International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.2 (2025) pp. 3196-3206
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Co-Bi-LSTM: Opinion Mining on Twitter Data Using Convolutional Neural Network with Optimized Bidirectional LSTM Model

Bikshapathy Peruka[1]*, K. Shahu Chatrapati[2]

[1] Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, Kukatpally, India.
* **Corresponding Author Email:** bpp.mec.research@gmail.com - **ORCID:** 0009-0000-8304-0865

[2] Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, Kukatpally, India.
Email: shahujntu@gmail.com- **ORCID :** 0009-0006-8037-8389

Article Info:

DOI: 10.22399/ijcesen.2452
Received : 20 January 2025
Accepted : 13 May 2025

Keywords :

Sentiment Analysis
Twitter Data
Sarcasm Detection
Convolutional Optimized Bidirectional Long Short-Term Memory and Self-Improved

Abstract:

The proliferation of social networking platforms has generated a substantial volume of user-generated content, posing significant challenges for text classification due to its diverse nature. Sentiment analysis or opinion mining, is crucial for extracting insights from user opinions and emotions regarding various entities and events. This research classifies tweets into positive and negative sentiments using Twitter data for predictive analysis in domains such as consumer behaviour and election outcomes. Two Kaggle datasets like Sentiment140 and News Headlines Dataset for Sarcasm Detection are used. The pre-processing phase includes cleaning, tokenization, and padding. Word embedding like skip-gram can capture semantic relationships and are used in neural network architectures for word2vec conversion. This paper proposes a hybrid model called Convolutional Optimized Bidirectional LSTM (CO-Bi-LSTM), combining Convolutional Neural Networks (CNN) with an Optimized Bidirectional Long Short-Term Memory (O-Bi-LSTM) network, enhanced by the Hybrid Hippopotamus based Zebra Optimization Algorithm (HH-ZOA). The model's performance is evaluated using metrics such as accuracy, F-measure, and precision, demonstrating its efficacy in sentiment analysis of Twitter data.

1. Introduction

Nowadays, the internet enables people to express their views through weblogs and tweets. It has become convenient to evaluate a service, a product, or even a celebrity and write a review on them. As a result, opinion mining categorization has emerged as an important research topic for automating the analysis of such large amounts of comments and blogs. Fine-grained opinion mining may identify if an opinion is extremely positive, positive, very negative, negative, or neutral [1,2]. Several researches have been completed for opinion mining based on other languages such as Arabic, French, Spanish, and Italian; nonetheless, a few works are focused in the English language.

Traditional sentiment analysis research is separated into two approaches: sentiment lexicon and machine learning. Sentiment lexicon-based methods often necessitate the human creation of unique sentiment lexicons for each field [3,4]. Sentiment analysis quality is highly connected to emotional lexicon quality and coverage. The creation and maintenance of a sentiment lexicon also demand a significant amount of manpower. With the introduction of new words on the Internet, a feeling lexicon-based method is no longer appropriate. Machine learning-based methods rely on manual feature selection. Sentiment analysis findings might vary depending on the feature chosen. Model generalization is not without its difficulties.

Deep learning-based sentiment analysis has recently emerged as a popular research issue [5]. Deep learning methods are more expressive than typical machine learning approaches, and they do not require manual feature selection and building. They work well for sentiment analysis. RNNs are a type of deep learning algorithm that is geared for learning data sequences and is typically used for textual data categorization. However, when dealing with lengthy datasets, RNNs face the vanishing

gradient problem. LSTM neural networks were proposed as a solution to this challenge [6,7], and they have shown to be useful in a number of real-world applications. However, obtaining a sentiment is strongly dependent on the context of the research. This study makes use of two datasets from Kaggle: Twitter sentiment and emotion dataset. This study offers a novel hybridized model that combines the benefits of deep learning methods CNN-OBiLSTM model with word embedding techniques. The Zebra Optimization Algorithm (ZOA) is useful in improving Bi-LSTM performance. The performance of several word embedding approaches is examined to identify the optimal embedding for implementation in the proposed model. The main contributions of the paper are as follows,

- The performance of the ZOA is improved by introducing the defence strategy of the Hippopotamus Optimization algorithm.
- The basic parameters of the Bi-LSTM model like LSTM units & Recurrent Dropout are optimized using the HH-ZOA optimization algorithm.
- Introduction of a novel hybrid model called as CO-Bi-LSTM, combining CNN with an O-Bi-LSTM network.

The remaining portion of this work is arranged in the following manner. The second section introduces similar works. Section 3 provides a comprehensive description of our strategy. Section 4 shows how our strategy and experiments were implemented, while Section 5 provides conclusions of the work.

2. Literature Review

In 2022, Setyanto, *et. al.,* [8] utilized the LSTM for opinion mining in Arabic. The paper analyzes two word embedding methods using an Arabic sentiment assessment dataset. Overall, fastText is an improved vector for encoding and comprehending emotion in Arabic texts. Sentences pre-processed with fastText outperformed those pre-processed with GloVe in terms of correctness. Second, stacked LSTM layers led to improved accuracy, but also slowed the pace of training and testing.

In 2022, Eshmawi, *et. al.,* [9] designed an An automated opinion mining approach based on an optimized Fuzzy Neural Network (FNN). The current study focuses on developing an automatic sentiment analysis model that integrates the Deer Hunting Optimization Algorithm (DHOA) with an FNN, known as the DHOA-FNN model. Along with those above, the suggested DHOA-FNN model includes two stages of the feature extraction method: glove and N-gram techniques. Furthermore, the FNN model is utilized as a classification model, and the GTOA is used for parameter optimization.

In 2022, Kokab, *et. al.,* [10] introduced a deep learning models based on transformers are used to analyse social media sentiment. This paper develops an efficient Convolution Bi-directional Recurrent Neural Network (CBRNN) model for analysing morphological and semantic knowledge, as well as emotional and historical context processing. Initially, the zero-shot categorization was employed to identify reviews by computing their polarity ratings. Next, a pre-trained BERT model captures sentence-level meanings and historical context from the input and creates embeddings. The contextualized embedding vectors were then sent to the neural network, which was made up of Bi-LSTM and dilated convolution.

In 2022, Ye, *et. al.,* [11] used Twitter comments, a stacking ensemble deep learning model was developed to forecast Bitcoin values. The study offers a unique ensemble deep-learning model for predicting Bitcoin's upcoming 30-minute pricing based on price data, technical signs, and sentiment indices. The model incorporates two types of neural networks, LSTM and gate recurrent unit (GRU), as well as a stacking ensemble approach, to increase decision accuracy. It is then analysed using linguistic statistical methods to create sentiment indices. Meanwhile, as a financial market forecasting model, the model also chooses the technical indicators as input.

In 2023, Bengesi, *et. al.,* [12] A machine learning-sentiment investigation of the monkeypox outbreak: a big dataset used to uncover the divisions of the general public in Twitter messages. The present research employed NLP techniques to change the datasets such that they were relevant to the purpose of the study. It used VADER and TextBlob to annotate the pre-processed data, which was subsequently vectorized with Count Vectorizer and TF-IDF. We used several machines learning methods, including K-Nearest Neighbour (KNN), Multilayer Perceptron (MLP), Support Vector Machine (SVM), Logistic Regression, Naïve Bayes, Random Forest, and XGBoost, to categorize sentiment as positive, negative, and neutral.

Sentiment classification in user-generated social media content remains difficult due to the data's diversity and dynamic nature. Traditional models frequently fail to capture the nuanced semantics of brief, informal messages, such as tweets. Previous research has found mixed efficacy in various word embeddings and neural network architectures, with fastText surpassing GloVe in Arabic sentiment analysis and BERT-based models effectively

capturing syntactic and semantic information. There is still a need for more robust and optimized methods. This study tries to address these issues by presenting a hybrid model for improving sentiment classification accuracy and efficiency when studying Twitter data.

3. Proposed Methodology

This study proposes a novel approach for sentiment analysis of Twitter data using a hybrid model termed CO-Bi-LSTM, optimized using the HH-ZOA optimization. Leveraging the Sentiment140 [13] and Sarcasm Detection [14] datasets, the methodology involves comprehensive data pre-processing to clean, tokenize, and pad tweets. After that, apply the skip-gram word embeddings to capture semantic relationships. The model architecture combines CNN for feature extraction with an O-Bi-LSTM network for contextual understanding. Figure 1 shows the block diagram for the suggested opinion mining model.

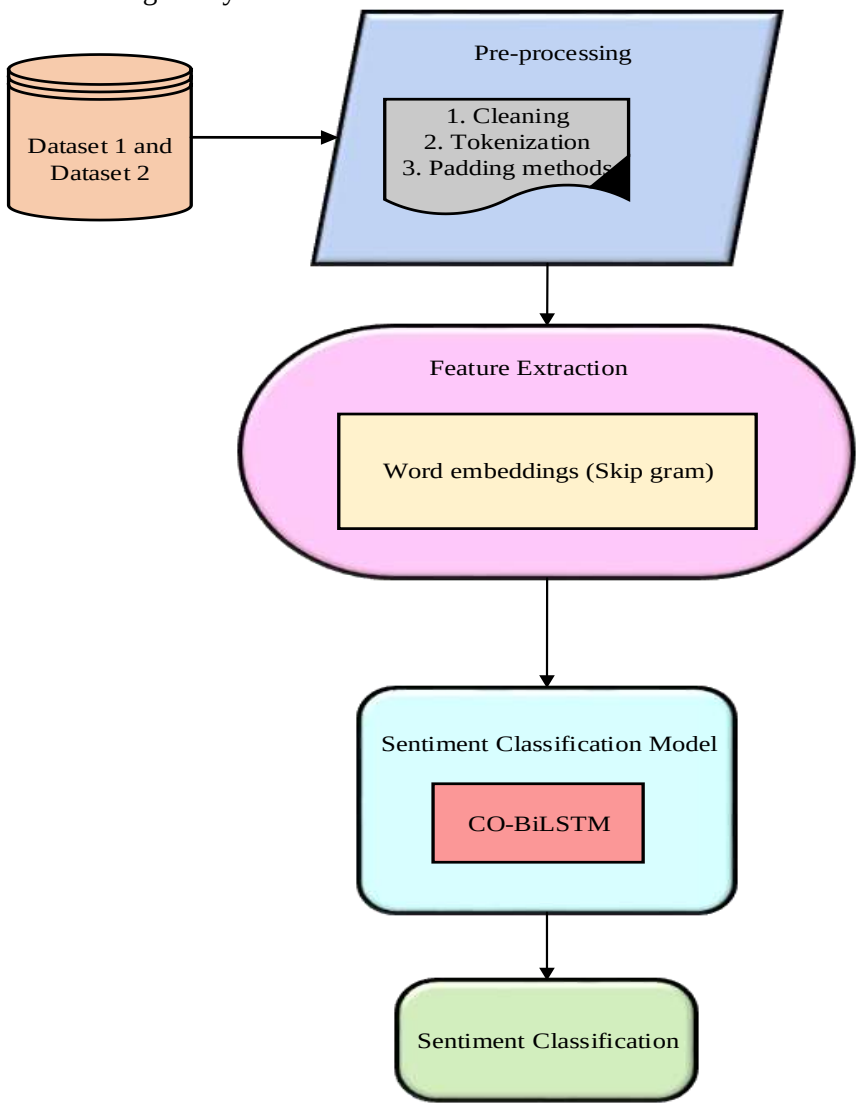


Figure 1. *Block diagram of the proposed sentiment analysis model*

3.1. Pre-processing

Employ NLP techniques like Cleaning, Tokenization, and Padding methods to prepare the data for analysis.

Text cleaning

To reduce bias and increase model performance, missing values must be discovered and managed effectively. Text cleaning involves eliminating numerals, punctuation, and special characters, as well as changing the text to lowercase, to maintain uniformity and readability. These pre-processing approaches turn unstructured input into an intelligible, organized format, enabling the NLP token classification model in the next layer.

Text tokenization

Tokenization is the initial stage in pre-processing text. To convert the text into tokens, do the following two steps: Divide the passage into sentences, and then into tickets for each sentence. In this study, the tokenization model divides sections into sentences using the delimiters full stop (-), question mark (?), and exclamation mark (!). To separate sentences into tokens, white spaces (), commas (,), and semicolons (;) are used.

Padding

There are three popular padding algorithms: fixed, dynamic, and uniform.

i) Fixed Padding

The input text $X = \{x_1, x_2, \ldots, x_n\}$ is tokenized and inserted with two special tokens [CLS] and [SEP], which mark the start of the sequence and separate it from another series, respectively. Next, these tokens are mapped to an index matrix $D = d_1, d_2, \ldots, d_n$, with unique token IDs assigned to each x_i. The attention masks are then created by putting [PAD] tokens (usually with the value 0) at the end of each d_i vector. The function returns a token ID for indexing tokens from input X, an attention mask, and a list of labels for mapping token ID to the input ground truth label.

ii) Dynamic Padding

Unlike Fixed Padding, which generates a collection of all d_i with the same maximum length, Dynamic Padding estimates the number of [PAD] that must be added in the same batch. This approach ensures that every di in each batch have the same length but are distinct between batches. This strategy dramatically decreases the amount of [PAD] tokens that must be inserted.

iii) Uniform Padding

Uniform Padding is an improvement over Dynamic Padding. The approach sorts the d_i in increasing order of size. Next, clusters of di are formed according on the number of batches. Finally, the [PAD] tokens are inserted into the di to ensure that all di in the same batch are of equal size.

3.2. Word embeddings using Skip gram

To extract the word vectors, the pre-processed data is sent into the skip-gram framework. The skip-gram structure is composed of a basic three-layer neural network, which contains an input layer, hidden layer, and output layer.

The skip-gram approach is used to acquire the word vector illustration of concepts by increasing the mean logarithmic conditional probability q_i for every idea to be taught W_i in the corpus, as expressed as follows:

$$q_i = \frac{1}{m}\sum_{i=1}^{m}\sum_{-h\le k\le h, k\neq 0} \lg q(W_{i+k}|W_i) \quad (1)$$

Where W_{i-k} and W_{i+k} are the first and last ideas of W_i, respectively, h is the size of the training text window, and m is the total number of concepts in the training sentence. The SoftMax function defines $q(W_{i+k}|W_i)$ in the following way:

$$q(W_{i+k}|W_i) = \frac{\exp\left(u_{w_{i+k}}^{m} u_{w_i}\right)}{\sum_{W=1}^{n} \exp(u_w^m u_{w_i})} \quad (2)$$

Where u_w^m represents the transpose of each concept vector in the corresponding table, and n indicates the total number of concepts. For basic corpus training, the skip-gram model generates concept vectors for each subject in the corpus.

Word2Vec Representation using Skip-Gram

The skip-gram model selects a word vector from the wider context of the desired word (w) as its underlying description. Words are combined with their surroundings to make a word, which may be expressed as follows:

$$c_l(w_i) = \phi\left(w^{(l)} c_l(w_{i-1}) + w^{(sl)} e(w_{i-1})\right) \quad (3)$$

Where w_i is indicated above as $c_l(w_i)$,

$$c_r(w_i) = \phi\left(w^{(r)} c_r(w_{i+1}) + w^{(sr)} e(w_{i+1})\right) \quad (4)$$

Where $e(w_{i+1})$ is the word vector for the word w_{i-1}. The $w^{(l)}$ matrix moves the previous word's hidden layer to the following word's hidden layer. The activation function φ, is nonlinear. In this study, the word wi's representation x_i is specified by the formula below. The preceding expression is $c_l(w_i)$, the next one is $c_r(w_i)$, and the word vector splicing is $e(w_i)$.

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)] \quad (5)$$

3.3. Sentiment Classification using CO-Bi-LSTM model

The proposed approach recommends a hybrid model termed CO-Bi-LSTM, which combines CNN with an O-Bi-LSTM network. CNNs detect local patterns in text, but Bi-LSTMs process data bidirectionally to grasp context. The HH-ZOA improves the model's performance by fine-tuning hyperparameters to ensure quick training and robust performance. This combination allows the CO-Bi-LSTM model to achieve high accuracy in sentiment analysis of Twitter data while efficiently dealing with the complexity of user-generated information. The architecture of the CO-Bi-LSTM model is shown in Figure 2.

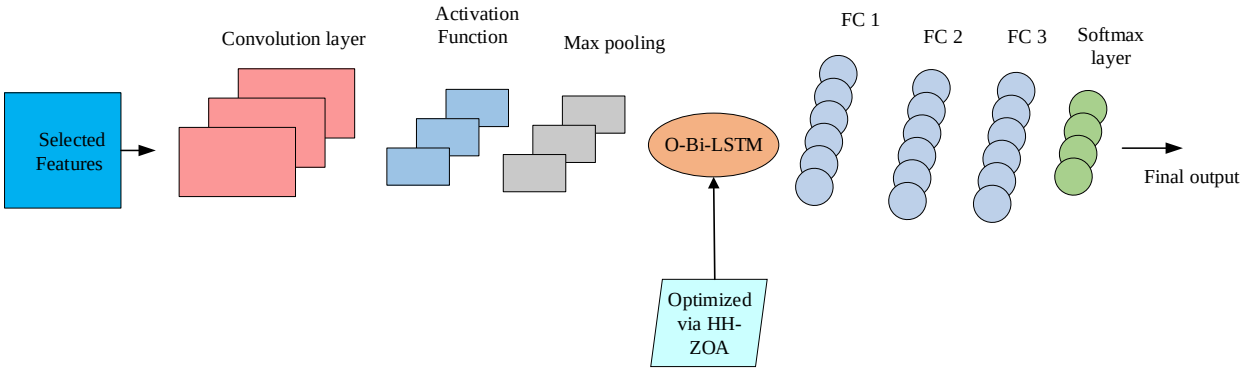


Figure 2. *Layered architecture of the CO-Bi-LSTM model*

3.3.1. CNN

A CNN is a deep learning system that can-do complex tasks with text, pictures, audio, videos, and other types of input. With visual inputs, this multi-layer neural network can perform tasks such as segmentation, object detection, and picture categorization. It can discriminate between various things in the data by using programmed weights and biases. CNN generates neurons with learnable weights and biases.

Convolutional Layer

CNNs often receive input in the form of tensors, whose shape is dictated by the volume, diversity, height, and depth of data. The width, height, and number of channels construct a feature map, which is generated from the input after it has gone through a convolutional layer. Each convolutional layer in a neural network is defined by certain properties, such as the height and width of the convolutional kernels (hyper-parameters provided during network construction). Furthermore, the layer defines the number of input and output channels, which are two crucial hyperparameters that influence the network's architecture and ability to extract features effectively. Because the convolution filter represents the input channels, the total number of channels in the input feature map must equal the depth of the convolution filter to ensure compatibility and proper convolution operations inside the network structure. The output of a convolutional layer's input is sent to the following layer. The i^{th} filter kernel's weights and bias are represented by K_i^l in Layer L, whereas the j^{th} local area is represented by $x^{l(r^j)}$. Consequently, Eq. (7) that follows describes the convolution process.

$$y^{l(i,j)} = K_i^l * x^{l(r^j)} = \sum_{j=0}^{W} K_i^l(j') * x^{l(j+j')} \quad (7)$$

The notations $K_i^l(j')$ and * denote the j'^{th} weights in frame I of layer $l + 1$, respectively, while W stands for the kernel's width. The symbol * represents the dot product of the local regions and the kernel.

Activation Layer

The activation function used to each convolutional layer's output increases positive vectors while suppresses negative vectors from the layer before it. In the article, we utilize an activation function known as Leaky ReLU. When compared to other activations, ReLU can improve feature sparsity while decreasing the likelihood of vanishing gradients. Eq. (8) gives the formula for Leaky ReLU,

$$LeakyReLU(x) = \begin{cases} x, & x \geq 0 \\ ax, & x < 0 \end{cases} \quad (8)$$

When 'a' has a value of 0.2, it indicates the hyper-parameters.

Max Pooling Layer

Convolutional networks can be enhanced with local or global pooling layers to improve fundamental computing efficiency. Pooling layers minimize data dimensionality by integrating neuron group responses from the first layer into a single neuron in the succeeding layer. The local pooling strategy combines tiny clusters, generally 2 × 2. global pooling impacts all convolutional layer neurons. Pooling can also be used to calculate maximums or averages. When max pooling is enabled, each neural cluster in the layer above contributes its maximum value. A pooling layer is frequently used instead of a convolutional layer because of its ability to reduce feature spatial size and expedite learning. Although there are other types of pooling processes, max-pooling is the most often used. To generate location-invariant features, the max-pooling layer applies a local max algorithm to the input features dependent on the kernel/pool size. This explains the improvement in maximum pooling:

$$P^{l(i,j)} = \max_{(j-1)W+1 \leq t \leq jW} \{a^{l(i,t)}\} \quad (9)$$

Bi-LSTM

The LSTM classifier is composed of four fundamental components: the memory unit, the forget gate, the input gate, and the output gate. The memory cell may store data for a long or short period of time. The input gate of an LSTM cell regulates the quantity of data input, while the forget gate regulates the retention of data. The data in the LSTM layer cell can be modified to calculate and generate the output activation for the Output Gate. Back-propagation through time (BPTT) is used to prepare extended successions, which results in exploding/vanishing gradients that are difficult to generate with regular RNNs. To address this, a gated cell is employed as a Bi-LSTM cell rather than an RNN cell. Three entry points are used to enter data into a cell case. As illustrated in the equation below, a sigmoid layer determines whether data should be eliminated from the cell state via the first entry.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \quad (10)$$

As shown in the following equations, the next gate is an input entry path that incorporates a sigmoid layer for selecting the variables that need to be updated and a tanh layer to generate a vector with the new revised values.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (11)$$

$$\underline{C_t} = \tanh(W_c.[h_{t-1}, x_t] + b_c) \quad (12)$$

Eq. (10)– Eq. (12) are then used to update the state of the cell.

$$C_t = f_t.C_{t-1} + i_t.\underline{C_t} \quad (13)$$

The modified cell state and a sigmoid activation function that selects the components of the cell state will be used to decide the output of the current state, which will be the final output displayed.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \quad (14)$$

$$h_t = o_t * \tanh(C_t) \quad (15)$$

In this case, the sigmoid activation function is symbolised by σ. The symbol for the tangent activation function is tanh. The weight matrices are denoted by W. The input vector is x_t. The former hidden state is represented by h_{t-1}. Bias exists in b_f, b_i, b_c, and b_o.

This study makes use of the Bi-LSTM that was constructed. The input sequence is fed into two distinct deep learning networks, one that works in reverse chronological order and the other in standard temporal order. The stacked layer Bi-LSTM structure collects both background and forward sequence data at every time step, leading to high classification accuracy. Eq. (16) through Eq. (18) demonstrate how the bi-LSTM classifier processes data backwards.

$$\vec{h}_t = f(w_1 x_t + w_2 \vec{h}_{t-1}) \quad (16)$$

$$\overleftarrow{h}_t = f(w_3 x_t + w_5 \vec{h}_{t+1}) \quad (17)$$

$$O_t = g(w_4 \vec{h}_t + w_6 \overleftarrow{h}_t) \quad (18)$$

HH-ZOA Algorithm

The defence strategy of the HHO is incorporated within the defence property of the ZOA optimization to improve the performance of the Zebra.

- **Initialization**

HH-ZOA is a population-based optimizer since it includes zebras. The plain where the zebras are placed is the problem's search space, and each zebra represents a potential mathematical solution to the problem.

The positions of each zebra (search agent) inside the search region define the selection parameters' values. As a consequence, each zebra may be modelled as an HH-ZOA member using a vector, with the vector's elements representing the values of the problem variables. A matrix can be used to mathematically represent the zebra population. The zebras' initial positions inside the search area are picked at random. Eq. (19) gives the HH-ZOA population matrix.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,j} & \cdots & x_{N,M} \end{bmatrix}_{N\times m} \quad (19)$$

Where X represents the zebra population, X_i is the i^{th} zebra, m is the number of option variables, N is the number of population members (zebras), and $x_{i,j}$ is the value for the j^{th} problem variable that the i^{th} zebra provided. Each zebra symbolizes a possible solution to the optimization issue. As a consequence, the objective function may be evaluated using each search agent's suggested values for the issue variables. The values acquired for the objective function are expressed as a vector in Eq. (20).

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times 1} \quad (20)$$

Where F_i is the fitness function value attained for the i^{th} search agent and F is the vector of objective function values. The values of the objective function are compared to determine which candidate solution is the best match for the given problem. This procedure successfully examines the quality of possible solutions related to the given challenge. The members of the HH-ZOA population are therefore updated at two discrete intervals during each repetition.

Phase 1: Foraging behaviour

During the initial phase of foraging, zebra behaviour models are used to update population members. Within HH-ZOA, the most remarkable individual in the population is known as the pioneer zebra, and it guides the other members of the population to its allotted place within the search space. As a consequence, Eq. (21) and Eq. (22) may be utilized to quantitatively model how zebras' locations vary during the foraging time.

$$x_{i,j}^{new,P1} = x_{i,j} + r.(PZ_j - I.x_{i,j}) \quad (21)$$

$$X_i = \begin{cases} X_i^{new,P1} \\ X_i, \ else \end{cases}, F_i^{new,P1} < F_i \quad (22)$$

Where PZ is the top member and the pioneering zebra; PZ_j is its j^{th} dimension; r is a random number in the interval [0, 1]; $I = round\,(1 + rand)$, where rand is a random value in the interval [0, 1]; and $x_{i,j}^{new,P1}$ is its j^{th} dimension value. Based on the first phase, this represents the new status of the i^{th} zebra. As a result, $I \in \{1, 2\}$ and if parameter $I = 2$, then the population movement is vary significantly more.

Phase 2: Hippopotamus based defence strategies against predators

The second step involves simulating zebra defense systems against predator assaults in order to update the position of HH-ZOA individuals in the search region. Zebras have a variety of defense techniques based on the type of predator they encounter. Zebras defend themselves against lion assaults by sprinting in a zigzag pattern and occasionally turning sideways.

It is expected in the HH-ZOA design that one of the following two scenarios is occur with an equal chance:

- The zebra decides on an escape route after the lion assaults it.
- When the zebra is attacked by another predator, it is decided how to go on the offensive.

The first tactic involves the zebras attacking the lions and running away from the attack in the area around their current location. The mode S_1 in Eq. (23) can be used to mathematically represent this tactic. This stratergy is replaced by the defence strategy of the Hippopotamus algorithm. As a predator attacks a zebra, the herd's closer members go in that direction as part of the second technique, which aims to confuse and frighten the attacker by forming a protective structure. In mathematics, the mode S_2 in Eq. (23) represents the zebras' approach. If the objective function has a higher value at the new site after zebras are relocated, then they accept the change. Using Eq. (24), this update condition is modelled.

$$x_{i,j}^{new,P2} = \begin{cases} S_1: \overrightarrow{RL} \oplus Predator_j + \left(\frac{f}{\sigma d \times \cos(2\pi g)}\right).\left(\frac{1}{\vec{D}}\right), & P_s \le 0.5 \\ S_2: x_{i,j} + r.\left(AZ_j - I.x_{ij}\right), & else \end{cases} \quad (23)$$

$$X_i = \begin{cases} X_i^{new,P2} \\ X_i, \ else \end{cases}, F_i^{new,P2} < F_i \quad (24)$$

Where $\overrightarrow{RL}$ is a Levy-distributed random vector that is used when a predator suddenly shifts positions during an attack. Whereas AZ_j represents its j^{th} dimension value. The objective function value is $F_i^{new,P2}$ and its j^{th} dimension value is $x_{i,j}^{new,P2}$. One uniform random value f, is between 2 and 4 in Eq. (24). A uniform random integer between 2 and 3 is represented by $\vec{D}$. An even random number between -1 and 1 is represented by the symbol g and $r_{i,j}$ is a random vector.

$$\vec{D} = \left|Predator_j - x_{i,j}\right| \quad (25)$$

Eq. (25) shows how far away the predator is from the i^{th} zebra.

Dropout Layer

The dropout regularization methodology provides an approximate way for training many neural networks with different architectures at the same time. During the training process, specific layer outputs are "dropped out," or arbitrarily ignored. As a result, the layer will act and seem differently from one with a different number of nodes and a weaker link to the previous layer. In other words, for every layer change during training, a new "view" of the current layer is used. Dropout creates noise to the training process, therefore nodes in a layer are

required to bear a probabilistic degree of responsibility for the inputs.

Fully Connected Layer

Neural networks get input from a fixed number of neurons in the layer above each one. When the layer above it is fully connected, every neuron may receive input from every other neuron. Only a tiny subset of the neurons in the previous layer provide data to the convolutional layer. The subarea is frequently square in form. The receptive field is the area of a neuron that receives information. As a result, the layer in front of a completely connected layer serves as its receptive field. The total receptive area of a convolutional layer is larger than that of prior layers. A fully connected layer anticipates accurate feature extraction by adding weights to the input supplied by the feature analysis.

Softmax

The softmax function is an extension of a multidimensional logistic function. It is widely employed as the very last activation function of a neural network in multinomial logistic regression, modifying the network's output to a probability distribution over the predicted output class. A multi-class classification neural network's final linear layer logit produces numerical results. Softmax normalizes each number and converts it into probabilities by summing the exponents of each output. This ensures that the full output vector and the probability sum are identical. Cross-entropy loss is typically used for this type of multi-class classification problem. The output has three labels like positive, negative and neutral.

4. Result And Discussion

In this part, the proposed model's outcomes are compared to current approaches including LSTM, FNN, CBRNN, and GRU. Python is used for implementation. Table 1 shows the metrics for evaluating the various machine learning approaches used for opinion mining on Sentiment data. Table 2 compare the values for Sarcasm data. Each approach is evaluated using a variety of metrics, including negative predictive value (NPV), precision, recall, specificity, sensitivity, false positive rate (FPR), Matthews correlation coefficient, F-measure, accuracy, and false negative rate (FNR). The suggested technique outperforms competitors across most parameters and demonstrates higher efficacy in reliably identifying opinions within the dataset.

Table 1: *Overall comparison of the proposed opinion mining on sentiment data*

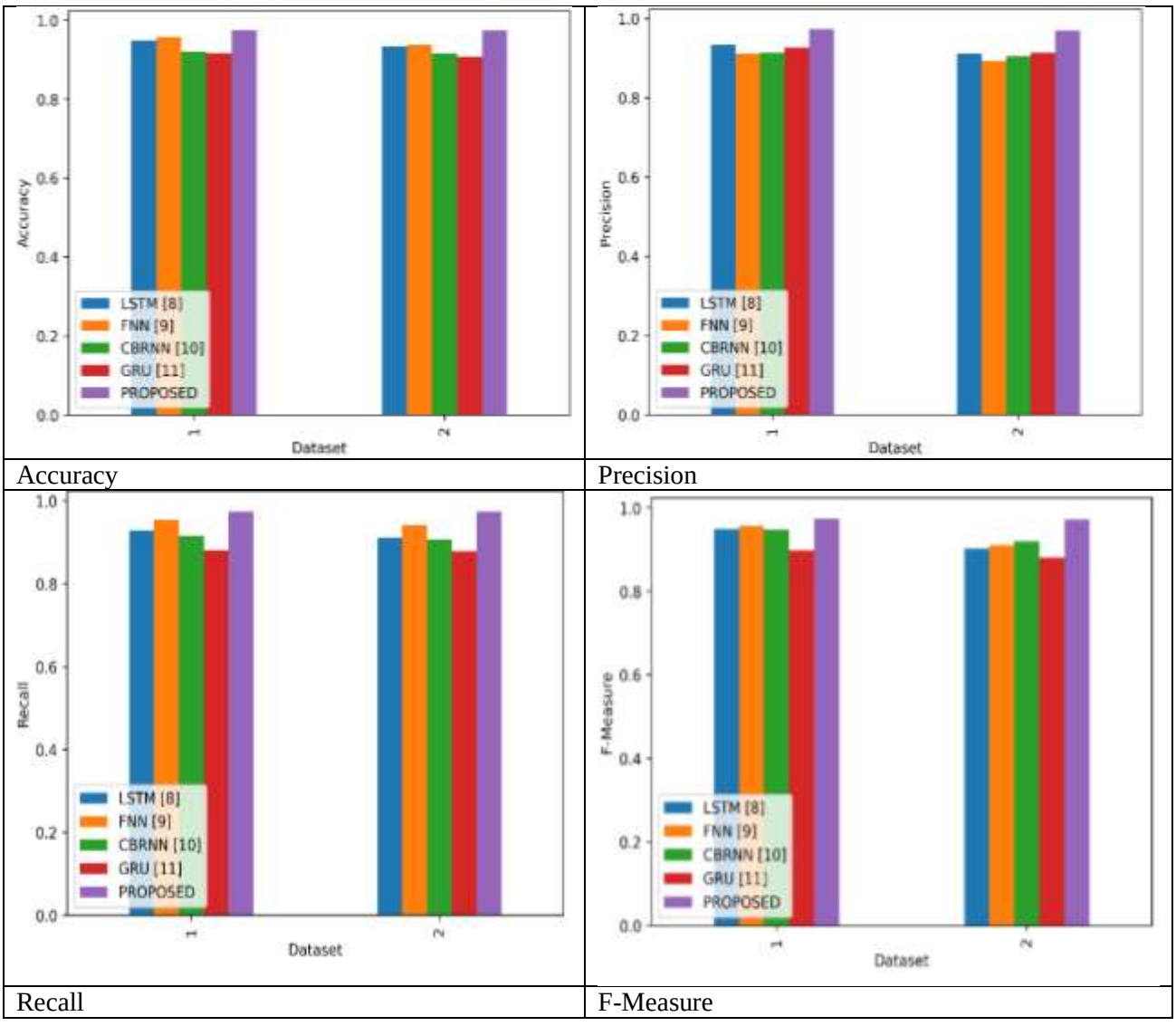
Metrics	LSTM [8]	FNN [9]	CBRNN [10]	GRU [11]	PROPOSED
Accuracy	0.9486	0.9580	0.9217	0.9174	0.9748
Precision	0.9333	0.9123	0.9136	0.9269	0.9745
Sensitivity	0.9181	0.9016	0.9061	0.8899	0.9751
Specificity	0.9672	0.9658	0.9650	0.9529	0.9745
Recall	0.9289	0.9545	0.9158	0.8818	0.9751
F-Measure	0.9492	0.9571	0.9477	0.8981	0.9748
MCC	0.9371	0.9243	0.9051	0.8909	0.9630
NPV	0.9388	0.9404	0.9293	0.9186	0.9752
FPR	0.0459	0.0218	0.0328	0.0453	0.0195
FNR	0.0424	0.0601	0.0665	0.0833	0.0249

Table 1 provides a comprehensive comparison of sentiment analysis models, including LSTM, FNN, CBRNN, GRU, and a proposed model, across multiple evaluation metrics. The proposed model stands out with the highest accuracy of 97.48%, demonstrating its superior ability to correctly classify sentiment compared to LSTM (94.86%), FNN (95.80%), CBRNN (92.17%), and GRU (91.74%). It also achieves high precision (97.45%), sensitivity (97.51%), specificity (97.45%), recall (97.51%), and F-measure (97.48%), indicating robust performance in identifying both positive and negative sentiments accurately. Additionally, the model shows strong Matthews Correlation Coefficient (MCC) of 96.30% and Negative Predictive Value (NPV) of 97.52%, alongside low False Positive Rate (FPR) of 1.95% and False Negative Rate (FNR) of 2.49%, highlighting its reliability in distinguishing true sentiment categories. Table 2 compares the performance of sentiment analysis models, including LSTM, FNN, CBRNN, GRU, and a proposed model, specifically in sarcasm detection. The proposed model achieves superior results across various metrics, notably recording the highest accuracy at 97.40% compared to LSTM (93.31%), FNN (93.78%), CBRNN (91.64%), and GRU (90.74%). It also demonstrates

Table 2: *Overall comparison of the proposed opinion mining on sarcasm data*

Metrics	LSTM [8]	FNN [9]	CBRNN [10]	GRU [11]	PROPOSED
Accuracy	0.9331	0.9378	0.9164	0.9074	0.9740
Precision	0.9121	0.8933	0.9048	0.9132	0.9704
Sensitivity	0.8807	0.9016	0.8996	0.8705	0.9753
Specificity	0.9472	0.9389	0.9408	0.9492	0.9728
Recall	0.9119	0.9427	0.9083	0.8790	0.9753
F-Measure	0.9023	0.9105	0.9207	0.8811	0.9729
MCC	0.9127	0.9133	0.8811	0.8727	0.9480
NPV	0.9208	0.9316	0.9104	0.9014	0.9774
FPR	0.0582	0.0618	0.0619	0.0653	0.0272
FNR	0.0516	0.0501	0.0640	0.0719	0.0247

strong precision (97.04%), sensitivity (97.53%), specificity (97.28%), recall (97.53%), and F-measure (97.29%), indicating its robust ability to accurately identify sarcasm in text. Additionally, the model shows high MCC of 94.80% and NPV of 97.74%, alongside low FPR of 2.72% and FNR of 2.47%. Figure 3 displays the comparison graphs for the performance measures that were covered.



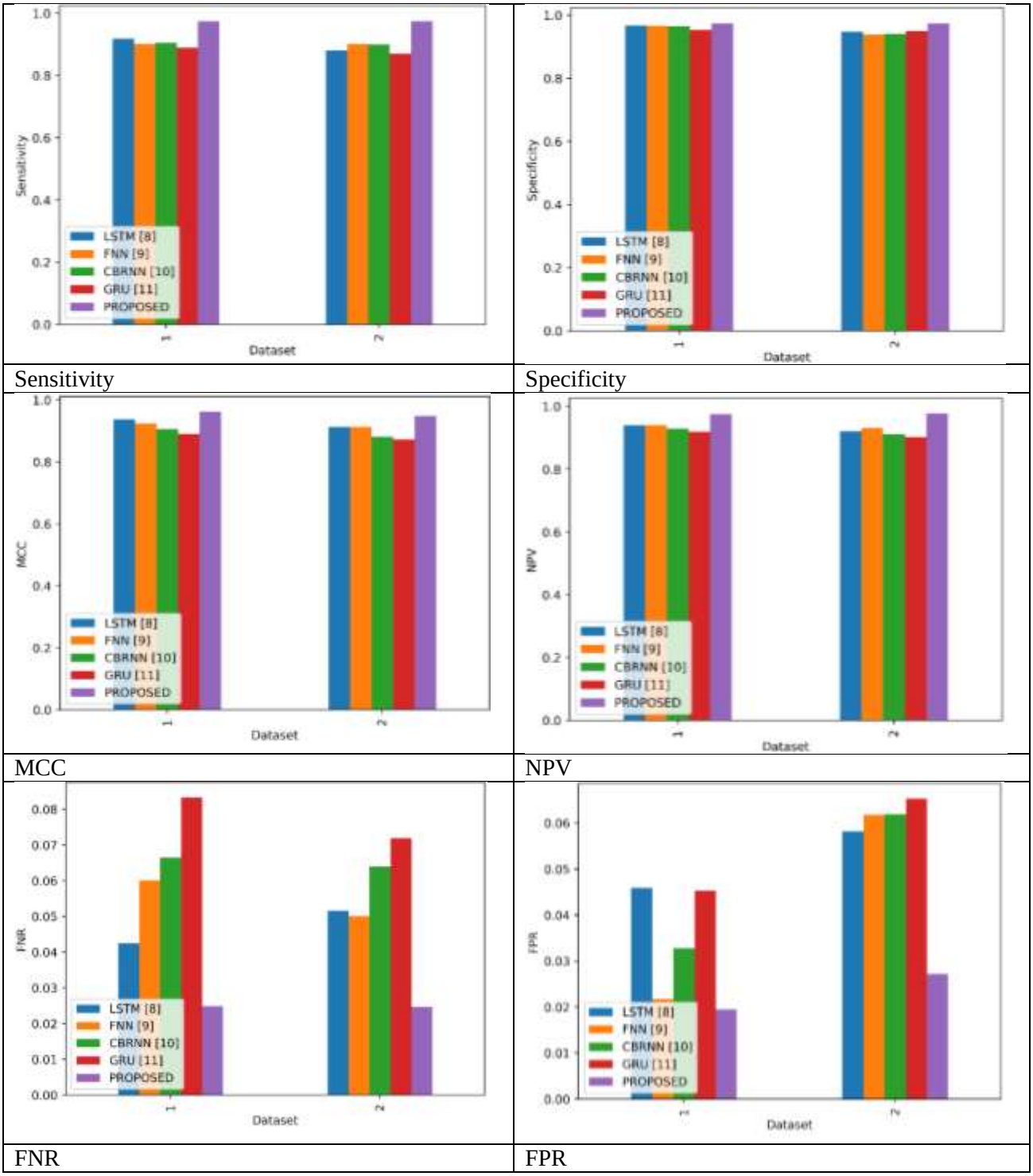


Figure 3: *Comparison of the proposed and existing methods for opinion mining*

5.Conclusion

In conclusion, the study introduces a novel CO-Bi-LSTM model augmented with the HH-ZOA for sentiment analysis of Twitter data. Through meticulous data pre-processing involving noise removal, tokenization, and padding, followed by skip-gram word embedding, the model effectively captures semantic relationships in tweets. The architecture combines CNN for local feature extraction and an O-Bi-LSTM for capturing long-range dependencies in sequential data. HH-ZOA optimizes key hyperparameters such as learning rates and batch sizes, enhancing model performance. Evaluation metrics including accuracy, F-measure, and precision validate the model's efficacy in sentiment classification tasks, demonstrating superior results compared to conventional approaches. This research underscores the model's applicability in extracting nuanced sentiment insights from social media data, critical for domains such as consumer behaviour analysis

and predictive analytics in electoral contexts. Future directions may explore advancements in optimization algorithms and the integration of additional data modalities to further elevate model performance and applicability in real-world scenarios.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Xiao, Y., Li, C., Thürer, M., Liu, Y., & Qu, T. (2022). Towards lean automation: Fine-grained sentiment analysis for customer value identification. *Computers & Industrial Engineering, 169*, 108186. https://doi.org/10.1016/j.cie.2022.108186

[2] Li, Y., Wang, F., & Zhong, S. H. (2023). A more fine-grained aspect–sentiment–opinion triplet extraction task. *Mathematics, 11*(14), 3165. https://doi.org/10.3390/math11143165

[3] Almosawi, M. M., & Mahmood, S. A. (2022). Lexicon-based approach for sentiment analysis to student feedback. *Webology, 19*(1). https://doi.org/10.14704/webology.v19i1.XXX *(DOI eksikse kaldırılabilir)*

[4] Cero, I., Luo, J., & Falligant, J. M. (2024). Lexicon-based sentiment analysis in behavioral research. *Perspectives on Behavior Science*, 1–28. https://doi.org/10.1007/s40614-024-00391-z *(Varsa DOI ile güncellenebilir)*

[5] Mendi, A. F. (2022). A sentiment analysis method based on a blockchain-supported long short-term memory deep network. *Sensors, 22*(12), 4419. https://doi.org/10.3390/s22124419

[6] Alwehaibi, A., Bikdash, M., Albogmi, M., & Roy, K. (2022). A study of the performance of embedding methods for Arabic short-text sentiment analysis using deep learning approaches. *Journal of King Saud University - Computer and Information Sciences, 34*(8), 6140–6149. https://doi.org/10.1016/j.jksuci.2021.03.003

[7] Alsayat, A. (2022). Improving sentiment analysis for social media applications using an ensemble deep learning language model. *Arabian Journal for Science and Engineering, 47*(2), 2499–2511. https://doi.org/10.1007/s13369-021-05888-6

[8] Setyanto, A., Laksito, A., Alarfaj, F., Alreshoodi, M., Oyong, I., Hayaty, M., Alomair, A., Almusallam, N., & Kurniasari, L. (2022). Arabic language opinion mining based on long short-term memory (LSTM). *Applied Sciences, 12*(9), 4140. https://doi.org/10.3390/app12094140

[9] Eshmawi, A. A., Alhumyani, H., Khalek, S. A., Saeed, R. A., Ragab, M., & Mansour, R. F. (2022). Design of automated opinion mining model using optimized fuzzy neural network. *Computers, Materials & Continua, 71*(2). https://doi.org/10.32604/cmc.2022.022402

[10] Kokab, S. T., Asghar, S., & Naz, S. (2022). Transformer-based deep learning models for the sentiment analysis of social media data. *Array, 14*, 100157. https://doi.org/10.1016/j.array.2022.100157

[11] Ye, Z., Wu, Y., Chen, H., Pan, Y., & Jiang, Q. (2022). A stacking ensemble deep learning model for Bitcoin price prediction using Twitter comments on Bitcoin. *Mathematics, 10*(8), 1307. https://doi.org/10.3390/math10081307

[12] Bengesi, S., Oladunni, T., Olusegun, R., & Audu, H. (2023). A machine learning-sentiment analysis on Monkeypox outbreak: An extensive dataset to show the polarity of public opinion from Twitter tweets. *IEEE Access, 11*, 11811–11826. https://doi.org/10.1109/ACCESS.2023.3268706

[13] Kaggle. (2024, May 19). *Sentiment140 dataset*. https://www.kaggle.com/datasets/kazanova/sentiment140

[14] Kaggle. (2024, May 19). *News headlines dataset for sarcasm detection*. https://www.kaggle.com/datasets/rmisra/news-headlines-dataset-for-sarcasm-detection