INTRODUCTION

Sentiment analysis is the process of extracting emotional tones from text data. It has become a powerful tool for understanding public opinion on social media platforms like Twitter. The ongoing COVID-19 pandemic has further increased the significance of sentiment analysis. By analysing the sentiments expressed in tweets related to COVID-19, we can gain valuable insights into various aspects of the pandemic.

Firstly, we can gauge public perception by understanding the emotions and opinions surrounding the pandemic. This can reveal public trust in authorities, concerns about the virus, and anxieties related to social and economic impacts.

Secondly, sentiment analysis can help highlight specific areas where public anxiety is high. This can allow policymakers to address misinformation and target communication efforts effectively.

Thirdly, understanding public sentiment can significantly improve adherence to guidelines and boost vaccination rates. This can help tailor public health messaging to better resonate with the target audience.

One of the most significant applications of sentiment analysis is identifying vaccine hesitancy and misinformation. By pinpointing tweets expressing doubts or negativity towards vaccines, policymakers can develop targeted campaigns to address these concerns and promote evidence-based information.

This study aims to evaluate and compare different techniques for vectorizing tweets, which involves converting text into numerical formats. It also aims to classify their sentiment as positive, negative, or neutral, using machine learning algorithms. Through this research, we hope to identify the most effective methods for analysing public sentiment on social media during the pandemic.

BACKGROUND

The Importance of Text Representation:

Effective sentiment analysis using machine learning algorithms hinges on robust text representation techniques (Mutinda,2022). Word embeddings, a form of vectorization, are crucial for converting textual data (like tweets) into numerical formats suitable for machine learning algorithms (Dhanani et al.,2021). These techniques transform tweets into numerical

feature vectors that significantly enhance the accuracy and efficiency of sentiment analysis tasks (Dhanani et al.,2021).

Classification Techniques:

Once tweets are represented numerically, classification techniques categorize them into sentiment classes (positive, negative, or neutral). Research in this area is ongoing, with various approaches showing promise.

Kuer et al. (2021) proposed a Hybrid Heterogeneous Support Vector Machine (H-SVM) for classifying COVID-19 tweet sentiment, achieving an accuracy of 86% - outperforming RNN and standard SVM models.

However, limitations exist. Hama Aziz et al. (2021) explored sentiment analysis using SentiXGboost, an ensemble XGBoost classifier that combined Bag-of-Words (BoW) and TF-IDF vectorization. While achieving high accuracy (0.93), they highlighted the challenges of using deep learning models with limited datasets.

Deep Learning Approaches:

Studies have explored the use of deep learning architectures like Bidirectional Long Short-Term Memory Networks (Bi-LSTM) for sentiment classification tasks, demonstrating promising performance in analysing sentiment (Purwarianti & Crisdayanti, 2019).

Problem Statement and Scope

This project aims to evaluate the effectiveness of different text vectorization and classification techniques for sentiment analysis of COVID-19 tweets. The primary focus is to identify the most efficient approach for classifying these tweets into sentiment categories (e.g., positive, negative, neutral).

- The project will explore various text vectorization techniques, including:
    - **CBOW (Continuous Bag-of-Words):** Captures word meaning based on surrounding words in a window.
    - **TF-IDF (Term Frequency-Inverse Document Frequency):** Considers both word importance within a tweet and its rarity across the entire dataset.
    - **GloVe Embeddings:** Pre-trained word embeddings capturing semantic relationships between words.

- We will investigate different classification algorithms, such as:
    - Naive Bayes: A probabilistic classifier suitable for text data.
    - Random Forest: An ensemble learning method that combines multiple decision trees for improved performance.
    - Long Short-Term Memory Networks (LSTM): Recurrent neural networks capable of capturing long-term dependencies within sequences (important for understanding context in tweets).
    - Gated Recurrent Unit (GRU) Networks: Another type of RNN suitable for sequential data like tweets, with a simpler architecture compared to LSTMs.
- The project will utilize a dataset of labelled COVID-19 tweets for training and testing the models.
- Evaluation will be based on metrics like accuracy, precision, recall, and F1-score.

Limitations:

- The project may be limited by the size and quality of the available COVID-19 tweet dataset.
- Computational resources may restrict the exploration of complex deep learning architectures.

Objectives:

- **Compare Vectorization Techniques**

Specific: Analyze the performance of three text vectorization techniques (TF-IDF, CBOW, and GloVe embeddings) on a dataset of COVID-19 tweets.

Measurable: Evaluate the performance of each technique using the F1-score, aiming for a score higher than the baseline accuracy of 85%.

Achievable: Use standard libraries and methods in Python (e.g., Scikit-learn for TF-IDF and CBOW, Gensim for GloVe) to implement the vectorization techniques.

Relevant: Understanding which vectorization technique best captures the sentiment in tweets will improve the accuracy of the sentiment analysis models.

Time-bound: Complete the evaluation and comparison of vectorization techniques by May 16th.

- **Compare Classification Techniques**

Specific: Evaluate the performance of four classification algorithms (Naive Bayes, Random Forest, LSTM, and GRU) on the vectorized COVID-19 tweet dataset.

Measurable: Aim to achieve an F1-score of over 85% for at least one of the classification algorithms. Compare all algorithms using metrics such as F1-score, precision, recall, and accuracy.

Achievable: Implement the classifiers using well-established machine learning and deep learning libraries (e.g., Scikit-learn for Naive Bayes and Random Forest, TensorFlow/Keras for LSTM and GRU).

Relevant: Identifying the most effective classification algorithm will improve the accuracy of categorizing the sentiments expressed in COVID-19 tweets.

Time-bound: Complete the evaluation and comparison of classification techniques by May 16th.

- **Data Preparation and Preprocessing**

Specific: Preprocess the collected COVID-19 tweet dataset by cleaning the text, removing noise, and balancing the dataset to ensure equal representation of sentiment classes.

Measurable: Ensure that the final dataset has balanced classes with equal representation for positive, negative, and neutral tweets, and is free from noise and irrelevant data.

Achievable: Use Python libraries like Pandas, NLTK, and Scikit-learn to preprocess the data and ensure it is suitable for analysis.

Relevant: Properly pre-processed data is crucial for building accurate sentiment analysis models.

Time-bound: Complete data preprocessing by May 16th.

Model Training and Evaluation

Specific: Train and evaluate sentiment analysis models using the pre-processed and vectorized tweet dataset.

Measurable: Track the training and validation accuracy and loss for each model. Evaluate the final models using metrics such as confusion matrix, classification report, and F1-score.

Achievable: Utilize GPU resources and efficient libraries to train the models within a reasonable time frame.

Relevant: Evaluating the models' performance will determine the most effective approach for sentiment analysis of COVID-19 tweets.

Time-bound: Complete model training and evaluation by May 16th.

Dataset Description

The COVID-19 tweet dataset used for analysis was sourced from Kaggle https://www.kaggle.com/datasets/arunavakrchakraborty/covid19-twitter-dataset/data and collected between April to June 2020. It underwent various transformations, including oversampling to balance classes and dropping irrelevant columns. An analysis of the dataset's properties, suitability, strengths, and weaknesses follows:

1. **Size and Shape**:

The initial dataset had 143,903 rows and 17 columns. After oversampling, balancing, and dropping irrelevant columns, it expanded to 172,755 rows. Each sentiment class now has an equal number of samples, which is 57,585 rows.

2. **Structured:**

Data is structured with one row per tweet and columns for features like text content and sentiment labels, making it easy to manipulate, analyse, and train models.

3. **Suitability for the Problem:**

This dataset is suitable for sentiment analysis, specifically for classifying tweets into positive, negative, or neutral categories. With many samples and well-balanced classes, this dataset provides ample data for training and evaluating machine learning models for sentiment classification.

4. Strengths:

- The dataset has balanced classes, a structured format, and a large sample size. Balancing techniques ensure equal distribution of samples, improving model accuracy. The structured format streamlines machine learning and the large sample size captures diverse language patterns.

5. Weaknesses:

- Dropping irrelevant columns during preprocessing may loss valuable information that could improve model performance.

- Random oversampling with replacement may introduce synthetic or duplicated samples, leading to overfitting if not properly controlled.

EXPLORATORY DATA ANALYSIS

For the preprocessing stage of this study a function ‘preprocess text’ was created and takes the text and performs the following preprocessing steps:

1. **Cleaning Text**: The "original text" column was pre-processed to remove URLs, special characters, numbers, and meaningless words such as "RT" (indicating retweets).
2. **Lemmatization**: Tokenized words were lemmatized to reduce inflected words to their base or dictionary form.
3. **Stopword Removal**: Common English stopwords were removed from the text to focus on meaningful content.
4. Additional Preprocessing: Certain substrings like "http," "ha," and "amp" were removed from the text.
5. Sentiment labels (‘pos’, ‘neu’, ‘neg’) are replaced with their full forms (‘positive’, ‘neutral’, ‘negative’) using ’.replace’.
6. **Stemming**: The stemming step was applied to further normalize the text
7. After preprocessing, a new "clean_text" column was generated to replace the original "clean_text" column. This ensured consistency and allowed for tailoring the preprocessing steps to better suit the analysis objectives.

- **Oversampling for Class Balance**: The function balances the dataset by oversampling the positive and negative sentiment classes to match the size of the neutral class. Random oversampling is performed using resample from the sklearn. utils module. The oversampled data is concatenated with the original neutral data to create a balanced Data Frame called balanced_df.
- **Plotting**: Two bar plots are created side by side: The first plot shows the value counts of sentiment labels before oversampling. The second plot shows the value counts of sentiment labels after oversampling as shown I figure 1.

- **Word Cloud:** Figure 2 shows Word clouds which are graphical representations of word frequency that give greater prominence to words that appear more frequently in a source text were plotted for the positive, Negative and Neutral sentiments. The word Covid19 was most frequent in the tweets across sentiments. While wors like Support, death was one of the most used words in positive and negative sentiment tweets respectively.

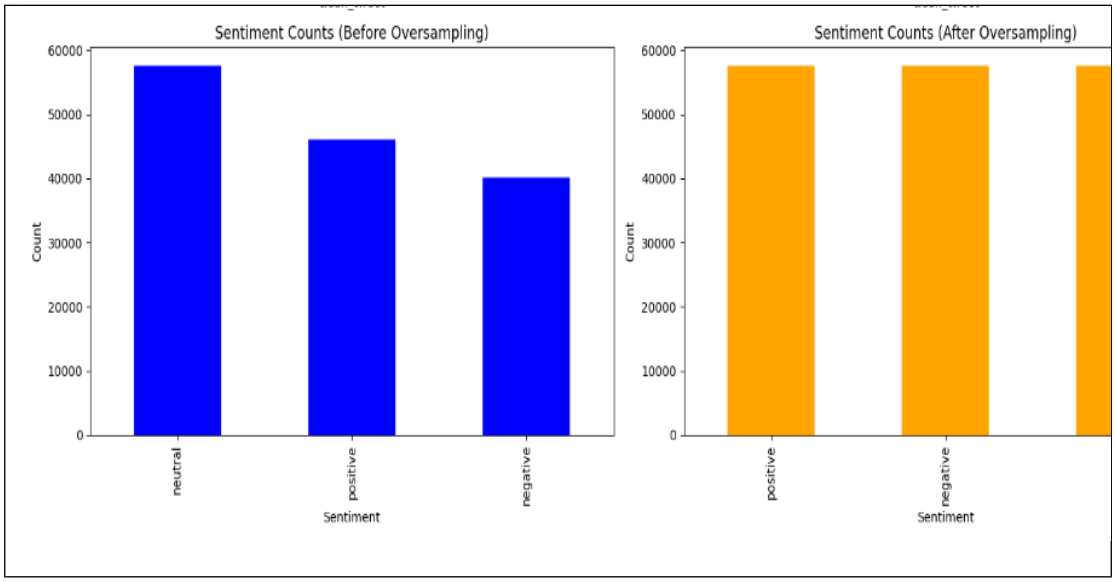


Figure 1: Oversampling for class balancing.

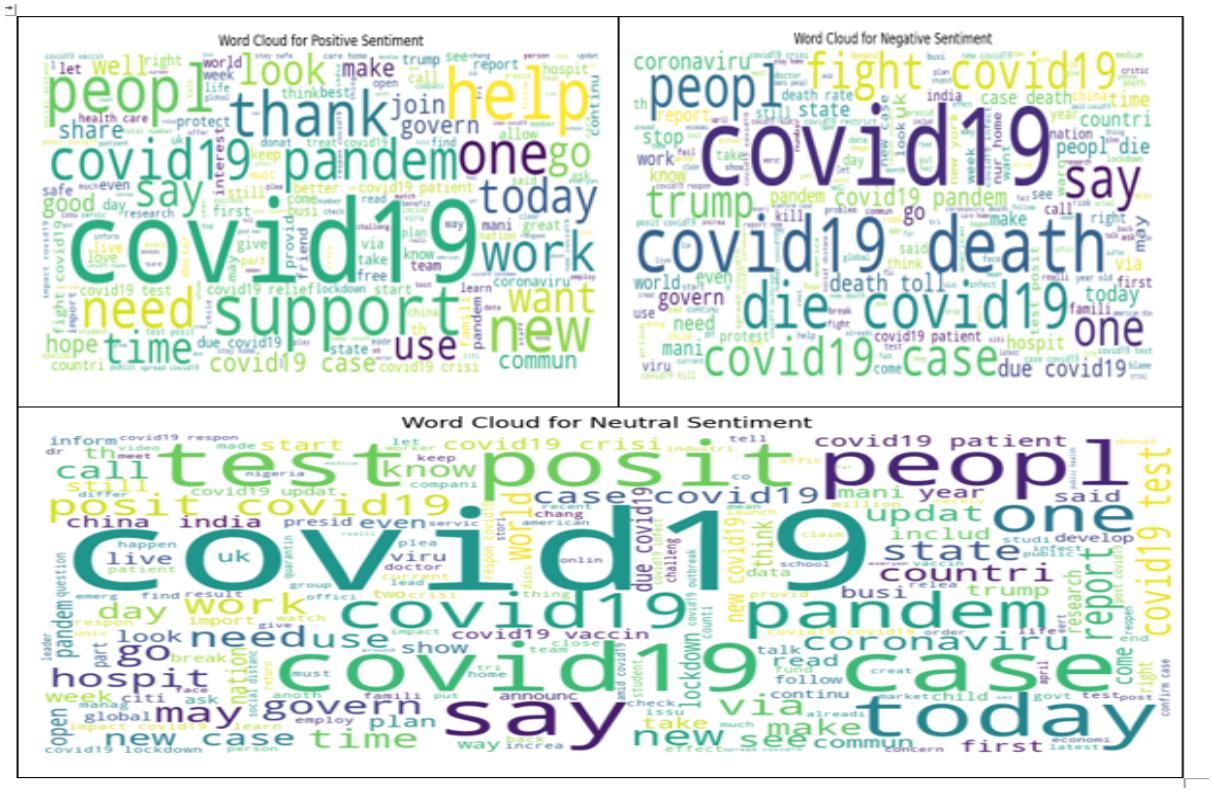


Figure 2: Word cloud.

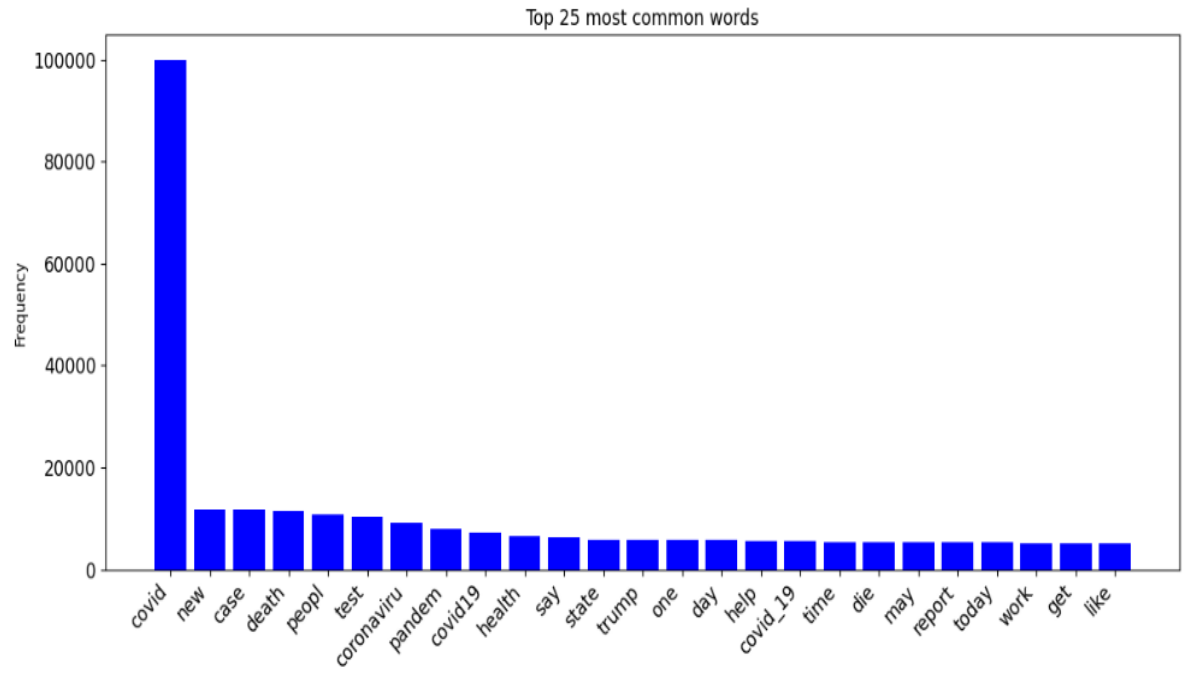


Figure 3:Plot of top 25 common words

TRADITIONAL MACHINE LEARNING MODELS

For this study, five traditional machine learning models are discussed from which two are chosen for implementation.

Logistic Regression

Logistic regression is a simple and interpretable linear classification model that is efficient but has limitations in handling complex data and non-linear relationships.

Strength

Simple to understand and implement.

Suitable for large dataset due to its computational efficiency.

Does not assume linear relationship between features. Useful for sentiment analysis, text categorization, and spam detection,

Weakness

Assumes feature independence which may limit its performance with interdependent text data.

Can be sensitive to imbalanced data.

Can overfit in the presence of many corelated features.

Naïve Bayes

Naive Bayes is a widely used algorithm in NLP due to its efficiency and simplicity. It is a top machine learning method for quick and accurate predictions Ji & Kwon (2023) and often performs better than other classification methods.

Strength

Performs well even with small dataset.

Effective for text classification tasks

Performs well with high-dimensional data.

Weakness

However, Naive Bayes assumes feature independence, which is a significant limitation. (Chavan, 2019).

Sensitive to imbalance data

K-Nearest Neighbours (KNN)

K-Nearest Neighbours (KNN) is a simple and effective algorithm for classification tasks in NLP. It can handle noisy data and outliers (Hodge & Austin, 2004).

Strength

Doesn't rely on strong assumptions about data distribution.

KNN is easy to implement and interpret, making it a popular choice for beginners and quick prototyping.

Weakness

However, as discussed by (Berrett et al., 2019) its computational inefficiency,

sensitivity to parameter choices, and

performance limitations in high-dimensional spaces

Support Vector Machines (SVM)

Support Vector Machines (SVM) are useful tools in Natural Language Processing (NLP) for their ability to handle high-dimensional data and capture complex patterns effectively.

Strength

SVMs can handle large feature spaces efficiently (Tian et al., 2012), making them suitable for tasks like text classification and sentiment analysis.

Works well with sparse data

Weakness

However, SVMs have some limitations in NLP, such as computational complexity (Zhang et al., 2006) and sensitivity to noisy data.

Suited more for binary classification, requires modifications to handle multi-class classification problem.

Random Forests

Random Forests are a powerful ensemble learning method that combines multiple decision trees to create a robust classifier.

Strength

In NLP, they handle high-dimensional data, Breiman (2001).

Prevents overfitting, making them a reliable choice for various applications (Cronin et al., 2017).

Provide high accuracy in classification tasks.

They outperform single decision trees,

Weakness

While their computational complexity (Waleed et al., 2021)

interpretability challenges.

Requires Significant Memory

In summary, Multinomial Naive Bayes is used for multi-class classification, especially for text and high-dimensional data. Random Forest is great for sentiment analysis and multi-class classification. Logistic Regression is best for binary classification, SVM is powerful but computationally expensive, and KNN is simple but can suffer from high-dimensional data. Hence for this study I use Naïve Bayes and Random Forest.

DEEP LEARNING MODELS

Recurrent Neural Networks (RNNs)

Strength

Recurrent Neural Networks (RNNs) are efficient in processing sequential data, making them suitable for tasks like language modelling, sentiment analysis, and machine translation Omara et al. (2022).

They are useful in capturing long-range dependencies in text data, which is crucial for understanding context and meaning in language.

Weakness

However, RNNs have limitations in NLP, including the vanishing gradient problem and struggles with handling very long sequences.

computational inefficiency especially with large datasets.

Convolutional Neural Networks (CNNs)

Strength

CNNs are effective in image processing tasks, but they can also be used for NLP tasks such as text classification and sentiment analysis, Koupaee & Wang (2018).

They can learn hierarchical features from text data, making them suitable for tasks that require understanding textual semantics.

Pooling layers in CNN reduce dimension of data, helping in preventing overfitting.

Weakness

However, they have limited ability to capture long-range dependencies and sequential information compared to RNNs.

Additionally, their interpretability can be challenging due to the hierarchical feature learning process.

Long Short-Term Memory (LSTM) networks

Strength

LSTM models in NLP have several advantages: they can capture long-term dependencies, learn sequential patterns, and outperform traditional models in certain tasks (Maarouf et al., 2021 & Khano, 2023).

Addresses the vanishing gradient problem through their gating mechanism.

Excels at capturing sequential information.

Weakness

computational complexity and

limitations in handling long-range dependencies

sensitive to Hyper parameters.

Transformer models

Transformer models have revolutionized Natural Language Processing (NLP) ,

Strength

utilize self-attention mechanisms to capture long-range dependencies in text data, Vaswani et al. (2017).

Availability of pre-trained models like BERT.

They scale well with data.

Transformers can process words in parallel and have demonstrated exceptional performance across various NLP tasks, making them highly effective for multi-task learning scenarios.

Weakness

However, Transformer models have some limitations, including their computational complexity and scalability for long sequences (Qin et al., 2022).

Gated Recurrent Units (GRUs)

GRUs are popular in NLP due to their unique architecture that enables efficient training and processing of sequential data (Khano, 2023).

Their simpler gating mechanism accelerates training and makes them computationally efficient for tasks that require processing long sequences of text data.

GRUs capture dependencies in sequential data effectively and are adept at modelling long-term dependencies in text data.

weakness

However, they may struggle with understanding context in extremely long sequences, which can affect their performance in tasks that require a deep understanding of text semantics.

For this study LSTM and GRU are preferred. This is due to their ability to handle sequential data, capture long-term dependencies, and address challenges like the vanishing gradient problem. Their complementary strengths make them ideal choices for sentiment analysis of COVID-19 tweets, providing a robust solution for understanding public sentiment during the pandemic.

Libraries Used

For this study, the following Libraries were used:

1. **NumPy (np)**:
2. **Pandas (pd)**: .
3. **re**: Regular expression operations for pattern matching in strings.
4. **Warnings**:
5. **Seaborn (sns)**:
6. **Matplotlib.pyplot (plt)**:
7. **NLTK (nltk)**:
8. **scikit-learn (sklearn)**:

- **ensemble:** RandomForestClassifier.
- **feature_extraction.text**: CountVectorizer and TfidfVectorizer.
- **Metrics**: accuracy_score, classification_report, and confusion_matrix
- **Model_selection**: GridSearch, train_test_split
- **Naïve Bayes**: MultinomialNB
- **Preprocessing**: LabelEncoder
- **Utils:resample**

9. **TensorFlow (tensorflow)**:
    - **keras.layers**: , Embedding, GRU, LSTM, and Dense.

- **keras.models**: Sequential.
- **keras.preprocessing.sequence**: pad_sequences.
- **keras.preprocessing.text**: Tokenizer.
- **keras. utils**: e to_categorical.

10. **Gensim**: Word2Vec model.
11. **WordCloud**: Library for creating word clouds from text data.

Implementation and refinement

1. **Data Loading and Exploration**: After loading the dataset with pandas, I conducted exploratory data analysis to visualize word cloud etc.
2. **Preprocessing**: Tokenized, lemmatized, and cleaned text by removing stopwords and irregular substrings using regex.
3. **Balancing Dataset**: The minority classes were oversampled to match the majority class, resulting in a balanced dataset.
4. **Modelling with Traditional ML Algorithms**: trained and evaluated models such as Naive Bayes and Random Forest using both Count-based Bag-of-Words (CBOW) and TF-IDF vectorization.
5. **Modelling with Deep Learning Algorithms**: LSTM and GRU models along with pre-trained Glove word embeddings to classify the sentiment.
6. **Evaluation**: The models were evaluated using various metrics such as accuracy, confusion matrix, and classification report, and the training history was also plotted for analysis.

Strategies for Fine-tuning:

1. **Hyperparameter Tuning**: Hyperparameter tuning was performed on models such as Naive Bayes and LSTM and GRU to improve performance by finding the best combination of hyperparameters. Batch size was increased to 64 , with a learning rate of 0.001.However, there wasn't any improvement in the results.

MODEL EVALUATION

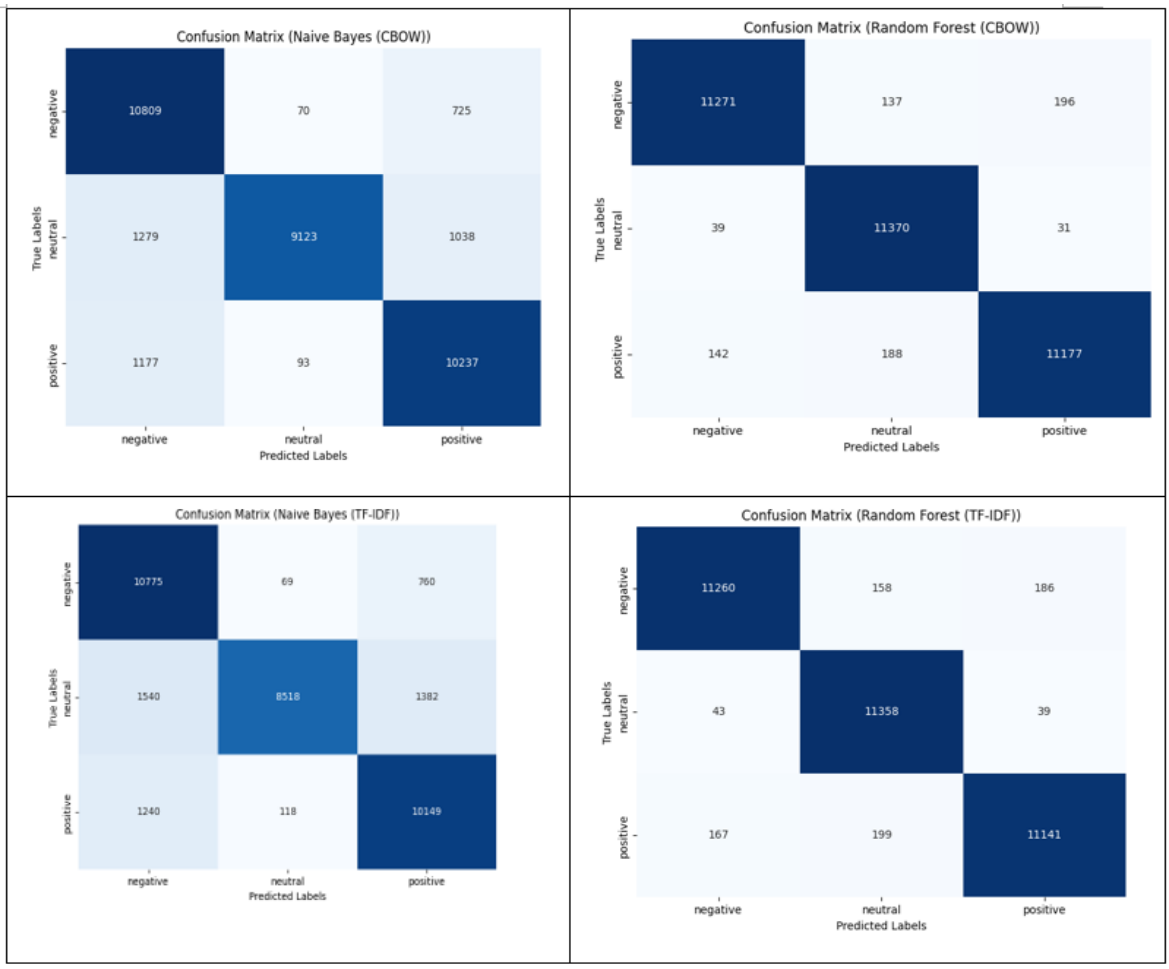
Traditional, machine learning models

Model Performance in classifying positive tweets.

Table 1:Positive Tweets analysis with traditional machine learning models

Model	vectorizer	Precision	Recall	F1-score	Accuracy
Naïve Bayes	CBOW	0.85	0.89	0.87	0.87
Naïve Bayes	TF-IDF	0.83	0.88	0.85	0.85
Random Forest	CBOW	0.98	0.97	0.98	0.98
Random Forest	TF-IDF	0.98	0.97	0.97	0.98

Table 1 displays the performance of traditional machine learning models in predicting positive tweets. The Random Forest algorithm achieves excellent results, with an average accuracy of 97% using both the CBOW and TF-IDF vectorizers. The F1-score, precision, and recall metrics are all high, indicating accurate classification of positive tweets. Overall, Random Forest is the strongest model for identifying positive tweets, with exceptional and consistent performance across all metrics regardless of the vectorizer used.



Model Performance in classifying Negative tweets.

Table 2: Negative tweets analysis with traditional machine learning models.

Model	vectorizer	Precision	Recall	F1-score	Accuracy
Naïve Bayes	CBOW	0.81	0.93	0.87	0.87
Naïve Bayes	TF-IDF	0.79	0.93	0.86	0.85
Random Forest	CBOW	0.98	0.97	0.98	0.98
Random Forest	TF-IDF	0.98	0.97	0.98	0.98

Table 2 shows that Random Forest is the most effective model for accurately identifying negative tweets. Both CBOW and TF-IDF vectorizers perform well with Random Forest. Naive Bayes also performs well, but CBOW outperforms TF-IDF for negative tweet classification. Overall, Random Forest with either CBOW or TF-IDF is the best choice for identifying negative tweets. If maximizing the number of negative tweets identified is the priority, Naive Bayes with CBOW might be a viable alternative.

Performance Table: Neutral Tweets

Table 3: Neutral Tweets analysis for traditional Machine learning models

Model	vectorizer	Precision	Recall	F1-score	Accuracy
Naïve Bayes	CBOW	0.98	0.80	0.88	0.87
Naïve Bayes	TF-IDF	0.98	0.74	0.85	0.85
Random Forest	CBOW	0.97	0.99	0.98	0.98
Random Forest	TF-IDF	0.97	0.99	0.98	0.98

From table 3, Both Random Forest and Naive Bayes with CBOW/TF-IDF can effectively classify neutral tweets. The recall values for all models are high, indicating that they excel at capturing a large portion of neutral tweets. Naive Bayes has very high precision, while Random Forest has high recall. If minimizing false positives is crucial, Naive Bayes might be preferable, and if capturing all or most neutral tweets is most important, Random Forest is a strong choice.

Table 4:Overall Accuracy for Traditional machine learning models

Model	**vectorizer**	**Accuracy**
Naïve Bayes	CBOW	0.87
Naïve Bayes	TF-IDF	0.85
Random Forest	CBOW	0.98
Random Forest	TF-IDF	0.98

From table 4, Naïve Bayes achieved an accuracy of 87% with CBOW against 85% with TFIDF.from tis we deduce that for dataset, the simpler representation of word counts might be more effective for the Naïve Bayes classifier. On the other hand, Random Forest performed exceptionally well with both vectorizers, achieving high accuracy of 98 %. This suggests that Random Forest is very effective in handling the features extracted by both CBOW and TF-IDF for sentiment classification.

DEEP LEARNING

Performance Table: Positive Tweets

Table 5: positive tweets analysis for deep learning models with Glove embedings

Model	**Precision**	**Recall**	**F1-score**	**Accuracy**
LSTM	0.98	0.98	0.98	0.98
GRU	0.97	0.97	0.97	0.98

From table 5: Both LSTM and GRU models have exceptional results (Precision: 0.98, Recall: 0.98, F1-score: 0.98) in identifying positive tweets when Glove embeddings are used in vectorising LSTM and GRU.

Performance Table: Negative Tweets

Table 6:Negative tweets analysis for deep learning models with Glove embedings

Model	Precision	Recall	F1-score	Accuracy
LSTM	0.98	0.98	0.98	0.98
GRU	0.98	0.98	0.98	0.98

From table 6, Both LSTM and GRU models perform exceptionally well in classifying negative tweets, with identical values in all metrics when Glove embeddings are used in vectorising LSTM and GRU.

Performance Table: Neutral Tweets

Table 7:Neutral tweets analysis for deep learning models with glove embedings

Model	Precision	Recall	F1-score	Accuracy
LSTM	0.99	0.98	0.98	0.98
GRU	0.98	0.98	0.98	0.98

Exceptional Performance for Neutral Tweets:

From table 7 Both LSTMs and GRUs perform well in identifying neutral tweets, achieving high values across all metrics.

Table 8:Positive tweets analysis for deep leaning without glove embeddings

Model	Precision	Recall	F1-score	Accuracy
LSTM	0.99	0.98	0.98	0.98
GRU	0.98	0.97	0.97	0.97

Table 9:Negative analysis for deep learnig without glove embedings

Model	**Precision**	**Recall**	**F1-score**	**Accuracy**
LSTM	0.98	0.98	0.98	0.98
GRU	0.97	0.97	0.97	0.98

Table 10:Neutral tweets analysis for deep learning with glove embedings

Model	**Precision**	**Recall**	**F1-score**	**Accuracy**
LSTM	0.98	0.98	0.98	0.98
GRU	0.97	0.98	0.97	0.98

Table 11 Overall: Accuracy for deep learning models

Model	**vectorizer**	**Accuracy**
LSTM	WITHOUT GLOVE EMBEDINGS	0.98
GRU	WITHOUT GLOVE EMBEDINGS	0.97
LSTM	WITH GLOVE EMBEDINGS	0.97
GRU	WITH GLOVE EMBEDINGS	0.98

The LSTM models achieved a high accuracy of 0.98 without pre-trained embeddings. The GRU models achieved an accuracy of 0.97 without pre-trained embeddings. When LSTM models used GloVe embeddings, the accuracy was 0.97. GRU models with GloVe embeddings achieved an accuracy of 0.98.

Summary and Conclusions

Although Naive Bayes is a strong baseline model for text classification due to its simplicity and effectiveness with small datasets, its performance is lower compared to Random Forest for this specific task. The slight difference in accuracy between CBOW and TF-IDF suggests that CBOW might capture the necessary features slightly better for Naive Bayes.

The high accuracy of Random Forest with both vectorizers indicates its robustness and capability in handling text data for sentiment analysis. It effectively captures complex patterns in the data, leading to superior performance.

For Deep learning, LSTM without GloVe embeddings and GRU with GloVe embeddings, both achieving 0.98 accuracy.

Recommendations

Consider Naïve Bayes with CBOW for simplicity and speed, or Random Forest with CBOW/TF-IDF for higher accuracy. Explore hybrid approaches for optimized performance. Depending on resources, use LSTMs without embeddings for ample data and time, or GRUs with embeddings for faster training and pre-existing word relationships.

REFERENCING

Referencing styles are vital for academic writing, ensuring consistency, giving credit to original authors, and facilitating source location.

Here's a comparison of MLA, APA, Chicago, Harvard, and Vancouver styles:

1. MLA: Primarily used in humanities.

2. APA: Common in social sciences.

3. Chicago: Versatile, used in various disciplines.

4. Harvard: Widely used in many fields, especially in the UK and Australia.

5. Vancouver: Predominantly used in medical and scientific papers.

Key Differences:

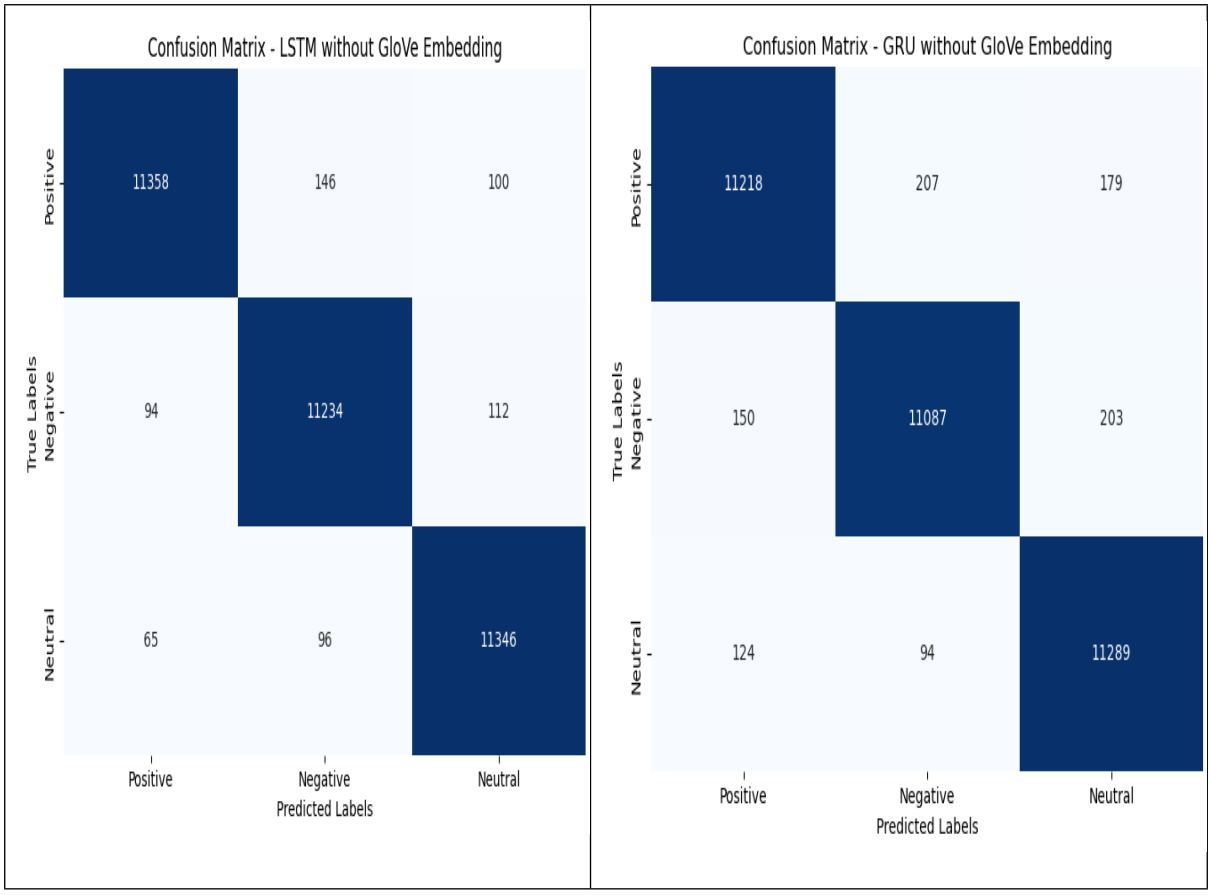
- In-text Citation: Varies across styles.

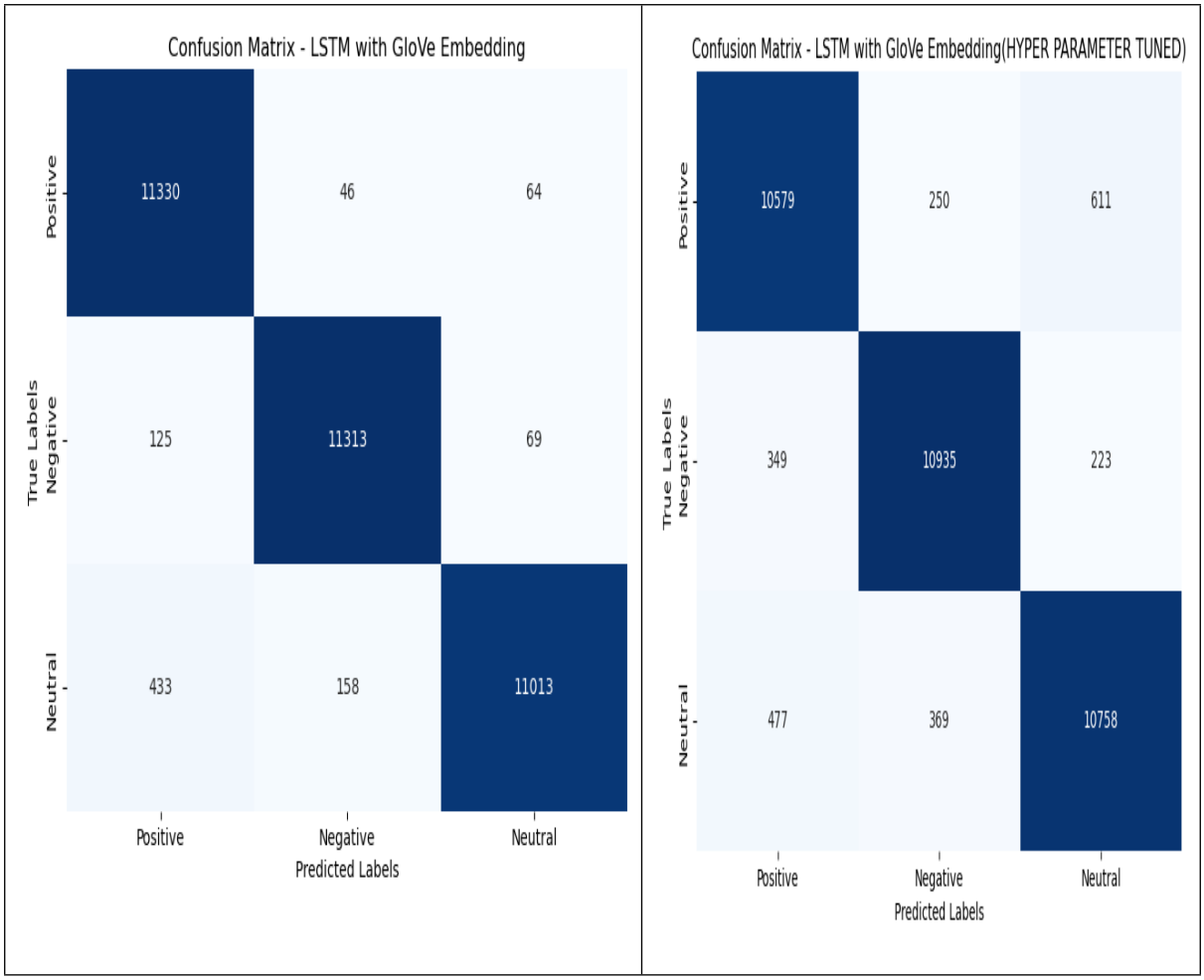
- Reference List Name: Different for each style.

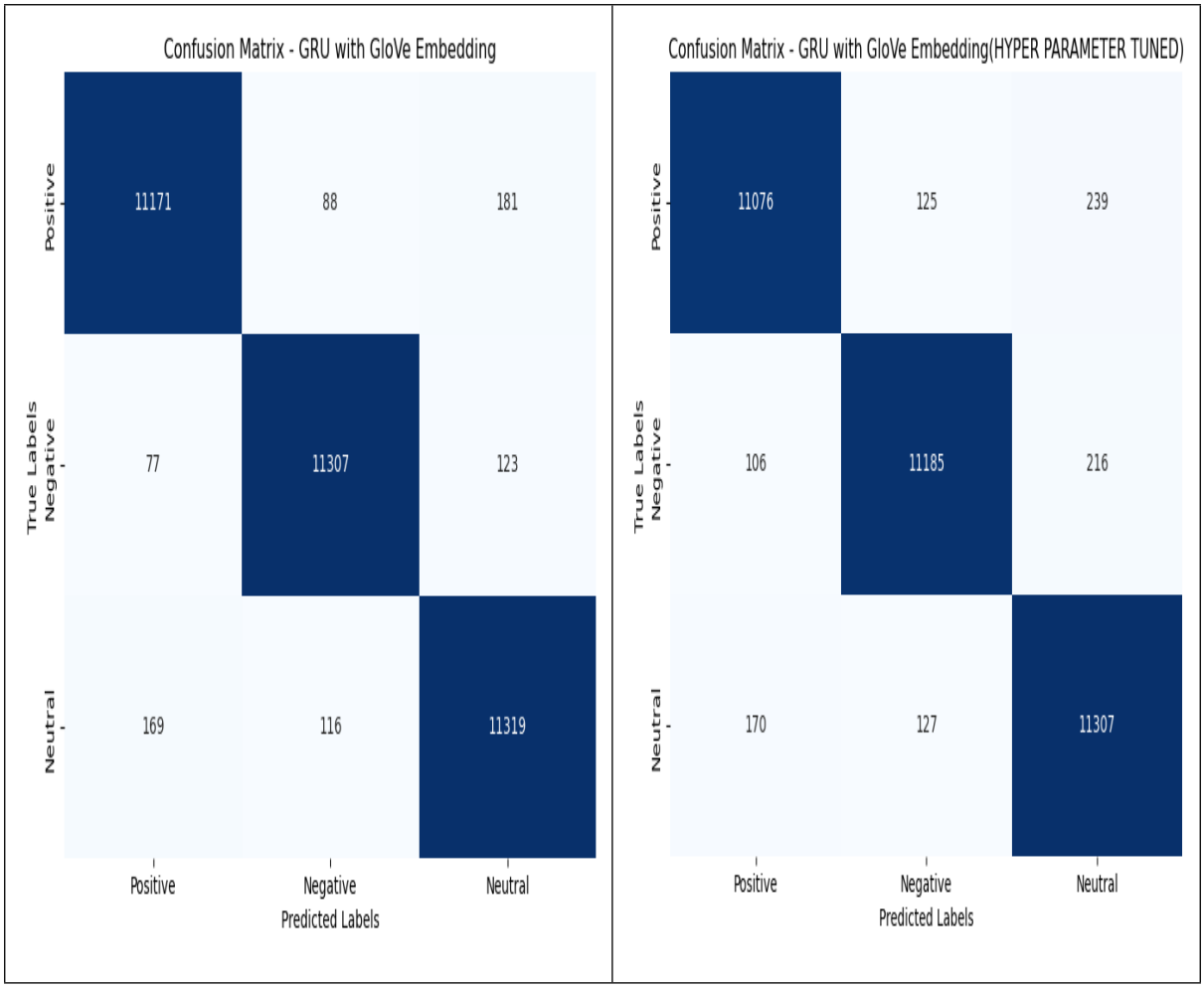
- Order in Reference List: Varies across styles.

- Detail Level: Varies across styles.

For this study the Harvard referencing style is adopted. Its widely used across many fields of study.


Confusion Matrix - LSTM without GloVe Embedding
True Labels
Positive
Negative
Neutral
11358
146
100
94
11234
112
65
96
11346
Positive
Negative
Neutral
Predicted Labels
Confusion Matrix - GRU without GloVe Embedding
True Labels
Positive
Negative
Neutral
11218
207
179
150
11087
203
124
94
11289
Positive
Negative
Neutral
Predicted Labels


Confusion Matrix - LSTM with GloVe Embedding
True Labels
Positive
Negative
Neutral
11330
46
64
125
11313
69
433
158
11013
Positive
Negative
Neutral
Predicted Labels
Confusion Matrix - LSTM with GloVe Embedding(HYPER PARAMETER TUNED)
True Labels
Positive
Negative
Neutral
10579
250
611
349
10935
223
477
369
10758
Positive
Negative
Neutral
Predicted Labels


Confusion Matrix - GRU with GloVe Embedding
True Labels
Positive
Negative
Neutral
11171
88
181
77
11307
123
169
116
11319
Positive
Negative
Neutral
Predicted Labels
Confusion Matrix - GRU with GloVe Embedding(HYPER PARAMETER TUNED)
True Labels
Positive
Negative
Neutral
11076
125
239
106
11185
216
170
127
11307
Positive
Negative
Neutral
Predicted Labels

REFERENCES

1. Aziz, R. H. H., & Dimililer, N. (2020). Twitter Sentiment Analysis using an Ensemble Weighted Majority Vote Classifier. In 2020 International Conference on Advanced Science and Engineering (ICOASE) (pp. 103-109). Duhok, Iraq. doi: 10.1109/ICOASE51841.2020.9436590
2. Berrett, T., Samworth, R., & Yuan, M. (2019). Efficient multivariate entropy estimation via k-nearest neighbor distances. The Annals of Statistics, 47(1). doi: 10.1214/18-aos1688
3. Breiman, L. (2001). Untitled. Machine Learning, 45(1), 5-32. doi: 10.1023/a:1010933404324
4. Chavan, R. (2019). A literature review on hierarchical naive Bayes classifier. International Journal for Research in Applied Science and Engineering Technology, 7(4), 179-180. doi: 10.22214/ijraset.2019.4032
5. Cronin, R., Fabbri, D., Denny, J., Rosenbloom, S., & Jackson, G. (2017). A comparison of rule-based and machine learning approaches for classifying patient portal messages. International Journal of Medical Informatics, 105, 110-120. doi: 10.1016/j.ijmedinf.2017.06.004
6. Dhanani, J., Mehta, R., & Rana, D. (2021). Sentiment weighted word embedding for big text data. International Journal of Web-Based Learning and Teaching Technologies, 16(6), 1-17. doi: 10.4018/ijwltt.20211101.oa2
7. Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. Artificial Intelligence Review, 22(2), 85-126. doi: 10.1023/b:aire.0000045502.10941.a9
8. Ji, K., & Kwon, Y. (2023). New spam filtering method with Hadoop tuning-based MapReduce naive Bayes. Computer Systems Science and Engineering, 45(1), 201-214. doi: 10.32604/csse.2023.031270
9. Kaur, H., Ahsaan, S. U., Alankar, B., et al. (2021). A Proposed Sentiment Analysis Deep Learning Algorithm for Analyzing COVID-19 Tweets. Inf Syst Front, 23, 1417–1429. doi: 10.1007/s10796-021-10135-7

10. Khano, M. (2023). Sentiment analysis with long-short term memory (lstm) and gated recurrent unit (Gru) algorithms. Barekeng Jurnal Ilmu Matematika Dan Terapan, 17(4), 2235-2242. doi: 10.30598/barekengvol17iss4pp2235-2242

11. Koupaee, M., & Wang, W. (2018). Analysing and interpreting convolutional neural networks in NLP. doi: 10.48550/arxiv.1810.09312

12. Maarouf, O., Ayachi, R., & Bahaj, M. (2021). Amazigh part-of-speech tagging with machine learning and deep learning. Indonesian Journal of Electrical Engineering and Computer Science, 24(3), 1814. doi: 10.11591/ijeecs. v24.i3.pp1814-1822

13. Mutinda, J. (2022). Sentiment analysis on text reviews using lexicon selected-bert embedding (lebert) model with convolutional neural network.

14. Omara, E., Mousa, M., & Ismail, N. (2022). Character-gated recurrent neural networks for Arabic sentiment analysis. Scientific Reports, 12(1). doi: 10.1038/s41598-022-13153-w

15. Purwarianti, A., & Crisdayanti, I. (2019). Improving bi-lstm performance for Indonesian sentiment analysis using paragraph vector. doi: 10.1109/icaicta.2019.8904199

16. Qin, G., Feng, Y., & Durme, B. (2022). The nlp task effectiveness of long-range transformers. doi: 10.48550/arxiv.2202.07856

17. Tian, Y., Shi, Y., & Liu, X. (2012). Recent advances on support vector machine research. Technological and Economic Development of Economy, 18(1), 5-33.