An Empirical Analysis of Sentiment Detection Using Deep Learning and Machine Learning Techniques

Dr G Yogeswararao[1], D. Ramya Satya Sree Devi[2], B. Santhosh Kumar [3], Ch. Vidya[4], G. Anand Rao[5]Associate Professor[1], Student[2,3,4,5]

Department of Computer Science and Engineering(Data Science)[1,2,3,4,5]

Avanthi Institute of Engineering and Technology (AVEN), Anakapalli, Andhra Pradesh, India

{yogi.gurubelli@gmail.com1,ramyadwarampudi2004@gmail.com[2], santhoshkumarbonnada@gmail.com[3], vidyachowduvada@gmail.com[4], anandgurram13@gmail.com[5]}@aiet.ac.in

ABSTRACT

Sentiment analysis is a Natural Language Processing (NLP) technique used to identify the emotional tone in textual data. This project focuses on analyzing and comparing the performance of various Machine Learning and Deep Learning algorithms for sentiment classification. Models such as K-Nearest Neighbors, Naive Bayes, Decision Tree, Random Forest, Logistic Regression, and Recurrent Neural Network are used to detect sentiments from text data. The dataset is processed using TF-IDF vectorization for machine learning models and tokenization with padding for the deep learning model. The system provides real-time sentiment prediction through a simple Streamlit web interface. It also displays confidence percentages and applies majority voting to determine the final sentiment. This approach helps improve the reliability of sentiment detection by comparing multiple models. The system can be used in applications like social media monitoring, product review analysis, and customer feedback evaluation.

I. INTRODUCTION

Sentiment analysis, also known as opinion mining, is an important application of Natural Language Processing that focuses on identifying the emotional tone or opinion expressed in textual data. With the rapid growth of digital platforms, a massive amount of text data is generated daily through social media posts, product reviews, blogs, forums, and online feedback systems. Analyzing this data helps organizations understand public opinion and customer attitudes. Businesses use sentiment analysis to evaluate customer feedback, monitor brand reputation, and improve products and services by identifying customer satisfaction levels from textual opinions. Earlier sentiment analysis methods mainly relied on rule-based and lexicon-based approaches, which often struggled with sarcasm, context understanding, and complex sentence structures. With advancements in Machine Learning and Deep Learning, sentiment detection has improved significantly. Machine Learning models can learn patterns from data, while Deep Learning models such as Recurrent Neural Networks (RNN) capture sequential relationships in text. In this project, multiple Machine Learning models are implemented and compared with an RNN model for sentiment classification. The system uses TF-IDF vectorization for ML algorithms and tokenization with padding for the RNN model, and a Streamlit-based interface allows users to input text and obtain real-time sentiment predictions from different models.

II. LITERATURE SURVEY

This section reviews key prior works in sentiment analysis and outlines the evolution of techniques used for extracting opinions from textual data. Sentiment analysis, a subfield of Natural Language Processing, focuses on identifying subjective information such as opinions, emotions, and attitudes from text. With the rapid growth of online platforms, massive amounts of user-generated data from social media, reviews, and feedback systems have driven extensive research in this domain.

Early approaches to sentiment analysis relied on statistical text representation methods. The Bag of Words model represents text as a collection of word frequencies, ignoring grammar and word order. While simple and computationally efficient, it fails to capture contextual relationships between words. To address this limitation, TF-IDF was introduced, which assigns higher importance to rare but meaningful words in a document. Although these methods improved feature representation, they still lacked the ability to understand semantic meaning.

Subsequently, machine learning techniques were widely adopted for sentiment classification. Algorithms such as K-Nearest Neighbors, Naive Bayes, Decision Tree, Random Forest, and Logistic Regression learn patterns from labeled data to classify sentiments. Among these, Naive Bayes is particularly effective due to its probabilistic nature and efficiency in handling high-dimensional text data, while Random Forest improves performance by reducing overfitting through ensemble learning. However, these methods are limited in capturing complex linguistic structures and contextual dependencies.

To overcome these limitations, deep learning approaches have been introduced. Models such as Recurrent Neural Networks are capable of processing sequential data and capturing contextual relationships between words. By considering word order and dependencies, these models learn deeper semantic patterns, leading to improved performance in sentiment analysis tasks compared to traditional machine learning approaches.

In addition to algorithmic advancements, the availability of powerful development tools has facilitated the implementation of sentiment analysis systems. Libraries such as TensorFlow, Keras, and Scikit-learn are widely used for building models, while Pandas and NumPy support data preprocessing. Visualization tools like Matplotlib and application frameworks such as Streamlit enable the development of interactive systems for real-time sentiment prediction.

III. METHODOLOGY

1. Data Collection and Preprocessing

The dataset is cleaned by removing noise, stop words, and unnecessary characters. Text is converted into numerical form using TF-IDF for machine learning models. For deep learning, tokenization and padding are applied. This ensures structured input for effective model training.

2. Model Training and Evaluation

Models such as Naive Bayes, Random Forest, and Recurrent Neural Networks are trained on the dataset. Each model learns patterns from text data for sentiment classification. Their performance is evaluated using accuracy and confidence scores. The best-performing models are identified through comparison.

3. System Design and Implementation

The system consists of preprocessing, model processing, and output layers. It is developed using Python with libraries like Scikit-learn and TensorFlow. A user interface is built using Streamlit. The final prediction is generated using a majority voting mechanism.

4. Deployment and Maintenance

The system is deployed as a web application using Streamlit for easy browser access. It provides real-time sentiment prediction to users. Maintenance is handled using GitHub for updates and bug fixes. Future improvements focus on performance optimization and feature enhancement.

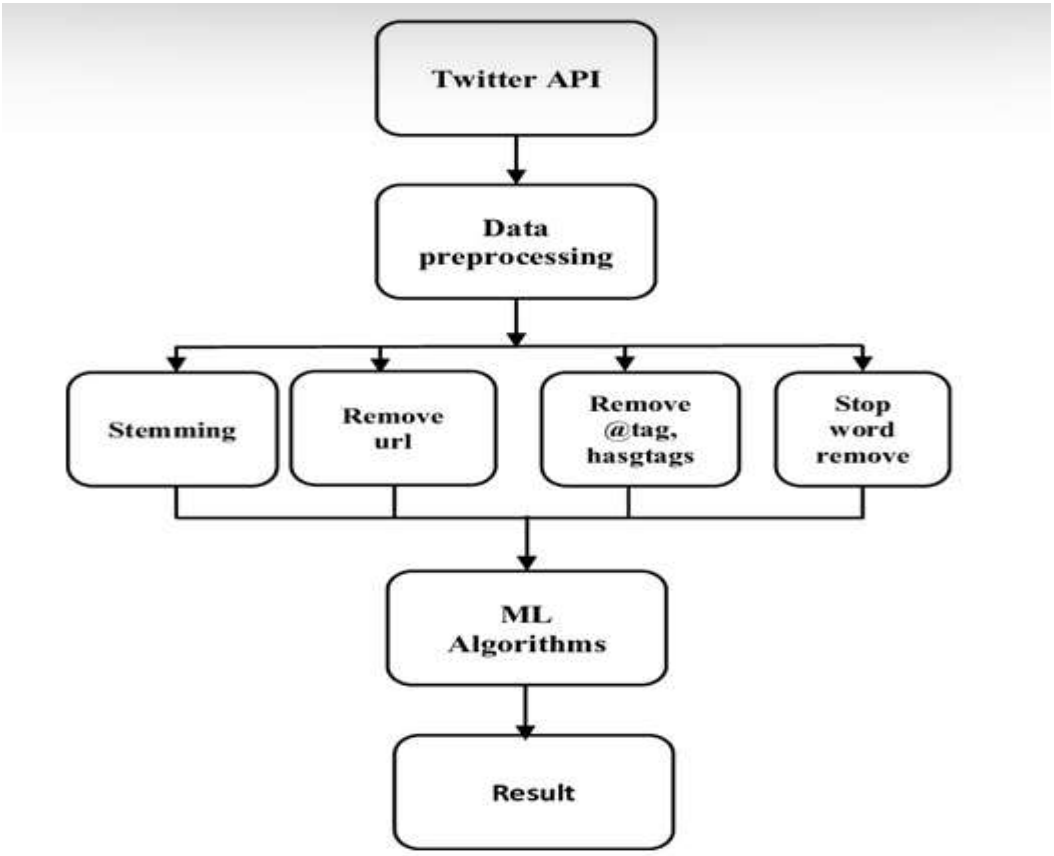
IV. SYSTEM ARCHITECTURE

A. System Architecture

The system follows a layered architecture designed for efficient sentiment analysis. It begins with the user interface developed using Streamlit, where users input text for analysis. The input is passed to the preprocessing layer, which cleans the text by removing noise, stop words, and special characters. The cleaned text is then transformed into numerical features using TF-IDF for machine learning models and tokenization with padding for deep learning.

The processed data is forwarded to the model layer, where multiple algorithms such as Naive Bayes, Random Forest, and Recurrent Neural Networks perform sentiment classification. Each model generates its prediction along with a confidence score.

These outputs are then passed to the decision layer, where a majority voting mechanism determines the final sentiment. The result is sent back to the interface layer for visualization. The system also includes a comparison module to analyze model performance based on confidence levels. Overall, the architecture ensures accurate, real-time sentiment prediction with an interactive user experience.


Twitter API
Data preprocessing
Stemming
Remove url
Remove @tag, hasgtags
Stop word remove
ML Algorithms
Result

V. ALGORITHMS

K-Nearest Neighbors (KNN)

K-Nearest Neighbors classifies text based on similarity with nearby data points. It uses distance metrics to find the closest neighbors. The majority class among neighbors determines the sentiment. It is simple but slow for large datasets.

Naive Bayes

Naive Bayes is a probabilistic algorithm based on Bayes' theorem. It assumes feature independence and works well for text data. It is fast, efficient, and widely used in sentiment analysis. However, the independence assumption is a limitation.

Decision Tree

Decision Tree splits data into branches based on feature conditions. It forms a tree structure to make decisions. It is easy to understand and interpret. However, it can overfit if not controlled.

Random Forest

Random Forest is an ensemble of multiple decision trees. It improves accuracy by combining predictions. It reduces overfitting compared to a single tree. However, it increases computational complexity.

Logistic Regression

Logistic Regression is used for binary classification problems. It predicts probability using a sigmoid function. It is simple, fast, and effective for linear data. However, it struggles with complex patterns.

Recurrent Neural Networks (RNN)

Recurrent Neural Networks process sequential data by maintaining memory of previous inputs. They capture context and word order in text. This makes them powerful for sentiment analysis. However, they require more data and computational power.

VI. SYSTEM MODULES

1. User Interface Module : The user interacts with the system through a web interface built using Streamlit. It allows users to input text such as reviews or sentences. It also displays the predicted sentiment and confidence scores. The interface is designed to be simple and user-friendly.

2. Data Preprocessing Module : This module cleans and prepares the input text for analysis. It removes stop words, punctuation, and unwanted characters. The text is then converted into numerical format using TF-IDF and tokenization. This step ensures better model performance.

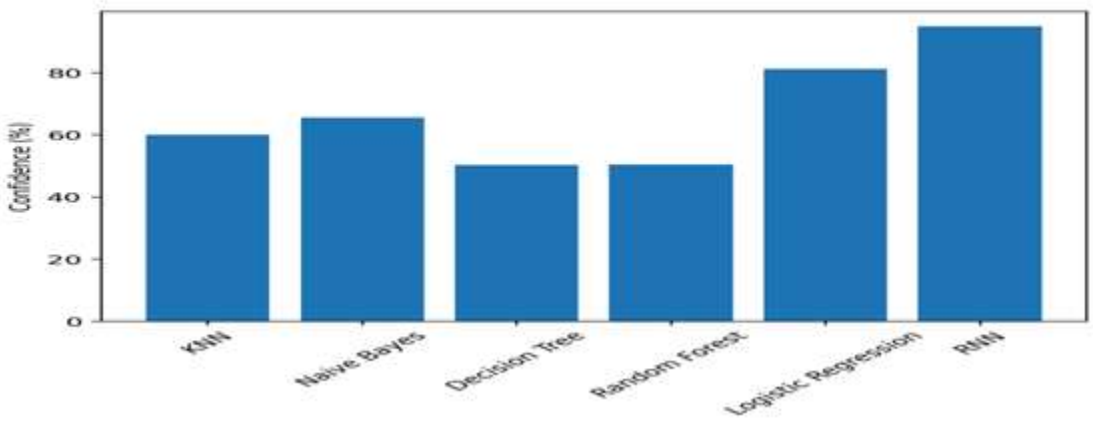
3. Feature Extraction Module : This module transforms processed text into meaningful feature vectors. TF-IDF is used for machine learning models, while tokenization and padding are used for deep learning. It helps models understand the importance of words. Proper feature extraction improves accuracy.

4. Model Processing Module : This module applies multiple algorithms such as Naive Bayes, Random Forest, and Recurrent Neural Networks. Each model predicts sentiment independently. It generates outputs along with confidence scores. This allows comparison between models.

5. Decision-Making Module : This module combines predictions from all models. A majority voting technique is used to determine the final sentiment. It ensures more reliable and balanced results. This improves overall system accuracy.

6. Result Visualization Module : The final output is displayed to the user through the interface. It shows sentiment labels such as positive, negative, or neutral. Confidence scores and comparisons are also presented. This helps users understand the prediction clearly.

VII. RESULTS



The system generates sentiment predictions based on the text input provided by the user through the Streamlit interface. Once the user enters a sentence or review and clicks the analyze button, the input text is processed through preprocessing and feature extraction stages before being passed to multiple trained models.

Each model independently predicts the sentiment of the given text as positive, negative, or neutral. Along with the predicted label, the system displays a confidence score, which represents the probability or certainty of the prediction made by that model. These confidence scores help users understand how strongly a model supports its prediction rather than just providing a final label.

To improve reliability, the system applies a majority voting mechanism, where the final sentiment is determined based on the most frequent prediction among all models. This approach reduces the dependency on a single model and provides a more balanced and accurate result.

In addition to the final output, the system also presents a comparison of model performance, showing how different algorithms respond to the same input. This comparison is typically visualized using charts or graphs, allowing users to easily identify which model performs better in terms of confidence and consistency.

Overall, the output is designed to be informative, transparent, and user-friendly, enabling users not only to view the predicted sentiment but also to understand the reasoning behind the decision through confidence values and model comparisons

VIII. CONCLUSION AND FUTURE WORK

Replace with transformer-based models like **BERT**, **RoBERTa**. These capture context better (e.g., sarcasm, long sentences). Expand to Indian languages: Telugu, Hindi, Tamil.Use multilingual models like **mBERT**

References

[1] ISCA, “Interspeech Proceedings,” Available: https://www.isca-speech.org/archive/interspeech_2010

[2] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed., Stanford University, Available: https://web.stanford.edu/~jurafsky/slp3/

[3] B. Pang and L. Lee, “Opinion Mining and Sentiment Analysis,” Foundations and Trends in Information Retrieval, Available: https://www.cs.cornell.edu/home/llee/papers/sentiment.pdf

[4] A. Maas et al., “Large Movie Review Dataset,” Stanford AI Lab, Available: https://ai.stanford.edu/~amaas/data/sentiment/