Sentiment Analysis in the Age of Machine Learning

*Shivam Rawat
**Prof. M.M.S. Rauthan

Abstract

Sentiment analysis, or opinion mining, is a key branch of natural language processing (NLP) that focuses on extracting subjective information from text. It plays a critical role in understanding emotions, opinions, and attitudes within the vast landscape of user-generated content, such as social media and reviews. This study explores the evolution of sentiment analysis, emphasizing the transformative impact of machine learning (ML). While early methods relied on lexicon-based and statistical approaches, modern advancements in deep learning—such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and transformers like BERT and GPT—have achieved remarkable progress. Applications span diverse fields, including social media monitoring, politics, healthcare, and business. Persistent challenges, such as sarcasm, contextual ambiguity, multilingual processing, and ethical considerations, are also examined. The study concludes by proposing future directions, including explainable AI, cross-domain adaptation, and multimodal sentiment analysis, to advance this dynamic field.

Keywords: Convolutional Neural Networks (CNNs),Recurrent Neural Networks (RNNs),Natural Language Processing (NLP),Machine learning (ML)

I. INTRODUCTION

The shift from Web 1.0 to Web 2.0 has greatly simplified the process for individuals to share and distribute their ideas, opinions, and perspectives online. This transition has led to an explosion of subjective content on the internet, fueling interest in collecting, analyzing, and leveraging this information. This has given rise to Sentiment Analysis (SA), which focuses on extracting and categorizing opinions from textual data. These insights serve diverse purposes, such as helping businesses understand customer preferences and improve product quality, guiding politicians in refining strategies based on public feedback, enabling stakeholders to assess activities through event reviews, and supporting public relations initiatives.

With the proliferation of user-generated content expressing emotions online, especially on social media, the sheer volume has become unmanageable for manual analysis. This has heightened the importance of automated opinion analysis across web platforms to support effective decisionmaking. However, this task remains complex, as it involves interpreting context and uncovering hidden sentiments within text. Therefore, developing innovative methodologies in this domain is essential.

The paper is structured as follows: Section II reviews existing literature, Section III describes the proposed system, Section IV outlines the experimental process and results, and Section V concludes with key findings and recommendations for future research.

II. RELATED WORK

The rapid growth of online reviews has brought significant attention to sentiment analysis, prompting numerous studies in this field. Jinyan Li et al. [1] conducted experiments on various filters during data preprocessing to determine their effects on sentiment analysis algorithms. Their results showed that sentiment-related trending words significantly influence predictions, and removing high-frequency words, especially class-specific ones, reduced prediction accuracy.

Huma Parveen and Shikha Pandey [2] studied the role of emoticons in preprocessing. They found that including emoticons improved accuracy since tweets often convey emotions through them. Conversely, removing emoticons increased the detection of neutral sentiments, while their inclusion reduced neutral tweets by converting emoticon-based sentiments into emotional categories.

Soumya S. and Pramod K.V. [3] used machine learning algorithms like Naive Bayes (NB), Support Vector Machines (SVM), and Random Forest (RF) to classify Malayalam tweets as positive or negative. They evaluated features such as Bag of Words (BOW), TF-IDF, Unigrams with SentiWordNet, and Unigrams with negation words. The RF classifier achieved the highest precision (95.6%) when using Unigrams with SentiWordNet and negation words.

Rashedul Amin Tuhin et al. [4] investigated sentiment analysis for Bangla text using Naive Bayes and a topical approach. These methods were applied at both the article and sentence levels. Their findings showed that Naive Bayes had lower precision at the article level due to its multiplication-based operations, while the topical approach performed consistently better by using summation and comparison techniques.

Poornima A. and K. Sathiya Priya [5] compared three machine learning algorithms—SVM, Multinomial Naive Bayes, and Logistic Regression—for Twitter data classification. Logistic Regression performed best, achieving 86.23% precision with a bigram model, followed by SVM (85.69%) and Multinomial Naive Bayes (83.54%).

Another study [6] explored sentiment analysis using the Yelp dataset. The authors applied methods including Naive Bayes, Multinomial Naive Bayes, Bernoulli Naive Bayes, Logistic Regression, and Linear SVC. While the models showed satisfactory precision during training, the authors emphasized the need for larger datasets to improve performance in real-world applications.

Shamantha Rai B. and Sweekriti M. Shetty [7] examined Twitter opinion mining for specific keywords, comparing Naive Bayes, SVM, and Random Forest. They observed that larger datasets increased execution time and noted that Random Forest was significantly slower than Naive Bayes and SVM.

Meylan Wongkar and Apriandy Angdresey [8] analyzed Twitter sentiment about Indonesia's 2019–2024 presidential candidates using a web crawler. Their comparison of Naive Bayes, SVM, and K-Nearest Neighbor (KNN) revealed that Naive Bayes had the highest accuracy (80.90%), outperforming KNN (75.58%) and SVM (63.99%).

Md. Golam Sarowar et al. [9] created a sentiment analysis classifier for Bengali e-commerce sites. They developed a Bengali StopWords database with 900 entries and used Python web scraping to collect Bangla comments and reviews. The data was tokenized using the NLTK library, filtered with StopWords, and converted into digital form using TF-IDF. They classified the data with KNN and SVM, comparing them to Logistic Regression, a CNN-based PCA model, and Random Forest. Their hybrid approach proved to be three times faster and more efficient than other methods.

III. Proposed System

Sentiment analysis is performed by ranking sentences using machine learning algorithms. The procedure starts with collecting tweets, followed by preprocessing, data preparation, classification, and concludes with the evaluation phase.

We opted for Python as the programming language due to its powerful tools and straightforward syntax. Anaconda was chosen as the development environment because it simplifies the installation of machine learning libraries. For natural language processing (NLP) tasks, we used the Natural Language Toolkit (NLTK) package, while machine learning tasks were handled using the open-source Scikit-learn library.

A) Data Collection Phase

This study utilizes a dataset of English tweets sourced from the NLTK package. The NLTK Twitter corpus comprises 20,000 neutral tweets, referred to as "twitter samples," obtained through the Twitter Streaming API. Additionally, it includes 10,000 sentiment-labeled tweets, evenly divided into 5,000 positive and 5,000 negative tweets.

B) Preprocessing Phase

Tweets often include slang, punctuation, and unstructured language, making them unsuitable for machine processing in their raw state. To prepare tweets for supervised machine learning algorithms, the following preprocessing steps are applied:

- **Data Tokenization**: This technique splits the text into sentences and further breaks each sentence into individual words, creating tokens.
- **Stop Word Removal**: Common words such as "is," "the," and "a" are removed, as they add little value to language processing.
- **URL Removal**: All URLs in the text, including those starting with "http," "https," or "pic://," are removed and replaced with an empty string.
- **Removing @ Mentions**: Twitter handles prefixed with "@" are excluded since they do not contribute meaningful sentiment information.
- **Lowercase Conversion**: The entire text is converted to lowercase to ensure uniformity and simplify processing.
- **Lemmatization**: This process normalizes words by analyzing their structure and context to align them with their root form. It uses vocabulary and morphological analysis for normalization. Before lemmatization, a tagging algorithm determines each word's context by identifying its grammatical role (e.g., noun, verb, adjective, adverb) based on its position in the sentence.

Figure 1 provides an illustration of the preprocessing process.

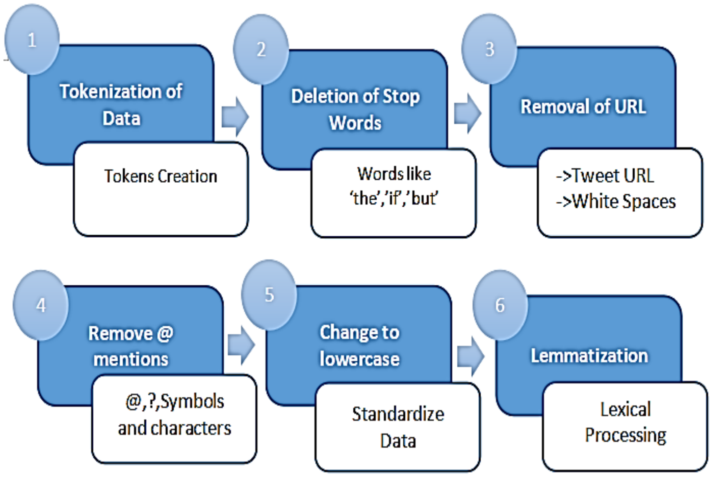


Figure 1 : Pretreatment Process

C) Data Preparation

In the data preparation stage, the tokens are converted into a Python dictionary format, where each word acts as a key with a value of True, and the data is randomized. The randomized dataset is then divided into two portions: one for model training and the other for performance evaluation. This division follows a 70:30 ratio, with 70% allocated for training and 30% for testing. Since the dataset comprises 10,000 tweets, the first 7,000 tweets from the randomized dataset are used to train the model, while the remaining 3,000 are reserved for testing its performance.

D) Classification Stage

After splitting the data into training and testing sets, machine learning algorithms can be applied to train the model using the training data. The following algorithms were employed:

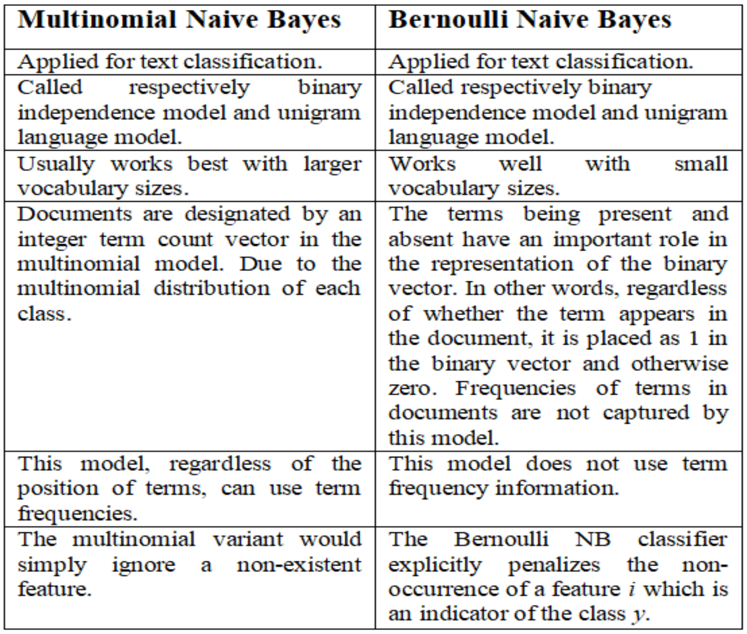
a) Naive Bayes (NB) Classification : Naive Bayes classifiers are straightforward yet highly effective supervised machine learning algorithms based on Bayes' theorem, proposed by Thomas Bayes for calculating probabilities. These classifiers support probabilistic predictions by determining the likelihood of an instance belonging to a particular class. A core assumption in Naive Bayes is that each feature is independent of others, given the class variable.

Two widely used variations of Naive Bayes were applied for classification: **Multinomial Naïve Bayes** and **Bernoulli Naive Bayes.**

- **Multinomial Naive Bayes:** This method is commonly applied in text classification and is also known as the binary independence model or unigram language model. It requires discrete features as input and performs best with larger vocabulary sizes. Each feature is assumed to follow a multinomial distribution, represented by a feature vector that captures the frequency of feature occurrences in a given instance.
- **Bernoulli Naive Bayes :** This variation treats features as independent binary variables, representing presence or absence rather than probabilities, as in the Multinomial Naïve Bayes. Often referred to as the binary independence model or unigram language model in literature, it is particularly suitable for datasets with smaller vocabulary sizes. Bernoulli Naïve Bayes is ideal for scenarios involving multiple entities, where each entity is treated as a binary variable. In text classification, word occurrence vectors are used for both training and classification.

Table I provides a comparison of Multinomial Naïve Bayes and Bernoulli Naïve Bayes.

Table I : Comparison of MNB and BNV



Multinomial Naive Bayes	Bernoulli Naive Bayes
Applied for text classification.	Applied for text classification.
Called respectively binary independence model and unigram language model.	Called respectively binary independence model and unigram language model.
Usually works best with larger vocabulary sizes.	Works well with small vocabulary sizes.
Documents are designated by an integer term count vector in the multinomial model. Due to the multinomial distribution of each class.	The terms being present and absent have an important role in the representation of the binary vector. In other words, regardless of whether the term appears in the document, it is placed as 1 in the binary vector and otherwise zero. Frequencies of terms in documents are not captured by this model.
This model, regardless of the position of terms, can use term frequencies.	This model does not use term frequency information.
The multinomial variant would simply ignore a non-existent feature.	The Bernoulli NB classifier explicitly penalizes the non-occurrence of a feature *i* which is an indicator of the class *y*.

b) Support Vector Machine (SVM) (Supervised Linear or Non-Linear Classification) : SVM is a popular supervised machine learning algorithm developed by Vapnik in 1992. It can be used as either a linear or non-linear classifier. Known as a "large margin classifier," SVM identifies the hyperplane with the maximum margin, which is the greatest distance between the hyperplane and the nearest data points on either side.

Several variations of SVM are available:

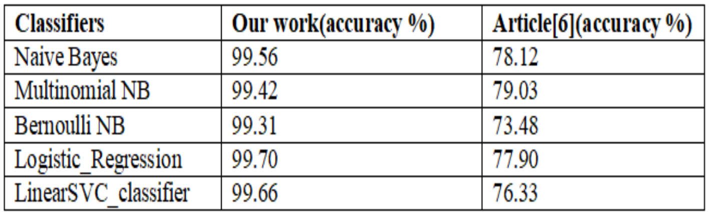
- SVC (Support Vector Classifier): A versatile method for support vector classification with adjustable parameters and a general mathematical framework.
- NuSVC (Nu-Support Vector Classification): Similar to SVC but employs a different set of parameters and mathematical formulation.
- LinearSVC (Linear Support Vector Classifier): A specific implementation of SVM tailored for cases requiring a linear kernel, optimized for linear classification scenarios.

IV. Experimentation and Results

The experiments were performed on an Intel® Core™ i3-6006U processor with a 2.00 GHz CPU and 4.00 GB of RAM. Using supervised learning techniques, we achieved a satisfactory level of accuracy in classifying Twitter reviews.

Compared to the findings in [6], our approach achieved superior precision. Table II presents a detailed comparison of the results.

Table II- Result Comparisons



Classifiers	Our work(accuracy %)	Article[6](accuracy %)
Naive Bayes	99.56	78.12
Multinomial NB	99.42	79.03
Bernoulli NB	99.31	73.48
Logistic_Regression	99.70	77.90
LinearSVC_classifier	99.66	76.33

The main goal of this study is to develop a system that can interpret and understand English sentences typed by humans and classify them as expressing either positive or negative sentiments.

The initial results show promising accuracy on the test dataset, although there is potential for improvement. While the model effectively classifies positive and negative sentiments, it struggles with more nuanced emotions, such as sarcasm, due to limited training data.

For this study, a dataset of sample tweets was prepared using the NLTK package for natural language processing (NLP). Various data cleansing techniques were applied to refine the dataset. Afterward, a model was trained on pre-classified tweets and used to classify new samples as positive or negative sentiments.

Python's NLTK package was used for all NLP tasks. The first step involved installing NLTK and downloading the sample tweets with the command `nltk.download('twitter_samples')`. Once the samples were downloaded, data processing began.

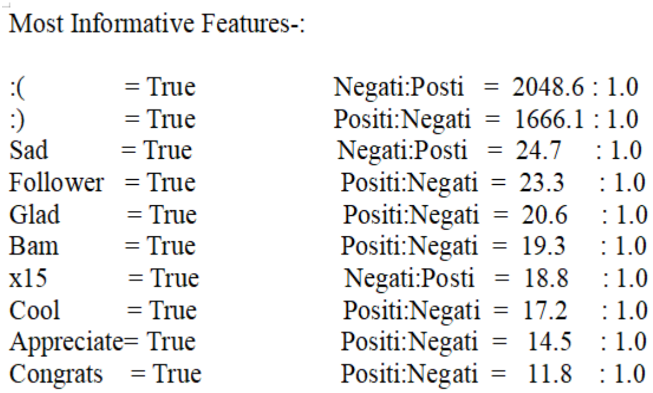
The initial stage of data processing involved tokenization—breaking down strings into smaller components called tokens. Tokenization typically splits text based on spaces and punctuation. To facilitate this, the `punkt` module was downloaded. NLTK provides several tokenization methods, such as `word_tokenize`, `TreebankWordTokenizer`, and `RegexpTokenizer`. For this task, the `tweets.Tokenized ()` method in NLTK was customized.

In the next stage, noise was removed from the dataset. Noise refers to irrelevant text elements that do not contribute useful information and varies depending on the project. Using Python's `re` library, URLs (e.g., beginning with `http://` or `https://`) were identified and removed by replacing them with an empty string via the `.sub()` method. Similarly, mentions (e.g., beginning with `@`) and punctuation were eliminated. The built-in NLTK stopword set was used to filter out common, uninformative words. Additionally, all text was converted to lowercase using the `.lower()` method, ensuring consistency and reducing data size.

The sentiment analysis model associates tweets with either positive or negative sentiments. The dataset was split into two parts: one for training the model and another for testing its performance. Since the dataset initially grouped positive tweets before negative ones, a randomization step was implemented using the `.shuffle()` method from Python's `random` library to avoid bias.

Finally, supervised machine learning algorithms were used to develop the model. The `.train()` method trained the model, and its performance was evaluated using the `.accuracy()` method on the test data. Figure 2 showcases the most informative features, with each row indicating the occurrence ratio of a token in positive and negative tweets within the training dataset.

Most Informative Features in Figure 2 -:



```
Most Informative Features-:

:(          = True          Negati:Posti  =  2048.6 : 1.0
:)          = True          Positi:Negati =  1666.1 : 1.0
Sad         = True          Negati:Posti  =  24.7   : 1.0
Follower    = True          Positi:Negati =  23.3   : 1.0
Glad        = True          Positi:Negati =  20.6   : 1.0
Bam         = True          Positi:Negati =  19.3   : 1.0
x15         = True          Negati:Posti  =  18.8   : 1.0
Cool        = True          Positi:Negati =  17.2   : 1.0
Appreciate= True            Positi:Negati =  14.5   : 1.0
Congrats    = True          Positi:Negati =  11.8   : 1.0
```

Figure 2. Collection of words and its probabilities.

The terms "Negati: Positi" and "Positi: Negati" refer to probability ratios that indicate whether a specific word is more likely to occur in a negative or positive sentiment context. For example, the first row of data shows that tweets containing the emoticon ":(" have a negative-to-positive sentiment ratio of 2048.6 to 1. This highlights that emoticons are some of the most influential elements in sentiment analysis. Words like "sad" are strongly linked to negative emotions, while words such as "cool" and "glad" are commonly associated with positive emotions.

V. Conclusion and Future Work

Sentiment detection has become an important area of study, posing several challenges. This research investigates methods and approaches to automatically classify sentiments into positive or negative categories. It utilizes various techniques, including the latest methods applied to data from NLTK's Twitter corpus, which contains a sample of 30,000 tweets.

Data preprocessing involves steps such as tokenization, lemmatization, and the removal of stop words, URLs, @ mentions, punctuation, and special characters. All tweets are then converted to lowercase. After cleaning, the tokens are transformed into a Python dictionary where words serve as keys and True as their corresponding values. The dataset is subsequently divided into training and testing sets in a 70:30 ratio.

Supervised machine learning algorithms are used to classify the tweets into positive and negative sentiments, with model performance evaluated based on accuracy.

Future efforts aim to improve the model’s ability to detect sarcasm, which will require a sufficient amount of training data. Currently, the study focuses on positive, negative, and neutral sentiment polarities for labeling tweets. However, additional labels that offer more detailed sentiment interpretations could be included. Considering the large volume of tweets generated every minute, including many in Arabic, future research plans to explore the effectiveness of hybrid classification techniques on Arabic tweets.

***Masters of Technology**
****Professor**
Department of Computer Sciences & Engineering
Hemvati Nandan Bahuguna Garhwal University
Uttarakhand, India

REFERENCES

[1] J. Li, S. Fong, Y. Zhuang, and R. Khoury, "Hierarchical Classification in Text Mining for Sentiment Analysis," presented at the 2014 International Conference on Soft Computing and Machine Intelligence, September 2014, pp. 46–51. DOI: 10.1109/ISCMI.2014.37.

[2] H. Parveen and S. Pandey, "Sentiment Analysis on Twitter Dataset Using the Naive Bayes Algorithm," in the 2016 2nd International Conference on Applied and Theoretical Computing

and Communication Technology (iCATccT), July 2016, pp. 416–419. DOI: 10.1109/ICATCCT.2016.7912034.

[3] S. S. and P. K. V., "Sentiment Analysis of Malayalam Tweets Using Machine Learning Techniques," ICT Express, April 2020. DOI: 10.1016/j.icte.2020.04.003.

[4] R. A. Tuhin, B. K. Paul, F. Nawrine, M. Akter, and A. K. Das, "An Automated System of Sentiment Analysis from Bangla Text Using Supervised Learning Techniques," presented at the 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS), February 2019, pp. 360–364. DOI: 10.1109/CCOMS.2019.8821658.

[5] A. Poornima and K. S. Priya, "A Comparative Sentiment Analysis of Sentence Embedding Using Machine Learning Techniques," presented at the 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), March 2020, pp. 493–496. DOI: 10.1109/ICACCS48705.2020.9074312.

[6] H. S. and R. Ramathmika, "Sentiment Analysis of Yelp Reviews Using Machine Learning," in the 2019 International Conference on Intelligent Computing and Control Systems (ICCS), May 2019, pp. 700–704. DOI: 10.1109/ICCS45141.2019.9065812.

[7] R. B. Shamantha, S. M. Shetty, and P. Rai, "Sentiment Analysis Using Machine Learning Classifiers: Evaluation of Performance," presented at the 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS), February 2019, pp. 21–25. DOI: 10.1109/CCOMS.2019.8821650.

[8] M. Wongkar and A. Angdresey, "Sentiment Analysis Using the Naive Bayes Algorithm for Data Crawler: Twitter," presented at the 2019 Fourth International Conference on Informatics and Computing (ICIC), October 2019, pp. 1–5. DOI: 10.1109/ICIC47613.2019.8985884.

[9] Md. G. Sarowar, M. Rahman, Md. N. Yousuf Ali, and O. F. Rakib, "An Automated Machine Learning Approach for Sentiment Classification of Bengali E-Commerce Sites," in the 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), March 2019, pp. 1–5. DOI: 10.1109/I2CT45611.2019.9033741.