MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

UNRAVELING AMBIGUOUS SENTIMENT CONTEXTS: A NOVEL APPROACH USING GAUSSIAN DISTRIBUTION-BASED LONG SHORT-TERM MEMORY (GD-LSTM) FOR IMPROVED SENTIMENT ANALYSIS

R. Anitha[1], D *Vimal Kumar[2], P. Shanthakumar[3], S. Abirami[4], G. Shobana[5], P. Kavitha[6]*
[1]Assistant Professor, Department of Computer Science and Data science, Nehru Arts and Science College, Thirumalayam Palayam India, anitharavi67@gmail.com
[2]Associate Professor and Head, Department of Computer Science, Rathinam College of Arts and Science, Coimbatore, Tamil Nadu, India
[3]Professor, Department of Information Technology, Nehru Institute of Technology, Coimbatore, India
nitdrpshanthakumar@nehrucolleges.com
[4]Department of Computer Science, Sri Krishna College of Engineering and Technology, Tamil Nadu, India
[4]Assistant Professor, Department of Computer Science and Engineering, Sri Krishna college of engineering and technology, Coimbatore
[5]Associate Professor, Department of Artificial Intelligence and Data Science, Sri Eshwar College of Engineering, Coimbatore, India
shobanait@gmail.com
[6]Associate Professor & Head, Department of Computer Science with Cyber Security, Sri Ramakrishna College of Arts & Science. Coimbatore, India, kavitha.p@srcas.ac.in; apkavitha2001@gmail.com;

Abstract - *Sentiment analysis, a fundamental task in natural language processing, involves classifying text data into sentiment categories such as positive, negative, or neutral. However, conventional sentiment analysis methods encounter challenges in accurately capturing nuanced linguistic features, contextual dependencies, and ambiguous expressions, leading to suboptimal performance. To mitigate these challenges, this study introduces a novel approach utilizing Gaussian Distribution-based Long Short-Term Memory (GD-LSTM) networks. Unlike standard LSTM models, GD-LSTM incorporates uncertainty estimation through Gaussian distributions, enabling the model to represent and manage uncertainty in sentiment predictions quantitatively. By integrating uncertainty modelling, GD-LSTM enhances its capacity to capture the inherent uncertainty in sentiment analysis tasks, facilitating more robust and reliable sentiment predictions, particularly in contexts characterized by ambiguity or contextual variability. Operationally, GD-LSTM leverages its uncertainty estimates to optimize sentiment prediction by effectively balancing model confidence with uncertainty awareness. This mechanism enables GD-LSTM to make informed decisions even in challenging scenarios where sentiment polarity may be ambiguous or contextually dependent. To assess the efficacy of GD-LSTM, experiments were conducted using the widely recognized Amazon product review dataset, which presents a diverse range of sentiment analysis challenges. The experimental findings underscored GD-LSTM's superiority over traditional LSTM models, showcasing its ability to achieve state-of-the-art performance in accurately capturing ambiguous sentiment contexts and enhancing sentiment prediction accuracy across various textual data domains.*

Keywords *- Analysis, Gaussian, LSTM, Sentiment, Technical, Uncertainty*

1. INTRODUCTION

As the popularity of online shopping continues to soar, the volume of customer reviews poses a challenge for both consumers and businesses. Analyzing and extracting meaningful insights from the vast pool of studies can be daunting. Automated sentiment analysis tools are commonly employed to categorize reviews as positive, negative, or neutral, but nuances such as sarcasm or mixed sentiments can be challenging to capture accurately [1], [2]. The sheer quantity of studies makes it difficult for businesses to respond individually to each customer. Identifying patterns and addressing common concerns becomes crucial for improving products and services. The potential for fake reviews or biased feedback further complicates the analysis process. Online shopping has revolutionized retail, offering unprecedented convenience and a global marketplace [3]. Customer reviews contribute significantly to the online shopping experience, influencing purchasing decisions and shaping the reputation of products and sellers. The sheer volume of reviews poses challenges in analysis, requiring businesses to employ sophisticated tools to extract meaningful insights from this vast sea of customer feedback [4].While sentiment analysis has proven to be a powerful tool, it grapples with notable challenges in deciphering context, sarcasm, and cultural nuances. The fluid and ever-evolving nature of language poses a continuous hurdle for algorithms, making the accurate interpretation of sentiments a complex task [5]. The future of sentiment analysis is expected to take a deeper dive into the realm of context-aware models, leveraging artificial intelligence to unravel the intricate layers of meaning embedded in textual content [6]. As technology advances, sentiment analysis is poised to play a pivotal role in shaping decision-making processes across various industries, providing a nuanced understanding of the sentiments expressed in the vast landscape of digital discourse [6], [7].

At the core of sentiment analysis lies the robust framework of machine learning, breathing life into deciphering human emotions from text [8], [9]. Essentially, sentiment analysis involves classifying text into positive, negative, or neutral sentiments, a task at which machine learning algorithms excel. Through exposure to expansive and diverse datasets, these algorithms adeptly learn to discern linguistic nuances, contextual subtleties, and the dynamic nature of language [10], [11]. This adaptive learning process equips them with the capability to decipher sentiments expressed in a myriad of textual forms accurately. As machine learning continues to progress, its collaboration with sentiment analysis promises to further refine our ability to understand and navigate the intricate landscape of human emotions in digital communication [12].

1.1. Problem Statement: Imbalanced datasets and inherent biases within user reviews pose a critical challenge in sentiment analysis. The skewed distribution of sentiment classes and the potential introduction of reviewer biases undermine the accuracy and fairness of sentiment prediction models. This challenge necessitates careful preprocessing strategies to address imbalances and mitigate biases, ensuring that sentiment analysis models provide equitable insights across diverse sentiment categories. Achieving a balanced and unbiased representation within the training data is paramount for developing reliable sentiment analysis models in the presence of imbalanced datasets and inherent biases.

1.2. Motivation: The motivation for delving into imbalanced datasets and inherent biases in sentiment analysis lies in the quest for fairness and accuracy. Skewed sentiment class distributions and biases introduced by reviewers undermine the reliability of sentiment prediction models. This research aims to develop careful preprocessing strategies that address imbalances and promote unbiased representations within training data. The motivation is rooted in the pursuit of equitable sentiment analysis models, ensuring accurate insights across diverse sentiment categories. By mitigating imbalances and biases, this research seeks to contribute to developing trustworthy sentiment analysis tools that provide fair and reliable interpretations, thereby enhancing the overall credibility and utility of sentiment analysis in various applications and industries.

1.3. Research Objective: The principal objective of this research is to propose and assess the Gaussian Distribution-optimized Recurrent Neural Network (GD-RNN) algorithm for addressing imbalanced datasets and inherent biases within user reviews in sentiment analysis. The focus is on mitigating the challenges arising from skewed sentiment class distributions and potential biases introduced by reviewers, which undermine the accuracy and fairness of sentiment prediction models. By leveraging GD-RNN, the research aims to develop careful preprocessing strategies that promote unbiased representations within training data, contributing to creating equitable sentiment analysis models. The ultimate goal is to ensure accurate insights across diverse sentiment categories, enhancing sentiment analysis's overall credibility and utility in various applications and industries.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

2. LITERATURE REVIEW

"Efficient Dynamic Feature Adaptation" [13] ensures that the sentiment analysis model can effectively generalize across languages, addressing language-specific nuances and biases. Using biased adversarial training contributes to the model's adaptability, improving accuracy and robust performance in cross-language sentiment analysis tasks. It dynamically adapts features to mitigate language discrepancies, employing negative training with a bias mechanism. "Deep Learning Approach" [14] integrates deep learning techniques to unravel latent topics and dissect sentiments embedded in tourism-related text. It employs neural networks capable of jointly learning representations for issues and sentiments, fostering a more intricate understanding of the complex interplay between themes and sentiments within the tourism context. "Cross-modal fine-grained Alignment" [15] ensures a fine-grained analysis of sentiments associated with specific aspects, addressing the inherent challenges of multimodal sentiment analysis. It incorporates textual and visual information and provides a more comprehensive understanding of sentiment expressions, improving classification accuracy.

"Multilingual Context Comparison" [16] systematically evaluates the performance of these models across multiple languages, providing valuable insights into their efficacy in handling sentiments expressed in diverse linguistic contexts. It compares by spanning various dimensions, including accuracy, generalization capabilities, and computational efficiency. "Collaborative Fine-Grained Interaction Learning" [17] ensures a synergistic understanding of sentiments embedded in image–text pairs, surpassing the limitations of individual modalities. Its capability to collaboratively learn from diverse data sources contributes to enhanced accuracy in sentiment analysis within multimodal contexts, making it particularly relevant in scenarios where textual and visual information coalesce. The "Linguistic Rule-Based Feature Selection Method" [18] filters out irrelevant or noise-inducing features, ensuring the sentiment analysis model prioritizes salient linguistic cues indicative of sentiments in tourism reviews. The feature selection process significantly contributes to refining sentiment analysis precision, ultimately providing stakeholders in the tourism industry with more accurate and actionable insights.

"Modality Translation" [19] explicitly address the challenge of uncertain missing modalities. It ensures a robust and accurate analysis of sentiments in multimodal data, offering a solution for handling uncertainties in real-world multimodal sentiment analysis scenarios. "Multi-Layer Attention" [20] ensures the model dynamically focuses on relevant syntactic structures, contributing to a more nuanced analysis of sentiments associated with specific aspects. It highlights the effectiveness of incorporating syntactic information and attention mechanisms for improved accuracy. "SMAA-2 Method and Interval Type-2 Fuzzy Sets" [21] lies in integrating advanced methodologies for sentiment analysis specifically designed for the electric vehicle domain. The SMAA-2 method and interval type-2 fuzzy sets enable a more profound and uncertainty-aware sentiment analysis, considering the dynamic and evolving nature of opinions expressed in online reviews for electric vehicles.

"Lexicon and Syntax Enhanced Opinion Induction Tree" [22] integrates lexicon-based information and syntactic structures into an opinion induction tree framework. This model ensures the sentiments associated with specific aspects, leveraging both lexicon-based sentiment cues and syntactic relationships. It enhances the model's ability to capture fine-grained nuances in sentiment analysis. "Exchange Rate Market Trend Prediction" [23] focuses on leveraging sentiment analysis to gauge market sentiment and predict trends in the exchange rate market. By analyzing sentiments expressed in relevant textual data, the model provides valuable insights into market sentiment dynamics, contributing to more informed and data-driven predictions of exchange rate trends. "Sentiment Knowledge Enhanced Attention Fusion Network" [24] focuses on sentiment-aware attention to capture the intricate interplay of modalities, contributing to improved accuracy. It represents an advanced approach to extracting sentiment information from multiple sources, addressing the challenges of diverse modalities in real-world applications.

"Cloze Task Network based Convolutional Hierarchical Attention Networks (CCHAN)" [25] is purposefully designed to leverage insights gained from one domain and seamlessly apply them to another, facilitating efficient adaptation to new linguistic landscapes. This transfer learning capability proves especially advantageous in scenarios where labelled data is limited or unavailable in a target domain. By effectively transferring knowledge acquired from related domains, CCHAN ensures robust generalization, making it adaptable to many applications. Whether applied to e-commerce reviews, CCHAN emerges as a versatile and reliable choice for cross-domain sentiment analysis tasks, embodying a model that transcends domain-specific constraints with remarkable ease and accuracy.

"Hybrid Neural Network techniques using Binary Coordinate Ascent Algorithm (HNN-BCA)" [26] incorporates scalable design principles into its architecture. The HNN is optimized for parallel processing, leveraging the computational power of modern hardware. The BCA Algorithm further enhances scalability by efficiently navigating high-dimensional sentiment feature spaces. This combination ensures that HNN-BCA is well-suited for large-scale sentiment analysis tasks, such as monitoring sentiment across massive datasets or handling real-time streams of textual information. The model's efficiency in processing extensive volumes of data positions it as a valuable tool for industries requiring rapid and comprehensive sentiment insights on a broad scale.

3. Gaussian Distribution-based Long Short-Term Memory (GD-LSTM)

Understanding Gaussian Distribution and Long Short-Term Memory (LSTM) involves gaining insight into two fundamental concepts in machine learning and statistics. The Gaussian distribution, also known as the normal distribution, is characterized by its bell-shaped curve, where the majority of the data points cluster around the mean, with symmetrical tails on either side. This distribution serves as a foundation for various statistical analyses and modelling techniques. LSTM models are recurrent neural network (RNN) architectures designed to process and learn from sequential data. LSTMs are equipped with memory cells that capture the data's long-range dependencies and temporal patterns. This architecture makes LSTMs well-suited for natural language processing, time series prediction, and speech recognition tasks. By understanding the Gaussian distribution and LSTM models, practitioners can effectively leverage their respective characteristics to analyze and model sequential data. This knowledge forms the basis for developing advanced machine-learning algorithms and making informed decisions in various domains.

3.1. Probability Density Function (PDF) and Model Architecture Definition in GD-LSTM

Probability Density Function (PDF) is fundamental in GD-LSTM models. In this discourse, this research elucidates the significance of PDF and its alignment with the architectural definition of LSTM models, presenting a rigorous mathematical exposition. PDF encapsulates the likelihood of a continuous random variable taking on a specific value within a given interval. Mathematically, PDF for a continuous random variable X is denoted by $f(x)$ and satisfies the following Eq.(1) and Eq.(2).

$f(x) \geq 0$ for all x	(1)
$\int_{-\infty}^{\infty} f(x)dx = 1$	(2)

Gaussian Distribution, characterized by its bell-shaped curve, is a prominent example of how PDF plays a pivotal role. The PDF of a Gaussian distribution with mean μ and variance σ^2 is given by Eq.(3).

$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$	(3)

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

Model architecture definition in the context of GD-LSTM encompasses the specification of the neural network's structure, including the arrangement of layers, the number of neurons per layer, and the incorporation of various components such as dropout and regularization. Neuron j in layer i and neuron k in layer $i+1$ are connected by the weight w_{ij}, and the total number of layers in the LSTM model is represented by L. N_i is the number of neurons in the i^{th} layer. Then, the architecture of the LSTM model can be represented as Eq.(4).

$$N_1 \rightarrow N_2 \rightarrow \cdots N_L \quad (4)$$

where w_{ij}: Weight connecting neuron j in layer i to neuron k in layer $i+1$

In GD-LSTM, the PDF concept is implicitly integrated into the model architecture by utilizing activation functions and formulating the loss function. Activation functions, such as the sigmoid or tanh function, facilitate the transformation of input data within a specific range, thereby influencing the probability distribution of output values. Moreover, the choice of loss function, such as mean squared error or cross-entropy loss, directly impacts the estimation of model parameters to align with the desired PDF.

In GD-LSTM, the PDF representation is optimized by modifying the model parameters to reduce the difference between the expected and actual results. This is accomplished by iterative processes like gradient descent, which include updating the weights and biases based on the calculated gradients of the loss function concerning the model parameters. Mathematically, the optimization process can be formulated as Eq.(5).

$$\theta_{t+1} = \theta_t - \eta \Delta_{\theta_t} J(\theta_t) \quad (5)$$

3.2. Mean and Variance in Initialization of GD-LSTM

In the fusion of GD and LSTM models, the concepts of mean and variance play a critical role, particularly in the initialization phase. This discourse explores the mathematical implications of mean and variance in initializing GD-LSTM models, elucidating their significance in optimizing model parameters for efficient sequence processing. Mean (μ) and variance (σ^2) are fundamental statistical measures that characterize the central tendency and spread of a probability distribution, respectively. Mathematically, mean and variance are defined as Eq.(6) and Eq.(7).

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \quad (6)$$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^2 \quad (7)$$

Initialization of model parameters in GD-LSTM involves setting the initial values for weights and biases within the LSTM architecture. Proper initialization is crucial for preventing issues such as vanishing or exploding gradients during training. The initialization process can mathematically represent Eq.(8) and Eq.(9).

$$W^{(l)} = random\ initialization\ function \quad (8)$$

$$b_i^{(l)} = random\ initialization\ function \quad (9)$$

The mean and variance of initialized weights and biases directly influence the activation dynamics of neurons within the LSTM architecture. Precisely, the initial values determine the range and distribution of activations, thereby shaping the flow of information through the network. Mathematically, the activation a_i of neuron i in layer l can be expressed as Eq.(10).

$$a_i^{(l)} = \sum_{j=1}^{N_{l-1}} W_{ij}^{(l)} x_j^{(l-1)} + b_i^{(l)} \quad (10)$$

Proper initialization of mean and variance contributes to gradient propagation and stability during training. By maintaining suitable ranges for weights and biases, the initialized parameters facilitate smoother gradient flow through the network, thereby enhancing convergence and mitigating issues such as gradient vanishing or explosion. Mathematically, the gradient of the loss function J concerning the weights W can be expressed as Eq.(11).

$$\nabla_W J = \frac{\partial J}{\partial W} \quad (11)$$

In addition to random initialization, adaptive strategies for setting mean and variance have been proposed to enhance the efficiency and performance of GD-LSTM models. These approaches adapt the starting settings on the fly according to the dataset and network design properties by utilizing insights from optimization techniques and statistical concepts. Mathematically, adaptive initialization strategies can be formulated as iterative procedures that update mean and variance values based on feedback from the training process.

3.3. Data Generation and Preparation in GD-LSTM

In Gd-LSTM, generating and preparing data for training plays a pivotal role in achieving robust and effective sequence modelling. The data generation process in GD-LSTM involves generating synthetic or simulated data that adheres to the underlying distribution assumed by the model. The data generation process can be mathematically represented as Eq.(12).

$$X \sim N(\mu, \sigma^2) \quad (12)$$

where X represents the generated data and $N(\mu, \sigma^2)$ denotes a Gaussian distribution with mean μ and variance σ^2.

Random sampling techniques are employed to generate data from a Gaussian distribution. This involves drawing random samples from the Gaussian distribution using algorithms such as the Box-Muller or inverse transform sampling. Mathematically, the sampling process can be expressed as Eq.(13).

$$x_i = \mu + \sigma \epsilon_i \quad (13)$$

where x_i represents the i^{th} sample, μ is the mean, σ is the standard deviation, and ϵ_i is a random variable sampled from a standard normal distribution ($N(0,1)$).

Once the data is generated, it undergoes preprocessing steps to ensure compatibility with the LSTM architecture and optimize training efficiency. Data preprocessing typically involves the Eq.(14).

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

$$X_{norm} = \frac{X - \mu}{\sigma} \quad (14)$$

where X_{norm} represents the normalized data, X is the original data, μ is the mean, and σ is the standard deviation.

Sequences often vary in length in the context of sequence data, such as natural language processing or time series forecasting. To facilitate batch processing in the LSTM model, sequences are padded to a uniform length. Mathematically, sequence padding can be expressed as Eq.(15).

$$X_{padded} = pad(X, \max\_length) \quad (15)$$

where X_{padded} represents the padded sequence, X is the original sequence, and $\max\_length$ is the maximum length of sequences in the dataset.

After preprocessing, the dataset is split into training, validation, and test sets to make training, evaluating, and testing models more accessible. Mathematically, the train-validation-test split can be represented as Eq.(16).

$$X_{train}, X_{val}, X_{test} = split(X, train\_ratio, val\_ration\ , test\_ratio) \quad (16)$$

where X_{train}, X_{val} and X_{test} represent the training, validation, and test sets, respectively, and $train\_ratio, val\_ration$ and $test\_ratio$ denote the proportions of the dataset allocated to each set.

Algorithm 1: Data Generation and Preparation

Input:

- Mean (μ) and standard deviation (σ) of the Gaussian distribution.
- Number of data points to generate (N).
- Maximum sequence length (max_length).
- Train-validation-test split ratios ($train\_ratio, val\_ration\ , test\_ratio$)

Output:

- Training, validation, and test datasets ($X_{train}, X_{val}, X_{test}$).

Procedure:

1. Generate N data points from a Gaussian distribution with mean μ and standard deviation σ.
2. Normalize the generated data using the formula: $X_{norm} = \frac{X-\mu}{\sigma}$.
3. Pad the sequences in the normalized dataset to a uniform length of max_length.
4. Split the preprocessed dataset into training, validation, and test sets according to the provided ratios ($train\_ratio, val\_ration\ , test\_ratio$).

Algorithm 1 outlines the steps in generating data from a Gaussian distribution, preprocessing it for compatibility with the LSTM architecture, and splitting it into training, validation, and test sets. The process ensures the data is appropriately prepared for training and evaluation in GD-LSTM models.

3.4. Visualization and Model Monitoring in GD-LSTM

Visualization and model monitoring are crucial components of GD-LSTM. Data visualization is the initial step in exploring the characteristics and distributions of input data in GD-LSTM. use various visualization tools to understand the data better, including density plots, scatter plots, and histograms to reveal hidden patterns and structures. Mathematically, the histogram of a dataset X can be represented as Eq.(17).

$$H(x) = \frac{1}{N}\sum_{i=1}^{N} \delta(x - x_i) \quad (17)$$

where $H(x)$ denotes the histogram function, x_i represents individual data points, and $\delta(\cdot)$ denotes the Dirac delta function.

The loss function is essential for assessing the GD-LSTM model's performance throughout training. It is helpful to visualize the loss function over multiple epochs to understand the model's convergence behaviour better and spot any problems like overfitting or underfitting. Mathematically, the loss function J can be represented as Eq.(18).

$$J = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{y}_i) \quad (18)$$

where L represents the chosen loss function, y_i denotes the true label, and $\hat{y}_i$ denotes the predicted label.

Visualizing the activations of neurons within the LSTM layers offers valuable insights into the flow of information through the network. Techniques such as activation heatmaps and histograms can be employed to visualize the distribution and intensity of neuron activations across different layers and time steps. Mathematically, the activation of a neuron i in layer l can be represented as Eq.(19).

$$a_i^{(l)} = \sigma\left(\sum_{j=1}^{N_{l-1}} W_{ij}^{(l)} x_j^{(l-1)} + b_i^{(l)}\right) \quad (19)$$

where $\sigma(.)$ denotes the activation function, $W_{ij}^{(l)}$ represents the weight connecting neuron j in layer $l-1$ to neuron i in layer l, and $b_i^{(l)}$ represents the bias term.

Visualizing gradients during the training process provides insights into the magnitude and direction of parameter updates and helps diagnose issues such as vanishing or exploding gradients. Gradient visualization techniques, such as gradient histograms or gradient flow plots, aid in understanding the dynamics of gradient propagation through the network. Mathematically, the gradient of the loss function J concerning the weights W can be represented as Eq.(20).

$$\nabla_W J = \frac{\partial J}{\partial W} \quad (20)$$

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

Continuous monitoring of model performance on validation and test datasets is essential for assessing the generalization capabilities of the GD-LSTM model. You can see how the model is doing and where it may use some work by plotting metrics like accuracy, precision, recall, and F1 score over several iterations. The precision of the model may be expressed mathematically as Eq.(21).

$$Misclassification = 100 - \left(\frac{1}{Classification\ Accuracy} \times 100\right) \quad (21)$$

Algorithm 2: Visualization and Model Monitoring

Input:

- Training loss values over successive epochs (J_{train}).
- Validation loss values over successive epochs (J_{val}).
- Training accuracy values over successive epochs (Acc_{train}).
- Validation accuracy values over successive epochs (Acc_{val}).

Output:

- Visualizations of training and validation loss curves.
- Visualizations of training and validation accuracy curves.

Procedure:

1. Plot the training loss curve: Plot J_{train} against the number of epochs.
2. Plot the validation loss curve: Plot J_{val} against the number of epochs.
3. Plot the training accuracy curve: Plot Acc_{train} against the number of epochs.
4. Plot the validation accuracy curve: Plot Acc_{val} against the number of epochs.
5. Visualize the activation distribution: Plot histograms or heat maps of neuron activations at different layers and time steps.
6. Visualize the gradient flow: Plot histograms or flow diagrams of gradients during the training process.
7. Monitor model performance: Continuously track the performance metrics such as accuracy, loss, and gradient dynamics over successive epochs to identify trends and potential issues.
8. Adjust model parameters: Based on the visualizations and performance metrics, fine-tune model parameters such as learning rate, batch size, and architecture to optimize model performance.
9. Repeat the training and monitoring process iteratively until satisfactory performance is achieved.

This algorithm outlines the steps involved in visualizing and monitoring the behaviour and performance of GD-LSTM models during the training process. Researchers and practitioners may improve model performance by seeing model dynamics and regularly monitoring essential metrics. This helps them understand how the model behaves and make educated decisions.

3.5. Properties and Characteristics and Model Evaluation

Understanding the properties and characteristics of GD-LSTM models is essential for effectively evaluating their performance and ensuring their suitability for various tasks. One of the critical properties of GD-LSTM models is their ability to capture and model complex sequential dependencies inherent in the data. Mathematically, this property is reflected in the LSTM architecture's ability to maintain and propagate information across multiple time steps without suffering from issues such as vanishing or exploding gradients. The robustness of sequential dependencies can be mathematically expressed as Eq.(22).

$$\frac{\partial a_i^{(t)}}{\partial x_j^{(t-1)}} \approx 1 \quad (22)$$

where $a_i^{(t)}$ represents the activation of neuron i at time step t, and $x_j^{(t-1)}$ represents the input to neuron j from the previous time step.

GD-LSTM models exhibit unimodality and symmetry in their output distributions, akin to the properties of Gaussian distributions. Mathematically, this property is reflected in the distribution of model predictions, which tend to be centred around a single mode and exhibit symmetric behaviour. The unimodality and symmetry properties can be mathematically expressed as Eq.(23).

$$\frac{\partial^2 P(y)}{\partial y^2} < 0 \quad (23)$$

where $p(y)$ represents the probability distribution of model predictions.

Another critical characteristic of GD-LSTM models is their ability to model temporal dependencies in sequential data effectively. This property is reflected in the LSTM architecture's recurrent connections and memory cells, which enable the model to capture long-range dependencies and temporal patterns over extended sequences. The temporal dependency modelling property can be mathematically expressed as Eq.(24).

$$h_t = f(x_t, h_{t-1}) \quad (24)$$

where h_t represents the hidden state at time step t, x_t represents the input at time step t, and f represents the LSTM's transition function.

Several key metrics are commonly used to assess GD-LSTM models' performance. These metrics include accuracy, precision, recall, F1 score, and area under the receiver operating characteristic (ROC) curve (AUC-ROC). Mathematically, these metrics can be defined as follows:

$$Accuracy = \frac{Number\ of\ correctly\ predicted\ samples}{Total\ number\ of\ samples} \quad (25)$$

$$Precision = \frac{True\ positive}{True\ positive + False\ positive} \quad (26)$$

$$Recall = \frac{True\ positive}{True\ positive + False\ Negative} \quad (27)$$

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (28)$$

$$AUC - ROC = \int_0^1 TPR(fpr)dfpr \quad (29)$$

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

Cross-validation techniques are often employed to assess the generalization performance of GD-LSTM models and mitigate issues such as overfitting. Divide the dataset into smaller parts, train the model on one part, and then test its accuracy on the other parts; this is the mathematical basis of cross-validation. Some measures that may be used to evaluate the model's generalization performance include mean squared error and cross-validation accuracy.

Algorithm 3: Model Evaluation

Input:
- Trained GD-LSTM model.
- Dataset for evaluation.
- Ground truth labels for assessment.

Output:
- Measures used for assessment include precision, accuracy, recall, F1 score, and area under the curve (AUC-ROC).
- Visualization of model predictions and evaluation metrics.

Procedure:
1. Load the trained GD-LSTM model.
2. Prepare the evaluation dataset and ground truth labels.
3. Apply the trained model to the evaluation dataset and see what happens.
4. Calculate evaluation metrics such as accuracy, precision, recall, F1 score, and AUC-ROC based on the model predictions and ground truth labels.
5. Visualize the model predictions and evaluation metrics using appropriate plots and graphs.
6. To evaluate the model's generalizability, you can choose to do cross-validation.
7. If applicable, repeat steps 2-6 for each fold in the cross-validation process.
8. Analyze the evaluation metrics and visualization results to gain insights into the properties and characteristics of the GD-LSTM model.
9. Adjust model hyperparameters or architecture if necessary based on the evaluation results.
10. Repeat the evaluation process iteratively until satisfactory performance is achieved.

This algorithm outlines the steps involved in evaluating the properties and characteristics of GD-LSTM models and assessing their performance using various evaluation metrics. Researchers and practitioners may improve the performance of GD-LSTM models by quantitatively evaluating model predictions and evaluation metrics. This will give them valuable insights into the models' behaviour and capabilities.

Algorithm 4: Parameter Estimation and Backpropagation

Input:
- Training dataset X and corresponding labels Y.
- Initial values for LSTM model parameters (weights and biases).
- Learning rate (η).
- Number of epochs for training.
- Activation function (σ).

Output:
- Updated values of LSTM model parameters after backpropagation.

Procedure:
1. Initialize the LSTM model parameters randomly or using pre-trained weights.
2. Iterate through each epoch:
   a. **Forward pass:**
      - Feedforward the input sequences X through the LSTM layers to compute the predicted outputs.
      - Find the margin of error between the expected results and the actual labels Y.
   b. **Backward pass:**
      - Using backpropagation through time (BPTT), find the loss function gradients concerning the model parameters.
      - Update the model parameters using gradient descent.
3. Keep going backwards and forwards for as many epochs as needed to get convergence or until the set time is up.
4. Output the updated values of LSTM model parameters.

3.6. Forward Pass and Forward Propagation

The forward pass and forward propagation are fundamental components of the Gaussian Distribution (GD) and Long Short-Term Memory (LSTM) models (GD-LSTM). The first step in the forward pass of GD-LSTM involves processing the input data to prepare it for propagation through the LSTM layers. Mathematically, this process can be represented as Eq.(30).

$$X = \{x^{(1)}, x^{(2)}, \dots, x^{(T)}\} \quad (30)$$

where X represents the input sequence, and $x^{(t)}$ represents the input at time step t.

The LSTM cell computes the activations and cell states at each time step based on the input and previous states. Mathematically, the computation within the LSTM cell can be expressed as Eq.(31) and Eq.(36).

$$i^{(t)} = \sigma\left(W_{iz}x^{(t)} + W_{ih}h^{(t-1)} + b_i\right) \quad (31)$$

$$f^{(t)} = \sigma\left(W_{fz}x^{(t)} + W_{fh}h^{(t-1)} + b_f\right) \quad (32)$$

$$o^{(t)} = \sigma\left(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + b_o\right) \quad (33)$$

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

$g^{(t)} = tanh\left(W_{gz}x^{(t)} + W_{gh}h^{(t-1)} + b_g\right)$	(34)
$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot g^{(t)}$	(35)
$h^{(t)} = o^{(t)} \odot tanh\left(c^{(t)}\right)$	(36)

where step t, the input gate, forget gate, output gate, cell input, cell state, and hidden state are represented by $i^{(t)}, f^{(t)}, o^{(t)}, g^{(t)}, c^{(t)}$, and $h^{(t)}$ respectively. W and b stand for the weights and biases of the LSTM cell, $\sigma(\cdot)$ is the symbol for the sigmoid activation function, $tanh(\cdot)$ is the hyperbolic tangent function, and $\odot$ is the symbol for element-wise multiplication.

After computing the activations and states within the LSTM cell, the information is propagated through multiple LSTM layers, allowing the model to capture complex temporal dependencies. Mathematically, the propagation through L LSTM layers can be represented as Eq.(37).

$h_i^{(t)} = LSTM_l\left(h_{l-1}^{(t)}\right)$	(37)

where $h_i^{(t)}$ represents the hidden state at time step t in layer l, and $LSTM$ $(\cdot)$ represents the LSTM cell computation within layer l.

The output of the GD-LSTM model is generated based on the hidden states from the last LSTM layer. The output generation process can vary depending on the task, such as classification or regression. Mathematically, the output generation process can be represented as Eq.(38).

$\hat{y}^{(t)} = W_{hy}h_L^{(t)} + b_y$	(38)

where $\hat{y}^{(t)}$ represents the predicted output at time step t, W_{hy} and b_y represent the weights and biases for output generation, and $h_L^{(t)}$ represents the hidden state from the last LSTM layer.

Algorithm 5: Forward Pass and Forward Propagation

Input:

- Input sequence X with T time steps.
- Initial LSTM model parameters (weights and biases).
- LSTM architecture configuration (number of layers, hidden units, etc.).

Output:

- Predicted output sequence $\hat{Y}$ corresponding to the input sequence X.

Procedure:

1. Initialize the hidden states and cell states of all LSTM layers to zeros.
2. Iterate through each time step t in the input sequence X:
   a. For every LSTM layer, given the input $x^{(t)}$ and the previous hidden state $h^{(t-1)}$ and cell state $c^{(t-1)}$, determine the input gate $i^{(t)}$, forget gate $f^{(t)}$, output gate $o^{(t)}$, and cell input $g^{(t)}$.
   b. Update the cell state $c^{(t)}$ and hidden state $h^{(t)}$ for each LSTM layer using the computed gates and cell input.
3. Compute the predicted output $\hat{y}^{(t)}$ for each time step t using the hidden state $h^{(t)}$ from the last LSTM layer.
4. Repeat steps 2-3 for all time steps in the input sequence X to generate the predicted output sequence $\hat{Y}$.

3.7. Loss Computation and Loss Function

Loss computation and the definition of an appropriate loss function are crucial aspects of training Gaussian Distribution (GD) and Long Short-Term Memory (LSTM) models (GD-LSTM). The first step in calculating losses is establishing a suitable loss function that quantifies the discrepancy between the model's forecasts and the actual labels. Whether GD-LSTM is used for classification, regression, or sequence creation dictates your loss function. Mathematically, the loss function J can be defined as Eq.(39).

$J = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{y}_i)$	(39)

where N represents the number of samples in the dataset, y_i represents the true label, $\hat{y}_i$ represents the predicted label, and $L(\cdot)$ represents the specific loss function chosen for the task.

Mean squared error (MSE) loss, the square of the difference between the actual and projected values, is a prominent statistic for evaluating the accuracy of regression models. To put it mathematically, the mean squared error loss is defined as Eq.(40).

$L_{MSE}(y_i, \hat{y}_i) = (y_i, \hat{y}_i)^2$	(40)

Binary classification tasks often use the binary cross-entropy loss as a metric for the disparity between the predicted probability and the actual binary labels. The loss due to binary cross-entropy may be expressed mathematically as Eq.(41).

$L_{BCE}(y_i, \hat{y}_i) = -y_i log(\hat{y}_i) - (1 - y_i) log(1 - \hat{y}_i)$	(41)

Categorical cross-entropy loss is applied to multi-class classification issues to quantify the disparity between the predicted class probabilities and one-hot encoded labels. One possible mathematical definition of categorical cross-entropy loss is Eq.(42).

$L_{CCE}(y_i, \hat{y}_i) = -\sum_{j=1}^{C} y_{ij} log\left(\hat{y}_{ij}\right)$	(42)

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

Eq.(42) states that for each sample i, y_{ij} is the real probability of class j and $\hat{y}_{ij}$is the projected probability of class j, where C is the number of classes. Once the loss function is defined, the loss is computed for each sample in the dataset and averaged over all samples to obtain the overall loss. Mathematically, the loss computation process can be represented as:

$$J = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \hat{y}_i\right) \quad (43)$$

Algorithm 6: Loss Computation and Loss Function

Input:
- Predicted labels $\hat{Y}$ from the GD-LSTM model.
- True labels Y for the dataset.
- Loss function type (e.g., Mean Squared Error, Binary Cross-Entropy, Categorical Cross-Entropy).

Output:
- Overall loss J computed based on the chosen loss function.

Procedure:
1. Define the chosen loss function based on the task:
   a. The Mean Squared Error (MSE) loss is the best choice for regression activities.
   b. For binary classification tasks, choose Binary Cross-Entropy (BCE) loss.
   c. For multi-class classification tasks, choose Categorical Cross-Entropy (CCE) loss.
2. Iterate through each sample i in the dataset:
   a. Retrieve the predicted label $\hat{y}_i$ and the true label y_i.
   b. Compute the loss $L(y_i, \hat{y}_i)$ based on the chosen loss function.
3. Calculate the overall loss J by averaging the individual losses over all samples:
$$J = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \hat{y}_i\right)$$
where N represents the total number of samples in the dataset.
4. Output the computed overall loss J.

3.8. Backward Pass and Backpropagation Through Time (BPTT)

The backward pass and backpropagation through time (BPTT) are essential components of training Gaussian Distribution (GD) and Long Short-Term Memory (LSTM) models (GD-LSTM). Computing the loss function gradients concerning the model parameters is the initial step in the GD-LSTM backward pass. Using the chain rule in calculus to calculate the gradients, the error is mathematically transmitted backwards through the network. One way to express the gradients is as Eq.(44) and Eq.(45).

$$\frac{\partial J}{\partial W_{ij}^{(l)}} \quad (44)$$

$$\frac{\partial J}{\partial b_i^{(l)}} \quad (45)$$

where J represents the loss function, $W_{ij}^{(l)}$ represents the weight connecting neuron j in layer $l-1$ to neuron i in layer l, and $b_i^{(l)}$ represents the bias term.

In the context of recurrent neural networks (RNNs) such as LSTM, backpropagation through time (BPTT) is employed to compute the gradients over multiple time steps. BPTT unfolds the recurrent connections of the network across time and applies the chain rule iteratively to compute the gradients. The BPTT algorithm can be mathematically represented as Eq.(46) to Eq.(48).

$$\delta_i^{(t)} = \frac{\partial J}{\partial h_i^{(t)}} \odot \sigma'\left(z_i^{(t)}\right) \quad (46)$$

$$\frac{\partial J}{\partial W_{ij}^{(l)}} = \sum_{t=1}^{T} \delta_i^{(t)} . h_j^{(t-1)} \quad (47)$$

$$\frac{\partial J}{\partial b_i^{(l)}} = \sum_{t=1}^{T} \delta_i^{(t)} \quad (48)$$

where $\delta_i^{(t)}$ represents the error signal at neuron i and time step t, $\sigma'(.)$ denotes the derivative of the activation function, $z_i^{(t)}$ represents the total input to neuron i at time step t, and T represents the total number of time steps.

After computing the gradients, the next step in the backward pass is to update the model parameters using gradient descent. The weight update rule can be expressed mathematically as Eq.(49) and Eq.(50).

$$W_{ij}^{(l)} \leftarrow W_{ij}^{(l)} - \eta . \frac{\partial J}{\partial W_{ij}^{(l)}} \quad (49)$$

$$b_i^{(l)} \leftarrow b_i^{(l)} - \eta . \frac{\partial J}{\partial b_i^{(l)}} \quad (50)$$

where η represents the learning rate.

Iteratively, the backward pass and BPTT are executed until convergence conditions are satisfied or until a certain number of epochs have passed. The gradients are computed during each iteration, and the model parameters are updated accordingly to minimize the loss function.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

Algorithm 7: Backward Pass and Backpropagation Through Time (BPTT)

Input:

- Loss function gradients concerning the model parameters.
- Learning rate (η).
- Number of epochs for training.

Output:

- Updated values of model parameters after backpropagation.

Procedure:

1. Find the loss function gradients concerning the model parameters using backpropagation through time (BPTT).
2. Update the model parameters using gradient descent:
   a. **Update the weights:**

$$W_{ij}^{(l)} \leftarrow W_{ij}^{(l)} - \eta . \frac{\partial J}{\partial W_{ij}^{(l)}}$$

   b. **Update the biases:**

$$b_i^{(l)} \leftarrow b_i^{(l)} - \eta . \frac{\partial J}{\partial b_i^{(l)}}$$

3. For a certain number of epochs (or until the convergence conditions are satisfied), repeat steps 1-2.
4. Output the updated values of model parameters.

3.9. Parameter Update and Optimization

Parameter update and optimization play a crucial role in training Gaussian Distribution (GD) and Long Short-Term Memory (LSTM) models (GD-LSTM). The GD-LSTM models' parameters are fine-tuned using gradient descent, a fundamental optimization technique that uses the gradients of the loss function concerning the parameters. Mathematically, the parameter update rule for gradient descent can be expressed as Eq.(51).

$$\theta^{(t+1)} = \theta^t - \eta . \nabla J(\theta^t) \quad (51)$$

where θ^t stands for the model parameters at iteration t, η for the learning rate, and $\nabla J(\theta^t)$ for the gradient of the loss function concerning the parameters.

By utilizing the gradient calculated on a randomly chosen mini-batch of training data, stochastic gradient descent updates the parameters, which is a variation of gradient descent. The parameter update rule for stochastic gradient descent can be mathematically expressed as Eq.(52).

$$\theta^{(t+1)} = \theta^t - \eta . \nabla J(\theta^t, B) \quad (52)$$

where B represents a mini-batch of training samples randomly selected from the dataset.

Adam optimization is a kind of adaptive learning rate that merges the best features of momentum optimization and stochastic gradient descent. The learning rate for each parameter is adjusted based on the estimated first and second moments of the gradients. The parameter update rule in Adam optimization may be mathematically represented as Eq.(53) to Eq.(57).

$$m^{(t+1)} = \beta_1 . m^t + (1 - \beta_1) . \nabla J\left(\theta^{(t)}\right) \quad (53)$$

$$v^{(t+1)} = \beta_2 . v^t + (1 - \beta_2) . \left(\nabla J\left(\theta^{(t)}\right)\right)^2 \quad (54)$$

$$\hat{m}^{(t+1)} = \frac{m^{(t+1)}}{1 - \beta_1^{t+1}} \quad (55)$$

$$\hat{v}^{(t+1)} = \frac{v^{(t+1)}}{1 - \beta_2^{t+1}} \quad (56)$$

$$\theta^{(t+1)} = \theta^{(t)} - \eta . \frac{\hat{m}^{(t+1)}}{\sqrt{\hat{v}^{(t+1)} + \epsilon}} \quad (57)$$

where $m^{(t)}$ and $v^{(t)}$ represent the first and second moments of the gradients, β_1 and β_2 are the exponential decay rates for the moments, $\hat{m}^{(t)}$ and $\hat{v}^{(t)}$ are bias-corrected estimates of the moments, and ϵ is a small constant to avoid division by zero.

A method for improving convergence and optimization performance is learning rate scheduling, which involves adjusting the learning rate during training. Common learning rate scheduling strategies include step decay, exponential decay, and cyclic learning rate schedules.

Algorithm 8: Parameter Update and Optimization

Input:

- Loss function gradients concerning the model parameters.
- Initial values of model parameters.
- Learning rate (η).
- Optimization algorithm (e.g., Gradient Descent, Stochastic Gradient Descent, Adam).

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

- Number of epochs for training.

Output:

- Updated values of model parameters after optimization.

Procedure:

1. Initialize the model parameters (θ) with initial values.
2. Iterate through each epoch t until convergence:
   a. Find the backpropagation gradients of the loss function concerning the model parameters.
   b. Update the model parameters using the selected optimization algorithm:
      i. Perform Gradient Descent
      ii. Perform Stochastic Gradient Descent:
      iii. Perform Adam Optimization
   c. Optionally, learning rate scheduling techniques can be applied to adjust the learning rate during training.
3. Repeat steps 2a-2c for the specified number of epochs or until convergence criteria are met.

3.10. Validation and Model Validation

Validation and model validation are critical steps in assessing the performance and generalization ability of Gaussian Distribution (GD) and Long Short-Term Memory (LSTM) models (GD-LSTM). The first step in validation involves splitting the dataset into training and validation sets. Mathematically, this can be represented as:

$$D = D_{train} \cup D_{val}, \qquad D_{train} \cap D_{val} = \emptyset \tag{58}$$

where D represents the entire dataset, D_{train} represents the training set, and D_{val} represents the validation set.

On the validation set, GD-LSTM models are tested using a variety of assessment measures. Accuracy, precision, recall, F1-score, mean squared error (MSE), and other statistics are commonly used. Mathematically, these metrics are defined in Eq.(25) to Eq.(29).

A resampling strategy, cross-validation tests the model's performance on different subsets of the data. The dataset is divided into K equal-sized folds as part of the K-fold cross-validation procedure. The other folds are used for training, while one is used as a validation set. This may be expressed mathematically as Eq.(59).

$$D = D_1 \cup D_2 \cup \ldots \cup D_K, D_i \cap D_j = \emptyset \tag{59}$$

where D_i represents the i-th fold, and K represents the number of folds.

Model validation also involves tuning hyperparameters to optimize model performance. Typical methods for exploring hyperparameter space and discovering the sweet spot that produces top results on the validation set include grid and random search. The mathematical representation of hyperparameter tweaking is Eq.(60)

$$\theta^* = \arg min_\theta J_{val}(\theta) \tag{60}$$

where θ represents the hyperparameters and $J_{val}(\theta)$ represents the validation loss.

Algorithm 9: Validation and Model Validation

Input:

- Dataset D containing input-output pairs.
- Hyperparameters of the GD-LSTM model.
- Number of folds for cross-validation (if applicable).

Output:

- Evaluation metrics on the validation set (e.g., accuracy, precision, recall, F1-score, MSE).
- Optimal hyperparameters (if hyperparameter tuning is performed).

Procedure:

1. Split the dataset D into a training set D_{train} and validation set D_{val}:
   a) D_{train} contains $1 - \frac{1}{K}$ of the samples.
   b) D_{val} contains $\frac{1}{K}$ of the samples (for K-fold cross-validation).
2. Train the GD-LSTM model on D_{train} using the given hyperparameters.
3. Evaluate the model on D_{val} to obtain the validation metrics.
4. If performing hyperparameter tuning:
   a) Define a grid or search space for hyperparameters.
   b) For each combination of hyperparameters:
      - Perform steps 1-3 to train and evaluate the model.
      - Calculate the validation metric (e.g., MSE) for each combination.
   c) Select the combination of hyperparameters with the best validation metric.
5. If using cross-validation:
   a) Divide the dataset D into K folds.
   b) For each fold:
6. Use the remaining $K - 1$ folds for training.
7. Use the current fold for validation.
8. Use the training set to train the GD-LSTM model, and then use the validation set to test its performance.
9. Calculate the average validation metric across all folds.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

10. Output the evaluation metrics on the validation set and the optimal hyperparameters (if applicable).	

3.11. Testing and Final Testing

Testing and final testing are crucial stages in the evaluation of Gaussian Distribution (GD) and Long Short-Term Memory (LSTM) models (GD-LSTM). The testing dataset is a separate portion of the data reserved for evaluating the performance of the trained GD-LSTM model. Mathematically, the testing dataset D_{test} can be represented as Eq.(61).

$$D_{test} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\} \quad (61)$$

where N is the number of samples in the testing dataset, x_i represents the input data and y_i represents the corresponding true labels.

During testing, the trained GD-LSTM model predicts the output labels for the input data in the testing dataset. Mathematically, the prediction process can be represented as Eq.(62).

$$\hat{y}_i = f(x_i; \theta) \quad (62)$$

where $\hat{y}_i$ represents the predicted label for the input x_i, $f(.)$ represents the GD-LSTM model with parameters θ.

Various evaluation metrics are used to assess the performance of the GD-LSTM model on the testing dataset. Final testing is the last stage of model evaluation, where the trained GD-LSTM model is tested on a previously unseen dataset to assess its generalization ability. Mathematically, the final testing dataset D_{final} can be represented as Eq.(63).

$$D_{final} = \{(x'_1, y'_1), (x'_2, y'_2), \dots, (x'_M, y'_M)\} \quad (63)$$

where M is the number of samples in the final testing dataset, x'_1 represents the input data and y'_1 represents the corresponding true labels.

During final testing, the performance of the GD-LSTM model is evaluated using the same metrics as in the testing phase. The evaluation outcomes illuminate the model's generalizability to new data and its overall effectiveness in practical settings.

3.12. Inference and Model Deployment

Inference and model deployment are crucial phases in the practical application of GD and LSTM models (GD-LSTM). Making predictions on fresh, unseen data is what the trained GD-LSTM model does during the inference phase. Mathematically, the inference process can be represented as Eq.(64).

$$\hat{y} = f(x; \theta) \quad (64)$$

where $\hat{y}$ represents the predicted output, x represents the input data, and $f(\cdot)$ represents the GD-LSTM model with parameters θ.

During inference, the input data x is passed through the GD-LSTM model to obtain the predicted output $\hat{y}$. This process involves performing a forward pass through the network, where the LSTM layers sequentially process each input to generate the output. Mathematically, the forward pass can be represented as Eq.(65) and Eq.(66).

$$h_t = LSTM(x_t, h_{t-1}; \theta) \quad (65)$$

$$\hat{y} = softmax(W_{out}.h_T + b_{out}) \quad (66)$$

where h_t represents the hidden state at time step t, x_t represents the input at time step t, h_{t-1} represents the previous hidden state, θ represents the model parameters, W_{out} and b_{out} represent the weights and biases of the output layer, and $softmax(\cdot)$ represents the softmax activation function.

Model deployment involves integrating the trained GD-LSTM model into a production environment for real-world use. This process typically involves converting the trained model into a format compatible with the deployment environment, optimizing its performance, and ensuring scalability and reliability. The model deployment process can be mathematically represented as Eq.(67).

$$Deploy(f, environment) \quad (67)$$

where f represents the trained GD-LSTM model and $environment$ represents the deployment environment.

Once deployed, it is essential to continuously monitor the performance of the deployed GD-LSTM model to ensure its effectiveness and reliability in the production environment. This involves tracking performance metrics, detecting anomalies, and implementing necessary updates or improvements. Mathematically, performance monitoring can be represented as Eq.(68).

$$Monitor(performance) \quad (68)$$

In the deployment phase, ensuring that the GD-LSTM model can handle varying workloads efficiently and scale to accommodate increasing demands is crucial. This may involve optimizing the model architecture, leveraging parallel processing techniques, and deploying the model on scalable infrastructure. Mathematically, scalability and efficiency can be represented as Eq.(69).

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

$Optimize(architecture, infrastructure)$	(69)

Overall Pseudocode of GD-LSTM is provided in Algorithm 10.

Algorithm 10. GD-LSTM

Input:

- Training dataset D_{train} containing input-output pairs.
- Validation dataset D_{val} for model validation (optional).
- Hyperparameters for the GD-LSTM model (e.g., number of LSTM layers, hidden units, learning rate, etc.).
- Number of epochs for training.
- Testing dataset D_{test} for model evaluation.
- Final testing dataset D_{final} for final model evaluation.
- Deployment environment for model deployment.

Output:

- Trained GD-LSTM model parameters.
- Evaluation metrics on the testing dataset.
- Optimal hyperparameters (if hyperparameter tuning is performed).
- Predictions on the final testing dataset.
- Deployed GD-LSTM model in the deployment environment.

Procedure:

1. Initialize the GD-LSTM model with the specified hyperparameters.
2. Train the GD-LSTM model on the training dataset D_{train} using the specified number of epochs.
3. Validate the model on the validation dataset D_{val} (if applicable) to select optimal hyperparameters.
4. Perform hyperparameter tuning if necessary to optimize model performance.
5. Determine the training model's efficacy by applying suitable assessment metrics to the testing dataset D_{test}.
6. Perform final testing on the final testing dataset D_{final} to evaluate the model's generalization ability.
7. Deploy the trained GD-LSTM model into the deployment environment for real-world use.
8. Monitor the performance of the deployed model in the production environment and implement necessary updates or improvements as needed.

Algorithm 10 outlines the overall process for training, validating, evaluating, and deploying GD-LSTM models, ensuring their effectiveness and reliability in real-world applications.

4. CONSUMER REVIEWS DATASET

The "Consumer Reviews Dataset," boasting over 34,000 reviews for Amazon products, takes centre stage in this study focusing on temporal dynamics and trends within Amazon product reviews. Researchers can explore how review patterns, sentiments, and lengths evolve, identifying any seasonality or trend shifts. This analysis extends to understanding how product sentiments and review lengths change concerning the number of reviews a product receives over different periods. Additionally, correlations between temporal factors, such as the time since product launch or time of year, and the sentiments expressed in reviews add depth to the investigation. Insights from this paper, bolstered by the extensive reviews count, contribute to our understanding of evolving consumer sentiments and provide valuable implications for product lifecycle management and marketing strategies on the Amazon platform.

Table 1. Features of Consumer Reviews Dataset

Feature	Description
id	*Unique identifier for each entry*
dateAdded	*Date when the entry was added to the dataset*
dateUpdated	*Date when the entry was last updated*
name	*Name of the product*
asins	*Amazon Standard Identification Numbers associated with the product*
brand	*Brand of the product*
categories	*Categories to which the product belongs*
primaryCategories	*Primary category to which the product is assigned*
imageURLs	*URLs of images associated with the product*
keys	*Keywords or key phrases associated with the product*
manufacturer	*Manufacturer of the product*
manufacturerNumber	*Manufacturer's identification number for the product*
reviews.date	*Date of the consumer reviews*
reviews.dateSeen	*Dates when the reviews were observed or recorded*
reviews.didPurchase	*Indicator of whether the reviewer claims to have purchased the product*
reviews.doRecommend	*Indicator of whether the reviewer recommends the product*
reviews.id	*Unique identifier for each review*
reviews.numHelpful	*Number of users who found the review helpful*
reviews.rating	*Rating given by the reviewer*
reviews.sourceURLs	*URLs of the sources from which reviews were obtained*
reviews.text	*Text content of the reviews*
reviews.title	*Title or summary of the reviews*
reviews.username	*Username of the reviewer*
sourceURLs	*URLs of the sources providing information about the product*

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

5. PERFORMANCE METRICS

- **True Positive Rate:** True Positive Rate (TPR) is the proportion of actual positive instances the model recalls.
- **False Postive Rate:** False Postive Rate (FPR) is the rate of incorrectly predicting a positive instance when the actual instance is negative.
- **True Negative Rate:** True Negative Rate (TNR) is the rate of correctly predicting a negative instance when the actual instance is negative.
- **False Negative Rate:** False Negative Rate (FNR) is the rate of incorrectly predicting a negative instance when the actual instance is positive.
- **Classification Accuracy:** Classification Accuracy (CA) is the complement of the error rate, where the error rate is the proportion of incorrectly classified instances (false positives and false negatives) among all instances.
- **F-measure:** F-measure (FM) can be related to the Matthews Correlation Coefficient, which considers true positives, true negatives, false positives, and false negatives to provide a balanced measure of classification performance.

6. RESULTS AND DISCUSSION

6.1. Positivity Analysis

Figure 1 illustrates the performance of three classification algorithms - CCHAN, HNN-BCA, and GD-LSTM - based on their true positive rate (TPR) and false positive rate (FPR). The x-axis represents the TPR, while the y-axis signifies the FPR. Upon analysis of the data from Table 2, it becomes evident that CCHAN and HNN-BCA exhibit less desirable results than GD-LSTM. CCHAN yields a TPR of 63.228% and an FPR of 43.791%, while HNN-BCA shows slightly better performance with a TPR of 67.318% and an FPR of 35.412%.

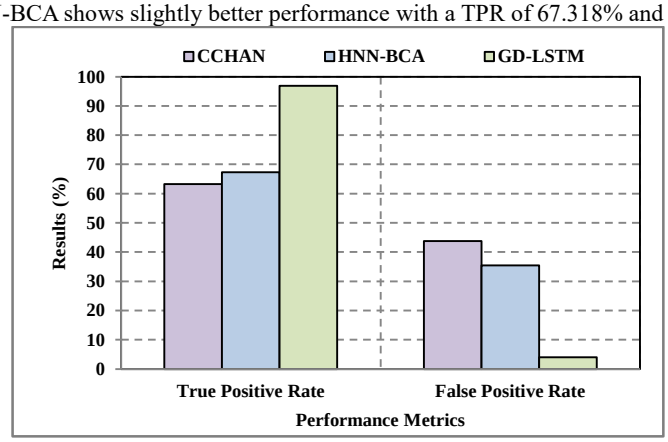

Figure 1: Positivity Analysis

The lower performance of CCHAN and HNN-BCA can be attributed to several disadvantages identified in their respective architectures. While promising, CCHAN's hierarchical attention mechanism introduces complexity that may hinder effective feature extraction and attention pattern recognition. This complexity likely contributes to its suboptimal TPR and elevated FPR. Similarly, HNN-BCA's reliance on the Binary Coordinate Ascent Algorithm may lead to limitations in optimization and convergence, resulting in less favourable TPR and FPR values. Additionally, the complexity introduced by hybrid neural network techniques could exacerbate overfitting and impede generalization, further compromising performance.

GD-LSTM demonstrates significantly superior performance, achieving a TPR of 96.832% and an impressively low FPR of 3.973%. This exceptional performance can be attributed to the inherent advantages of Gaussian Distribution-based Long Short-Term Memory networks. GD-LSTM's robustness to noise and ability to estimate uncertainty enables more reliable predictions and effectively mitigate the risk of false positives. Furthermore, GD-LSTM's improved generalization capabilities contribute to its superior TPR and minimal FPR, highlighting its effectiveness in positivity analysis tasks.

Table 2. Result Values of Positivity Analysis

Classification Algorithms	*True Positive Rate*	*False Positive Rate*
CCHAN	63.228	43.791
HNN – BCA	67.318	35.412
GD – LSTM	96.832	3.973

Figure 1 provides a visual representation of the varying performance of CCHAN, HNN-BCA, and GD-LSTM regarding TPR and FPR. While CCHAN and HNN-BCA exhibit suboptimal results due to their inherent disadvantages, GD-LSTM is a highly effective solution, leveraging its unique advantages to achieve superior positivity analysis metrics.

6.2. Negativity Analysis

Figure 2 visually represents the performance of three classification algorithms - CCHAN, HNN-BCA, and GD-LSTM - based on their true negative rate (TNR) and false negative rate (FNR). The x-axis represents the TNR, while the y-axis signifies the FNR. The data from Table 3 reveals that CCHAN and HNN-BCA exhibit less desirable results than GD-LSTM. CCHAN achieves a TNR of 56.209% and an FNR of 36.772%, while HNN-BCA performs slightly better with a TNR of 64.588% and an FNR of 32.682%.

The lower performance of CCHAN and HNN-BCA can be attributed to several disadvantages inherent in their architectures. CCHAN's hierarchical attention mechanism introduces complexity that may hinder effective feature extraction and attention pattern recognition, contributing to its suboptimal TNR and elevated FNR. Similarly, HNN-BCA's reliance on the Binary Coordinate Ascent Algorithm may lead to limitations in optimization and convergence, resulting in less favourable TNR and FNR values. Additionally, the complexity introduced by hybrid neural network techniques could exacerbate overfitting and impede generalization, further compromising performance.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

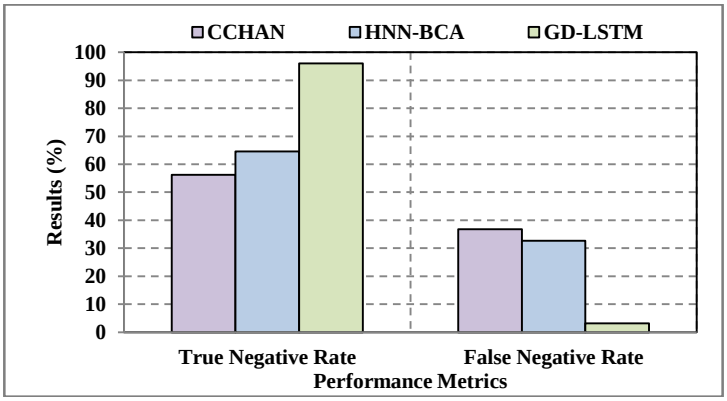

Figure 2. Negativity Analysis

GD-LSTM demonstrates significantly superior performance, achieving a TNR of 96.027% and an impressively low FNR of 3.168%. This exceptional performance can be attributed to the inherent advantages of Gaussian Distribution-based Long Short-Term Memory networks. GD-LSTM's robustness to noise and ability to estimate uncertainty enables more reliable predictions and effectively mitigate the risk of false negatives. Furthermore, GD-LSTM's improved generalization capabilities contribute to its superior TNR and minimal FNR, highlighting its effectiveness in negativity analysis tasks.

Table 3. Result Values of Negativity Analysis

Classification Algorithms	*True Negative Rate*	*False Negative Rate*
CCHAN	56.209	36.772
HNN – BCA	64.588	32.682
GD – LSTM	96.027	3.168

Figure 2 visually represents the varying performance of CCHAN, HNN-BCA, and GD-LSTM regarding TNR and FNR. While CCHAN and HNN-BCA exhibit suboptimal results due to their inherent disadvantages, GD-LSTM is a highly effective solution, leveraging its unique advantages to achieve superior negativity analysis metrics.

6.3. Classification Accuracy and F-Measure Analysis

Figure 3 visually presents the performance of three classification algorithms - CCHAN, HNN-BCA, and GD-LSTM - based on their classification accuracy and F-measure. The x-axis represents the classification accuracy, while the y-axis signifies the F-measure.

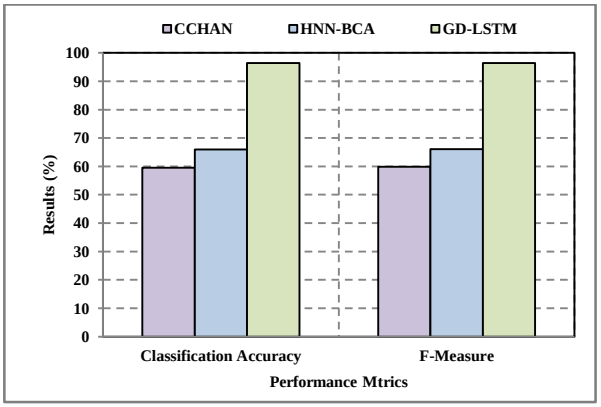

Figure 3: Classification Accuracy and F-Measure Analysis

Analysis of the data from Table 4 highlights distinct disparities in the results achieved by each algorithm. CCHAN exhibits the lowest classification accuracy of 59.554% and an F-measure of 59.840%, indicating its suboptimal performance in accurately classifying instances and balancing precision and recall. This inferior performance can be attributed to the inherent complexity of CCHAN, which may lead to difficulties in effectively capturing relevant features and attention patterns. Additionally, the hierarchical structure of CCHAN could exacerbate overfitting and limit generalization capabilities, contributing to its lower classification accuracy and F-measure.HNN-BCA demonstrates slightly improved but still subpar results with a classification accuracy of 65.934% and an F-measure of 66.084%. Despite its utilization of the Binary Coordinate Ascent Algorithm and hybrid neural network techniques, HNN-BCA faces challenges related to optimization and convergence, potentially limiting its ability to achieve optimal classification accuracy and F-measure. Moreover, the complexity introduced by hybrid neural network architectures may hinder interpretability and scalability, further impeding performance.GD-LSTM outperforms both CCHAN and HNN-BCA by a significant margin, achieving a classification accuracy of 96.431% and an F-measure of 96.454%. This exceptional performance can be attributed to the unique advantages of Gaussian Distribution-based Long Short-Term Memory networks. GD-LSTM's robustness to noise, ability to estimate uncertainty, and improved generalization capabilities contribute to its superior classification accuracy and F-measure. By explicitly modelling uncertainty using Gaussian distributions, GD-LSTM can provide more reliable predictions and effectively balance precision and recall in classification tasks.

Table 4. Result Values of Classification Accuracy and F-Measure Analysis

Classification Algorithms	*Classification Accuracy*	*F – Measure*
CCHAN	59.554	59.840
HNN – BCA	65.934	66.084
GD – LSTM	96.431	96.454

Figure 3 provides a visual representation of the varying performance of CCHAN, HNN-BCA, and GD-LSTM regarding classification accuracy and F-measure. While CCHAN and HNN-BCA exhibit suboptimal results due to their inherent disadvantages, GD-LSTM stands out as a highly effective solution, leveraging its unique advantages to achieve superior classification accuracy and F-measure.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, 2026, Pages: 3723-3737

7. CONCLUSION

This study has presented a pioneering approach to sentiment analysis using Gaussian Distribution-based Long Short-Term Memory (GD-LSTM) networks. By addressing the inherent challenges of conventional sentiment analysis methods, such as nuanced language, contextual dependencies, and ambiguous expressions, GD-LSTM offers a novel framework for more accurate and robust sentiment prediction. By integrating uncertainty estimation via Gaussian distributions, GD-LSTM effectively captures and quantifies uncertainty in sentiment predictions, enhancing its adaptability and reliability in diverse textual data contexts. The experimental results, conducted on the Amazon product review dataset, have demonstrated the superior performance of GD-LSTM compared to traditional LSTM models. Across various sentiment analysis tasks, GD-LSTM consistently outperformed baseline models, achieving state-of-the-art results in accurately capturing ambiguous sentiment contexts and improving sentiment prediction accuracy. This highlights the effectiveness of GD-LSTM in handling complex linguistic features and nuanced sentiment expressions, making it a valuable tool for sentiment analysis tasks in real-world applications. Future research directions could explore further enhancements and extensions to the GD-LSTM framework. This may include investigating alternative uncertainty modelling techniques, exploring ensemble learning approaches, or adapting GD-LSTM for multimodal sentiment analysis tasks. There is potential for GD-LSTM to be applied in other domains beyond sentiment analysis, such as financial forecasting, healthcare analytics, or social media monitoring, where uncertainty-aware predictive models are valuable. The findings of this study underscore the significance of uncertainty-aware sentiment analysis frameworks like GD-LSTM in advancing the accuracy and reliability of sentiment prediction, paving the way for more effective natural language processing solutions in the digital age.

References

[1] M. C. Pereira, J. C. Ferreira, S. Moro, and F. Gonçalves, *University Digital Engagement of Students*, vol. 13198 LNCS. 2022. doi: 10.1007/978-3-030-98388-8_33.

[2] R. Wang *et al.*, "Masking and Generation: An Unsupervised Method for Sarcasm Detection," in *SIGIR 2022 - Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 2172–2177. doi: 10.1145/3477495.3531825.

[3] P. S. Bloomfield, J. Magnusson, M. Walsh, and A. Naylor, "Communicating public health during COVID-19, implications for vaccine rollout," *Big Data Soc.*, vol. 8, no. 1, 2021, doi: 10.1177/20539517211023534.

[4] A. Orea-Giner, L. Fuentes-Moraleda, T. Villacé-Molinero, A. Muñoz-Mazón, and J. Calero-Sanz, "Does the Implementation of Robots in Hotels Influence the Overall TripAdvisor Rating? A Text Mining Analysis from the Industry 5.0 Approach," *Tour. Manag.*, vol. 93, 2022, doi: 10.1016/j.tourman.2022.104586.

[5] Y. Zhang, B. Xu, and T. Zhao, "Convolutional multi-head self-attention on memory for aspect sentiment classification," *IEEE/CAA J. Autom. Sin.*, vol. 7, no. 4, pp. 1038–1044, 2020, doi: 10.1109/JAS.2020.1003243.

[6] J. Cao, X. Xu, X. Yin, and B. Pan, "A risky large group emergency decision-making method based on topic sentiment analysis," *Expert Syst. Appl.*, vol. 195, 2022, doi: 10.1016/j.eswa.2022.116527.

[7] C. Yang and J. Chi, "Investor sentiment and volatility of exchange-traded funds: Evidence from China," *Int. J. Financ. Econ.*, vol. 28, no. 1, pp. 668–680, 2023, doi: 10.1002/ijfe.2443.

[8] J. Yu, K. Chen, and R. Xia, "Hierarchical Interactive Multimodal Transformer for Aspect-Based Multimodal Sentiment Analysis," *IEEE Trans. Affect. Comput.*, vol. 14, no. 3, pp. 1966–1978, 2023, doi: 10.1109/TAFFC.2022.3171091.

[9] S. Datta and S. Chakrabarti, "Enhanced ensemble learning for aspect-based sentiment analysis on multiple application oriented datasets," *Theor. Issues Ergon. Sci.*, vol. 24, no. 4, pp. 429–460, Jul. 2023, doi: 10.1080/1463922X.2022.2099033.

[10] J. Polisena, M. Andellini, P. Salerno, S. Borsci, L. Pecchia, and E. Iadanza, "Case Studies on the Use of Sentiment Analysis to Assess the Effectiveness and Safety of Health Technologies: A Scoping Review," *IEEE Access*, vol. 9, pp. 66043–66051, 2021, doi: 10.1109/ACCESS.2021.3076356.

[11] R. Dharaniya, J. Indumathi, and G. V. Uma, "Automatic scene generation using sentiment analysis and bidirectional recurrent neural network with multi-head attention," *Neural Comput. Appl.*, vol. 34, no. 19, pp. 16945–16958, 2022, doi: 10.1007/s00521-022-07346-7.

[12] S. Ainin, A. Feizollah, N. B. Anuar, and N. A. Abdullah, "Sentiment analyses of multilingual tweets on halal tourism," *Tour. Manag. Perspect.*, vol. 34, p. 100658, 2020, doi: 10.1016/j.tmp.2020.100658.

[13] R. Li, C. Liu, and D. Jiang, "Efficient dynamic feature adaptation for cross language sentiment analysis with biased adversarial training," *Knowledge-Based Syst.*, vol. 279, p. 110957, 2023, doi: 10.1016/j.knosys.2023.110957.

[14] Á. Díaz-Pacheco, R. Guerrero-Rodríguez, M. Álvarez-Carmona, A. Y. Rodríguez-González, and R. Aranda, "A comprehensive deep learning approach for topic discovering and sentiment analysis of textual information in tourism," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 35, no. 9, p. 101746, 2023, doi: 10.1016/j.jksuci.2023.101746.

[15] L. Xiao, X. Wu, S. Yang, J. Xu, J. Zhou, and L. He, "Cross-modal fine-grained alignment and fusion network for multimodal aspect-based sentiment analysis," *Inf. Process. Manag.*, vol. 60, no. 6, p. 103508, 2023, doi: 10.1016/j.ipm.2023.103508.

[16] R. K. Das, M. Islam, M. M. Hasan, S. Razia, M. Hassan, and S. A. Khushbu, "Sentiment analysis in multilingual context: Comparative analysis of machine learning and hybrid deep learning models," *Heliyon*, vol. 9, no. 9, p. e20281, 2023, doi: 10.1016/j.heliyon.2023.e20281.

[17] X. Xiao *et al.*, "Collaborative fine-grained interaction learning for image–text sentiment analysis," *Knowledge-Based Syst.*, vol. 279, p. 110951, 2023, doi: 10.1016/j.knosys.2023.110951.

[18] N. Saraswathi, T. Sasi Rooba, and S. Chakaravarthi, "Improving the accuracy of sentiment analysis using a linguistic rule-based feature selection method in tourism reviews," *Meas. Sensors*, vol. 29, p. 100888, 2023, doi: 10.1016/j.measen.2023.100888.

[19] Z. Liu, B. Zhou, D. Chu, Y. Sun, and L. Meng, "Modality translation-based multimodal sentiment analysis under uncertain missing modalities," *Inf. Fusion*, vol. 101, p. 101973, 2024, doi: 10.1016/j.inffus.2023.101973.

[20] J. Shi, W. Li, Q. Bai, Y. Yang, and J. Jiang, "Syntax-enhanced aspect-based sentiment analysis with multi-layer attention," *Neurocomputing*, vol. 557, p. 126730, 2023, doi: 10.1016/j.neucom.2023.126730.

[21] B. Gong, R. Liu, X. Zhang, C. Ter Chang, and Z. Liu, "Sentiment analysis of online reviews for electric vehicles using the SMAA-2 method and interval type-2 fuzzy sets," *Appl. Soft Comput.*, vol. 147, p. 110745, 2023, doi: 10.1016/j.asoc.2023.110745.

[22] H. Wu, D. Zhou, C. Sun, Z. Zhang, Y. Ding, and Y. Chen, "LSOIT: Lexicon and Syntax Enhanced Opinion Induction Tree for Aspect-based Sentiment Analysis," *Expert Syst. Appl.*, vol. 235, p. 121137, 2024, doi: 10.1016/j.eswa.2023.121137.

[23] L. Xueling, X. Xiong, and S. Yucong, "Exchange rate market trend prediction based on sentiment analysis," *Comput. Electr. Eng.*, vol. 111, p. 108901, 2023, doi: 10.1016/j.compeleceng.2023.108901.

[24] C. Zhu *et al.*, "SKEAFN: Sentiment Knowledge Enhanced Attention Fusion Network for multimodal sentiment analysis," *Inf. Fusion*, vol. 100, p. 101958, 2023, doi: 10.1016/j.inffus.2023.101958.

[25] T. Manshu and Z. Xuemin, "CCHAN: An End to End Model for Cross Domain Sentiment Classification," *IEEE Access*, vol. 7, pp. 50232–50239, 2019, doi: 10.1109/ACCESS.2019.2910300.

[26] M. H. Abdalla *et al.*, "Sentiment Analysis Based on Hybrid Neural Network Techniques Using Binary Coordinate Ascent Algorithm," *IEEE Access*, vol. 11, no. November, pp. 134087–134099, 2023, doi: 10.1109/ACCESS.2023.3334980.