Journal of Theoretical and Applied Information Technology
15th April 2025. Vol.103. No.7
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMIZING SENTIMENT INTERPRETATION OF COURSERA COURSE REVIEWS USING AN ADAPTIVE FISH SWARM OPTIMIZATION-INSPIRED RECURRENT NEURAL NETWORK (AFSO-RNN)

J SAHITHA BANU[1], G PREETHI[2]
[1]Research Scholar, Department of Computer Science, Ponnaiyah Ramajayam Institute of Science and Technology (PRIST), Thanjavur,Tamilnadu
[2] Associate Professor, Ponnaiyah Ramajayam Institute of Science and Technology (PRIST), Thanjavur,Tamilnadu
E-mail:[1] shahithak@gmail.com,[2] mgpreethi@gmail.com

ABSTRACT

This study introduces an innovative approach, Adaptive Fish Swarm Optimization-Inspired Recurrent Neural Network (AFSO-RNN), to optimize sentiment interpretation of Coursera course reviews. The AFSO-RNN model combines the adaptive capabilities of fish swarm optimization with the power of recurrent neural networks. The recurrent neural network component processes the textual data of course reviews by capturing semantic meaning and context. It learns from sequential dependencies to extract relevant features for sentiment analysis. The adaptive fish swarm optimization component enhances the learning process of the recurrent neural network. Inspired by collective behaviour in fish swarms, it dynamically adjusts network parameters during training. The optimisation process explores and exploits optimal solutions by mimicking fish swarm movement and communication patterns, improving sentiment interpretation accuracy. Extensive experiments on a large dataset of Coursera course reviews demonstrate the superior performance of AFSO-RNN compared to traditional sentiment analysis techniques. The model's optimized sentiment interpretation provides valuable insights into learner sentiments, enabling informed decision-making for instructors, administrators, and learners regarding course selection and improvement. This research contributes to sentiment analysis in online learning environments by showcasing the effectiveness of the AFSO-RNN model. By combining recurrent neural networks with adaptive fish swarm optimization, AFSO-RNN offers a promising avenue for enhancing the accuracy and efficiency of sentiment analysis for Coursera course reviews

Keywords: *Adaptive, Courseera, Fish Swarm optimization, Recurrent Neural Network, Sentiment Analysis, Optimization*

1. INTRODUCTION

Online learning has disrupted traditional educational models, leveraging technological advancements to create an innovative and flexible learning ecosystem [1]. Through Learning Management Systems (LMS) and cloud-based platforms, online learning provides learners a wide range of educational resources, including multimedia content, e-books, and interactive learning modules. Incorporating Artificial Intelligence (AI) and machine learning algorithms within online learning platforms enables personalized learning experiences through adaptive assessments and intelligent tutoring systems [2]. Learners can receive immediate feedback, track their progress, and access supplementary materials tailored to their learning needs. The asynchronous nature of online learning allows for self-paced learning, granting learners the freedom to set their schedules and progress at their preferred speed. Online learning fosters a global community of learners, connecting individuals from different backgrounds, cultures, and geographic locations through virtual classrooms and online discussion forums [3]. The availability of massive open online courses (MOOCs) further democratizes education, making high-quality educational content accessible to a vast audience. Embracing online-learning empowers learners to develop digital literacy, adaptability, and lifelong learning skills required in today's rapidly evolving knowledge-based society [4].

In the digital age, where information is at our fingertips, understanding the sentiment behind textual data has become crucial. Sentiment analysis, also known as opinion mining, is a field that focuses on extracting emotions, attitudes, and opinions expressed in a text [5]. By applying computational techniques and machine learning algorithms, sentiment analysis enables us to analyse and interpret the sentiment conveyed in large volumes of text data. One of the primary applications of sentiment analysis is customer feedback analysis [6]. With the rise of e-commerce platforms and online reviews, businesses can collect valuable customer feedback. Sentiment analysis allows companies to automatically process and categorize this feedback, providing insights into customer satisfaction levels, common pain points, and areas for improvement. This information can guide businesses in enhancing their products, services, and overall customer experience [7]. Another significant application of sentiment analysis is in social media monitoring. Social media platforms serve as a rich source of public opinion and sentiment. By analysing tweets, posts, comments, and other user-generated content, sentiment analysis can help track public sentiment towards brands, events, or trending topics. This information is beneficial for marketers, advertisers, and decision-makers in understanding the public's perception and tailoring their strategies accordingly [8].

Sentiment analysis also finds applications in political analysis and public policy making. Researchers and policymakers can gauge public opinion on policies, politicians, or societal issues by analysing sentiments expressed in news articles, blogs, or social media posts [9]. This insight can inform decision-making processes, help identify areas of concern, and facilitate more effective communication strategies. Sentiment analysis has implications for market research and competitive analysis. Businesses can gain valuable insights into consumer preferences, sentiment towards competitors, and emerging trends by analysing online reviews, forum discussions, or survey responses. This knowledge allows companies to adapt their marketing strategies, develop new products, and gain a competitive edge in the market [10]. As sentiment analysis techniques advance, researchers are exploring new dimensions, such as aspect-based sentiment analysis and emotion detection. Aspect-based sentiment analysis aims to identify sentiments towards specific aspects or features of a product or service, providing more granular insights. Emotion detection focuses on categorizing emotions expressed in text, enabling a deeper understanding of user experiences and reactions [11]. Bio-inspired optimization techniques are versatile and can be applied in various domains [12]–[16], including sentiment analysis, to achieve better results [17], [18], [27], [28], [19]–[26]. These techniques provide a means to enhance sentiment analysis performance by leveraging nature-inspired algorithms and adapting them to the specific requirements and characteristics of different domains, such as social media, product reviews, or customer feedback. Using bio-inspired optimization, sentiment analysis can be more accurate, robust, and adaptable to different textual data types.

1.1. Problem Statement

Negation and ambiguity present significant challenges in sentiment analysis, as they can lead to incorrect sentiment classification. Sentences containing negation, double negatives, or ambiguous expressions can completely reverse the sentiment expressed or introduce uncertainty. Existing sentiment analysis models often struggle to handle these linguistic complexities, resulting in erroneous sentiment predictions. This problem calls for developing advanced algorithms and techniques to accurately identify and handle negations, resolve ambiguities, and infer the correct sentiment in complex sentence structures. Overcoming the challenge of negation and ambiguity in sentiment analysis requires innovative approaches that can effectively capture the subtle linguistic cues and disambiguate sentiment expressions, leading to more reliable and precise sentiment classification.

1.2. Motivation

The motivation behind addressing the challenge of negation and ambiguity in sentiment analysis is to improve the precision and reliability of sentiment classification. We can avoid misinterpretations and erroneous sentiment predictions by developing advanced algorithms and techniques to accurately identify negations, resolve ambiguities, and infer the correct sentiment in complex sentence structures. This motivation arises from enhancing customer satisfaction by accurately capturing sentiments in customer feedback, improving sentiment analysis accuracy in social media monitoring, and providing more reliable insights to guide business strategies and decision-making.

1.3. Objective

Devise an advanced sentiment analysis algorithm that effectively identifies and handles negations, resolves ambiguities, and accurately infers the correct sentiment in complex sentence structures. This objective aims to enhance the precision and reliability of sentiment classification by mitigating the impact of linguistic complexities on sentiment analysis results. It involves developing techniques to accurately interpret sentiments expressed in negated sentences, double negatives, and ambiguous expressions, thereby avoiding misinterpretations and erroneous sentiment predictions.

- Identify and handle negations, ambiguities, and complex sentence structures.
- Enhance precision and reliability of sentiment classification.
- Avoid misinterpretations and erroneous sentiment predictions.

2. LITERATURE REVIEW

.

"Semantic Conceptualization" [29] involves representing text documents as a bag of concepts, where each concept is associated with a specific sentiment. The concepts are derived from semantic resources or domain-specific knowledge bases. The model captures the underlying sentiment expressed in the text by tagging the concepts with sentiment labels. This approach enables a more nuanced understanding of sentiment, focusing on concepts' meaning and context rather than individual words. By leveraging tagged bag-of-concepts, sentiment analysis models can provide more accurate and context-aware predictions, facilitating applications such as opinion mining, social media sentiment analysis, and customer feedback analysis. "Urdu Sentiment Analysis" [30] capture complex patterns and relationships in multimodal data. Fusing textual and non-textual information allows for more comprehensive sentiment analysis, considering linguistic and visual cues. This approach can be valuable in understanding sentiment in social media content, user reviews, or multimedia data in the Urdu language. By applying deep learning algorithms to multimodal data, Urdu sentiment analysis can provide insights into public opinion, customer feedback, and user experiences in various domains, aiding decision-making processes and enhancing user engagement.

"Attention-Emotion-Enhanced Convolutional LSTM" [31] is a powerful model for sentiment analysis that combines attention mechanisms, emotion-enhanced features, and the Convolutional LSTM architecture. It effectively captures important text parts, incorporates emotional cues, and captures spatial and temporal dependencies in the data. This comprehensive approach improves sentiment analysis performance and can be applied to various tasks, including social media sentiment monitoring, customer feedback analysis, and opinion mining, providing a more accurate understanding of sentiment in textual data. "Broad Multitask Transformer Network" [32] employs a multitask learning approach, simultaneously handling various sentiment analysis subtasks, such as document-level, sentence-level, and aspect-level sentiment analysis. The model learns to generalize well across different sentiment analysis scenarios by joint training on multiple tasks. This multitask framework transfers shared knowledge and enhance the model's overall performance. BMT-Net demonstrates promising results in sentiment analysis applications, providing a versatile and efficient solution for understanding sentiment in text data across multiple levels of granularity.

"Dynamic Bayesian Network" [33] combines the flexibility of Bayesian networks with the ability to capture temporal dynamics in sentiment and topic transitions. The DBN model considers the dependencies between topics and sentiments over time, allowing for the exploration of how sentiments change within different topics or themes. The DBN approach comprehensively explains the evolving relationships between topics and sentiments by incorporating dynamic factors, such as time intervals and sequential patterns. This enables researchers and analysts to track sentiment shifts, identify emerging trends, and gain insights into the evolution of public opinion in various domains, such as social media, customer reviews, or political discourse. The DBN approach offers a valuable tool for studying the complex dynamics of topic-sentiment relationships and their evolution over time. "Efficient Adaptive Transfer Network" [34] focuses on effectively transferring knowledge across different domains or tasks to enhance performance. EATN employs an adaptive transfer learning approach that dynamically adjusts the transferability of knowledge from source domains to target domains based on their similarities. This adaptive mechanism allows the model to leverage relevant information while mitigating the adverse effects of domain differences. EATN performs aspect-level sentiment analysis tasks better by efficiently utilizing transfer learning, particularly in scenarios with limited labelled data. This model provides a valuable solution for sentiment analysis in various domains, allowing for efficient knowledge transfer and better utilization of available resources.

"Supervised Machine Learning-Based Sentiment Analysis" [35] approaches consider various factors such as domain, dataset characteristics, and feature representations to provide a more accurate prediction. By analysing the context in which the sentiment analysis model will be applied, such as social media, customer reviews, or news articles, contextual-based approaches can account for the specific challenges and nuances of the target domain. This enables researchers and practitioners to make informed decisions regarding model selection, feature engineering, and hyperparameter tuning, improving sentiment analysis performance. "Deformable CNN and Attention" [36] addresses the challenges of capturing aspect-specific information and modelling the relationships between aspects and sentiment expressions in the text. By using deformable CNNs, ADeCNN effectively captures local context and aspect-specific features, allowing for more precise sentiment analysis at the aspect level. Incorporating attention mechanisms further improves the model's ability to focus on essential aspects and sentiment-bearing words within the text. ADeCNN provides a robust solution for aspect-level sentiment analysis, offering improved accuracy and capturing fine-grained sentiment information.

"Affective Knowledge Augmented Interactive Graph Convolutional Network" [37] incorporates affective knowledge, such as emotion lexicons or sentiment intensifiers, to enhance sentiment analysis in Chinese text. Leveraging interactive graph convolutional networks effectively captures the contextual relationships between aspects and sentiments, considering both syntactic and semantic dependencies. This interactive graph-based approach allows the model to learn and propagate sentiment information through the aspect-sentiment graph, enabling a more comprehensive understanding of aspect-level sentiment in Chinese text. It demonstrates superior performance in capturing fine-grained sentiment nuances. It is a valuable tool for aspect-based sentiment analysis in Chinese language applications, such as product reviews, social media content, or customer feedback analysis. "Multitask Multiview Neural Network" [38] addresses multiple subtasks within the sentiment analysis process, such as aspect extraction and sentiment classification, in a unified framework. By leveraging multi-view representations, which incorporate multiple perspectives or modalities of the input data, the model captures a more comprehensive understanding of the aspects and their associated sentiments. The multitask paradigm enables shared learning across subtasks, enhancing the model's performance and generalization capabilities. This approach allows for a holistic analysis of aspect-based sentiment, enabling more accurate and nuanced sentiment analysis results. The multitask multi-view neural network is a valuable tool for extracting and analysing sentiments towards specific aspects in various domains, including product reviews, social media discussions, and customer feedback analysis.

"Naive Bayes Classification Algorithm (NBCA)" [39] is widely utilized in sentiment analysis, but it has certain limitations to be mindful of. A fundamental assumption of NBCA is feature independence, which may not hold in real-world text data. This oversimplification can limit its ability to capture complex sentiment patterns accurately. Additionally, NBCA treats all features equally, disregarding the varying impact of different words or phrases on sentiment. This may result in the suboptimal weighting of feature importance. The assumption of feature independence may lead to misclassifications when sentiment depends on contextual or interactive effects between features. Despite these limitations, NBCA can still generate valuable results in many sentiment analysis tasks, especially in more uncomplicated cases or as a baseline approach. It is essential to understand the assumptions and considerations of NBCA when applying it to sentiment analysis and to evaluate its performance against other algorithms in more complex scenarios. "Random Forest Classification Algorithm (RFCA)" [40] is a versatile machine learning algorithm that offers several advantages for sentiment analysis. RFCA can handle both numerical and categorical text features, allowing effective processing of textual data by encoding it into numerical representations. RFCA excels at detecting complex feature interactions, enabling it to capture nuanced sentiment patterns that rely on combinations of words and phrases. RFCA also provides feature importance measures, facilitating feature selection and guiding analysts to focus on the most influential factors for sentiment prediction. With its ability to handle text data, capture feature interactions, and offer insights into feature importance, RFCA enhances sentiment analysis by providing accurate predictions and valuable interpretability, making it an indispensable tool for sentiment analysis practitioners.

3. ADAPTIVE FISH SWARM OPTIMIZATION-INSPIRED RECURRENT NEURAL NETWORK (AFSO-RNN)

3.1. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) offer distinct advantages over primary neural networks like Multi-Layer Perceptions (MLPs) due to their ability to process information in both forward and backward directions. RNNs can store and utilize information temporarily by iterating through various layers. At each time step i, the RNN performs two primary operations: hidden state update and output calculation. The hidden state denoted as h_i, is computed by applying an activation function σ to a combination of the current input p_i and the previous hidden state $h_{\{i-1\}}$. This operation can be expressed as Eq.(1).

$$h_i = \sigma\left(w_{hh} * h_{\{i-1\}} + w_{hp} * p_i + b_h\right) \quad (1)$$

where w_{hh} represents the weight matrix for the hidden-to-hidden connections, w_{hp} represents the weight matrix for the input-to-hidden connections, and b_h represents the bias vector for the hidden state.

Once the hidden state is updated, the RNN calculates the output l_i bypassing the hidden state h_i through a typical neural network represented by TT, which can be written as Eq.(2):

$$l_i = TT(h_i) \quad (2)$$

where TT denotes the function that computes the output based on the hidden state.

The RNN can process the input sequence in either the forward or backward direction. Forward processing involves sequentially calculating the hidden states and outputs from time step 1 to m. Conversely, backward processing computes the hidden states and outputs in reverse order, from time step m to 1. RNNs are categorized as deep neural networks because they can process information across multiple layers. However, a challenge they face is the vanishing gradient problem, which hinders their performance in handling long-term dependencies. Alternative RNN architectures such as Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) have been developed to overcome this issue, introducing specialized mechanisms to mitigate the vanishing gradient problem. To train the sRNN (Simple RNN) model, a technique called forward-backwards propagation (FFP) is employed. FFP, commonly used in MLPs, is adapted to the context of sRNN by extending the process over multiple time steps. The formulation for the FFP process in the sRNN can be described as follows:

Considering the input value p_m, which serves as the input to the sRNN model, the input layer (represented by p_m) is connected to the current hidden layer (represented by n_{lp}) through connection weights. Similarly, the connection weight between the present hidden layer and the next hidden layer is denoted as n_{ll}. The connection strength between the last hidden layer and the output layer is given by n_{lq}. The biases implemented in the links between the hidden layer and the result layer are represented by m_l and m_q. At time step m, the output can be expressed as l_m. In this context, the sRNN can be mathematically formulated as Eq.(3) to Eq.(5).

$$l_m = j(l_{m-1}, p_m) \quad (3)$$

$$l_m = j\left(\left[n_{lp}.p_m + n_{ll}.p_{m-1}\right] + v_l\right) \quad (4)$$

$$\bar{q}_m = j\left(\left[n_{lp}.p_m + n_{ll}.p_{m-1}\right] + v_l\right) \quad (5)$$

where j denotes the activation function that combines the inputs, weights, and biases to compute the output, and the symbol * represents the multiplication operation.

The predicted result is represented by $\bar{q}_m$in the given context. This formulation illustrates how the sRNN processes the input at each time step, updating the hidden state and calculating the output based on the current and previous inputs and their corresponding connection weights and biases. It is important to note that the specific form of the activation function j and the definitions of the variables, such as $n_{lp}, n_{ll}, n_{lq}, v_l, m_l$ and m_q can vary depending on the specific implementation and variations of the sRNN model. The training of the sRNN involves a sequential process where, at each time step m, the error (H) between the predicted output values $(\bar{q}_m)$ and the valid output values (q_m) is calculated. In machine learning terminology, this error is commonly referred to as the loss and is denoted as Z. The primary objective of the training process is to minimize Z, indicating that an ideal model would exhibit significantly reduced loss. To quantify the overall loss produced by training a single set of data starting from time step 1 to T, Eq.(6) is applied.

$$Z(\bar{q}_m, q) = \sum_{m=1}^{T} Z(\bar{q}_m, q_m) \tag{6}$$

where T represents the maximum duration of the training process, indicating the number of time steps considered during training. The sum over m indicates that the loss at each time step is accumulated to compute the overall loss.

One challenge sRNN encounter during training is the vanishing gradient problem or the gradient explosion issue. These issues tend to arise as the size of the dataset increases and as the training duration T extends. The vanishing gradient problem refers to the phenomenon where the gradients used for updating the network's parameters diminish exponentially as they propagate back through time. This can result in slower convergence or difficulty in capturing long-term dependencies. On the other hand, the gradient explosion occurs when the gradients grow exponentially, leading to unstable training and divergence. The vanishing gradient and gradient explosion problems are particularly relevant to sRNN due to their recurrent nature and the repeated application of weight matrices and activation functions over multiple time steps. Researchers have developed alternative RNN architectures, such as LSTM and GRU, incorporating specialized gating mechanisms to alleviate these gradient-related issues and improve training stability.

As the size of the dataset increases, or the training duration extends, the vanishing gradient and gradient explosion problems become more pronounced. Researchers continue to explore advanced optimization techniques, weight initialization strategies, and architectural modifications to address these challenges and improve the training efficiency and effectiveness of sRNNs. Considering the variable Z, which represents the loss (H) produced by a batch of data V within the given training period T, the partial derivative of the loss concerning a specific weight N can be calculated. This is expressed in Eq.(7).

$$\frac{\varepsilon Z}{\varepsilon N} = \sum_{m=1}^{T} \frac{\varepsilon Z_m}{\varepsilon N} \tag{7}$$

Eq.(7) signifies that the derivative of the overall loss Z concerning a particular weight N is the sum of the derivatives of the loss at each time step, denoted as $\frac{\varepsilon Z_m}{\varepsilon N}$, for m ranging from 1 to T.To simplify Eq.(7), we can apply the chain rule of differentiation. The reformulation is achieved as Eq.(8).

$$\frac{\varepsilon Z}{\varepsilon N} = \sum_{m=1}^{T} \frac{\varepsilon Z_m}{\varepsilon q_m} \frac{\varepsilon q_m}{\varepsilon l_m} \frac{\varepsilon l_m}{\varepsilon q_t} \frac{\varepsilon q_t}{\varepsilon N} \tag{8}$$

where

- $\frac{\varepsilon Z_m}{\varepsilon N}$ represents the partial derivative of the loss at time step m concerning the predicted output q_m. This derivative quantifies the impact of a small change in q_m on the overall loss. It reflects how errors in the prediction at a specific time step contribute to the overall loss.
- $\frac{\varepsilon q_m}{\varepsilon l_m}$ denotes the derivative of the predicted output q_m concerning the hidden state l_m at the same time step. This derivative capture how changes in the hidden state affect the predicted output. It demonstrates how information flows through the network, connecting the hidden and output states.
- $\frac{\varepsilon l_m}{\varepsilon q_t}$ symbolizes the derivative of the hidden state l_m at time step m concerning the hidden state q_t at a later time step t. This term illustrates the ability of the sRNN to retain and propagate information across time. It reveals how the hidden state at a particular time step influences the hidden state at a later time step.
- $\frac{\varepsilon q_t}{\varepsilon N}$ represents the derivative of the hidden state q_t at time step t for the weight N. This term quantifies the impact of a change in weight N on the hidden state q_t. It

demonstrates how weights affect the hidden state and, subsequently, the overall prediction of the sRNN.

By considering all these terms together, Eq.(8) showcases the flow of gradients through the sRNN during the training process. The derivative of the loss for a particular weight N accumulates contributions from various factors, including the prediction error at each time step, the relationship between the hidden state and the predicted output, and the propagation of information across time steps.

Understanding these gradient dependencies is crucial for tackling challenges like the vanishing gradient problem and optimizing the training of sRNNs. Researchers have developed techniques such as gradient clipping, which limits the magnitude of gradients, and initialization strategies, to alleviate gradient-related issues. Additionally, advancements like LSTM and GRU architectures, incorporating specialized gating mechanisms, have effectively mitigated the vanishing gradient problem and captured long-term dependencies. Continuous exploration and development of techniques to improve gradient flow and address the challenges inherent in training sRNNs are essential for enhancing their performance and enabling them to model and learn from sequential data effectively.

Eq.(9) highlights the dependence of the hidden state at time m on the hidden state at a later time t. The derivative expresses this $\frac{\varepsilon l_m}{\varepsilon l_t}$. This derivative capture how changes in the hidden state at time m affect the hidden state at time t. To understand this relationship, Eq.(9) involves the utilization of the Jacobian matrix, which represents the derivative of the hidden state at time m for the hidden state at time t. The Jacobian matrix can be decomposed into a sequence of derivatives, as shown in Eq.(9):

$$\frac{\varepsilon l_m}{\varepsilon l_t} = \frac{\varepsilon l_m}{\varepsilon l_{m-1}} \frac{\varepsilon l_{m-1}}{\varepsilon l_{m-2}} \cdots \frac{\varepsilon l_{t+1}}{\varepsilon l_t} = \sum_{s=t+1}^{m} \frac{\varepsilon l_s}{\varepsilon l_{s-1}} \quad (9)$$

In Eq.(9), the summation over s from $t+1$ to m indicates that the derivative of the hidden state at time s concerning the hidden state at the time $(s-1)$ is accumulated. This demonstrates the progressive impact of each hidden state on the following hidden states in the sequence, allowing information to flow through time steps. By combining Eq.(7) and Eq.(9), an Eigen Decomposition Vector (EDV) can be obtained, represented as Eq.(10).

$$EDV = N^F diag[j'(l_{m-1})] \quad (10)$$

In Eq.(10), N represents the weight matrix, F represents the activation function, 'j' represents the derivative of the activation function, and $l_{(m-1)}$ represents the hidden state at the time $(m-1)$. The EDV represents the eigenvalues and eigenvectors associated with the hidden state dynamics. It characterizes how the hidden state evolves and the impact of the weight matrix and activation function on this evolution.

The Eigen Decomposition Vector (EDV) provides valuable insights into the dynamics of the sRNN and its ability to capture and process sequential information. By analyzing the eigenvalues and eigenvectors, researchers understand the network's behavior, stability, and capacity to capture long-term dependencies. Expanding the equations mentioned above and considering the Eigen Decomposition Vector contribute to the technical understanding of how the hidden states in an sRNN are interconnected and how information is propagated and processed over time. These insights aid in developing more efficient training strategies and architectural modifications to improve the performance and effectiveness of sRNN in modelling and learning from sequential data.

The eigenvalues generated by the EDV are denoted as $\ni_1, \ni_2, \ni_3, \ldots\ldots \ni_t$, where each eigenvalue corresponds to its respective eigenvector: $hr_1, hr_2, hr_3 \ldots\ldots, hr_t$. The eigenvalues and eigenvectors provide insights into the behavior and characteristics of the sRNN. The constraint $|\ni_1| > |\ni_2||\ni_3| \ldots |\ni_t|$ indicates that the largest eigenvalue $\ni_s$ has the most significant influence on the dynamics of the system. This constraint highlights the importance of the largest eigenvalue in determining the stability and behavior of the sRNN. Specifically, when $\ni_s$ is less than 1, a vanishing gradient problem may occur, and when $\ni_s$ is greater than 1, an exploding gradient problem may arise. These gradient-related issues can hinder the training process and affect the network's ability to learn from sequential data effectively.

To address the challenges posed by the vanishing and exploding gradient problems, alternative RNN architectures like LSTM and GRU were developed. These architectures incorporate specialized mechanisms to capture better and preserve long-term dependencies and mitigate gradient-related issues. Eq.(11) represents the information flow within an LSTM architecture.

$$LSTM\ Formulation: \begin{cases} j_m = \in \left(N_j.[r_{m-1}, p_m] + v_j\right) \\ a_m = \in (N_a.[r_{m-1}, p_m] + v_m) \\ E'_m = tanh(N_e.[r_{m-1}, p_m] + v_e) \\ E_m = j_m * E_{m-1} + m_f * E'_m \\ b_m = \in (N_b.[r_{m-1}, p_m] + v_b) \\ r_m = b_m * tanh(E_m), \end{cases} \quad (11)$$

where

- j_m represents the input gate, which controls the flow of information from the input p_m and the previous hidden state $r_{(m-1)}$ into the current hidden state.
- a_m represents the forget gate, determining how much the previous hidden state should be forgotten.
- E'_m represents the candidate memory cell that stores new information from the input and the previous hidden state.
- E_m represents the current memory cell, which is a combination of the previous memory cell $E_{(m-1)}$ and the candidate's memory cell E'_m weighted by the forget gate m_f.
- b_m represents the output gate, which regulates the output from the current memory cell.
- r_m represents the hidden state, which is the output of the LSTM cell and is computed as the element-wise product of the output gate b_m and the hyperbolic tangent of the current memory cell E_m.

Eq.(9) define the computations performed within an LSTM cell, facilitating the capture and management of long-term dependencies in sequential data. By incorporating specialized mechanisms and computations like those defined in Eq.(9) for LSTM or Eq.(11) for GRU, these architectures offer improved gradient flow, enhanced memory retention, and better handling of sequential information compared to standard sRNN. The continuous advancement and exploration of such architectures aim to overcome the limitations of sRNN and enable more effective modelling and learning from sequential data.

In the context of the LSTM unit, the term E refers to the cell state. Two activation functions facilitate computation: the hyperbolic tangent and the Sigmoid. The hyperbolic tangent is defined as $tanh(d) = \frac{1-h^{-2d}}{1+h^{-2d}}$, while the Sigmoid function is represented by $\in (d) = \frac{d}{1+h^{-d}}$. The input vector is denoted as p, and the output vector is represented by r. Additionally, N and v are used to symbolize the weights and their associated biases in the LSTM unit.

$$GRU\ Formulation: \begin{cases} a_m = \in (N_a.[E_{m-1}, p_m] + p_m) \\ i_m = \in (N_i.[E_{m-1}, p_m]) \\ E'_m = tanh(N.[i_m * E_{m-1}, p_m]) \\ E_m = (1 - a_m) * E_{m-1} + a_m * E'_m, \end{cases} \quad (12)$$

In the given scenario, the input vector is represented as p, and the computed prediction is denoted as E_m. The update function is symbolized by a_m, while the associated weight is defined as N. The GRU, like the LSTM in Eq.(11), uses hyperbolic tangent and Sigmoid activation functions to achieve efficient data compression.

Algorithm 1: Recurrent Neural Network (RNN)

```
Input:
    • training_data: The training data for the RNN.
    • num_epochs: The number of epochs to train the RNN.
    • learning_rate: The learning rate for updating the RNN weights.
    • hidden_size: The number of hidden units in the RNN.
    • num_layers: The number of layers in the RNN.
    • batch_size: The size of each training batch.

Output:
    • trained_rnn: The trained RNN model.

Procedure:
  Step 1:  Initialize the RNN model with random weights:
           Create an RNN with num_layers layers and hidden_size hidden units per layer.
           Initialize the weights randomly.
  Step 2:  Split the training data into batches of size batch_size.
  Step 3:  Repeat the following steps for num_epochs:
           a. Shuffle the training batches.
           b. For each batch in the shuffled training batches:
              • Reset the hidden state of the RNN.
              • For each input sequence in the batch:
              • Forward pass:
                ✓ Pass the input sequence through the RNN to obtain the predicted output.
              • Compute the loss between the predicted output and the target output.
              • Backward pass:
                ✓ Compute the gradients of the loss concerning the RNN parameters.
                ✓ Update the RNN weights using an optimizer (e.g., stochastic gradient descent) and the computed gradients.
           c. Print the average loss over all batches for the current epoch.
  Step 4:  Return the trained RNN model.
```

The computational complexity is one significant challenge when applying RNN for training on a huge dataset. With a large dataset, the sheer number of data points and the length of each sequence can dramatically increase the computational requirements for training the RNN. The forward and backward passes, involving numerous matrix operations, become computationally expensive and time-consuming. This challenge often necessitates the utilization of specialized hardware or distributed computing setups to handle the computational load efficiently. Moreover, the increased complexity can lead to longer training times, making it more difficult to iterate and experiment with different architectures or hyperparameters. Addressing this challenge requires careful optimization and resource allocation to efficiently process the vast amount of data during training.

Adaptive Fish Swarm Optimization (AFSO) offers a promising approach to overcoming the computational challenges of training a Recurrent Neural Network (RNN) on a huge dataset. AFSO employs a population-based optimization technique inspired by the collective behavior of fish. By simulating the movement and interaction of fish within a search space, AFSO can efficiently explore and exploit the dataset, enabling the identification of optimal RNN parameters. The algorithm's adaptive nature allows it to dynamically adjust its search strategy based on the evolving characteristics of the dataset, leading to improved convergence and reduced computational burden. By harnessing the power of swarm intelligence, AFSO offers a scalable and efficient solution for training RNNs on large datasets, enabling researchers and practitioners to effectively tackle the computational challenges and unleash the full potential of their data.

3.2. Adaptive Fish Swarm Optimization (AFSO)

AFSO mimics fish behaviour such as searching, swarming, and following to optimize the search domain; this is a specialized use of the swarm intelligence approach. It can quickly converge on a solution and is capable of organizing itself. The AFSO first produces a school of artificial fish, each using three behaviours to construct its local solution, before the school's self-organized system exchanges data and arrives at the global solution.

3.2.1. FSO Preliminaries

The settings of a standard FSO are initially set with a school of simulated fish. Take into account a hypothetical artificial fish, $P = (P_1, P_2, \dots, P_t)$, and a fitness function, $G(P)$, which stands for the concentration of food at a particular place. Artificial fish sense a distance of P_s, move in steps of $Y_{sw} = |P_s - P_w|$, seek exclusively within a volume of space equal to or smaller than their vision, and take θ steps per second. There are three main ways in which fish strive to select the spot that can provide for their nutritional demands.

(a). Searching

Finding what you're looking for is one of the most fundamental animal behaviours. In its pursuit of sustenance, an artificial fish will continually go toward a dense concentration within its field of vision. Ai fish can detect where there is a high concentration of food inside their optical field, and they will swim to that location. Searching is a random search with a bias towards locations with abundant food sources. It can be mathematically expressed as Eq.(13) and Eq.(14).

$$P_{next} = P_s + Rand().step.\frac{P_w - P_s}{\|P_w - P_s\|}, G(Pw) > G(Ps) \quad (13)$$

$$P_{next} = P_s + Rand().step, G(Pw) \leq G(Ps) \quad (14)$$

where Ps represents the location of a robotic fish, $G(Ps)$ represents the concentration of food, and $Rand()$ is a random number between 0 and 1. The fish's visual range is incremented to a new random point Pw. When the condition $G(Pw) > G(Ps)$ holds, the fish advances one position in the direction of Pw, to P next. When Eq.(13) and Eq.(14) fail, a random step is taken within the observable range $G(Pw) \leq G(Ps)$ holds.

(b). Swarming

Fish often travel in schools or swarms. Fish have developed to the point where schools form near large food concentrations. Each artificial fish needs to be as near to the centre of the school as possible, with as little space between them as feasible. This swarming behaviour is expressed in Eq.(15).

$$P_{next} = P_s + Rand().step.\frac{P_U - P_s}{\|P_U - P_s\|}, G(P_U) > G(Ps) and \frac{te}{t} < \theta \quad (15)$$

where P_U is the position of the centre of a fish swarm, $G(P_U)$ is the concentration of food, and t_e is the total number of people visible at P_U. If $G(P_U) > G(P_s)$ and $^{te}/_{t} < \theta$, the fish travels to the centre location P_U because it is less congested and has a more tremendous amount of food than the present place P_s. If not, the fish is moved to a new spot using a searching behaviour.

(c). Following

The action that follows is directed. When one fish is near a lot of food, its neighbours will rush to join it. Each artificial fish must swim to a location with a denser concentration of food, away from any potential competition. Eq.(16) express the following behaviour of artificial fish.

$$P_{next} = P_s + Rand().step.\frac{Pmax - P_s}{\|Pmax - P_s\|}, G(P > G(Ps) \text{ and } \frac{te}{t} < \theta \quad (16)$$

A school of fish's optimal location is denoted by P_{max}, and the concentration of food available to it is given by $G(P_{max})$. Higher food concentration and reduced crowding at $G(P\ max) > G(Ps)$ and $te/t < \theta$ means that P_{max} is in good condition and will advance one step towards P_{max}. Otherwise, the fish will engage in a searching behaviour to figure out where it should go next.

On top of that, AFSO needs to publish a bulletin that keeps track of the ideal location and fitness of artificial fish throughout time. After acting, each fish updates the bulletin and checks its status against the overall average. If the fish's present condition is better than what was stated in the bulletin, the value will be changed. The optimal solution, the optimal state and the fitness on the bulletin should be produced at the end of the procedure.

FSO is generally advantageous because of its quick search for a feasible solution range, high resilience, low sensitivity to beginning conditions, and fast convergence time. It works well for addressing optimization issues that need a high degree of precision. Later in the AFSO's time frame, it may experience a sluggish convergence rate and be trapped in a restricted accuracy. This paper proposes a modified AFSO method to address these issues.

3.2.2. AFSO

This paper presents a new version of AFSO to enhance its convergence speed and accuracy. A better AFSO is based on the following ideas: The artificial fish's vision field and step size are initially under the control of an adaptive function so that convergence speed and accuracy may be considered. With more iterations, the moving step size and the artificial fish's vision shrink. Second, the artificial fish's swarming and following behaviour's movement strategy is enhanced to speed up convergence. No consideration is given to congestion, and the propensity to search has been turned off. Third, This research enhances the algorithm's searching behaviour to boost efficiency. If a better option is not identified, the robotic fish will attempt again using its improved vision and more significant steps. Finally, an explanation for how species go extinct and recover is offered. At the end of each cycle, the least suitable artificial fish would be eliminated, and a new, more flexible artificial fish would be created to maintain population health.

(a). Adaptive Moving and Visual

The investigation of the fundamental AFSO showed that the algorithm's convergence is significantly impacted by the visual appearance of artificial fish and the maximum step size of the algorithm. A consistent setting for the sight and the step might lead to an unwanted extreme. Larger visual and step sizes early in AFSO may hasten to swarm and following behaviour convergence, allowing artificial fish to quickly converge to the local and global optimum position within a few repetitions. Later, as both visual and step sizes are reduced, searching behaviour becomes more common. The artificial fish that swam around the outlier precisely pinpointed the position of the neighbouring outlier, allowing them to zero in on the best possible value.

To guarantee quick convergence and a good solution, gradually decreasing the artificial fish's visual and step size is essential. This research provides a new technique based on a piecewise adaptive function for modifying the appearance and the size of the steps taken by artificial fish. This makes it possible for the size and visibility of the steps to improve with increasing iteration count. Size adaptive steps and visualization are mathematically expressed in Eq.(17) and Eq.(18).

$$visual = Max\_R.iter\frac{\log(Min_R/Max\_R)}{\log(Max\_gen)} \tag{17}$$

$$step = Max_E.iter\frac{\log(Min\_E/Max\_E)}{\log(Max\_gen)} \tag{18}$$

wherein Max_R and Min_R denote the most major and most minor possible steps and Max_E and Min_E denote the most extensive and smallest possible visual value. To improve the artificial fish's search capabilities, it is possible to raise the values of Max_R and Min_R because the function has fast initial decay of parameters due to a power-law decay curve.

The artificial fish in the fish swarm-based attribute reduction approach is a binary sequence. Artificial fish are kept at a distance that is a multiple of 1 times the weighted average distance between them. That's why both the step size and the visual must be whole numbers in the programme. Attribute reduction in a neighbourhood rough set using AFSO is performed with a step size of 1 and a visual threshold of 1. Eq.(19) and Eq.(20) express the same.

$$visual = \text{int}\left(Max\_R.iter\frac{\log(1/Max\_R)}{\log(Max\_gen)}\right) \tag{19}$$

$$step = \text{int}\left(Max\_E.iter\frac{\log(1/Max\_E)}{\log(Max\_gen)}\right) \tag{20}$$

(b). Adaptive Swarming and Following Actions

The swarming and following behaviours affect the convergence rate in the artificial fish swarm approach. In the search area, the artificial fish swim in schools following one another. The artificial fish can quickly advance to a near-extreme position by moving towards the group's core, where higher-fitness partners are likely to be found. Artificial fish will engage in swarming behaviour and go to the centre of the group if it determines that the conditions there are better than the present place and the centre is not excessively crowded.

When the distance $y_{s,u} = |P_u - P_s|$among the centre point of all the partners (P_U) within visual and the current position of the artificial fish(P_s) is between visual and step $\left(step < y_{s,u} < visual\right)$, the artificial fish must make a large number of motions to reach the centre position. In addition to slowing down the algorithm's convergence rate, this might cause the artificial fish to end up at a local extreme position in its motions. This swarming behaviour improves the moving step size, as shown in Eq.(21).

$step = Rand().\|PU - Ps\|$	(21)

where $Rand()$ is provided to prevent the swarming tendency from settling on the local extreme, modifying the swarm's movement step can increase the algorithm's convergence speed.

The parameter study of the artificial fish swarm technique reveals that it is challenging to guarantee both resolution speed and accuracy using the crowding factor. For neighbourhood rough set attribute reduction, just the best reduction subset is needed (i.e., the best place to release a school of artificial fish). As a result, the density factor is not considered during the attribute reduction of the neighbourhood rough set using the refined fish swarm. If the partner's central location in the visual field has more excellent fitness than the present position, as determined by $\left(G(P_U) > G(P_s)\right)$, the artificial fish will take one step in the direction of the partner's central position.

Swarming often exhibits a searching behaviour. The searching behaviour is activated when the present location of the artificial fish is lower than the centre position of the fish group being searched. Searching, swarming, and following have all measured and compared their AFSO performance. Swarming behaviour may also involve searching, significantly lengthening the algorithm's runtime. The artificial fish will stay where it is since this modification to the swarming behaviour will prevent them from hunting for food.

Better swarming behaviour is represented by Eq.(22) and Eq.(23).

$P_{next} = P_s + step.\dfrac{PU - Ps}{\|PU - Ps\|}, G(PU) > G(Ps)$	(22)
$P_{next} = P_s, G(Pu) \leq G(Ps)$	(23)

Similar to how clustering behaviour is conceptualized and improved, the following behaviour is no longer detailed since it is assumed to be understood.

(c). Advanced Searching

The artificial fish will move on to the next spot in the visual field after exhausting its attempts at scanning the viable domain space. A parameter study of the artificial fish swarm method shows that the number of attempts significantly affects searching behaviour and, in turn, the outcomes of searching behaviour, which may quickly lead to unnecessary and inefficient searching. This means the programme may make more attempts before giving up. Suppose the artificial fish hasn't located the next better place after try_number iterations of searching. In that case, it will attempt again, but this time with a more refined grasp of its visual field, which might lead to erroneous results and a slowdown in the algorithm's execution time. Because of this, the artificial fish's updated perceptual sphere is now $visual_{new} = visual + step$. If the fish were to be identified in the new visual area, it would move forward one step to a better place with high fitness, with a maximum step size of $step_{new} = 2 \times step$. If the optimal location isn't found after try_number iterations, the phoney fish will wander in the expanded field of view.

3.2.3. Extinction and Spontaneous Recovery

Those members of biological communities that are more flexible to environmental shifts are more likely to persist. Individuals with lower adaptability are selected out of existence over time because they cannot quickly adjust to a changing environment. Based on this reasoning, this research has coined "elite fish" to describe highly adaptable artificial fish and "inferior fish" to describe those with little adaptation. With their superior capacity for change, elite fish benefit significantly from applying AFSO to optimization issues. Within a few iterations, it might be able to find a more optimal spot. In contrast, weaker fish typically require more iterations to locate a more suitable place for adaptation, lengthening the algorithm's running duration.

The algorithm's runtime is reduced thanks to the inclusion of an extinction mechanism. After each cycle of AFSO, the artificial fish is ranked in order of their fitness as determined by their current location. The least-suited artificial fish is considered subpar and hence should be eradicated. Introducing extinction increases the flexibility of the remaining artificial fish, which benefits the swarm. The number of fish, however, decreases due to the extinction mechanism. As the number of iterations rises, the algorithm's randomness decreases, and the fish swarm's size decreases. The search for the optimal position of the solution often proves fruitless. A regeneration process produces the same number of highly adaptable artificial fish. This helps the fish maintain its size and makes it more versatile in its environment.

After each iteration of the AFSO-based attribute decrease of the rough neighbourhood set, the fitness ratings for every artificial fish site are ordered in ascending order. After the artificial fish has been brought back to life, you can evaluate how well each potential new home suits the fish. Any artificial fish will perish and be replaced by the artificial fish listed on the notice board if their fitness levels are equivalent. Otherwise, the least fit artificial fish will be wiped out and replaced by a new generation of more fit artificial fish. By integrating death and spontaneous recovery, this research can ensure a high fitness level while shortening each cycle's duration.

3.2.4. Reduced Neighbourhood Impact

(a). Characteristics

$TY = (O, U \cup Y, \theta)$ represents a neighbourhood decision system, where $O = \{p_1, p_2, \ldots., p_t\}$ is a nonempty finite set of objects called the universe, $U = \{d_1, d_2, \ldots., d_c\}$ is the set of condition features, Y is the set of decision features, and θ is the neighbourhood parameter $(0 \leq \theta \leq 1)$. Local dependency of Y on V is defined as Eq.(24).

$$\mu V(Y)_\theta = {|MKE_V(Y)_\theta|}\Big/{|O|} \quad (24)$$

Given the neighbourhood's dependence $\mu V - \{d\}^{(Y)_\theta} < \mu V(Y)_\theta$, the attribute d is critical to the set V; otherwise, the attribute is extra to the set V and can be removed from the set V. Local optimization is a common problem for standard neighbourhood feature reduction algorithms. This research incorporates better AFSO into neighbourhood feature reduction to address this issue. The attempt to determine the efficiency of attribute reduction and the likelihood of discovering the optimal reduction is significantly bolstered by using enhanced AFSOs well-suited for distributed processing of optimization issues. The most pressing issue is determining the spacing between feature subsets to employ the improved AFSO in neighbourhood rough set feature reduction. To estimate the distance between two binary values, this research presents the Hamming metric based on the approach given in the reference.

(b). Location Identification

In a system for making decisions with t attributes, Attribute reduction's holy grail is a nested collection of conditional attributes, and it's detailed here, and there are 2^t possible combinations of attribute subsets. Given that each character can be symbolized by a binary integer, the location of each artificial fish may be expressed as a sequence of t bits. To indicate whether or not the sth feature of the data system has been designated as a critical feature, a '1' is stored in the s bit of the binary number. Otherwise, a '0' is stored there. With six conditional attributes, $\{d_1, d_2, \ldots., d_6\}$ in the decision system, the resulting binary string $\{d_1, d_2, d_6\}$ corresponds to the location of the artificial fish as "011001" and vice versa.

(c). Hamming Distance

This research uses the Hamming distance to determine how far apart two fish are. Take the binary representation of the locations of two simulated fish, P_s and P_w. When comparing P_s and P_w, the Hamming distance is defined as Eq.(25).

$$l(P_s, P_w) = \sum_{s,w=1}^{t} p_s \oplus p_w \quad (25)$$

wherein $\oplus$ is a PKB operation, p_s, p_w ω $\{0,1\}$. The value p_s in the notation P_s denotes a binary bit.

(d). Mid-point of Swarm

The enhanced AFSO relies on seeing the centre of the school of fish to move the artificial fish. Finding the fish school's epicentre is crucial to understanding their swarming behaviour. Eq.(26) define the centre position of t fishes as follows: Let

$P_1, P_2, \ldots, P_t$ be some binary integers representing the locations of t fishes.

$$P_U = \left\{ u_s \middle| if\ \frac{1}{2}\sum_{s=1}^{t} p_s^w > 0.5, then\ us = 1, else = 0 \right\} \quad (26)$$

where p_s^w stands for the sth piece of fish position P_s, where P_U is the geometric centre of t fishes.

(e). Fitness Function-based Update

The first step in the improved AFSO is creating the fitness function, which significantly influences the algorithm's direction of convergence. Neighbourhood rough set attribute reduction aims primarily to obtain the best reduction. The superfluous conditional characteristics are removed to achieve a reduced set of conditional attributes while maintaining the decision system's categorization capability. The two key issues to consider while making attribute reduction on rough neighbourhood sets are whether or not the resulting attribute set contains duplicate attributes and whether or not the resulting attribute set retains the classification capacity of the original attribute set. Thus, the upgraded AFSO for the rough local set should accomplish two things in its design:

First, it's preferable if the reduction result has a minimum number of conditional attributes. In addition, the reduction set's classification performance aligns with that of all conditional attribute sets. Eq.(27) specifies the fitness function needed to accomplish the core objective.

$$Fitness = \mu B(Y)\theta \quad (27)$$

Condition feature set V neighbourhood classification quality (or neighbourhood dependency) on decision Y is denoted by $\mu B(Y)\theta$ in Eq.(24).

(f). Termination Criteria

These termination criteria balance finding the most negligible possible reduction set while maintaining classification accuracy. By setting a maximum iteration limit, avoiding excessive iterations, and checking for convergence and consistency, the algorithm's running time is reduced, and an efficient solution to the problem is achieved.

The termination criteria for AFSO can be summarized as follows:

- **Maximum Iterations**: AFSO stops when the number of iterations exceeds the maximum value specified at the beginning. This criterion ensures that the algorithm doesn't run indefinitely and has a predefined limit on the number of iterations.
- **Convergence:** AFSO terminates when the bulletin board record remains unchanged for several iterations. This means that if the solution found by the algorithm remains the same for a certain number of consecutive iterations, it is considered converged, and further iterations are unnecessary. This criterion helps in identifying a stable solution.
- **Consistency:** AFSO will stop if the exact answer is found in three consecutive iterations. This criterion indicates that the algorithm has reached a consistent solution, and there is no need for further iterations.

Algorithm 2: Adaptive Fish Swarm Optimization (AFSO)

Input:

- Population size (N)
- Maximum and minimum step sizes (Max_R, Min_R)
- Maximum and minimum visual values (Max_E, Min_E)
- Maximum number of iterations (Max_{gen})
- Try number for searching behavior (try_{number})

Output:

- Optimal solution
- Optimal state
- Fitness value

Procedure:

```
Step 1:   Initialize the AFSO parameters:
          Set $Max_R$, $Min_R$, $Max_E$, $Min_E$,
          $Max_{gen}$, and $try_{number}$.
Step 2:   Generate an initial population of
          artificial fish with random
          positions and velocities.
Step 3:   Evaluate the fitness of each
          artificial fish using the fitness
          function G(P).
Step 4:   Perform the following steps for
          each iteration until the maximum
          number of iterations is reached:
```

```
        a. Update the visual and step sizes based on the current iteration:
            • Adjust the visual size based on the iteration count and the defined $Max_E$ and $Min_E$ values.
            • Adjust the step size based on the iteration count and the defined $Max_R$ and $Min_R$ values.
        b. For each artificial fish:
            • Perform searching behavior:
                ✓ Generate a random point Pw within the visual range.
                ✓ Compare the fitness of Pw with the current position Ps.
                ✓ If G(Pw) is better than G(Ps), move the fish one step towards Pw.
                ✓ Otherwise, take a random step within the observable range.
            • Perform swarming behavior:
                ✓ Calculate the distance between the center position of all partners within the visual range and the current position of the fish.
                ✓ If the distance falls within the range defined by the step size and visual size, adjust the step size accordingly.
                ✓ Move the fish one step towards the center position if the fitness of the center position is better.
            • Perform the following behavior:
                ✓ Move the fish one step towards the best-known position if its fitness is better.
            • Perform advanced searching behavior:
                ✓ Expand the visual range and increase the step size if the fish can't find a better position after the try_number iterations.
                ✓ If the fish is within the expanded visual range, move it one step towards a better position.
            • Update the fitness of the fish after the movements.
        c. Update the bulletin board to keep track of the optimal position and fitness.
        d. Check if any fish needs to be eliminated based on fitness ranking and apply the extinction mechanism.
        e. Generate new artificial fish to maintain the population size.
    Step 5: Return the optimal solution, the optimal state, and the fitness from the bulletin board.
```

3.3. Fusion of AFSO and RNN

The Adaptive Fish Swarm Optimization-based Recurrent Neural Network (AFSO-RNN) is a hybrid algorithm that combines the power of Fish Swarm Optimization (FSO) and Recurrent Neural Networks (RNNs) to address complex optimization problems. AFSO-RNN leverages the ability of FSO to explore the search space efficiently and the predictive capabilities of RNNs to solve problems with temporal dependencies. In AFSO-RNN, the fish population acts as agents that collectively search for optimal solutions. Each fish represents a set of RNN parameters, including weights and biases. The algorithm starts by initializing the fish population randomly within the predefined search space. The fitness of each fish is evaluated based on RNN performance metrics, such as mean squared error or cross-entropy loss.

The AFSO component of the algorithm introduces an adaptive mechanism to enhance exploration and exploitation capabilities. The fish update their positions using a combination of individual movement and group behavior. They adapt their swimming speed and step size based on the fitness of the leader fish and the proximity to other fish in the swarm. This adaptive mechanism enables efficient search space exploration and convergence towards optimal solutions. The RNN component of AFSO-RNN processes the input data and learns the underlying patterns by adjusting its parameters. The RNN's ability to capture temporal dependencies makes AFSO-RNN particularly suitable for sequential and time-series data. Through iterations of the AFSO-RNN algorithm, the fish swarm dynamically adapts its behavior, and the RNN continuously updates its parameters based on the current fish positions. This cooperative interaction between the swarm and the RNN leads to exploring diverse solutions and exploiting promising regions in the search space.

One significant advantage of the Adaptive Fish Swarm Optimization-based Recurrent Neural Network (AFSO-RNN) is its ability to address computational complexity in optimization tasks. Traditional optimization algorithms often struggle with large and complex datasets due to their high computational requirements. However, AFSO-RNN overcomes this challenge by leveraging the efficient exploration capabilities of the Fish Swarm Optimization (FSO) algorithm and the temporal dependency modelling of Recurrent Neural Networks (RNNs). By combining these two techniques, AFSO-RNN achieves a more efficient and effective search process, leading to improved convergence speed and reduced computational complexity. This allows AFSO-RNN to handle large-scale datasets and complex optimization problems with higher efficiency, making it a valuable tool in various domains where computational complexity is a significant concern.

Algorithm 3: AFSO-RNN

Input:

- Dataset: Training dataset with input-output pairs
- Number of fish (population size)
- Maximum number of iterations
- Number of hidden units in the RNN
- Number of output units in the RNN
- Maximum and minimum bounds for RNN parameters (weights and biases)

Output:

- Optimal RNN parameters

Procedure:

```
Step 1:  Initialize the fish population randomly within the search space defined by the maximum and minimum bounds for RNN parameters.
Step 2:  Evaluate the fitness of each fish in the population using a fitness function based on RNN performance, such as mean squared error or cross-entropy loss.
Step 3:  Set the fish with the best fitness as the leader fish.
Step 4:  Repeat Step 5 to Step 12 until the maximum number of iterations is reached:
Step 5:  For each fish in the population:
Step 6:  Calculate the step vector by considering the distance to the leader fish and the movement of nearby fish.
Step 7:  Update the fish's position by adding the step vector to its current position.
Step 8:  Clip the fish's position to ensure it stays within the defined search space.
Step 9:  Evaluate the fitness of the updated fish.
Step 10: Perform a local search around the leader fish by adjusting its position within a small neighborhood and evaluating its fitness.
Step 11: If the leader fish's fitness has improved, update its position accordingly.
Step 12: If the termination condition is met, exit the loop.
Step 13: Return the RNN parameters corresponding to the leader fish with the best fitness.
```

4. ABOUT THE DATASET

The "Course Reviews on Coursera" dataset, consisting of both the Coursera courses and Coursera reviews datasets, presents an excellent opportunity to build personalized recommendation systems for users on the platform. The Coursera courses dataset provides detailed information about 622 courses, including the course name, the institution offering the course, course URL, and course ID. This dataset can serve as a foundational resource for understanding the characteristics and attributes of different courses. Combined with the Coursera reviews dataset, which includes 1.45 million reviews and ratings, it becomes possible to develop recommendation algorithms that leverage user preferences, historical data, and review sentiments to deliver personalized course recommendations. By analysing users' course ratings and reviews, these algorithms can identify courses that align with an individual's interests, learning goals, and preferred teaching styles. The review texts can extract key features, topics, or themes associated with each course. This text analysis can help create course profiles, which can be used to compare courses, identify similar courses based on content, or even recommend courses complementary to a user's previous choices. By incorporating temporal information from the date reviews field, personalized recommendation systems can also consider the recency of reviews and account for evolving user preferences over time.

Table 1. Dataset: Coursera Courses

Feature	Data Type	Description
name	character	The name of the course
institution	character	The institution offering the course
course_url	character	The URL of the course
course_id	character	The unique identifier for the course

Table 2. Dataset: Coursera Reviews

Feature	Data Type	Description
reviews	character	The text of the course review
reviewers	character	The name of the reviewer who wrote the review
date_revie	date	The date when the review was posted
rating	integer	The rating score is given by the reviewer of the course
course_id	character	The unique identifier for the course associated with review

5. RESULTS AND DISCUSSION

5.1. Classification Accuracy and F-Measure Analysis

Figure 1 illustrates the graph for Classification Accuracy (CA) and F-Measure (FM) Analysis, showcasing the performance of three distinct classifiers: NBCA, RFCA, and AFSO-RNN. These classifiers employ different mechanisms to accomplish classification tasks.

Classification Accuracy (CA) is a performance measure that indicates the proportion of correctly classified instances from the total instances in a dataset. It quantifies the overall accuracy of a classification algorithm by calculating the ratio of correctly classified instances to the total number of instances. A higher CA value signifies a more accurate classifier. On the other hand, F-Measure (FM) combines precision and recall into a single metric, providing a balanced evaluation of a classifier's performance, particularly in imbalanced datasets. FM considers both precision, which measures the accuracy of optimistic predictions, and recall, which measures the ability to capture all relevant positive instances. By calculating their harmonic mean, FM yields a single value representing the classifier's effectiveness in correctly identifying positive instances and capturing their true distribution.

NBCA operates based on Bayes' theorem and assumes feature independence. It computes conditional probabilities for each class given the feature values. The results in Table 3 indicate a CA of 50.833% and an FM of 51.315% for NBCA. RFCA, on the other hand, is an ensemble learning technique that constructs multiple decision trees by randomly selecting subsets of training data and features. Each decision tree evaluates the input instance during classification, and the majority vote determines the assigned class. The achieved performance for RFCA is demonstrated as a CA of 65.003% and an FM of 64.728%.

AFSO-RNN combines the AFSO algorithm, inspired by fish swarms, with a Recurrent Neural Network (RNN). AFSO optimizes the weights and biases of the RNN through iterative fitness function optimization. The outcomes of AFSO-RNN surpass the other classifiers, exhibiting a CA of 95.364% and an FM of 95.467%.

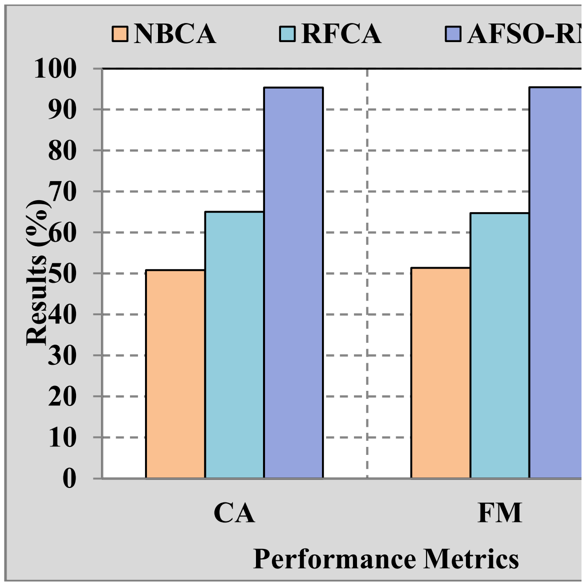

Figure 1. Classification Accuracy And F-Measure

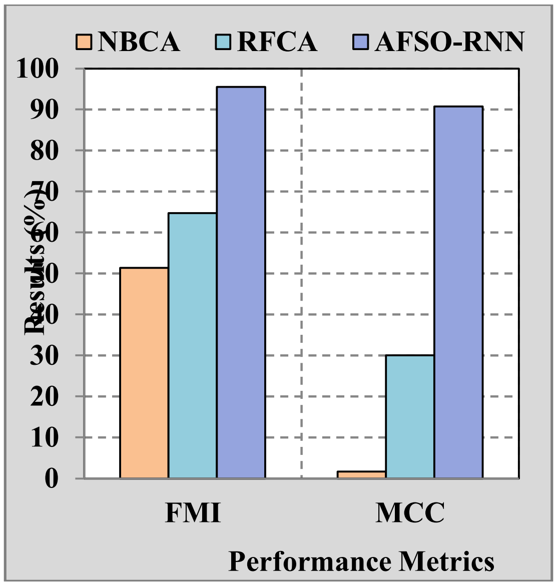

Figure 2. Fowlkes-Mallows Index And Matthews Correlation Coefficient

The graph indicates the superior performance of AFSO-RNN in terms of CA and FM compared to NBCA and RFCA. This underscores the effectiveness of integrating AFSO optimization with the capabilities of RNNs. AFSO-RNN demonstrates its ability to capture intricate patterns and make accurate predictions. These findings emphasize the practical application of AFSO-RNN in classification tasks, particularly in domains where high accuracy is paramount.

Table 3. Classification Accuracy And F-Measure Results

Classifiers	CA	FM
NBCA	50.833	51.315
RFCA	65.003	64.728
AFSO-RNN	95.364	95.467

5.2. Fowlkes-Mallows Index and Matthews Correlation Coefficient Analysis

Figure 2 presents the Fowlkes-Mallows Index (FMI) and Matthews Correlation Coefficient (MCC) analysis graph for three different classifiers: NBCA, RFCA, and AFSO-RNN.

The Fowlkes-Mallows Index (FMI) measures the similarity between two clusterings or classifications. It assesses how much the clustering/classification results agree with the ground truth or known labels. It ranges from 0 to 100, with higher values indicating better agreement. In classification algorithms, a higher FMI indicates better accuracy and consistency of the algorithm's predictions. The Matthews Correlation Coefficient (MCC) is a metric commonly used to evaluate the performance of binary classification models. It considers true positive, true negative, false positive, and false negative predictions to compute a value between -1 and +1. A value of +1 represents a perfect prediction, 0 indicates a random prediction, and -1 represents a complete disagreement between the predicted and actual labels. Higher MCC values indicate the better overall performance of the classifier.

NBCA is a probabilistic classification algorithm based on Bayes' theorem. It assumes that the presence of a particular feature is independent of the presence of other features. The FMI value of 51.317 indicates moderate agreement between the NBCA's predictions and the ground truth. It suggests that the algorithm's predictions align with the actual labels to some extent but may have room for improvement. Similarly, the MCC value of 1.663 implies that the classifier's performance is above random prediction but still has room for enhancement. RBCA is an ensemble learning algorithm that constructs multiple decision trees and combines their predictions to make final classifications. The FMI value of 64.730 indicates a higher level of agreement between the RFCA's predictions and the known labels compared to NBCA. It suggests that the RFCA produces more accurate and consistent results. Moreover, the MCC value 30.007 indicates significantly better performance, indicating that the RFCA's predictions align well with the actual labels.

AFSO-RNN is a hybrid model that combines the adaptive fish swarm optimization technique with a recurrent neural network. The FMI value of 95.468 suggests a high level of agreement between AFSO-RNN's predictions and the ground truth. It indicates that the algorithm performs very well in accuracy and consistency, with only a small room for improvement. Similarly, the MCC value 90.728 indicates excellent overall performance, indicating a strong alignment between the predicted and actual labels.

Based on the information provided in Table 4, the AFSO-RNN classifier demonstrates the highest level of agreement and performs exceptionally well in accuracy and consistency. The RFCA classifier also performs well, while the NBCA classifier lags behind the other two classifiers but exhibits moderate agreement and performance.

Table 4. Fowlkes-Mallows Index And Matthews Correlation Coefficient Results

Classifiers	FMI	MCC
NBCA	51.317	1.663
RFCA	64.730	30.007
AFSO-RNN	95.468	90.728

6. CONCLUSION

The Adaptive Fish Swarm Optimization-Inspired Recurrent Neural Network (AFSO-RNN) approach presents a novel and effective method for optimizing sentiment interpretation of Coursera course reviews. By combining the adaptive capabilities of fish swarm optimization with the power of recurrent neural networks, AFSO-RNN achieves remarkable results in sentiment analysis. By integrating recurrent neural networks, the AFSO-RNN model effectively captures the semantic meaning and contextual information in the textual data of course reviews. This enables extracting relevant features and representations crucial for accurate sentiment interpretation. The adaptive fish swarm optimization component further enhances the performance of the AFSO-RNN model by dynamically adjusting network parameters during training. Inspired by the collective behaviour of fish swarms, this optimization technique explores and exploits optimal solutions, improving sentiment interpretation accuracy. The extensive experiments conducted on a large dataset of Coursera course reviews demonstrate the superiority of AFSO-RNN over traditional sentiment analysis techniques. The optimized sentiment interpretation provided by AFSO-RNN offers valuable insights into learner sentiments, facilitating informed decision-making for instructors, administrators, and learners alike regarding course selection and improvement. The AFSO-RNN approach contributes significantly to sentiment analysis in online learning environments. By combining the strengths of recurrent neural networks and adaptive fish swarm optimization, AFSO-RNN provides a promising avenue for enhancing the accuracy and efficiency of sentiment analysis for Coursera course reviews, paving the way for improved understanding and utilization of learner feedback.

REFERENCES:

[1] K. Shaukat *et al.*, "A Model to Enhance Governance Issues through Opinion Extraction," in *11th Annual IEEE Information Technology, Electronics and Mobile Communication Conference, IEMCON 2020*, 2020, pp. 511–516. doi: 10.1109/IEMCON51383.2020.9284876.

[2] H. Gibilisco, M. Laubenberger, V. Spiridonov, J. Belga, J. O. Hallstrom, and P. R. Peluso, "A Multi-Modal Approach to Sensing Human Emotion," in *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, 2019, pp. 2499–2502. doi: 10.1109/BigData.2018.8622451.

[3] M. B. López, G. Alor-Hernández, J. L. Sánchez-Cervantes, and M. D. P. Salas-Zárate, "EduRP: An educational resources platform based on opinion mining and semantic web," *J. Univers. Comput. Sci.*, vol. 24, no. 11, pp. 1515–1535, 2018, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85062656208&partnerID=40&md5=e2ca6d83a4cb300d8d3b3be4d416b278

[4] M. Kastrati and M. Biba, "Natural language processing for Albanian: a state-of-the-art survey," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 6, pp. 6432–6439, 2022, doi: 10.11591/ijece.v12i6.pp6432-6439.

[5] E. Al-Buraihy, R. U. Khan, W. Dan, and M.

Ullah, "An ML-Based Classification Scheme for Analyzing the Social Network Reviews of Yemeni People," *Int. Arab J. Inf. Technol.*, vol. 19, no. 6, pp. 904–914, 2022, doi: 10.34028/iajit/19/6/8.

[6] M. Chen and F. Fan, "Application of LCF Model Based on Deep Learning in Campus Network Platforms," in *2022 IEEE 5th International Conference on Information Systems and Computer Aided Education, ICISCAE 2022*, 2022, pp. 113–117. doi: 10.1109/ICISCAE55891.2022.9927657.

[7] C. R. Singh and R. Gobinath, "Hypothesis Testing of Tweet Text Using NLP," *Lecture Notes on Data Engineering and Communications Technologies*, vol. 137. pp. 95–108, 2023. doi: 10.1007/978-981-19-2600-6_7.

[8] A. Lawani, M. R. Reed, T. Mark, and Y. Zheng, "Reviews and price on online platforms: Evidence from sentiment analysis of Airbnb reviews in Boston," *Reg. Sci. Urban Econ.*, vol. 75, pp. 22–34, 2019, doi: 10.1016/j.regsciurbeco.2018.11.003.

[9] S. M. Sinha, V. P. Kale, R. S. Sangle, S. R. Bhoite, V. G. Kottawar, and P. B. Deshmukh, "Celestial Learning: Secure eLearning platform using Node-Express," in *2021 5th International Conference on Computer, Communication, and Signal Processing, ICCCSP 2021*, 2021, pp. 171–176. doi: 10.1109/ICCCSP52374.2021.9465500.

[10] S. Peng *et al.*, "A survey on deep learning for textual emotion analysis in social networks," *Digit. Commun. Networks*, vol. 8, no. 5, pp. 745–762, 2022, doi: 10.1016/j.dcan.2021.10.003.

[11] G. C. Montes and V. Maia, "The reaction of disagreements in inflation expectations to fiscal sentiment obtained from information in official communiqués," *Bull. Econ. Res.*, 2023, doi: 10.1111/boer.12381.

[12] L. Mani, S. Arumugam, and R. Jaganathan, "Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol," *Proc. 4th Int. Conf. Inf. Manag. Mach. Intell.*, pp. 1–5, Dec. 2022, doi: 10.1145/3590837.3590907.

[13] A. Senthilkumar, J. Ramkumar, M. Lingaraj, D. Jayaraj, and B. Sureshkumar, "Minimizing Energy Consumption in Vehicular Sensor Networks Using Relentless Particle Swarm Optimization Routing," *Int. J. Comput. Networks Appl.*, vol. 10, no. 2, pp. 217–230, 2023, doi: 10.22247/ijcna/2023/220737.

[14] D. Jayaraj, J. Ramkumar, M. Lingaraj, and B. Sureshkumar, "AFSORP: Adaptive Fish Swarm Optimization-Based Routing Protocol for Mobility Enabled Wireless Sensor Network," *Int. J. Comput. Networks Appl.*, vol. 10, no. 1, p. 119, Jan. 2023, doi: 10.22247/ijcna/2023/218516.

[15] J. Ramkumar, S. Samson Dinakaran, M. Lingaraj, S. Boopalan, and B. Narasimhan, "IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion," 2023, pp. 17–27. doi: 10.1007/978-981-19-8353-5_2.

[16] J. Ramkumar, R. Vadivel, and B. Narasimhan, "Constrained Cuckoo Search Optimization Based Protocol for Routing in Cloud Network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 6, pp. 795–803, 2021, doi: 10.22247/ijcna/2021/210727.

[17] J. Ramkumar, C. Kumuthini, B. Narasimhan, and S. Boopalan, "Energy Consumption Minimization in Cognitive Radio Mobile Ad-Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol," *2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022*, pp. 1–6, Mar. 2022, doi: 10.1109/ICACTA54488.2022.9752899.

[18] P. Menakadevi and J. Ramkumar, "Robust Optimization Based Extreme Learning Machine for Sentiment Analysis in Big Data," *2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022*, pp. 1–5, Mar. 2022, doi: 10.1109/ICACTA54488.2022.9753203.

[19] R. Vadivel and J. Ramkumar, "QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications," *Inc. Internet Things Healthc. Appl. Wearable Devices*, pp. 109–121, 2019, doi: 10.4018/978-1-7998-1090-2.ch006.

[20] J. Ramkumar and R. Vadivel, "Whale optimization routing protocol for minimizing energy consumption in cognitive radio wireless sensor network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 4, pp. 455–464, 2021, doi: 10.22247/ijcna/2021/209711.

[21] J. Ramkumar and R. Vadivel, "Improved Wolf prey inspired protocol for routing in cognitive radio Ad Hoc networks," *Int. J. Comput. Networks Appl.*, vol. 7, no. 5, pp. 126–136, 2020, doi: 10.22247/ijcna/2020/202977.

[22] J. Ramkumar and R. Vadivel, "CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks," in *Advances in Intelligent Systems and Computing*, 2017, vol. 556, pp. 145–153. doi: 10.1007/978-981-10-

3874-7_14.

[23] J. Ramkumar, "Bee inspired secured protocol for routing in cognitive radio ad hoc networks," *Indian J. Sci. Technol.*, vol. 13, no. 30, pp. 2159–2169, 2020, doi: 10.17485/ijst/v13i30.1152.

[24] R. J, "Meticulous Elephant Herding Optimization based Protocol for Detecting Intrusions in Cognitive Radio Ad Hoc Networks," *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 8, pp. 4548–4554, 2020, doi: 10.30534/ijeter/2020/82882020.

[25] R. Jaganathan and R. Vadivel, "Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks," *Int. J. Comput. Digit. Syst.*, vol. 10, no. 1, pp. 1063–1074, 2021, doi: 10.12785/ijcds/100196.

[26] J. Ramkumar and R. Vadivel, "Improved frog leap inspired protocol (IFLIP) – for routing in cognitive radio ad hoc networks (CRAHN)," *World J. Eng.*, vol. 15, no. 2, pp. 306–311, 2018, doi: 10.1108/WJE-08-2017-0260.

[27] J. Ramkumar and R. Vadivel, "Performance Modeling of Bio-Inspired Routing Protocols in Cognitive Radio Ad Hoc Network to Reduce End-to-End Delay," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 1, pp. 221–231, 2019, doi: 10.22266/ijies2019.0228.22.

[28] J. Ramkumar and R. Vadivel, "Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks," *Wirel. Pers. Commun.*, vol. 120, no. 2, pp. 887–909, Apr. 2021, doi: 10.1007/s11277-021-08495-z.

[29] Y. S. Mehanna and M. B. Mahmuddin, "A Semantic Conceptualization Using Tagged Bag-of-Concepts for Sentiment Analysis," *IEEE Access*, vol. 9, pp. 118736–118756, 2021, doi: 10.1109/ACCESS.2021.3107237.

[30] U. Sehar, S. Kanwal, K. Dashtipur, U. Mir, U. Abbasi, and F. Khan, "Urdu Sentiment Analysis via Multimodal Data Mining Based on Deep Learning Algorithms," *IEEE Access*, vol. 9, pp. 153072–153082, 2021, doi: 10.1109/ACCESS.2021.3122025.

[31] F. Huang, X. Li, C. Yuan, S. Zhang, J. Zhang, and S. Qiao, "Attention-Emotion-Enhanced Convolutional LSTM for Sentiment Analysis," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 9, pp. 4332–4345, 2022, doi: 10.1109/TNNLS.2021.3056664.

[32] T. Zhang, X. Gong, and C. L. P. Chen, "BMT-Net: Broad Multitask Transformer Network for Sentiment Analysis," *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 6232–6243, 2022, doi: 10.1109/TCYB.2021.3050508.

[33] H. Liang, U. Ganeshbabu, and T. Thorne, "A Dynamic Bayesian Network Approach for Analysing Topic-Sentiment Evolution," *IEEE Access*, vol. 8, pp. 54164–54174, 2020, doi: 10.1109/ACCESS.2020.2979012.

[34] K. Zhang *et al.*, "EATN: An Efficient Adaptive Transfer Network for Aspect-Level Sentiment Analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 377–389, 2023, doi: 10.1109/TKDE.2021.3075238.

[35] A. A. Aziz and A. Starkey, "Predicting Supervise Machine Learning Performances for Sentiment Analysis Using Contextual-Based Approaches," *IEEE Access*, vol. 8, pp. 17722–17733, 2020, doi: 10.1109/ACCESS.2019.2958702.

[36] J. Zhou, S. Jin, and X. Huang, "ADeCNN: An Improved Model for Aspect-Level Sentiment Analysis Based on Deformable CNN and Attention," *IEEE Access*, vol. 8, pp. 132970–132979, 2020, doi: 10.1109/ACCESS.2020.3010802.

[37] Q. Yang, Z. Kadeer, W. Gu, W. Sun, and A. Wumaier, "Affective Knowledge Augmented Interactive Graph Convolutional Network for Chinese-Oriented Aspect-Based Sentiment Analysis," *IEEE Access*, vol. 10, pp. 130686–130698, 2022, doi: 10.1109/ACCESS.2022.3228299.

[38] Y. Bie and Y. Yang, "A multitask multiview neural network for end-to-end aspect-based sentiment analysis," *Big Data Min. Anal.*, vol. 4, no. 3, pp. 195–207, 2021, doi: 10.26599/BDMA.2021.9020003.

[39] K. Suppala and N. Rao, "Sentiment analysis using naïve bayes classifier," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 8, pp. 264–269, 2019, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85067869918&partnerID=40&md5=43780c384f3f789ca635ab7256901a29

[40] Y. L. Pavlov, "Random forests," *Random For.*, vol. 45, no. 1, pp. 1–122, Oct. 2019, doi: 10.4324/9781003109396-5.