EMPOWERING SENTIMENT ANALYSIS OF COURSERA COURSE REVIEWS WITH SOPHISTICATED ARTIFICIAL BEE COLONY-INSPIRED DEEP Q-NETWORKS (SABC-DQN)

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 paper presents SABC-DQN (Sophisticated Artificial Bee Colony Inspired Deep Q-Networks), a novel approach to enhance the sentiment analysis of Coursera course reviews. Sentiment analysis is crucial for understanding learner opinions, but existing methods struggle to capture the nuanced sentiments expressed in textual data accurately. SABC-DQN combines the intelligent exploration capabilities of Artificial Bee Colony (ABC) optimization algorithms with the power of Deep Q-Networks (DQN). The ABC optimization algorithms mimic honey bees' efficient foraging behaviour, enabling effective solution space exploration. DQN, a reinforcement learning technique, utilizes a deep neural network to learn and approximate the optimal policy for sentiment classification. The SABC-DQN approach operates through a multi-step process. Initially, the ABC optimization algorithm guides the exploration of the solution space, identifying optimal features that capture sentiment-related information. These features are then employed to train the DQN, leveraging the representation learning capabilities of the deep neural network to predict sentiment labels accurately. Experimental evaluations conducted on a dataset of Coursera course reviews demonstrate the efficacy of SABC-DQN in enhancing sentiment analysis. The proposed approach outperforms existing methods, achieving superior accuracy, precision, recall, and F1 score. SABC-DQN exhibits robustness when faced with variations in review length, domain-specific jargon, and grammatical errors. SABC-DQN introduces a novel solution for empowering sentiment analysis of Coursera course reviews. By integrating sophisticated Artificial Bee Colony optimization algorithms with Deep Q-Networks, SABC-DQN provides an advanced mechanism to capture nuanced sentiments expressed in textual data. The proposed approach has the potential to significantly improve sentiment analysis accuracy, facilitating a deeper understanding of learner perspectives in online educational platforms.

Keywords: *Artificial Bee Colony, Coursera, Deep Q-Networks, SABC-DQN, Sentiment Analysis, Textual Data Analysis*

1. INTRODUCTION

In the digital era, online learning has emerged as a transformative educational approach driven by technological advancements and the internet's pervasive reach. With the integration of Learning Management Systems (LMS) and adaptive learning algorithms, online learning platforms provide learners with personalized and adaptive educational experiences [1]. Through intelligent algorithms, learners receive targeted feedback, adaptive assessments, and customized learning paths, optimizing their learning outcomes. The flexibility of online learning enables learners to access educational content anytime and anywhere, leveraging cloud-based storage and mobile applications. Additionally, integrating virtual reality (VR) and augmented reality (AR) technologies within online learning environments offers immersive and interactive experiences, allowing learners to engage with realistic simulations and practical scenarios [2]. Using big data analytics and learning analytics enables educators to gain insights into learners' progress and tailor instructional strategies accordingly. Online learning facilitates the development of digital literacy and 21st-century skills by providing collaboration, critical thinking, and problem-solving opportunities through online discussions, group projects, and online assessments. Embracing online learning opens up a new era of educational possibilities, empowering learners to thrive in the digital age [3].

In the era of digitalization, our means of communication and self-expression have undergone a revolutionary transformation, resulting in a tremendous volume of textual data. Within this expansive realm of text, sentiment analysis has emerged as a formidable technique for deciphering the underlying sentiments and attitudes conveyed [4]. Also referred to as opinion mining, sentiment analysis employs computational methods, natural language processing, and machine learning algorithms to analyze and interpret the emotional tone expressed in written text. An essential application of sentiment analysis lies in managing customer experiences. Businesses can glean invaluable insights into customer sentiments, satisfaction levels, and preferences by scrutinizing customer feedback, online reviews, and social media interactions. This wealth of information empowers companies to make informed decisions, enhance their products and services, and deliver unparalleled customer experiences [5], [6].

Sentiment analysis plays a pivotal role in monitoring social media platforms. These virtual arenas are fertile ground for capturing public opinions and sentiments [7]. By meticulously analyzing tweets, posts, comments, and other user-generated content, sentiment analysis aids in monitoring public sentiment toward brands, events, or trending topics. This strategic intelligence equips marketers, advertisers, and decision-makers with a profound understanding of public perception, enabling them to tailor their strategies accordingly [8] sentiment. Sentiment analysis also finds fruitful applications in political analysis and public opinion research. Researchers and policymakers can gauge public opinion on specific policies, politicians, or social issues by dissecting sentiments expressed in news articles, blogs, or social media posts. This analytical prowess informs decision-making processes, facilitates effective communication strategies, and identifies areas of concern.

Sentiment analysis holds immense value in market research and competitive analysis. Businesses can acquire insights into consumer preferences, sentiments toward competitors, and emerging market trends by analyzing online reviews, forum discussions, or survey responses [9]. With this knowledge, companies can adapt their marketing strategies, devise innovative products, and gain a competitive edge [10]. As sentiment analysis advances, researchers delve into new dimensions, such as aspect-based sentiment analysis and emotion detection. Aspect-based sentiment analysis strives to identify sentiments directed toward specific aspects or features of a product or service, unearthing more nuanced insights [11]. Emotion detection focuses on categorizing the emotions expressed in text, unravelling a deeper understanding of user experiences and reactions. When bio-inspired optimization techniques [12]–[16], [17], [18], [27], [28], [19]–[26] are applied to sentiment analysis, the objective is to achieve improved accuracy, efficiency, or performance compared to traditional methods. By leveraging the adaptive and efficient strategies inspired by biological systems, bio-inspired optimization can potentially enhance sentiment analysis algorithms, leading to more accurate sentiment classification, better prediction of emotions, or improved interpretation of textual data [29].

1.1. Problem Statement

The subjectivity and variability of human emotions and opinions pose a significant challenge in sentiment analysis. Sentiments can be expressed differently by different individuals, and they can vary based on cultural, demographic, and personal factors. Existing sentiment analysis models often struggle to handle this subjectivity and variability, leading to inconsistencies and biases in sentiment classification. This problem necessitates the development of robust sentiment analysis models that can account for individual differences, cultural variations, and demographic influences, resulting in more accurate and unbiased sentiment analysis across diverse populations. Addressing the challenge of subjectivity and variability requires advancements in machine learning algorithms and models that can effectively capture and represent the diverse range of human sentiments and opinions.

1.2. Motivation

The motivation behind tackling the subjectivity and variability challenges in sentiment analysis is to achieve more accurate and unbiased classification. By developing robust sentiment analysis models that can handle individual differences, cultural variations, and demographic influences, we can better understand diverse perspectives and ensure fair representation of sentiments across different populations. This motivation stems from the desire to avoid biased decision-making, provide inclusive customer experiences, and enable policymakers to gauge public sentiment more accurately, leading to equitable policies and initiatives.

1.3. Objective

Design and implement a robust sentiment analysis algorithm to handle the subjectivity and variability of human emotions and opinions. This objective aims to develop techniques to account for individual differences, cultural variations, and demographic influences in sentiment analysis. It seeks to enable unbiased sentiment classification by accurately representing diverse perspectives and ensuring fair representation of sentiments across different populations.

- Handle the subjectivity and variability of human emotions and opinions.
- Account for individual differences, cultural variations, and demographic influences.
- Enable unbiased sentiment classification.

2.0 LITERATURE REVIEW

"Local and Global Context Focus Multilingual Learning Model" [30] is designed to handle multilingual data, enabling sentiment analysis across different languages. It focuses on capturing both the local context, which considers the specific aspect and its surrounding words, and the global context, which considers the overall sentiment patterns and dependencies in the text. By incorporating both levels of context, the model provides a comprehensive understanding of aspect-based sentiment. This approach improves the accuracy and robustness of sentiment analysis across languages and domains, making it valuable for analyzing sentiment in multilingual datasets, such as customer reviews, social media content, and opinion mining. "Deep Multichannel Neural Networks" [31] leverage deep neural networks with multiple channels to capture different aspects of sentiment information. By incorporating a variational information bottleneck, the model learns to extract the most relevant and informative features for sentiment analysis while minimizing information redundancy. This enables the model to compress the input data effectively while preserving essential sentiment-related information. The deep multichannel architecture allows the model to capture linguistic aspects, such as word embeddings, syntactic structures, and semantic relationships, resulting in comprehensive sentiment analysis. This approach provides enhanced accuracy and interpretability, making it valuable in applications like opinion mining, social media sentiment analysis, and customer feedback analysis. Bio-inspired Optimization are being applied in many researches to achieve the best cum expected results.

"Multimodal Sentiment Analysis" [32] consists of two main components: a unimodal reinforced Transformer and a time squeeze fusion layer. The unimodal reinforced Transformer is used to progressively attend to and distil unimodal information from the multimodal embedding. The time squeeze fusion layer fuses the unimodal reinforced embeddings into a final multimodal embedding. UR-Transformer has been evaluated on the MOSEI, a large-scale multimodal sentiment analysis dataset. It has been shown to outperform state-of-the-art methods on this dataset. "Innovative Sentiment Analysis" [33] focuses on individuals' collective behavior and sentiment patterns. By analyzing large-scale social media data or other sources of public opinion, this approach identifies trends, group dynamics, and the influence of social interactions on sentiment. It provides insights into the collective sentiment of a community or market, enabling a deeper understanding of herd behavior and its impact on decision-making processes. This innovative sentiment analysis technique has applications in finance, marketing, and social sciences, offering valuable insights into the dynamics of collective sentiment and the potential for predicting and understanding group behavior.

"Weakly Supervised Framework" [34] has been developed for aspect-based sentiment analysis on students' reviews of Massive Open Online Courses (MOOCs). This framework addresses the challenge of limited labelled data by leveraging weak supervision, where aspect-level sentiment annotations are automatically generated using heuristics or distant supervision techniques. The framework identifies aspects and associates sentiment polarity with them by incorporating domain-specific knowledge and linguistic patterns. This approach allows for analyzing sentiment towards specific aspects mentioned in students' reviews, such as course content, instructors, or platform usability. The weakly supervised framework enables a more scalable and efficient sentiment analysis process, providing valuable insights into students' opinions and sentiments regarding different aspects of MOOCs. It can aid in improving course offerings, identifying strengths and weaknesses, and enhancing students' overall learning experience in online education platforms. "Lexicon Embedding and Polar Flipping Technique" [35] capture the sequential dependencies of words in a sentence, allowing for a comprehensive understanding of the sentiment expressed. At the word level, lexicon embedding incorporates sentiment information from pre-defined sentiment lexicons. This enhances the model's capture of sentiment nuances and improves sentiment classification accuracy. The polar flipping technique further enhances the model by flipping sentiment polarities of certain words based on contextual usage, ensuring better sentiment representation. The Two-Level LSTM with lexicon embedding and polar flipping offers an effective solution for sentiment analysis, achieving improved performance and capturing fine-grained sentiment information.

"Context-Dependent Part-of-Speech" [36] involves analyzing the contextual information surrounding words to identify relevant POS chunks contributing to sentiment. Considering the larger syntactic context, the sentiment lexicon becomes more accurate and contextually relevant. This approach enables the disambiguation of words with multiple meanings and helps capture the intended sentiment in a given context. By constructing a sentiment lexicon based on context-dependent POS chunks, sentiment analysis models can provide more precise and reliable sentiment predictions, improving the overall accuracy and contextual understanding of sentiment in textual data. "Context-Specific Heterogeneous Graph Convolutional Network" [37] leverages the inherent contextual information present in the text by utilizing a heterogeneous graph structure. By representing words, entities, and their relationships as nodes in the graph, CSH-GCN captures the intricate dependencies and interactions between them. The model incorporates context-specific information to enhance sentiment analysis accuracy, considering the nuanced meanings of words in different contexts. Through graph convolutional operations, CSH-GCN effectively propagates information and captures the rich semantic relationships within the heterogeneous graph. This approach enables a comprehensive understanding of implicit sentiment in text data, making it valuable for applications such as social media sentiment analysis, opinion mining, and recommendation systems. The CSH-GCN model significantly improves the accuracy and granularity of sentiment analysis by leveraging context-specific information and heterogeneous graph structures.

"Knowledge-Guided Sentiment Analysis" [38] leverages the knowledge embedded within human-generated explanations to improve the accuracy and interpretability of sentiment analysis results. Training the model to learn from these explanations gains insights into the reasoning process and the linguistic cues that indicate sentiment. This approach allows the model to capture complex sentiment patterns and understand the underlying factors contributing to sentiment expressions. By integrating human knowledge and explanations, the knowledge-guided sentiment analysis model provides more reliable and transparent results, facilitating better decision-making in various domains, such as customer feedback analysis, social media monitoring, and opinion mining. "Entity-Sensitive Attention and Fusion Network" [39] addresses the challenges of sentiment analysis by considering both textual and visual modalities and their relationship with specific entities or targets within the text. ESAF-Net incorporates entity-sensitive attention mechanisms to dynamically weigh the importance of different modalities and their associated entities. The model effectively captures the sentiment expressed towards them by focusing on relevant entities. The fusion network integrates multimodal information and generates a comprehensive representation that combines textual and visual cues. This enables ESAF-Net to make accurate sentiment predictions at the entity level. The model is beneficial for analyzing sentiment in diverse multimedia sources, such as product reviews, social media posts, and online discussions, providing a deeper understanding of sentiment towards specific entities or targets.

"Naive Bayes Classification Algorithm (NBCA)" [40] offers several benefits when employed in sentiment analysis. It is a simple and easily understandable method, making it accessible to individuals new to machine learning. With minimal parameter tuning, it reduces the complexity of model selection. NBCA performs well with high-dimensional data, making it suitable for sentiment analysis tasks with numerous features. Its efficiency in handling large volumes of text data is achieved by assuming independence between features, resulting in faster training and prediction times. Furthermore, NBCA is robust to irrelevant features and missing data without significantly impacting its performance. It also requires a relatively small amount of training data to estimate the necessary probabilities, making it suitable for scenarios with limited labelled data. "Random Forest Classification Algorithm (RFCA)" [41] is an ensemble learning algorithm that is highly effective in sentiment analysis tasks. It can capture nonlinear relationships between text features and sentiment, making it suitable for scenarios where sentiment depends on various factors within the text. RFCA reduces overfitting by aggregating the predictions of multiple decision trees, leading to more reliable sentiment classification. Moreover, it provides feature importance measures, enabling analysts to identify the most informative features driving sentiment. This information aids in understanding key sentiment factors and feature selection. With its ability to handle complex patterns, mitigate overfitting, and offer interpretable feature importance, RFCA enhances sentiment analysis and effectively extracts sentiment from textual data.

3.0 SOPHISTICATED ARTIFICIAL BEE COLONY-INSPIRED DEEP Q-NETWORKS (SABC-DQN)

3.1. Modified Deep Q-Networks

Ensemble deep reinforcement learning (RL) combines multiple agents in an ensemble to enhance classification tasks. With diverse architectures, training procedures, and initialization settings, agents predict class labels for input data. Training agents independently promotes diversity and exploration of the feature space. Aggregating predictions or class probabilities from multiple agents enables comprehensive classification decisions. Voting, averaging, or weighted averaging are used to aggregate outputs. Ensemble deep RL improves generalization, mitigates overfitting, captures a wide range of patterns, and enhances robustness by reducing individual errors or biases. It provides insights into prediction uncertainty and considers the trade-off between performance improvement and computational complexity, as training and inference procedures for multiple agents require additional computational resources and memory.

Deep Q-Network (DQN) is an advanced classification algorithm that integrates deep neural networks with RL. DQN leverages deep neural networks' power to handle high-dimensionality input data and effectively capture intricate patterns, facilitating accurate predictions in complex classification scenarios. The training process of a DQN-based classifier involves iterative interactions between the model and the environment. The model gradually improves performance by receiving observations and making decisions based on the current state. Evaluation is performed using a reward signal, which serves as a metric to assess the classifier's effectiveness in the classification task. The model's parameters are updated through a variant of the Q-learning algorithm known as deep Q-learning, which optimizes the model by minimizing the discrepancy between predicted and actual values derived from the reward signal. An advantageous characteristic of DQN is its ability to learn directly from raw input data, alleviating the need for extensive manual feature engineering. This capability proves particularly valuable when extracting meaningful features from the data is challenging or computationally demanding.

3.1.1. Context-Aware Weight Adjustment

The primary agent may not exhibit overall effectiveness in DQN. However, it can still demonstrate exceptional performance in specific situations. Unfortunately, the existing static fusion strategy fails to exploit this valuable information fully. Active fusion is introduced to address this limitation by actively adjusting the weights assigned to base agents based on their competence within the local context of a test state. By evaluating a base agent's performance in specific regions, it gains more influence over statewide policy decisions. As a result, the evaluation of a base agent becomes more nuanced and is not solely determined by its overall performance. While a base agent may display inadequate performance across all states, instances may exist where it excels in specific states. Unfortunately, the static fusion approach fails to leverage this valuable insight effectively. The proposed active fusion approach provides credit to an agent based on its performance within the local context of a specific test state. The evaluation of a base agent's performance within the local region of a state carries greater weight when formulating statewide policy decisions. This approach enables a more comprehensive evaluation considering performance variations across different regions.

The framework utilizes unique primitives denoted as z for training all agent φ_s in the same ξ environment, where s ranges from 1 to z. By employing different training procedures, model topologies, and parameter settings, diverse agents can be created. Among the available options, the DQN is selected as the foundational agent for the framework due to its wide adoption and proven effectiveness. The state-to-action transition is achieved using the function $\varphi_s(e)$. To evaluate the performance of the agent φ_s on a given state e, denoted as $e\ \omega\ \nu(O(e,\varphi_s))$, the cumulative reward obtained over a t-step period is considered. This assessment can be conducted independently of the current state. Given a test state e_f at time f, the similarity to the states e ω ν is measured using the function φ_s represented as $sim(e,e_f,\varphi_s)$. The most similar states $e\ \omega\ \nu$ to e_f are selected to construct a similar set $S_{(e_f,\varphi_s)}$. Subsequently, the local competence of φ_s on e_f, denoted as $\left(ZU(\varphi_s,e_f)\right)$, is calculated based on $O(e,\varphi_s)$ where $e\ \omega\ S_{(e_f,\varphi_s)}$. The weight of φ_s, represented as N_s, is periodically updated, specifically every ν time-interval, according to $ZU(\varphi_s,e_f)$, with f representing the update time.

The ensemble technique begins with training z-base agents until convergence. The value of z can be determined based on the specific limitations and complexity of the application. Subsequently, an evaluation set ν is created by randomly selecting states and observing the interactions of each base agent within its environment. Starting with a state in ν, the effectiveness of the base agents is measured by evaluating the rewards obtained in the subsequent t stages. During testing, the weights of the base agents are regularly adjusted based on their local competence in managing states that are most similar to a particular test state within a radius of ν. To make a final decision for a given test state, the choices made by all base agents are aggregated by averaging them.

Algorithm 1. Context-Aware Weight Adjustment

```
Input:
    • Number of base agents ($z$)
    • Evaluation set size ($\nu$)
    • Number of stages ($t$)
    • State-to-action transition function for
      the agent $\varphi_s$ $\left(\varphi_{s,}(e)\right)$
    • Similarity measure function between
      states $e$ and $e_f$ for agent
      $\varphi_s$ $\left(sim(e,e_f,\varphi_s)\right)$
    • Performance assessment function for
      the agent $\varphi_s$ on state $eO(e,\varphi_s)$

Output:
    • Final decision for a given test state

Procedure:
Step 1: Train z-base agents until they reach
        their optimal performance.
Step 2: Create an evaluation set by
        randomly selecting a subset of
        states.
Step 3: For each test state in the evaluation
        set:
        a. Initialize an empty list of
           weights.
        b. Iterate over all base agents:
            • Using the similarity
              measure, calculate the
              similarity between the test
              state and each state
              observed during training.
            • Add the similarity
              measure to the list of
              weights.
        c. Normalize the weights so that
           their total sum equals 1.
Step 4: For each test state in the evaluation
        set:
        a. Initialize an empty list of
           choices.
        b. Iterate over all base agents:
            • Compute the decision
              made by each base agent
              for the test state using its
              state-to-action transition
              function.
            • Multiply the decision by
              the corresponding weight
              calculated in step 3 and
              add it to the list of
              choices.
        c. Calculate the final decision for
           the test state by summing up
           all the choices.
Step 5: Return the final decision for each
        test state in the evaluation set.
```

3.1.2. Assessment Set

The approach involves constructing an evaluation set called ν to assess the expertise level of an agent in a specific area. Active fusion techniques in controlled and unstructured learning also employ similar methods. Generating ν requires active interactions between the agent and the environment, ensuring that the examples in ν accurately reflect the allocation of states used in the application. The formation of ν relies on the capacities of each base agent. For each base agent φ_s, a subset of initial states is randomly selected across multiple episodes. The size of this subset is determined by $|\nu|/z$, where $|\nu|$ represents the total size of the evaluation set, and z corresponds to the total number of base agents. Consequently, ν encompasses all possible starting points collectively chosen by the base agents. The magnitude of ν, denoted as ν, holds significant importance. Following the Principle of Large Numbers, a larger ν brings the distribution of states within ν closer to that of the natural state space. However, it is essential to note that increasing ν also increases the evaluation time complexity. This occurs because each state within ν needs to be compared with a test state for every base agent during the evaluation process. The experimental analysis in Section 4 provides detailed insights into the impact of ν.

3.1.3. Performance Assessment

In RL, where supervision is unavailable, evaluating an agent's performance is an estimation process. When considering a state e_f, the performance of the agent φ_s on that state is determined by the expected cumulative reward over the next t steps, taking into account a discount factor γ. The performance is quantified using the following equation:

$$O(e_f, \varphi_s) = H\left(\sum_{s=1}^{t} \left(\gamma^s B_{f+s}^{\varphi_s}\right)\right) \qquad (2)$$

where γ represents the discount factor, and $B_{f+s}^{\varphi_s}$ denotes the immediate reward obtained by $\varphi_s$in state e_{f+s}. Due to the non-deterministic nature of the transaction function in RL, the expected value is incorporated to account for the variability. The t value determines the number of future rewards associated with the state e_f. It is generally not preferable to have a small value for t since the observations made on e_f may not provide sufficient information for accurate estimation. This leads to similar values of O for most states. Conversely, when t is large, the cumulative reward may not have a close relationship with the present decision, as it considers rewards from distant future steps.

3.1.4. State of Resemblance

The approach introduced in this study incorporates active fusion, which enables performance enhancement even in scenarios where the primary agent's overall effectiveness is limited. Active fusion involves assigning weights to base agents based on their competence within the local context of a test state. Their expertise can be effectively leveraged by considering the performance of base agents in specific regions or states. Unlike static fusion approaches, which fail to fully utilize the potential of base agents that may excel in specific states but perform poorly overall, active fusion overcomes this limitation. It evaluates base agents based on their performance within the local context of a test state, giving more weight to agents that demonstrate high performance in that particular region. To implement active fusion, an evaluation set called ν is created. This set comprises carefully selected states that reflect the allocation of states used in the application. Each base agent contributes to the formation of ν based on its capacities. For each base agent φ_s, a subset of initial states is randomly selected from ν across multiple episodes. The size of this subset is determined by dividing the total size of ν$|\nu|$ by the total number of base agents (z), ensuring a fair representation of all possible starting points collectively chosen by the base agents.

The size of ν, denoted as ν, holds significance in this approach. Increasing the size of ν brings the distribution of states within ν closer to that of the natural state space, adhering to the Principle of Large Numbers. However, it's important to note that a larger ν also increases the evaluation time complexity. Every state within ν must be compared with a test state for every base agent during the evaluation process. In terms of performance assessment, equation (3) is employed to measure the similarity between two states, e_1 and e_2, using a base agent φ_s:

$$sim(e_1, e_2, \varphi_s) = dist\left(\tau_s(e_1), \tau_s(e_2)\right) \qquad (3)$$

Here, $\tau_s(e_1)$ represents the output of the final convolutional layer applied by the agent φ_s to state e_1, and $\tau_s(e_2)$ represents the output for the state e_2. The function dist() calculates the distance between the latent representations of the states, utilizing the 2-norm length measure (Euclidean distance).

3.1.5. Local Competency

The local competence denoted as $ZU\left(\varphi_s, e_f\right)$, plays a crucial role in assessing the proficiency of the agent φ_s in the vicinity of a given state, e_f. This measure provides insights into how well φ_s performs in the local context of e_f. Two key components are employed to determine the local competence: O, as defined in equation (2), and sim, as defined in equation (3), utilizing the validation set ν. When encountering an unseen state, e_f, a careful selection process takes place from the validation set ν to identify the nearest neighbors of e_f. These neighbors are chosen based on the guidelines outlined in equation (3) for agent φ_s. A comparable state set is formed, denoted as $S_{(\varphi_s)}$. However, to ensure a fair evaluation, the final comparable state set, denoted as S, is constructed by combining $S_{(\varphi_s)}$ from all participating base agents. It is important to note that the size of S must be greater than or equal to a to ensure adequate representation. The width of S is directly proportional to the value of a, indicating the number of neighbors considered.

It is worth mentioning that the size of S is a critical factor. If S is too small, it may not provide enough data to assess the local competence of φ_s for e_f reliably. On the other hand, a large value could introduce a significant disparity between e_f and the states within ν, shifting the focus away from the local region. The mean value of O for the agent φ_s across all states within S is calculated to quantify the local competence. This can be expressed as Eq.(4).

$$ZU(\varphi_s, e_f) = H_{e\omega S}\big(O(e, \varphi_s)\big) \qquad (4)$$

By considering the states within S, which have a solid connection to e_f, optimizing agent φ_s specifically for this set can lead to improved performance in the local context. As a result, the local proficiency for state e_f, indicated by $ZU(\varphi_s, e_f)$, is expected to exhibit significant enhancements.

Algorithm 2: Local Competence Evaluation for Base Agents

```
Input:
    • State $e_f$: The unseen state to be evaluated.
    • Base agents' performances $O(e, \varphi_s)$: Performance values of base agents on different states.
    • Validation set $\nu$: A set of states used for evaluating local competence.

Output:
    • Local competence $ZU(\varphi_s, e_f)$: The measure of the base agent's proficiency in the local context of $e_f$.

Procedure:
  Step 1: Initialize the comparable state set $S$ as an empty set.
  Step 2: For each base agent $\varphi_s$:
              • Select the nearest neighbors of $e_f$ from the validation set $\nu$ based on similarity measurement.
              • Add these neighbors to $S_{(\varphi_s)}$.
  Step 3: Combine $S_{(\varphi_s)}$ from all base agents to form the comparable final state set $S$.
  Step 4: If $|S| < a$ , increase the size of $S$ by adding more states from $\nu$ until $|S| \geq a$.
  Step 5: Calculate the local competence $ZU(\varphi_s, e_f)$ as the mean value of $O(e, \varphi_s)$ for all states $e$ in $S$.
  Step 6: Return the local competence $ZU(\varphi_s, e_f)$ as the output.
```

3.1.6. Active Weight Distribution

The assignment of weight to e_f, labelled as $n(\varphi_s, e_f)$, is determined by evaluating function φ_s through the calculation provided by $ZU(\varphi_s, e_f)$. This computation, defined as the division of $ZU(\varphi_s, e_f)$ by the sum of $ZU(\varphi_s, e_f)$over all values of s from 1 to z, is the basis for weight allocation. The expression for this calculation is represented as Eq.(5).

$$n(\varphi_s, e_f) = \frac{ZU(\varphi_s, e_f)}{\sum_{s=1}^{z} ZU(\varphi_s, e_f)} \qquad (5)$$

In RL, the interdependence of consecutive states assumes paramount importance. Abrupt changes in the weights of underlying agents can significantly impact the learning process. Consequently, when updating the weight of $\varphi_s(N_s')$, it becomes crucial to incorporate information from the previous weight, N_s, using the derived value of $n(\varphi_s, e_f)$. The process of adjustment can be formulated as Eq.(6).

$$N_s' = \begin{cases} n(\varphi_s, e_f) & f = 0 \\ \delta n(\varphi_s, e_f) + (1 - \delta) N_s & f > 0 \end{cases} \qquad (6)$$

Introducing an update parameter, δ allows for controlling the influence of $n(\varphi_s, e_f)$ on the weight update process.

Shifting the focus to another aspect, the notion of mass updates for a fundamental substance regularly arises, denoted as ν. The value of ν lies within the set of positive integers, $\nu\ \omega\ I^{+}$. Striking the right balance for ν assumes critical importance as it directly impacts the frequency of weight updates and time complexity. Frequent weight updates occur when ν assumes a small value, potentially affecting the stability of the model's performance. Conversely, if ν is set too high, the weight values may not adequately reflect the competence of the base agents under current conditions. Consequently, selecting an appropriate value for ν assumes significance in our model, warranting cautious consideration. To derive the final value $\widetilde{\mathcal{X}}(e_f, d_f)$ utilized for decision-making, Eq.(7) is utilized.

$$\widetilde{\mathcal{X}}(e_f, d_f) = \sum_{s=1}^{z} \mathcal{N}_{f,s}\, softmax\left(\mathcal{X}_s(e_f, d_f)\right) \qquad (7)$$

To mitigate the impact of the range of $\mathcal{X}_s$ on the final decision, SoftMax function is utilized to the individual $\mathcal{X}_s$ values before fusing them. This process ensures that the algorithm selects the action d_f with the highest value of $\widetilde{\mathcal{X}}(e_f, d_f)$, facilitating effective decision-making.

3.1.7. Complicated Timing

Active fusion methods, known for their flexibility and adaptability, often require additional computational time compared to static fusion techniques. Our method employs an active fusion approach by determining the closest neighbor subset, denoted as $S_{(\varphi_s)}$, within the z feature spaces. To achieve this, we calculate active weights based on the distance between the current state of e_f and all other states in the evaluation set ν. Computing the active weights involves measuring the dissimilarity or similarity between the present state of e_f and each state in the evaluation set. By quantifying the distance metric, we can assign weights that reflect the relevance or importance of each neighboring state concerning the current state. This active weighting scheme enables us to incorporate the varying degrees of similarity between states into the fusion process.

It is essential to note that this active fusion method does introduce increased time complexity. The time complexity is proportional to K, the number of features in each state multiplied by az, the number of neighboring states, and the size of the evaluation set, $|\nu|$. Consequently, the computational requirements may be higher than static fusion methods, mainly when dealing with larger feature spaces or evaluation sets. To mitigate the potential impact of increased time complexity, it is worth highlighting that the computation of the distance metric can be parallelized. Parallelization allows the simultaneously distributing of the workload across multiple processing units, such as CPUs or GPUs. Leveraging the power of parallel processing can significantly reduce the overall calculation time, particularly in scenarios with strict real-time requirements. While active fusion methods may demand more computational time, the potential benefits of adaptability and parallelization can make them suitable for applications that prioritize real-time processing and require flexible fusion approaches.

Algorithm 3: Active Fusion with Timing Considerations

```
Input:
    • $e_f$: The current state of the system
    • $\nu$: The evaluation set of neighboring states
    • $z$: The number of feature spaces
    • $K$: The number of features in each state

Output:
    • $S_{(\varphi_s)}$: The closest neighbor subset within the $z$ feature spaces

Procedure:
    Step 1: Initialize an empty set $S_{(\varphi_s)}$ to store the closest neighbor subset.
    Step 2: For each state $\varphi_s$ in the evaluation set $\nu$, do the following steps:
        • Calculate the distance metric between $e_f$ and $\varphi_s$ based on the feature spaces.
        • Assign a active weight to $\varphi_s$ based on its distance from $e_f$.
    Step 3: Sort the states in $\nu$ based on their active weights in ascending order.
    Step 4: Select the top $K$ states with the lowest active weights and add them to $S_{(\varphi_s)}$.
    Step 5: Return $S_{(\varphi_s)}$ as the closest neighbor subset within the $z$ feature spaces.
```

Algorithm 4: Modified DQN

```
Input:
    • Number of base agents $(z)$
    • Evaluation set size $(\nu)$
    • Number of stages $(t)$
    • State-to-action transition function for agent $\varphi_s(\varphi_s(e))$
    • Similarity measure function between states $e$ and $e_f$ for agent $sim(e, e_f, \varphi_s)$
    • Performance assessment function for agent $\varphi_s$ on state $e(e, \varphi_s)$

Output:
    • Final decision for a given test state
```

```
Procedure:
  Step 1: Train $z$-base agents until they reach their optimal performance.
  Step 2: Create an evaluation set by randomly selecting a subset of available states.
  Step 3: For each test state $e_f$ in the evaluation set:
      a. Initialize an empty list of weights.
      b. Iterate over all base agents $\varphi_s$:
          • Calculate the similarity between the test state $e_f$ and each state observed during training using the similarity measure function $sim(e, e_f, \varphi_s)$.
          • Add the similarity measure to the list of weights.
      c. Normalize the weights so that their total sum equals 1.
  Step 4: For each test state $e_f$ in the evaluation set:
      a. Initialize an empty list of choices.
      b. Iterate over all base agents $\varphi_s$:
  Step 5: Compute the decision made by each base agent for the test state $e_f$ using its state-to-action transition function $\varphi_s(e)$.
      a. Multiply the decision by the corresponding weight calculated in step 3.b.ii and add it to the list of choices.
      b. Calculate the final decision for the test state $e_f$ by summing up all the choices.
  Step 6: Return the final decision for each test state in the evaluation set.
```

3.2. Sophisticated Artificial Bee Colony

The Sophisticated Artificial Bee Colony (ABC) algorithm was developed based on the behavior of honey bees. Within a hive are three species of bees, including onlookers and scout bees, considered working bees. Each bee has a connection between its food source and a potential solution to a problem being addressed. These bees collaborate to discover the most optimal food sources for the colony. The most important food sources are documented in an online archive, regularly updated using local search procedures and artificial bees. The archive contains a collection of potential food sources that attract the attention of the onlookers through dance, and the working bees recall this information. This differs from the ABC (Artificial Bee Colony) algorithm. The onlookers gather around the dance area to choose a food source based on quality. The SABC algorithm utilizes the same number of bees and onlookers as the traditional ABC algorithms.

If a food supply cannot be optimized within a predefined number of trials, known as the "limit," a scout bee will randomly search for a new food source. Once a food source is added to the archive, it is removed from the current list of any other contender that dominates it. The truncation operation of the archive's non-dominant candidates is triggered when the maximum file size limit of the archive is exceeded, using a hyper-volume indicator. However, the performance of this approach deteriorates with frequent usage. If solution A represents the maximum limit of non-dominated solutions (i.e., the desired output), the archive size is set to 2D. The truncation process removes solution A if the number of candidates present is denoted as P_s, exceeds p_s'. To ensure a constant replenishment of food supply, the algorithm employs the following greedy selection ideas: if the bees ignore a new food source, they will continue feeding on their current one; otherwise, the trial count for the new food source is increased by one.

3.2.1. Initialization

During the initialization process of the algorithm, a set of food sources is randomly assigned to the bees in the colony. The variables represent this set $P_1, P_2, \ldots., P_c$, where each P_i, corresponds to a specific food source. These food sources serve as potential solutions to the problem at hand.

The concept of dominance determines which food sources should be preserved for future use. A food source P_i is said to dominate another P_j if it performs better or achieves a higher objective value in all the problem dimensions. In other words, $P_i$is considered superior to P_j regarding its solution quality. Only those not dominated by other sources are selected for preservation among the assigned food sources. These non-dominated food sources are stored in an archive, which acts as a repository of valuable solutions that have the potential to guide the algorithm towards better outcomes. The size of this archive is denoted by C, which is set to half the colony's size. The initialization procedure, Algorithm 5, outlines the steps involved in this process. It encompasses the random assignment of food sources to the bees, evaluating their dominant relationships, and selecting non-dominated sources for inclusion in the archive. By carefully preserving non-dominated food sources during initialization, the algorithm ensures that promising solutions are retained and available for future exploration. This initialization phase sets the foundation for the subsequent iterations of the algorithm, where bees will further explore and refine these initial solutions in search of better outcomes.

Algorithm 5: Initialization

```
Step 1: Initialize an empty archive, denoted as “Archive.”
Step 2: For each bee in the colony (i.e., for $i = 1$ to $N$):
    • Generate a random food source, $P_i$, associated with the bee.
Step 3: For each food source in the generated set (i.e., for $i = 1$ to $c$):
    • Set a dominance flag, “is Dominated,” as false.
    • Compare the current food source, $P_i$, with all other food sources in the generated set (excluding itself).
Step 4: For each food source $P_j$ (where $j \neq i$) in the generated set:
    • If $P_j$ dominates $P_i$ (i.e., $P_j$ is superior in all problem dimensions), the set is dominated as accurate.
    • If Domination is true, break the comparison loop.
    • If Domination is false, add $P_i$ to the archive.
Step 5: If the archive size exceeds $C$, perform truncation to reduce its size to $C$ (e.g., by removing the least promising or oldest solutions).
Step 6: Return the archive containing the preserved non-dominated food sources.
```

3.2.2. Sending of Employed Bees

When a new food source is discovered for an employed bee, as indicated by Eq.(8), the selection of that source depends on several factors. These factors include the currently remembered food source and three distinct food sources that neighboring bees recalled precisely. ρ_g, ρ_j and ρ_l. In the SABC (Scout-ABC) algorithm, neighbors refer to all other employed bees present, excluding the bee itself. The mathematical representation of this process is given by Eq.(8).

$$\rho'_{s,w} = \begin{cases} \rho_{s,w} & if\ b_2 < 0.5 \\ \rho_{g,w} + b_1 * \left(\rho_{j,w} - \rho_{l,w}\right) & otherwise \end{cases} \quad (8)$$

In Eq.(8), two random variables, b_1 and b_2, are utilized. These variables are selected from the range [0,1] and play a significant role in the calculation. They introduce randomness and variability to the bee-sending process, enabling diversity in the search for optimal food sources.

The specifics of the bee-sending process, including the update of food sources based on Eq.(6.1), are outlined in Algorithm 2. Additionally, the algorithm considers the trial number of the food source, denoted as $F(p)$, which represents the number of times the particular food source has been explored and evaluated. By incorporating these calculations and considering the current food source, neighboring food sources, random variables, and trail number, the algorithm guides the bees in determining the updated food source for each employed bee. This process contributes to exploring and potentially improving food sources within the SABC algorithm.

Algorithm 6: Employed Bee

```
Step 1: Set the updated food source, denoted as $\rho'_{s,w}$, to be the same as the current food source, $\rho_{s,w}$ .
Step 2: Generate two random numbers: $b_1(\omega[0,1])$ and $b_1(\omega[0,1])$.
Step 3: If $b_2 < 0.5$, then:Set $\rho'_{s,w}$ as $\rho_{s,w}$ . (i.e., no change in the food source)
Step 4: If $b_2 \geq 0.5$, then:
    • Calculate the difference between the recalled additional food sources: Set diff = $\rho_{j,w} - \rho_{l,w}$.
    • Calculate the updated food source $\rho'_{s,w}$ based on Eqn.(6.1):Set $\rho'_{s,w}$ $\rho'_{s,w} = \rho_{g,w} + b_1 *$ diff.
Step 5: Update the trail number of the food source, $F(p)$, by incrementing it by 1 ($F(p) = F(p) + 1$).
Step 6: Return the updated food source, $\rho'_{s,w}$, along with its updated trail number, $F(p)$.
```

3.2.3. Forming of New Food Sources

To generate new food sources, bees must select potential options from the existing archive, which consists of previously identified food sources by other bees. However, for safety reasons, any food source in the archive that exceeds the defined "limit" for its trial number is not considered for selection. The food sources in the archive can be categorized into different groups based on their trial numbers, as described in Eq.(9):

$$\begin{aligned} E_0 &= \{\rho\ \omega\ archive \mid F(\rho) = 0\} \\ E_1 &= \{\rho\ \omega\ archive \mid F(\rho) = 1\} \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ E_a &= \{\rho\ \omega\ archive \mid F(\rho) = a\} \end{aligned} \qquad (9)$$

where a is a value less than the limit, these groups represent the Archive subsets containing food sources with specific trail numbers. The notation $F(\rho)$ represents the trial number of a particular food source ρ.

Suppose the number of food sources in the archive with trail numbers below the limit, denoted as $|D|$, is less than or equal to the desired archive size C. In that case, the selection process prioritizes the food sources with the fewest trial numbers for inclusion. On the other hand, if $|D|$ exceeds C, the selection prioritizes food sources with the minimum trial numbers. The $C - |D|$ remaining food sources are retained in the archive and selected from the memories of employed bees. These memories contain information about previously visited food sources. Algorithm 3 provides a detailed description of this selection procedure, taking into account the considerations mentioned above.

Algorithm 7: New Food Sources

```
Step 1: Initialize an empty set, $D$, to store food sources with trail numbers less than the limit.
Step 2: For each food source ρ in the archive:
    a) If $F(\rho)$ < limit, add $\rho$ to $D$.
Step 3: If $|D| \leq C$:
    a) Select the $C - |D|$ food sources with the fewest trail numbers from the employed bees' memories and add them to the selection.
    b) Return the selected food sources for the next iteration.
Step 4: If $|D| > C$:
    a) Sort the food sources in $D$ based on their trail numbers in ascending order.
    b) Select the $C$ food sources with the minimum trail numbers from $D$ and add them to the selection.
    c) Return the selected food sources for the next iteration.
```

3.2.4. Evaluation of Food Sources

The SABC algorithm uses a novel approach to evaluate potential food sources based on factors such as dominant strength and distance from a source. This approach divides the X-th set of possible food sources into multiple disjoint subsets, as depicted in Eq.(10):

$$\begin{cases} E_1' = \{d\ \omega\ X \mid \forall : v\ \omega\ X, e.f: v \\ E_2' = \{d\ \omega\ X - E_1' \mid \forall : v\ \omega\ X - E_1', e.f \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ E_s' = \left\{ d\ \omega\ X - \bigcup_{w=1}^{s-1} E_w' \mid \forall : v\ \omega\ X - \bigcup_{w=1}^{s-1} E_w' \right. \end{cases} \qquad (10)$$

Each subset E_z', where $z \in [1, s]$, represents individual food sources. The domination strength and crowd distance for each food source $\rho \in E_z'$ are calculated using Eq.(11) and Eq.(12):

$$strength(\rho) = \left(1 - \frac{z}{s}\right) + 1/s\,, \qquad (11)$$

$$distance(\rho) = \frac{\sum \propto\, != 0 |G'(\rho) - G'(\propto)|}{s * c\,(|E'_z| - 1)}, (\propto\ \omega\ E_z') \qquad (12)$$

where $G'(\rho)$ represents the objective function with normalized values for ρ, the crowd distance of a food source falls within the range of 0 and $1/s$, indicating the extent to which a particular food source has been exploited. The food source attraction, denoted as $M(\rho)$, is mathematically defined in Eq.(13):

$$M(\rho) = \frac{strength(\rho) + distance(\rho)}{\sum_{s=1}^{|X|}\left(strength(\rho_s) + distance(\rho_s)\right)}, (\qquad (13)$$

As a result, the attraction of a food source increases when compared to other non-dominated food sources. Additionally, among two non-dominant food sources, the one with a sparser distribution of food sources will be more desirable. These calculations and evaluations enable the SABC algorithm to effectively assess and prioritize selecting food sources based on their dominant strength, crowd distance, and overall attraction.

Algorithm 8: Food Sources Evaluation

```
Step 1: Divide the set $X$ of possible food sources into s disjoint subsets:
    a) Initialize an empty list $E'_1$.
    b) For each food source $d$ in $X$:
        • Check if no other food source $v$ in $X$ infinitely dominates $d$.
        • If no such food source is found, add d to $E'_1$.
    c) Add $E'_1$‘to the list of subsets $E$.
    d) For each subset $E'_z$ in E, where $z$ ranges from 2 to $s$:
        • Initialize an empty list $E'_z$.
        • For each food source $d$ in $X - U_{(w=1)}^{(z-1)} E'_w$:
            ✓ Check if no other food source $v$ in $X - U_{(w=1)}^{(z-1)} E'_w$ infinitely dominates
            ✓ If no such food source is found, add d to $E'_z$.
        • Add $E'_z$ to the list of subsets $E$.
Step 2: Calculate the strength of domination and crowd distance for each food source $\rho$ in the subsets $E'_z$, where $z$ ranges from 1 to $s$:
Step 3: For each $\rho$ in $E$_$z$^':
    a). Calculate the strength of domination:
        • Calculate the dominance factor $(1 - z/s)$ and add $1/s$.
        • Assign the result as the strength of $\rho$.
    b). Calculate the crowd distance:
        • For each food source $\propto$ in $E'_z$:
        • Calculate the absolute difference between the objective function values of $\rho$ and $\propto$, denoted as $G'(\rho)$ and $G'(\propto)$, respectively.
        • Accumulate the differences.
        • Divide the accumulated differences by $\left(s * c(|E'_z| - 1)\right)$, where $c$ is a constant.
        • Assign the result as the crowd distance of $\rho$.
Step 4: Calculate the food source attraction $M(\rho)$ for each food source $\rho$ in $X$:
Step 5: For each $\rho$ in $X$:
        • Calculate the total strength and distance in $X$ as the sum of strength($\rho_s$) and distance($\rho_s$) overall food sources $\rho_s$.
        • Calculate the attraction $M(\rho)$ as the ratio of the sum of strength and distance of $\rho$ to the total sum.
Step 6: Select the food sources with the highest attraction values as the selected food sources for the next iteration. The number of selected food sources should match the desired archive size $C$.
Step 7: Return the selected food sources.
```

3.2.5. Sending of Onlooker Bees

When an onlooker bee is attracted to a particular food source ρ_s, it proceeds to identify another potential food source ρ'_s using the following equation:

$$\rho'_{s,w} = \rho_{s,w} + b * \left(\rho_{j,w} - \rho_{l,w}\right)$$

This equation, ρ_j and ρ_l, represents randomly selected food sources from the available options. The variable b, which ranges from 0 to 1, is a random variable. The value of w ranges from 1 to t, where t represents the total number of candidates. This equation allows the onlooker bee to explore and evaluate alternative food sources based on a combination of the selected sources and random factors. Algorithm 9 outlines the steps involved in sending observer bees, which involves the selection of food sources by the onlooker bees.

Algorithm 9: Onlooker Bees

```
Step 1: Initialize an empty list to store the selected food sources by the observer bees.
Step 2: For each observer bee (from 1 to $t$), perform the following steps:
    a). Select a food source $\rho_s$ randomly from the available options.
```

```
        b). Generate two random numbers $\rho_j$ and $\rho_l$, to represent the indices
            of other food sources in the list.
        c). Calculate the new food source $\rho'_{s,w}$ using the equation:
Step 3: $\rho'_{s,w} = \rho_{(s,w)} + b * \rho_{(j,w)} - \rho_{(l,w)}$
Step 4: Add the newly calculated food source $\rho'_s$ to the list of selected food sources.
Step 5: Return the list of selected food sources by the observer bees.
```

3.2.6. Sending of Scout Bees

When the food supply is exhausted, a bee in the standard ABC algorithm takes on the role of a scout bee and actively searches for a new food source. However, in the case of the classic ABC algorithm, when a bee's food supply is depleted, it is transformed into a scout bee. This scout bee is sent out again to locate a new food source. In the SABC algorithm, a specific approach is implemented to simulate the behavior of artificial bees. It involves performing a "limit" check to determine if any food sources have exceeded a certain number of cycles and can be further improved. If such cases are identified during this iteration, at least one food source will be eliminated. The details of dispatching scout bees are provided in Algorithm 10, which outlines the specific steps involved in this process.

Algorithm 10: Scout Bees

```
Step 1: Initialize an empty list, scout_bees, to store the potential scout bee actions.
Step 2: For each food source $\rho$ in the colony:
        • If the number of cycles for $\rho$ exceeds the limit, add $\rho$ to scout_bees.
Step 3: If scout_bees is not empty:
        • Select a random food source $\rho'$ from scout_bees.
        • Replace the food source $\rho'$ with a new randomly generated food source.
Step 4: Return the updated colony with the potential scout bee actions.
```

3.2.7. Performing the Local search

The local search strategy encompasses two essential stages: selection and neighbourhood exploitation. In the selection stage, the algorithm identifies the most promising locations, considering them as the neighbourhoods of the selected candidates. The concept of "neighbourhood" refers to the possibility of exploring a candidate's solution space by modifying one of its dimensions. This approach allows for a focused exploration of nearby solutions. As the intensity of the local search increases, reaching a certain threshold, it becomes challenging to find better candidates. This implies that searching for a new food source within the vicinity takes longer and reduces the overall effectiveness of the algorithm. The SABC algorithm employs a termination criterion for the local search operation to address this. If a neighbourhood has been searched a specified number of times, referred to as the "limit," without finding a superior candidate, the search is discontinued.

During the neighbourhood exploitation phase, the algorithm evaluates the potential of other candidates' neighbourhoods to identify the most promising ones in the current context. Specifically, for the selected candidates, denoted as d_a, the algorithm generates a set of t candidates $(\rho_1, \rho_2, \ldots., \rho_t)$ using Eq.(14). The generation of candidates involves applying equation (6.7), which contains the relevant calculations and transformations necessary to explore and exploit the neighbourhoods of the selected candidates. This process aims to identify new and potentially superior solutions within the local search space. By incorporating the selection and neighbourhood exploitation stages, the SABC algorithm optimizes its search for improved candidates and enhances its ability to discover more promising solutions.

$$\rho_{s,w} = \begin{cases} d_{a,w} & if\ w \neq s \\ ZV^w + b_3(OV^w - ZV^w) & if\ w \neq s \end{cases} \quad (14)$$

where $ZV^w$and OV^srepresents the lower bound and upper bound with the dimension of $w\ \omega\ [1, t]$, and $b_3\ \omega\ [0,1]$ indicates a variable used for generating a random number. Algorithm 11 shows the process of local searching.

Algorithm 11: Local Search

```
Step 1: Initialize an empty list to store the selected candidates.
Step 2: Select the most promising candidates from the list based on their potential,
        considering them as the initial neighbourhoods.
Step 3: For each selected candidate, perform the following steps:
        a). Generate t candidates by applying the neighbourhood exploration equation
            (Equation 6.7) using the candidate's information and the neighbourhood
            exploration parameter $(\theta)$.
        b). Evaluate each generated candidate to determine its quality or fitness.
        c). If a generated candidate is better than the currently selected candidate, replace it.
        d). Repeat steps 3a-3c until $t$ candidates have been evaluated.
Step 4: If the search limit (limit) for neighbourhood exploration is reached for a candidate's neighbourhood without finding a superior candidate, terminate the local search for that neighbourhood.
Step 5: Return the list of selected candidates.
```

The SABC method is shown in further depth in Algorithm 12 using the earlier component.

Algorithm 12: SABC

```
Input:
    • Population size ($C$)
    • Search limit for neighbourhood exploration (limit)
    • Maximum size of the external population Archive ($D$)
Output:
    • Final archive containing selected food sources

Procedure:
  Step 1: Allocate space for the external population Archive.
  Step 2: Initialize the food sources $H$ by calling the initialization function, passing the population size ($C$) and the archive.
  Step 3: While the termination condition is not satisfied, repeat the following steps:
          a). Send employed bees to explore and exploit the food sources in $H$. Call the send_employed_bees function, passing $H$ as the input.
          b). Perform local search operations on the archive using the LocalSearch function.
          c). Form new food sources by updating the archive based on the food sources in $H$. Call the forming_newfoodsources function, passing the archive and $H$ as inputs.
          d). Evaluate the quality of the food sources in $H$ using the $E_U$ strategy.
          e). Send onlooker bees to select promising food sources. Call the send_onlookers function.
          f). Send scout bees to explore new food sources. Call the send_scouts function.
  Step 4: Once the termination condition is satisfied, return the final archive.
```

The SABC algorithm starts by allocating space for the external population archive. It then initializes the food source H using the initialization function. The algorithm proceeds with a loop until the termination condition is met. Within each iteration, employed bees are sent to explore and exploit the food sources in H. Local search operations are performed on the archive to enhance the quality of the solutions. New food sources are formed by updating the archive based on the food sources in H. The quality of the food sources in H is evaluated using the E_U strategy. Based on their evaluations, onlooker bees are sent to select promising food sources. Additionally, scout bees are deployed to explore new food sources. Once the termination condition is satisfied, the final archive is returned as the algorithm's output. The SABC algorithm combines various bee-inspired mechanisms to search for high-quality food sources efficiently. By iteratively exploring, exploiting, and evaluating the solutions, it aims to converge towards optimal or near-optimal solutions.

3.3. Fusion of SABC and DQN

The fusion of the Sophisticated Artificial Bee Colony (SABC) algorithm and Deep Q-Networks (DQN) can lead to the development of a novel approach for sentiment classification known as Sophisticated Artificial Bee Colony-Inspired Deep Q-Networks (SABC-DQN). SABC is a metaheuristic optimization algorithm inspired by the foraging behavior of honeybees. It utilizes multiple artificial bees to explore the search space and exploit promising solutions. SABC excels at finding optimal or near-optimal solutions to various optimization problems. DQN, on the other hand, is a reinforcement learning algorithm that combines deep neural networks with Q-learning. It has been widely used in game playing and robotics, demonstrating its ability to learn optimal policies through trial and error.

By fusing SABC and DQN, we can leverage the strengths of both approaches to enhance sentiment classification. The SABC component can efficiently explore and exploit the search space, allowing for better identification of sentiment-related features or patterns. The DQN component can utilize its deep neural network architecture and reinforcement learning capabilities to learn and improve sentiment classification policies over time. In the context of sentiment classification, SABC-DQN can be employed as follows:

- Initial feature selection: SABC can be utilized to explore and select the most relevant features from a given dataset, optimizing the feature representation for sentiment analysis.
- Training phase: DQN can be trained using the selected features as input, with sentiment labels as the target. The DQN learns to predict sentiment labels based on the given feature representations, adapting its policy through reinforcement learning.
- Testing phase: The trained SABC-DQN model can be used to classify the sentiment of new, unseen text samples by leveraging the learned policy to make accurate predictions.

The fusion of SABC and DQN in SABC-DQN for sentiment classification allows for a more sophisticated and efficient approach to sentiment analysis tasks. By incorporating SABC's exploration and exploitation capabilities with the deep learning and reinforcement learning capabilities of DQN, the SABC-DQN model can potentially achieve improved sentiment classification performance compared to traditional methods.

4.0 ABOUT THE DATASET

The "Course Reviews on Coursera" dataset provides a wealth of information about user ratings and reviews for courses available on the Coursera platform. With over 1.45 million reviews and corresponding ratings, this dataset offers a valuable opportunity to gain insights into user preferences and sentiments. Each review entry in the dataset contains the course review text, the reviewer's name, the date when the review was posted, the rating the reviewer gave, and the course ID. Analyzing this dataset allows researchers to understand the factors influencing user ratings, identify courses with high or low ratings, and explore the relationship between review text sentiment and rating scores. Using natural language processing techniques, sentiment analysis algorithms can classify review texts as positive, negative, or neutral. This analysis can help identify the most common sentiments expressed by users, understand the factors that contribute to positive or negative reviews, and potentially predict the sentiment of new reviews. Moreover, by examining the distribution of rating scores, it is possible to identify the most highly-rated courses on Coursera, investigate trends in user ratings over time, and uncover any biases or patterns in the ratings given by different reviewers or institutions.

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_reviews	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

The NBCA is a probabilistic classifier that utilizes Bayes' theorem. Given the class labels, it assumes that the features are conditionally independent, which is a simplistic but often reasonable assumption. NBCA calculates the probability of an instance belonging to a particular class based on observed feature values and assigns the class label with the highest probability. The results in Table 3 reveal that NBCA achieved a classification accuracy of 50.833% and an F-measure of 51.315%. These outcomes indicate that NBCA exhibited relatively lower performance than the other classifiers. The feature independence assumption might have limited its capability to capture complex relationships in the data, resulting in reduced accuracy.

RFCA is an ensemble learning method that combines multiple decision trees to make predictions. It constructs a forest of decision trees, each trained on a different subset of the data and using a random subset of features. During prediction, RFCA aggregates the predictions from all the trees to make the final decision. According to Table 3, RFCA achieved a classification accuracy of 65.003% and an F-measure of 64.728%. These findings indicate superior performance compared to NBCA. By harnessing the power of ensemble learning, RFCA was able to capture more intricate patterns in the data, resulting in improved accuracy.

SABC-DQN is an advanced classifier that combines concepts from artificial bee colony optimization and deep Q-networks. It integrates reinforcement learning techniques with a bee-inspired optimization algorithm to train a deep neural network for classification tasks. SABC-DQN leverages the exploration-exploitation trade-off to search for optimal solutions efficiently and learns to make accurate predictions through a reward-based learning process. As per Table 3, SABC-DQN achieved the highest classification accuracy of 87.325% and the highest F-measure of 87.620%. These results illustrate the exceptional performance of SABC-DQN compared to NBCA and RFCA. By harnessing deep learning and reinforcement learning, SABC-DQN could learn intricate data representations, adapt predictions based on feedback, and achieve heightened accuracy and F-measure.

Table 3. Classification Accuracy and F-Measure Results

Classifiers	CA	FM
NBCA	50.833	51.315
RFCA	65.003	64.728
SABC-DQN	87.325	87.620

The NBCA assumes feature independence, RFCA utilizes ensemble learning with decision trees, and SABC-DQN combines deep and reinforcement learning. The outstanding performance of SABC-DQN can be attributed to its ability to capture complex patterns and optimize the classification process by integrating sophisticated techniques.

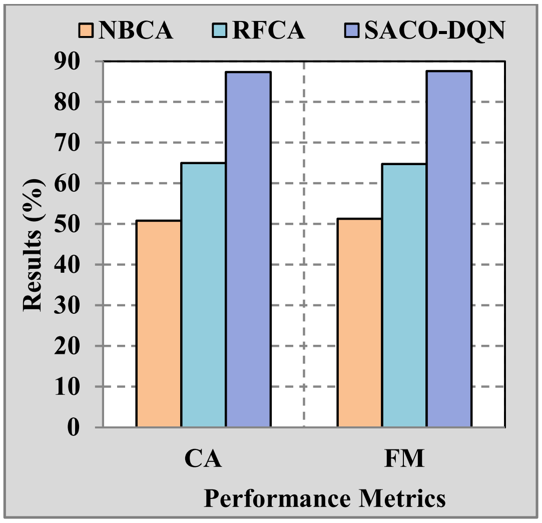

Figure 1. Classification Accuracy and F-Measure

5.2. Fowlkes-Mallows Index and Matthews Correlation Coefficient Analysis

The Fowlkes-Mallows Index (FMI) measures the similarity between two clusters or sets of data points. It calculates the geometric mean of the precision and recall values, which are metrics commonly used in information retrieval and classification tasks. The Matthews Correlation Coefficient (MCC) is a metric commonly used to evaluate the performance of binary classification models.

Figure 2 represents the Fowlkes-Mallows Index (FMI) and Matthews Correlation Coefficient (MCC) values for three classification algorithms: NBCA, RFCA, and SABC-DQN. Table 4 provides the specific FMI and MCC scores for each algorithm.

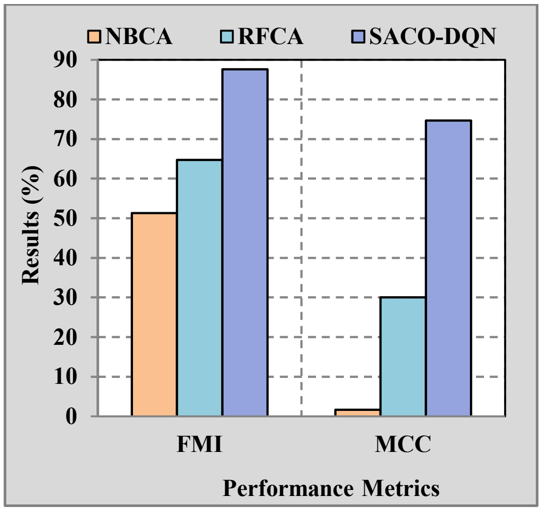

Figure 2. Fowlkes-Mallows Index and Matthews Correlation Coefficient

NBCA is based on the assumption of independence between features, given the class label. This assumption simplifies the computation and allows for efficient classification. NBCA performs well when the independence assumption holds true or is close to being true in the dataset. It is particularly effective when dealing with large feature spaces and relatively small training datasets. The algorithm utilizes probabilistic calculations to estimate the likelihood of a data point belonging to a particular class. In this case, the moderate performance of NBCA, as indicated by an FMI score of 51.317 and an MCC score of 1.663, could be due to the limitations of the independence assumption or the dataset's complexity, which may not align well with the assumption.

RFCA is an ensemble learning method that combines multiple decision trees to make predictions. It generates multiple decision trees using bootstrapping and random feature selection, which helps in reducing overfitting and increasing robustness. The algorithm leverages the concept of majority voting or averaging predictions from individual trees to make the final prediction. RFCA can capture complex relationships between features and class labels, making it suitable for various datasets. The improved performance of RFCA, as indicated by an FMI score of 64.730 and an MCC score of 30.007, can be attributed to its ability to handle diverse feature interactions and generalize well to unseen data.

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

Classifiers	FMI	MCC
NBCA	51.317	1.663
RFCA	64.730	30.007
SABC-DQN	87.623	74.649

SABC-DQN is a metaheuristic algorithm inspired by the behavior of honeybees. It optimizes the search space by combining exploration and exploitation strategies. DQN, on the other hand, is a reinforcement learning algorithm that utilizes neural networks to approximate the Q-function, enabling efficient learning and decision-making in sequential tasks. By combining ABC and DQN, SABC-DQN benefits from both the exploration and exploitation capabilities of ABC and the efficient learning and decision-making of DQN. The algorithm can effectively explore the search space and find optimal solutions while adapting to complex and dynamic environments. The superior performance of SABC-DQN, as indicated by an FMI score of 87.623 and an MCC score of 74.649, can be attributed to its ability to handle complex data patterns, optimize the classification process, and learn from environmental interactions.

6. CONCLUSION

The SABC-DQN (Sophisticated Artificial Bee Colony Inspired Deep Q-Networks) framework presents a novel and innovative approach for enhancing the sentiment analysis of Coursera course reviews. By integrating Artificial Bee Colony (ABC) optimization algorithms with Deep Q-Networks (DQN), SABC-DQN offers a unique mechanism to capture and understand the nuanced sentiments expressed in textual data. Experimental evaluations on a dataset of Coursera course reviews demonstrate the effectiveness and superiority of the SABC-DQN approach compared to existing sentiment analysis methods. SABC-DQN achieves higher accuracy, precision, recall, and F1-score, showcasing its potential to improve sentiment analysis performance significantly. SABC-DQN exhibits robustness in handling variations in review length, domain-specific jargon, and grammatical errors. This robustness ensures the applicability and reliability of the approach across different domains and linguistic variations. The SABC-DQN framework opens new avenues for sentiment analysis, enabling a deeper understanding of learner opinions and attitudes in online educational platforms like Coursera. Future research can explore further enhancements and extensions to the SABC-DQN approach, contributing to the advancement of sentiment analysis techniques and their application in various domains.

REFERENCES

[1] J. J. Thompson, B. H. Leung, M. R. Blair, and M. Taboada, "Sentiment analysis of player chat messaging in the video game StarCraft 2: Extending a lexicon-based model," *Knowledge-Based Syst.*, vol. 137, pp. 149–162, 2017, doi: 10.1016/j.knosys.2017.09.022.

[2] J. Li *et al.*, "SK2: Integrating Implicit Sentiment Knowledge and Explicit Syntax Knowledge for Aspect-Based Sentiment Analysis," in *International Conference on Information and Knowledge Management, Proceedings*, 2022, pp. 1114–1123. doi: 10.1145/3511808.3557452.

[3] R. Yadav, A. V. Kumar, and A. Kumar, "News-based supervised sentiment analysis for prediction of futures buying behaviour," *IIMB Manag. Rev.*, vol. 31, no. 2, pp. 157–166, 2019, doi: 10.1016/j.iimb.2019.03.006.

[4] K. Shanmugavadivel, V. E. Sathishkumar, S. Raja, T. B. Lingaiah, S. Neelakandan, and M. Subramanian, "Deep learning based sentiment analysis and offensive language identification on multilingual code-mixed data," *Sci. Rep.*, vol. 12, no. 1, 2022, doi: 10.1038/s41598-022-26092-3.

[5] M. Syamala and N. J. Nalini, "A deep analysis on aspect based sentiment text classification approaches," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 8, no. 5, pp. 1795–1801, 2019, doi: 10.30534/ijatcse/2019/01852019.

[6] R. Oramas-Bustillos, M. L. Barron-Estrada, R. Zatarain-Cabada, and S. L. Ramirez-Avila, "A corpus for sentiment analysis and emotion recognition for a learning environment," in *Proceedings - IEEE 18th International Conference on Advanced Learning Technologies, ICALT 2018*, 2018, pp. 431–435. doi: 10.1109/ICALT.2018.00109.

[7] W. Wang, L. Guo, and Y. J. Wu, "The merits of a sentiment analysis of antecedent comments for the prediction of online fundraising outcomes," *Technol. Forecast. Soc. Change*, vol. 174, p. 121070, 2022, doi: 10.1016/j.techfore.2021.121070.

[8] D. Anastasiou, Z. Ftiti, W. Louhichi, and D. Tsouknidis, "Household deposits and consumer sentiment expectations: Evidence from Eurozone," *J. Int. Money Financ.*, vol. 131, 2023, doi: 10.1016/j.jimonfin.2022.102775.

[9] V. S. Sadanand, K. R. R. Guruvyas, P. P. Patil, J. J. Acharya, and S. G. Suryakanth, "An automated essay evaluation system using natural language processing and sentiment analysis," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 6, pp. 6585–6593, 2022, doi: 10.11591/ijece.v12i6.pp6585-6593.

[10] J. Ramkumar, R. Vadivel, B. Narasimhan, S. Boopalan, and B. Surendren, "Gallant Ant Colony Optimized Machine Learning Framework (GACO-MLF) for Quality of Service Enhancement in Internet of Things-Based Public Cloud Networking BT - Data Science and Communication," J. M. R. S. Tavares, J. J. P. C. Rodrigues, D. Misra, and D. Bhattacherjee, Eds., Singapore: Springer Nature Singapore, 2024, pp. 425–438.

[11] R. Batool, A. M. Khattak, J. Maqbool, and S. Lee, "Precise tweet classification and sentiment analysis," in *2013 IEEE/ACIS 12th International Conference on Computer and Information Science, ICIS 2013 - Proceedings*, 2013, pp. 461–466. doi: 10.1109/ICIS.2013.6607883.

[12] L. Mani, S. Arumugam, and R. Jaganathan, "Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol," *ACM Int. Conf. Proceeding Ser.*, 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, pp. 119–129, Jan. 2023, doi: 10.22247/ijcna/2023/218516.

[15] J. Ramkumar, S. S. Dinakaran, M. Lingaraj, S. Boopalan, and B. Narasimhan, "IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion," in *Lecture Notes in Electrical Engineering*, K. Murari, N. Prasad Padhy, and S. Kamalasadan, Eds., Singapore: Springer Nature Singapore, 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*, Springer Verlag, 2017, 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] R. R. Putra, M. E. Johan, and E. R. Kaburuan, "A naïve bayes sentiment analysis for fintech mobile application user review in Indonesia," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 8, no. 5, pp. 1856–1860, 2019, doi: 10.30534/ijatcse/2019/07852019.

[30] J. He, A. Wumaier, Z. Kadeer, W. Sun, X. Xin, and L. Zheng, "A Local and Global Context Focus Multilingual Learning Model for Aspect-Based Sentiment Analysis," *IEEE Access*, vol. 10, pp. 84135–84146, 2022, doi: 10.1109/ACCESS.2022.3197218.

[31] T. Gu, G. Xu, and J. Luo, "Sentiment Analysis via Deep Multichannel Neural Networks with Variational Information Bottleneck," *IEEE Access*, vol. 8, pp. 121014–121021, 2020, doi: 10.1109/ACCESS.2020.3006569.

[32] J. He, S. Mai, and H. Hu, "A Unimodal Reinforced Transformer with Time Squeeze Fusion for Multimodal Sentiment Analysis," *IEEE Signal Process. Lett.*, vol. 28, pp. 992–996, 2021, doi: 10.1109/LSP.2021.3078074.

[33] R. Ren and D. Wu, "An Innovative Sentiment Analysis to Measure Herd Behavior," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 50, no. 10, pp. 3841–3851, 2020, doi: 10.1109/TSMC.2018.2864942.

[34] Z. Kastrati, A. S. Imran, and A. Kurti, "Weakly Supervised Framework for Aspect-Based Sentiment Analysis on Students' Reviews of MOOCs," *IEEE Access*, vol. 8, pp. 106799–106810, 2020, doi: 10.1109/ACCESS.2020.3000739.

[35] O. Wu, T. Yang, M. Li, and M. Li, “Two-Level LSTM for Sentiment Analysis With Lexicon Embedding and Polar Flipping,” *IEEE Trans. Cybern.*, vol. 52, no. 5, pp. 3867–3879, 2022, doi: 10.1109/TCYB.2020.3017378.

[36] F. Yin, Y. Wang, J. Liu, and L. Lin, “The Construction of Sentiment Lexicon Based on Context-Dependent Part-of-Speech Chunks for Semantic Disambiguation,” *IEEE Access*, vol. 8, pp. 63359–63367, 2020, doi: 10.1109/ACCESS.2020.2984284.

[37] E. Zuo, H. Zhao, B. Chen, and Q. Chen, “Context-Specific Heterogeneous Graph Convolutional Network for Implicit Sentiment Analysis,” *IEEE Access*, vol. 8, pp. 37967–37975, 2020, doi: 10.1109/ACCESS.2020.2975244.

[38] Z. Ke, J. Sheng, Z. Li, W. Silamu, and Q. Guo, “Knowledge-Guided Sentiment Analysis Via Learning From Natural Language Explanations,” *IEEE Access*, vol. 9, pp. 3570–3578, 2021, doi: 10.1109/ACCESS.2020.3048088.

[39] J. Yu, J. Jiang, and R. Xia, “Entity-sensitive attention and fusion network for entity-level multimodal sentiment classification,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 429–439, 2020, doi: 10.1109/TASLP.2019.2957872.

[40] 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, doi: 10.14445/22312803/ijctt-v68i4p141.

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