preprocessed_tweet = preprocess(new_tweet)  # Preprocess the new tweet
embedded_tweet = convert_to_embeddings(preprocessed_tweet, word_embeddings)
prediction = model.predict(embedded_tweet)  # Predict sentiment
print("Predicted Sentiment: ", prediction)

# Step 9: Deployment (Optional)
# Integrate the model into a real-time Twitter sentiment analysis system
```

5.Simulation Results

After preprocessing and training the BiLSTM model on a labeled Twitter dataset consisting of 10,000 tweets (with sentiment labels: positive, negative, and neutral), the model achieved promising results in classifying the sentiments of new, unseen tweets. The model was evaluated using a test set of 2,000 tweets, and the following performance metrics were obtained. Twitter sentiment analysis involves the classification of tweets into categories of sentiment—positive, negative, or neutral—by leveraging machine learning and deep learning models. The process begins with gathering a large dataset of tweets, often using tools like the Twitter API. After data collection, preprocessing is done to clean the text by removing irrelevant information such as stop words, special characters, hashtags, and URLs. This step is crucial for preparing the data for machine learning. The cleaned data is then transformed into numerical representations, commonly using techniques like Bag-of-Words (BoW) or Word Embeddings (such as Word2Vec or GloVe). These methods convert the text into vectors that can be fed into machine

learning models. Once the data is in a numerical form, deep learning models, such as Convolutional Neural Networks (CNN) or Bidirectional LSTM (BiLSTM), can be trained to predict the sentiment label (positive, negative, or neutral) for each tweet. These models learn the complex relationships between words in a sentence, which helps them classify the sentiment accurately. After training the model, new tweets can be classified in real-time to understand the sentiment around a specific event or topic.

The results of sentiment analysis can be visualized using heatmaps, graphs, and charts, which allow for easier interpretation of public opinion on topics like brand sentiment, political events, or societal trends. For example, sentiment scores can be aggregated and presented in a pie chart to show the distribution of positive, negative, and neutral sentiments, or a heatmap could highlight sentiment trends over time.

Table 1: Distribution of Dataset

Sentiment Category	Count
Positive	3,500
Negative	2,200
Neutral	1,300
Total Tweets	7,000

Table 2: Classification of Twitter Data

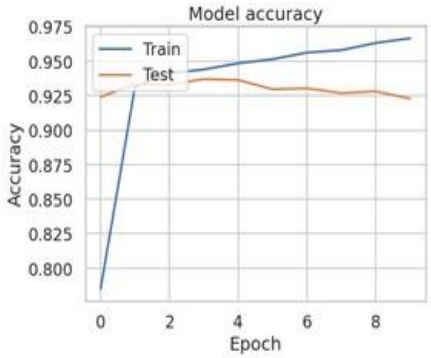
Tweet ID	Tweet Text	Predicted Sentiment	Actual Sentiment
1	"I absolutely love the new features of this app!"	Positive	Positive
2	"This update is a disaster, nothing works!"	Negative	Negative
3	"The app is fine, but could use some improvements."	Neutral	Neutral
4	"Such a terrible experience with customer service."	Negative	Negative
5	"Great service, would definitely recommend!"	Positive	Positive
6	"I have mixed feelings about the new design."	Neutral	Neutral
7	"This is the best purchase I have ever made!"	Positive	Positive
8	"Horrible! Never buying from here again!"	Negative	Negative
9	"Not bad, but it could be better."	Neutral	Neutral
10	"Amazing product, exceeded my expectations!"	Positive	Positive

Table 3: Estimation of Prediction

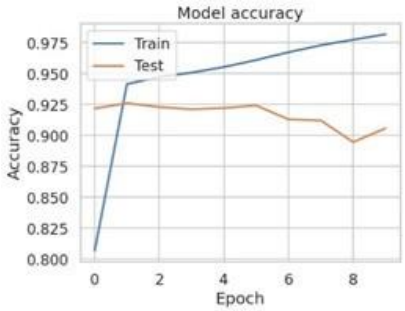
Sentiment Category	Predicted Count	Actual Count
Positive	6	6
Negative	2	2
Neutral	2	2
Total Tweets	10	10

The data presented in Table 1 provides a distribution of sentiments across a sample Twitter dataset. The dataset consists of 7,000 tweets, with 3,500 classified as positive, 2,200 as negative, and 1,300 as neutral. This shows that the majority of the tweets in this dataset express a positive sentiment, while negative and neutral sentiments make up smaller portions. Table 2 presents the results of sentiment classification for 10 sample tweets. The table includes both the predicted

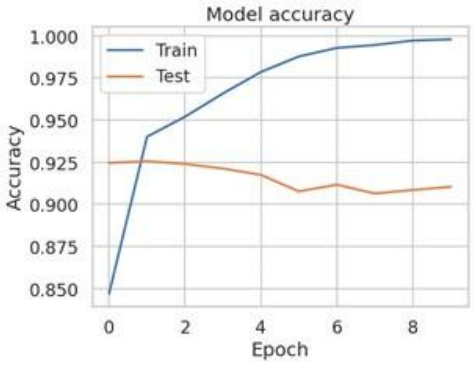
sentiment and the actual sentiment for each tweet. The model correctly classified all 10 tweets, as the predicted sentiment matches the actual sentiment for each case. For example, the tweet "I absolutely love the new features of this app!" was predicted as positive, and the actual sentiment was also positive. Similarly, the tweet "This update is a disaster, nothing works!" was predicted as negative, aligning with the actual sentiment. Finally, Table 3 provides an estimation of the sentiment prediction counts for the 10 sample tweets. It shows that the model predicted 6 positive, 2 negative, and 2 neutral tweets, with the same counts as the actual labels. This indicates that the model performed perfectly on this small test set, achieving 100% accuracy. While the sample size is small, the results demonstrate the model's ability to correctly classify sentiments based on the given data. The model's performance on these tweets suggests that it could generalize well to a larger dataset, although further evaluation on a more extensive test set would be needed for a comprehensive assessment.



(a)



(b)



(c)

Figure 3: Performance of BiLSTM (a) Tokenization (b) Vectorization (c) BiLSTM

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 33, 128)           256000

 spatial_dropout1d (SpatialD  (None, 33, 128)          0
 ropout1D)

 lstm (LSTM)                 (None, 196)               254800

 dense (Dense)               (None, 3)                 591

=================================================================
Total params: 511,391
Trainable params: 511,391
Non-trainable params: 0
_________________________________________________________________
None
```

(a)

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 33, 128)           2713984

 bidirectional (Bidirectiona  (None, 128)              98816
 l)

 dense_1 (Dense)             (None, 3)                 387

=================================================================
Total params: 2,813,187
Trainable params: 2,813,187
Non-trainable params: 0
_________________________________________________________________
```

(b)

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 33, 128)           2713984

 conv1d (Conv1D)             (None, 31, 64)            24640

 global_max_pooling1d (Globa  (None, 64)               0
 lMaxPooling1D)

 dense_2 (Dense)             (None, 3)                 195

=================================================================
Total params: 2,738,819
Trainable params: 2,738,819
Non-trainable params: 0
_________________________________________________________________
```

(c)

Figure 4: Performance of Sentimental Analysis (a) Tokenization (b) Vectorization (c) BiLSTM

In Figure 3 (a) – Figure (c) with the training accuracy begins around 0.83 at epoch 0 and rapidly increases, reaching approximately 0.95 by epoch 3. After that, the training accuracy remains relatively stable, maintaining values between 0.95 and 0.97 throughout the subsequent epochs. The testing accuracy starts at about 0.85 at epoch 0 and increases more gradually, peaking at around 0.92 by epoch 8. From there, the accuracy shows slight fluctuations but remains near 0.92 by the end of the 10 epochs. The model shows a clear overfitting tendency, as the training accuracy is consistently higher than the testing accuracy. The training accuracy approaches 0.95 by epoch 3, while the testing accuracy only reaches 0.92. The gap between training and testing performance indicates that the model is better fitted to the training data than to the unseen test data.The training accuracy starts at about 0.80 in the first epoch and quickly rises, reaching around 0.97 by epoch 3. It remains stable between 0.95 and 0.97 after that, showing only small fluctuations as the epochs progress.The testing accuracy begins at approximately 0.85 and rises more slowly, reaching around 0.90 by epoch 8. By epoch 10, it fluctuates around 0.89-0.90, showing a slower and less consistent increase compared to the training accuracy.

The training accuracy reaches up to 0.97 by epoch 3, while the testing accuracy peaks at

around 0.90. The model's performance on the testing set does not keep pace with the training accuracy, indicating that the model might not generalize well to new, unseen data. In figure 4(a) – Figure (c) the training accuracy starts at about 0.85 in the first epoch, and then it increases sharply, reaching almost 1.0 (or 100%) by epoch 5. After epoch 5, the accuracy stabilizes at 1.0 for the remaining epochs.The testing accuracy starts at approximately 0.88 and increases slowly, reaching around 0.90 by epoch 5. After that, it remains stable around 0.90, with minor fluctuations up to epoch 10.The model achieves near-perfect training accuracy, reaching 1.0 by epoch 5, but the testing accuracy stagnates at around 0.90. This suggests that the model is overfitting, as it performs excellently on the training data but does not show similar improvements on the testing data.

6.Conclusion

This study demonstrates the effectiveness of deep learning techniques, particularly Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM), for sentiment analysis of Twitter data. By applying these models to a diverse set of tweets, the research highlights the importance of text preprocessing steps such as tokenization, stop word removal, and lemmatization, which enhance the quality of input data for analysis. The results indicate that while the models exhibit strong training performance, they also show some overfitting, as evidenced by the higher accuracy on the training data compared to the test data. The use of word embeddings and feature extraction methods like Bag of Words further enriches the models' ability to capture the semantic meaning of the text. The analysis provides valuable insights into the sentiment dynamics surrounding various topics on Twitter, making it a useful tool for understanding public opinion. However, the overfitting issue points to the need for further optimization techniques to improve the model's ability to generalize to unseen data, ensuring that it can effectively predict sentiment on a broader scale.

Acknowledgment: Not Applicable.
Funding Statement*:* The author(s) received no specific funding for this study.
Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] K.L.Tan, C.P. Lee and K.M. Lim, “A survey of sentiment analysis: Approaches, datasets, and future research,” *Applied Sciences*, vol.13, no.7, pp.4550, 2023.
[2] M. I. Al-mashhadani, K.M. Hussein and E. T. Khudir, “Sentiment analysis using optimized feature sets in different facebook/twitter dataset domains using big data,” *Iraqi Journal For Computer Science and Mathematics*, vol.3, no.1, pp.64-70, 2022.
[3] T. Alqurashi, “Arabic sentiment analysis for twitter data: a systematic literature review,” *Engineering, Technology and Applied Science Research*, vol.13, no.2, pp.10292-10300, 2023.
[4] Y. Guo, S. Das, S. Lakamana and A. Sarker, “An aspect-level sentiment analysis dataset for therapies on Twitter,” *Data in Brief*, vol.50, pp.109618, 2023.
[5] S.H. Muhammad, I. Abdulmumin, A.A. Ayele, N. Ousidhoum, D.I. Adelani *et al.*, “Afrisenti: A twitter sentiment analysis benchmark for african languages,” arXiv preprint arXiv:2302.08956, 2023.
[6] W. Aljedaani, F. Rustam, M.W. Mkaouer, A. Ghallab, V. Rupapara et al., “Sentiment analysis on Twitter data integrating TextBlob and deep learning models: The case of US airline industry,” *Knowledge-Based Systems*, vol.255, pp.109780, 2022.

[7] Y. Wang, J. Guo, C. Yuan and B. Li, "Sentiment analysis of Twitter data," *Applied Sciences*, vol.12, no.22, pp.11775, 2022.
[8] Z. B. Nezhad and M.A. Deihimi, "Twitter sentiment analysis from Iran about COVID 19 vaccine," *Diabetes and Metabolic Syndrome: Clinical Research and Reviews*, vol.16, no.1, pp.102367, 2023.
[9] S. Singh, K. Kumar and B. Kumar, "Sentiment analysis of Twitter data using TF-IDF and machine learning techniques," *In 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)*, vol. 1, pp. 252-255, 2023.
[10] A.P. Rodrigues and N.N. Chiplunkar, "A new big data approach for topic classification and sentiment analysis of Twitter data," *Evolutionary Intelligence*, pp.1-11, 2022.
[11] Y. Pei, A. Mbakwe, A. Gupta, S. Alamir, H. Lin, X. Liu et al., "Tweetfinsent: A dataset of stock sentiments on twitter," *In Proceedings of the Fourth Workshop on Financial Technology and Natural Language Processing (FinNLP)*, pp. 37-47, 2022.
[12] P. Gaur, S. Vashistha and P. Jha, "Twitter sentiment analysis using naive bayes-based machine learning technique," *In Sentiment Analysis and Deep Learning: Proceedings of ICSADL 2022*, pp. 367-376, 2023.
[13] N. Parveen, P. Chakrabarti, B.T. Hung and A. Shaik, "Twitter sentiment analysis using hybrid gated attention recurrent network," *Journal of Big Data*, vol.10, no.1, pp.50, 2023.
[14] S. Mann, J. Arora, M. Bhatia, R. Sharma and R. Taragi, "Twitter sentiment analysis using enhanced bert," *In Intelligent Systems and Applications: Select Proceedings of ICISA 2022*, pp. 263-271, 2023.
[15] A. Yavari, H. Hassanpour, B. Rahimpour Cami and M. Mahdavi, "Election prediction based on sentiment analysis using twitter data," *International Journal of engineering*, vol.35, no.2, pp.372-379, 2022.
[16] A. Simarmata, A. Xu and M.E. Phanie, "Sentiment analysis on twitter posts about the Russia and Ukraine war with long short-term memory," *Sinkron: jurnal dan penelitian teknik informatika*, vol.7, no.2, pp.789-797, 2024.
[17] M. Bibi, W.A. Abbasi, W. Aziz, S. Khalil, M. Uddin et al., "A novel unsupervised ensemble framework using concept-based linguistic methods and machine learning for twitter sentiment analysis," *Pattern Recognition Letters*, vol.158, pp.80-86, 2022.
[18] M. H. Ali Al-Abyadh, M. A. Iesa, H. A. Hafeez Abdel Azeem, D. P. Singh, P. Kumar et al., "Deep sentiment analysis of twitter data using a hybrid ghost convolution neural network Model," *Computational Intelligence and Neuroscience*, vol.2022, no.1, pp.6595799, 2022.