Unraveling Sentiments: A Comprehensive Exploration of Emotion Detection from Text

Ahasan Habib Sourov
Dept. of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
ahasanhabib0503@gmail.com

Asadullah Al Muqeet
Dept. of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
twist61904@gmail.com

Jakwan Al-Din
Dept. of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
jakwanaldin@gmail.com

Tanvirul Islam Tajvir
Dept. of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
tanvirulislamtajvir@gmail.com

Rasel Jumar
Dept. of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
raseljumar@gmail.com

***Abstract*—This thesis explores emotion detection, a critical facet of natural language processing, aimed at discerning sentiments conveyed in textual data. Emotion detection holds profound significance in understanding human communication, offering insights into user experiences, sentiment analysis, and customer feedback. The study rigorously evaluates diverse machine learning models, including Logistic Regression (LR), Random Forest (RF), Convolutional Neural Network (CNN), Neural Network (NN), and Bidirectional Encoder Representations from Transformers (BERT), in the realm of emotion classification. Findings indicate that while LR, RF, CNN, and NN exhibit competitive performances, BERT achieves the highest accuracy of 62.0%, outperforming all other models.**

***Index Terms*—Logistic Regression (LR), Random Forest (RF), Convolutional Neural Network (CNN), and Neural Network (NN), Bidirectional Encoder Representations from Transformers (BERT)**

I. Introduction

In today's digital age, people share their emotions across a plethora of online platforms and communication channels. Some prominent avenues include Platforms like Facebook, Twitter, Whats App, Telegram, etc. Emotion recognition involves identifying human emotions, a task that humans naturally perform by observing facial expressions, verbal cues, body language, and other contextual cues. However, achieving the same level of accuracy using an automated system presents significant challenges. Emotion detection from text has emerged as a pivotal domain within the realm of natural language processing (NLP), harnessing the power of machine learning and deep learning models to decipher the intricate nuances of human emotions conveyed through written language. This research endeavors to delve into the intricacies of this fascinating field, aiming to develop robust and accurate models capable of discerning emotional states from a diverse textual source.

Understanding human emotions is fundamental to enhancing human-computer interaction, sentiment analysis, and personalized user experiences. As the digital landscape continues to evolve, with an overwhelming volume of text being generated on various platforms, the need for automated emotion detection becomes imperative. This research seeks to address the pressing demand for more nuanced and context-aware emotion recognition systems, fostering advancements in fields such as mental health monitoring, customer feedback analysis, and virtual assistant responsiveness.

II. Related Work

Before working on our research, we have reviewed some works that have been done related to our research.

In 2019, there was a study where the authors proposed a multi-task learning approach for hate speech detection combining related tasks of sentiment, emotion, and target analysis. Multiple emotion datasets were evaluated during model selection to determine which to use. Of all the emotion datasets, using the CrowdFlower corpus achieved the best Macro F1 score of 0.7870 on the HASOC 2019 benchmark hate speech detection task. The large labeled tweet dataset and coverage of relevant emotions like hate made CrowdFlower well-suited despite being considered noisier. The multi-task learning model combining CrowdFlower for emotion improved hate speech detection over the BERT baseline, demonstrating the utility of joint modeling with relevant affective phenomena.[1]

In 2022, there was a study where The authors proposed a multi-task learning approach for hate speech detection that jointly models related tasks of sentiment, emotion, and target analysis. For emotion analysis, the CrowdFlower dataset of 39,740 labeled tweets achieved the best results with a Macro-F1 score of 0.7870 for hate speech detection on the HASOC 2019 test set.[2]

In 2021, a study used machine learning and deep learning models like CNN, and BiLSTM[3]. The authors collected a corpus from online social platforms and labeled it through crowdsourcing. After preprocessing the data, they extracted features like TF-IDF and word embeddings. For sentiment analysis, CNN with word2vec gave the best accuracy of 83.18%. For emotion detection, BiLSTM with word2vec performed the best with 65.83% accuracy. The paper concludes that deep learning models outperform machine learning approaches for Bangla sentiment and emotion analysis.

In 2017, there was a study using Emotex, a supervised learning system to classify emotion in text[4]. It uses Twitter hashtags as labels to build a large training set automatically. A soft classification approach assigns probability scores over 4 emotion classes based on the Circumplex model. Emotex achieved 90% accuracy, outperforming lexical methods. It is applied to real-time tweet stream classification and emotion burst detection in live streams.

III. Dataset

People mainly express their emotions using text on Twitter. So, The primary source of data acquired was Twitter. The dataset consists of 39827 tweets tagged with 13 different attitudes. This public domain dataset was acquired via the data.world platform and supplied by @crowdflower, a data enrichment, mining, and crowdsourcing enterprise based in the United States.

TABLE I
Dataset for Experiment

Sentiment	*Number of Data*
Neutral	8598
Worry	8437
Happiness	5184
Sadness	5154
Love	3785
Surprise	2181
Fun	1775
Relief	1522
Hate	1322
Empty	822
Enthusiasm	758
Boredom	179
Anger	110
Total	39827

TABLE II
SOME EXAMPLES OF THE DATASET

Text	***Sentiment***
@kelcouch I'm sorry at least it's Friday?	Sadness
wants to hang out with friends SOON!	Enthusiasm
Hmmm. http:www.djhero.com is down	Worry
cant fall asleep	Nutral
@mrgenius23 You win ... SIGH Rakeem	Hate
Pats in philly at 2 am. I love it.Mmm cheesesteak. Miss my boyfriend but I love vacation.	Love
@softtouchme just answered you- never learned how to write in French- just basic stuff-	Empty
I'm at work	Relief
@havingmysay dude, that is my favorite sandwich place ever. ummm did you take PICTURES?	Happiess

IV. Methodology

A. Class Balancing

With 13 classes, Classes are imbalanced - each class is not evenly distributed, imbalance rate is less than 21.32% (anger - neutral). To reduce the imbalance rate we categorized the data into three primary emotions: Positive, Negative, and Neutral.

- Positive- Enthusiasm, Love, Fun, Happiness, Relief.
- Negative- Empty, Sadness, Worry, Hate, Boredom, Anger.
- Neutral- Surprise, Neutral.

By doing this, the imbalance rate came down to 13.17%, slightly above the general guideline.

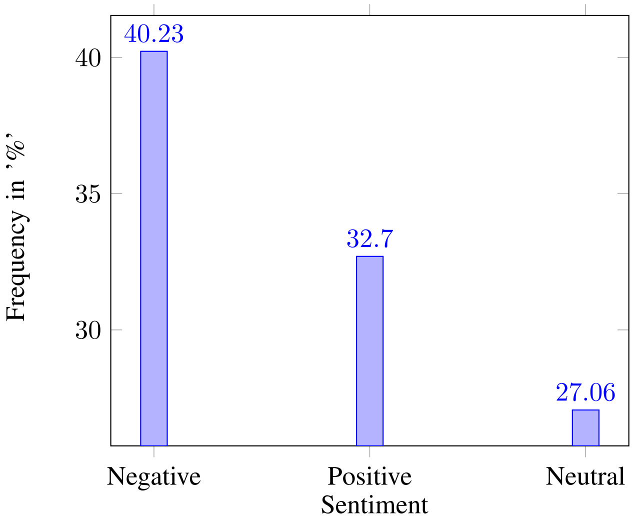

Fig. 1. Distribution of combined classes(Sentiment)

[twocolumn]article
graphicx

B. Data Cleaning

In this part we have removed all the special characters (any letter or a digit), removed all single characters (surrounded by

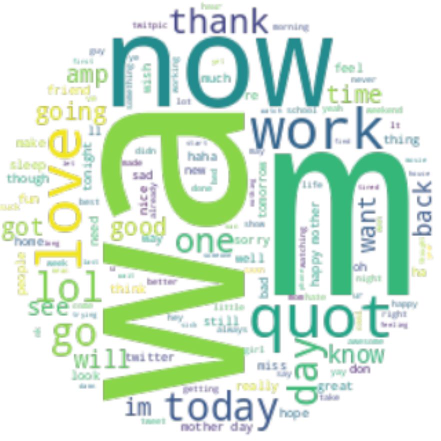

Fig. 2. Word Cloud

whitespace), removed single characters from the star, Substituted multiple spaces with single space, removed prefixed 'b', Converted all words to lowercase and lemmatization- splits into a list of words ['The', 'quick', ...]. we also delete all stopwords from data.

C. Tokenisation

Tokenization is a critical step in natural language processing (NLP) that entails dividing a chunk of text into smaller parts called tokens. Tokens might be words, phrases, sentences, or even single characters. We used NLTK's'sent_tokenize' method to tokenize the sentences.

D. Vectorization

Vectorizing is the process of turning non-numerical data, such as text or categorical variables, into a numerical representation that may be utilized as input to machine learning algorithms. We used "CountVectorizer" and "TF-IDF" to vectorize the data.

E. Models Architecture

We trained a Convolutional Neural Network (CNN) for text classification using the TensorFlow Keras library. The input text data was preprocessed by tokenizing and padding sequences. The CNN architecture comprised an embedding layer with a vocabulary size of max_words and an output dimension of 128. This was followed by a 1D convolutional layer with 64 filters of size 3 and a ReLU activation function. A global max-pooling layer was added to reduce spatial dimensions. Subsequently, a dense layer with 128 units and a ReLU activation function was employed, along with a dropout layer to mitigate overfitting. The final layer consisted of three units with a softmax activation function for multiclass classification.

The neural network architecture is composed of three layers: an input layer with 12 neurons and ReLU activation, a hidden layer with 8 neurons and ReLU activation, and an output layer with 3 neurons that use the softmax activation function for multi-class classification. The model is built using the categorical cross-entropy loss function and the Adam optimizer, using accuracy as the evaluation metric. Training takes place across 12 epochs, with a batch size of 50. The model is then tested on the training data, and its accuracy and F1 score (micro-average) are provided.

The BERT architecture integrates a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model with custom neural network layers tailored for natural language classification. Named BERT_Arch, the model leverages the rich contextual embeddings learned from the 'bert-base-uncased' variant, comprising 768-dimensional embeddings. Complementing the BERT backbone, custom layers include a dropout layer with a dropout rate of 0.3 to mitigate overfitting and two fully connected dense layers, with the first mapping the BERT pooled output to a hidden layer of 512 neurons and the second serving as the output layer with 3 neurons for class probability estimation. The training utilizes the AdamW optimizer with a learning rate of 1e-5 and the negative log-likelihood loss function with class weights, which are converted to tensors. During the forward pass, input sequences are tokenized and passed through BERT, with the pooled output sequentially processed through the custom layers, incorporating dropout and ReLU activation for regularization and non-linearity. This integrated architecture underscores the synergy between state-of-the-art transformer-based language representations and domain-specific neural network layers, facilitating effective natural language classification across various tasks.

TABLE III
MACHINE LEARNING MODEL ARCHITECTURES

Model	Architecture Description
CNN	• Embedding layer with a vocabulary size of 100 • Convolutional layer with 64 filters of size 3 • Followed by a max-pooling layer • A dense layer with 128 units and a ReLU activation function • Dropout for regularization • Softmax activation function for multiclass classification.
NN	• Simple NN structure with three layers • Input layer with 12 neurons and ReLU activation • hidden layer with 8 neurons and ReLU activation • Output layer with 3 neurons • Softmax activation function for multi-class classification
BERT	• 768-dimensional embedding layer • Followed by a dropout layer • A hidden layer of 512 neurons • Output layer with 3 neurons

The Logistic Regression model employed in this study follows an iterative optimization approach with a maximum of 1200 iterations. The logistic loss is minimized through

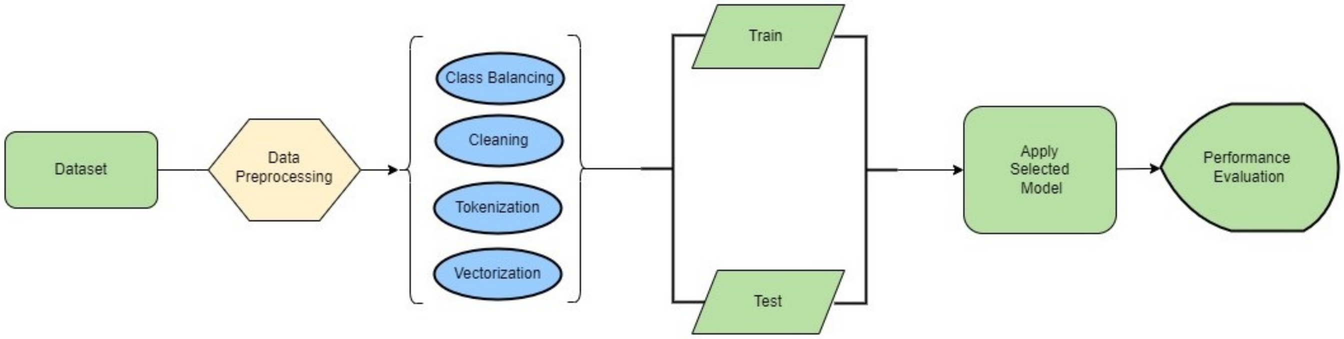

Fig. 3. FlowChart of The Methodology

the logistic regression algorithm, configured with a maximum iteration setting of 1200. Key components of the model include an input layer for accepting feature vectors, a linear transformation computing the weighted sum of input features, and an activation function applying the sigmoid function to generate class probabilities. This model, particularly suitable for binary and multi-class classification tasks, produces probabilistic outputs due to its linear nature.

The Random Forest model is utilized as an ensemble learning method with specific hyperparameter settings. In this research, the model is configured with default settings for the number of trees and employs the Gini impurity criterion for classification. Each decision tree is trained on a bootstrapped subset of the training data, ensuring diversity. Feature randomization further enhances the model's robustness by randomly selecting subsets of features for each tree. The final classification decision is determined by a majority vote across all decision trees. Random Forest, known for its versatility and ability to handle complex relationships in data, is well-suited for tasks that require ensemble-based learning and resilience to overfitting.

F. Evaluation Metrics

Multiple metrics were used to see how well each model sorts the emotions into groups: positive, negative, and neutral. Accuracy is the measurement of how many emotions from the dataset were identified correctly. This gives us a good sense of each of our model's performance. Precision examines the number of emotions that our model correctly classified as falling into a category. It indicates the accuracy of the model's category prediction. The number of real emotions in a category that our model was able to locate accurately is measured by recall. It indicates to us how well the model captures the emotion of a given category. Recall and precision are combined into one metric to create the F1 score. This lets us know how accurate and reliable the model is all around. All of these metrics combined provide us with a comprehensive understanding of how well each of our models classifies emotions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F1\text{-}score = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

V. Model Evaluation

A. Performance Evaluation

The Logistic Regression model yielded an accuracy of 55.2%, indicating a moderate level of overall correctness in its predictions. Precision and recall were consistent at 54.8% and 55.2%, respectively, suggesting a balanced performance in correctly identifying positive instances while minimizing false positives. The F1 score of 55.0% further reinforces the model's ability to strike a balance between precision and recall. While LR's linear nature might limit its capacity to capture complex relationships within the data, the model showcases reliability in its classification capabilities.

The Random Forest model achieved an accuracy of 54.4%, demonstrating comparable performance to LR. The precision, recall, and F1 scores were all around 54%, indicating a well-balanced trade-off between identifying true positives and avoiding false positives. The ensemble learning approach of RF, with its multiple decision trees, contributes to its versatility and resilience to overfitting. However, the model's performance might plateau when faced with intricate relationships within the dataset.

The CNN architecture, specifically designed for text classification, outperformed LR and RF with an accuracy of 55.8%. The precision, recall, and F1 scores were all approximately

55%, suggesting a balanced and effective classification performance. The CNN's ability to capture local patterns and relationships within the text sequences likely contributed to its superior performance. This model's slight advantage underscores the importance of leveraging deep learning architectures tailored to the characteristics of text data.

The Neural Network model, with its simple architecture comprising an input layer, a hidden layer, and an output layer, achieved an accuracy of 55.52%, the lowest among the models. The precision, recall, and F1 scores were consistently at 55.52%, indicating a balanced but modest performance. The simplicity of the model may have limited its ability to capture intricate patterns in the data compared to the more complex architectures like CNN. This suggests that for text classification tasks, a more intricate model may be necessary to extract and learn the nuanced features within the dataset.

The BERT model, leveraging the intricate architecture of Bidirectional Encoder Representations from Transformers, demonstrated exceptional performance with an accuracy of 61%, surpassing all other models in the evaluation. Precision, recall, and F1 scores were consistently high at 62.0%, 61.0%, and 61.0% respectively, reflecting its remarkable ability to discern and classify instances within the dataset accurately. The sophisticated architecture of BERT enabled it to capture nuanced semantic features, resulting in superior performance compared to simpler models. The model's comprehensive understanding of contextual relationships in language likely contributed to its outstanding predictive capability. These results underscore the efficacy of leveraging transformer-based models like BERT for text classification tasks, showcasing their potential to yield substantial improvements in accuracy and overall performance

In summary, the evaluation of text classification models highlights varying levels of efficacy. While simpler models yielded modest results, complex architectures like CNN demonstrated improved accuracy. Notably, BERT model's exceptional performance, leveraging sophisticated transformer-based architecture, underscores the importance of advanced techniques. These findings emphasize the necessity of selecting appropriate models to achieve optimal results in text classification tasks. Overall, this evaluation underscores the significance of leveraging state-of-the-art methodologies in natural language processing for superior performance.

TABLE IV
PERFORMANCE EVALUATION

Model	Accuracy	Precision	Recall	F1 score
LR	0.552	0.548	0.552	0.550
RF	0.544	0.540	0.544	0.542
CNN	0.558	0.554	0.558	0.555
NN	0.555	0.555	0.555	0.555
BERT	0.620	0.610	0.610	0.610

B. Effect of Data preprocessing on performance

This research emphasizes meticulous data preprocessing tailored for text-based machine learning models. Through rigorous text cleaning, including tokenization and removal of stopwords, the raw data was refined for enhanced pattern recognition. Techniques like stemming and lemmatization were applied for normalization, reducing dimensionality. Careful handling of missing data and noise, along with advanced vectorization methods like word embeddings, ensured a robust foundation for model training. This comprehensive preprocessing aimed to optimize model performance by providing a clean, structured, and representative dataset.

VI. FUTURE SCOPE

Further experimentation with hyperparameter tuning for each model could potentially enhance performance. Exploration of additional deep learning architectures or ensemble methods may provide insights into improving classification accuracy. Investigation into the impact of different preprocessing techniques and feature engineering on model performance.

VII. CONCLUSION

our study offers valuable insights into the effectiveness of various machine learning models utilized in our investigation of text classification. By rigorously evaluating Logistic Regression (LR), Random Forest (RF), Convolutional Neural Network (CNN), Neural Network (NN), and BERT, we have identified notable differences in their performances. Notably, BERT surpassed all other models, achieving the highest accuracy of 62%. This underscores the remarkable capability of transformer-based architectures to capture complex textual patterns. While LR and RF exhibited competitive performances, CNN demonstrated a slight advantage in capturing intricate patterns within the text data. Conversely, the NN model, with its simpler architecture, yielded more modest results. This study contributes to a comprehensive understanding of text classification models, empowering researchers and practitioners to navigate the diverse landscape of natural language processing with informed decision-making.

REFERENCES

[1] Anuradha Yenkikar and C. Narendra Babu, "SentiMLBench: Benchmark Evaluation of Machine Learning Algorithms for Sentiment Analysis" IJEEI, Vol. 11, No. 1, March 2023, pp. 318 336

[2] Plaza-del-Arco, Halat, Padó and Klinger, "Multi-Task Learning with Sentiment, Emotion, and Target Detection to Recognize Hate Speech and Offensive Language," arXiv:2109.10255v4 [cs.CL] 11 Jul 2022

[3] A. Al Jamil and R. Rahman, "Sentiment and Emotion Analysis from Textual Data in Bangla Language.", SEUSJCS, Vol. 01, No 01, June 2021.

[4] M. Hasan1, Elke R. and E. Agu1, "Automatic emotion detection in text streams by analyzing Twitter data," Int J Data Sci Anal 7, 35–51 (2019).