International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 1, pp 39-50, January-2026

https://doi.org/10.47001/IRJIET/2026.101006

Feature Engineering for Sentiment Analysis: Insights from Twitter Data

[1]Mansi A. Shah, [2]Ravi M. Gulati

[1,2]Department of Computer Science, Veer Narmad South Gujarat University, Surat, Gujarat, India

***Abstract* - One of the most popular social media sites, Twitter, is an essential source of information for opinion mining and sentiment analysis. With millions of tweets generated daily, analysing these tweets to extract opinions and sentiments on various topics has become a critical task. In a democratic country like India, Twitter is a prominent medium for expressing views on diverse subjects, such as newly released movies, political figures and events, current affairs, the stock market, and more. This paper utilizes a balanced collection of positive and negative tweets sourced from the Sentiment140 benchmark dataset on Kaggle. Two widely used feature extraction techniques—TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization and Count Vectorization—were implemented, incorporating unigram, bigram, trigram, and n-gram (1,3) approaches. Among these, TF-IDF with n-gram (1,3) modelling performed best on all evaluation metrics. For classification, Logistic Regression, a supervised machine learning model, was employed to capture sentiment patterns within the dataset effectively. This paper presents a well-structured pipeline for sentiment analysis, which can be used as a baseline method for future studies. It highlights the effectiveness of integrating advanced feature engineering techniques with robust machine learning algorithms to enhance sentiment classification accuracy on Twitter data.**

Keywords: Sentiment Analysis, Twitter, TF-IDF, Feature Extraction.

I. INTRODUCTION

Opinion mining and sentiment analysis play an essential role in modern data-driven decision-making by transforming unstructured textual data into useful plans of action. Since social media platforms have changed the way people communicate, and research has evolved from retrieving facts to "opinion mining."[1]. These techniques are equally employed by businesses, policymakers, and researchers to understand public opinion, track brand perception, and predict trends, which lead to better and more strategic decisions. Because of the volume, velocity, and diversity of social media information, sentiment analysis is essential when working with huge datasets, like tweets. Recently, researchers focused a lot of attention on Twitter Sentiment Analysis (TSA) as part of a text categorization task [2]. Each tweet gives real-time public reactions to various subjects. This makes them a useful source of information for consumer behaviour, social issues, and trends. This large-scale analysis helps organizations to adapt to the changing reality, identify patterns, and manage public opinion effectively. Additionally, Twitter opinions are vital information that reflects people's sentiments and has to be analysed [2]. Understanding machine learning and classification algorithms with the right language rules is necessary for the effective classification of opinions [1].

Data preprocessing helps to improve readability and consistency by removing noise from extracted text. In the field of sentiment analysis, normalisation is the process of cleaning or eliminating unnecessary data from a large dataset [1]. Regarding Twitter sentiment analysis, preprocessing, feature engineering, and sentiment classification involve various challenges. Preprocessing requires handling noise within unstructured text, such as abbreviations, emojis, clamorous text, and URLs, which require sophisticated cleaning methods and lemmatization to standardize the text while preserving its core meaning. Additionally, the use of TF-IDF and Count Vectorization with N-grams greatly impacts feature engineering, because it introduces high dimensionality to a contextually relevant feature space. Sentiment classification demands careful balancing of features to distinguish between positive and negative sentiments effectively. Overcoming these challenges emphasizes the need to establish a properly integrated pipeline to manage and represent the textual data to accurately predict sentiment. This paper establishes a well-structured pipeline for systematically implementing sentiment analysis. It guarantees a logical and efficient approach by precisely defining the order of preprocessing stages, feature engineering techniques, and the use of Count Vectorisation and TFIDF. This study provides a thorough framework that acts as a starting point for scholars who are interested in sentiment analysis.

The main aims of this study are to preprocess noisy and unstructured Twitter data by eliminating irrelevant elements, expanding contractions, and implementing lemmatization to standardize text before feature engineering. Feature engineering uses TF-IDF and Count Vectorization with

ISSN (online): 2581-3048
Volume 10, Issue 1, pp 39-50, January-2026
https://doi.org/10.47001/IRJIET/2026.101006

unigram, bigram, and trigram models to capture individual word importance and contextual relationships. Finally, the study evaluates the performance of classification models, focusing on Logistic Regression, to identify the most effective combination of features and n-gram configurations for accurate sentiment analysis.

The rest of this paper is organised as follows: The relevant work on this topic is reviewed in Section II. The approach used for tweet sentiment categorization is described in Section III. Section IV provides a full account of the experimental procedure, whereas Section V displays the experimental results. Finally, the study's conclusion is presented in Section VI.

II. LITERATURE REVIEW

Feature extraction and preprocessing are vital processes for sentiment analysis using machine learning. Preprocessing involves data cleaning to remove noise (e.g., URLs, punctuation), Tokenization for text segmentation into words and phrases, stemming/lemmatization to reduce words to their root forms, and removing stop words that do not significantly add meaning to focus on relevant terms. Common feature extraction approaches include TF-IDF for weighting terms based on importance to a document, Bag of Words (BoW) to convert text into numeric vectors, and N-grams to encapsulate sequential context among words. Sentiment lexicons determine if a word is positive or negative, and POS tagging classifies nouns and adjectives in the sentence to understand the structure. These steps enhance data quality for effective sentiment classification.

For efficient sentiment analysis of Turkish tweets, Alawi et al. [3] integrated machine learning models with customised pre-processing and feature extraction techniques, including TF-IDF and BoW. Bello et al. [4]used BERT, which outperforms conventional models like LSTMs by representing words as numerical vectors using transformer-based encoders and word embeddings. This allows for greater context understanding by processing full sentences at once. Devarapalli et al. [5] employed the BoW model for feature extraction by transforming text into numerical vectors based on word frequency. Preprocessing methods include stop word removal, stemming, tokenisation, and data cleaning. Then, logistic regression and Naïve Bayes are used to classify sentiment-labeled COVID-19 tweets. For better sentiment categorisation, Gupta et al. [6] presented a strong Twitter sentiment analysis approach that makes use of lexicon-based, morphological, n-gram, and POS-tagged characteristics. The key processing methods used are feature optimisation, dimensionality reduction, and improved negation handling.

To enhance sentiment analysis, Poornima et al. [7]used bigrams for feature extraction, capturing word pairs, and negation modelling. Preprocessing comprises text cleaning methods, including converting to lowercase and removing extra spaces, as well as normalisation and special character removal. Combining these techniques with machine learning algorithms like Logistic Regression improves the precision of tweet sentiment categorisation. Gupta et al. [8] employed preprocessing procedures such as tokenisation and feature selection using chi-square testing, along with the sentiment, punctuation, and pattern-based features in tweets. Sentiment and punctuation characteristics are used with TF-IDF to improve machine learning models, such as SVM and Random Forest.

For effective sarcasm recognition, Parmar et al. [9] integrated lexical and hyperbolic characteristics using a Map Reduce architecture, aided by preparation procedures including data cleaning and POS tagging. Using a hybrid algorithm in a Hadoop-based framework improves the precision of sarcasm detection in massive amounts of social media data. Yafeng Ren et. al. [10] utilized linguistic features (n-grams, punctuation, emoticons, sentiment) and contextual features (historical tweets, POS tags, dependency structures). Preprocessing involves pooling techniques and non-linear hidden layers to refine feature representation.

Rathan et. al. [11]employed an ontology-based system with attribute-specific lexicons and synonym identification for feature extraction, enhancing sentiment analysis in the smartphone domain. Preprocessing techniques include text cleaning, emoji detection, and grammatical relationship identification using Stanford NLP tools, followed by automated training data labelling and classification with an SVM model for accurate sentiment classification. Wandra et. al. [12]used pragmatic features (emoticons, mentions), n-gram features (unigrams, bigrams, trigrams), and hyperbolic features (intensified words, punctuation) for feature extraction. Preprocessing involves cleaning and normalizing tweets. Making use of machine learning classifiers such as SVM and Logistic Regression, the model is trained and tested. Lavanya et. al. [13]extracted text features like sentiment words and POS-tagged terms, along with non-text features like emoticons and punctuation. Preprocessing includes cleaning and stemming, and a topic-adaptive algorithm enhances sentiment classification across various topics.

Deshwal et. al. [14]used sentiment lexicons, word counts, emoticons, and punctuation for feature extraction and compared classification algorithms to improve sentiment analysis on Twitter data. Lima et. al. [15]employed n-gram with TF-IDF weighting for feature extraction and preprocessing tools like LIWC, MRC Database, and POS

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 1, pp 39-50, January-2026
https://doi.org/10.47001/IRJIET/2026.101006

Tagger. A hybrid framework combines knowledge-based methods and machine learning for accurate Twitter sentiment analysis. Nadia et. al. [16]employed feature extraction techniques like BoW and feature hashing to represent tweets efficiently while leveraging opinion lexicons, emoticon analysis, and sentiment word classification during preprocessing. To extract features more efficiently, it can be applied in multiple phases. Neetu et al. [17] extracted Twitter-specific features in two phases, like emoticons and hashtags, then removed them to extract text features using unigrams and preprocessing, including URL removal, misspelling correction, and slang replacement with a domain-specific dictionary. Pak et al. [18]extracted n-grams (unigrams, bigrams, trigrams) from tweets after preprocessing steps like removing URLs, usernames, special words, and stop words. Negations were handled by attaching them to adjacent words, improving sentiment classification accuracy.

III. METHODOLOGY

The methodology for sentiment analysis begins with collecting the dataset, such as a Twitter dataset, for processing and analysis. The first step involves data preparation, which includes removing special characters, stop words, and URLs to eliminate unnecessary elements from the text. After that, the entire text is changed to lowercase to maintain uniformity. The text is then divided into distinct tokens or words using tokenization, and words are subsequently reduced to their basic form using lemmatization. After preprocessing, feature extraction is conducted using techniques such as the Count Vectorizer, which converts text into word frequency-based numerical data, and TF-IDF, which uses term frequency-inverse document frequency to highlight key terms. The dataset is then split into subsets for testing and training to train and evaluate the model. Algorithms like logistic regression are used for classification to construct the sentiment analysis model. Lastly, to make sure the strategy is effective, the model's performance is assessed on the test data using metrics like accuracy, precision, recall, and F1-score. This systematic methodology ensures accurate and efficient sentiment analysis. The system's whole workflow is shown in Figure 1.

3.1 Dataset Description

The Sentiment140 dataset, a known benchmark for sentiment analysis research, is used in this study. The dataset can be accessed from Kaggle and consists of around a million tweets that have been extracted through the Twitter API. We have considered 50,000 tweets from this dataset. The dataset is balanced, containing **25,000 positive tweets** and **25,000 negative tweets**. Each tweet is annotated with a sentiment label, where **0 denotes negative sentiment** and **4 denotes positive sentiment.** In order to make the analysis more efficient and centric towards sentiment detection, unnecessary columns like UserID, Date, Flag, and Username were excluded, leaving only two key columns: Tweets and their respective Sentiment labels. The dataset was divided into two sets for training and assessing the model, 70% for training and 30% for testing.

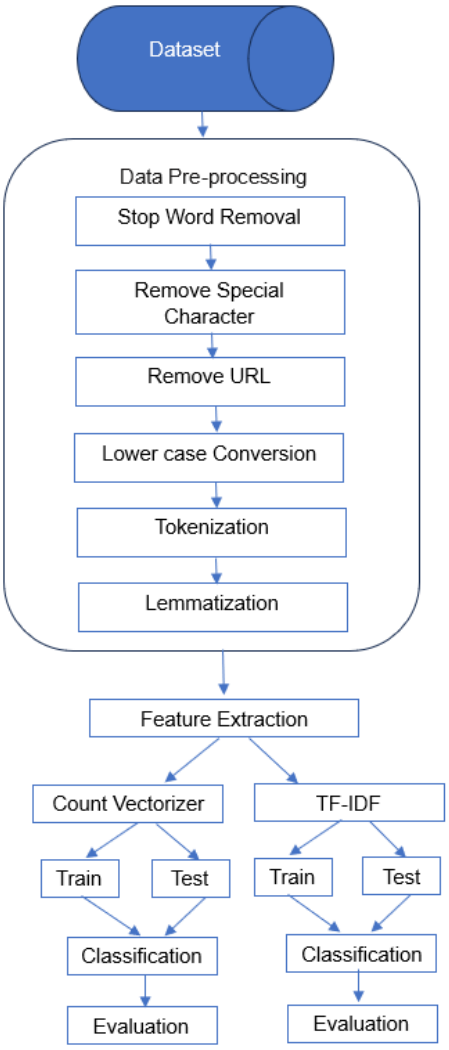


Figure 1: Flow of Work

The training set is used to build and improve machine learning models, while the testing set is used to evaluate the models' performance and generalisation abilities. The Sentiment140 dataset is a reliable resource for building and assessing sentiment analysis models because of its size and evenly distributed sentiment. Its extensive scope makes it possible to thoroughly experiment with feature extraction strategies, classification algorithms, and preprocessing procedures.

3.2 Pre-processing

Pre-processing is considered to be the first step in text classification, and using the right pre-processing techniques

ISSN (online): 2581-3048
Volume 10, Issue 1, pp 39-50, January-2026
https://doi.org/10.47001/IRJIET/2026.101006

can improve classification effectiveness [19]. Tweets usually contain a lot of noise, poorly constructed sentences, and incomplete statements. Tweets undergo several preprocessing steps to reduce noise before feature selection [20].This paper offers a well-defined preprocessing flow, guaranteeing a structured and efficient approach for optimal performance. The order of preprocessing steps is essential for attaining high accuracy in sentiment analysis, and any modification or omission of these steps can greatly affect the results.

Steps to perform pre-processing for Sentiment Analysis

Input: A dataset containing raw tweets in the 'tweets' column.
Output: A cleaned dataset, where the tweets_clean column contains the processed tweets.

Step 1: Initialization

- Import essential libraries & resources (nltk, re, and pandas)
- Download the NLTK stopword list to facilitate the removal of irrelevant words.

Step 2: Stopword Initialization

- Load the list of English stopwords using nltk.corpus.stopwords.
- Exclude the word "not" from the stopword list to preserve its sentiment-bearing role in the text.
- Convert the stopword list into a set for faster lookup.

Step 3: Define Helper Functions

- **Special Character Removal**
  - Define function: remove_special_character (info_content)
    - Remove special characters (e.g., [, @,!) using regular expressions.
    - Retain alphanumeric content only.
- **URL Removal**
  - Define function: remove_url(info_content)
    - Identify and remove URLs using a regular expression pattern (http\S+).
- **Stopword Removal**
  - Define function: remove_stopwords(info_content):
    - Tokenize the text into individual words.
    - Keep only alphabetic tokens and exclude words that are in the stopword list.
    - Retain only alphabetic words and change them to lowercase.
    - Return the cleaned text as a single string.
- **Contraction Expansion**
  - Define contraction_expansion(info_content)
    - Expand common English contractions (e.g., "can't" → "cannot") using re.sub () for each contraction pattern.

Step 4: Develop Unified Data Cleaning Function

Define data_cleaning(info_content):
Apply contraction_expansion(info_content)
Apply remove_special_character(info_content)
Apply remove_url(info_content)
Apply remove_stopwords(info_content)
Return processed info_content

Step 5: Apply the Cleaning Pipeline to the Dataset
Step 6: Return the cleaned Dataset

3.3 Visualization of Word Clouds

The word cloud is a visual depiction of text data in which each word's size corresponds to its significance or frequency in the dataset. It is frequently used to highlight prominent terms, providing a rapid and easy method for locating important themes or patterns in text.

Figures 2 & 3 define Word clouds for positive and negative tweets to analyse frequently used words.

Figure 2: Word Cloud of Positive words

Figure 3: Word Cloud of Negative words

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 1, pp 39-50, January-2026

https://doi.org/10.47001/IRJIET/2026.101006

3.4 Feature Engineering

One essential phase in Feature Engineering is text analysis, where textual data in its raw form is transformed into numerical representations to be used by machine learning models. In this research work, TF-IDF and Count Vectorizer were employed as vectorization techniques to convert text into numerical features. These methods were applied using various n-gram ranges, including unigram, bigram, trigram, and n-grams, to capture contextual relationships between words.

Prior to applying both vectorization techniques, lemmatization was incorporated into the preprocessing pipeline as a critical step. Lemmatization focuses on converting words to their base or root forms as found in a dictionary, such as changing "speaking" to "speak" or "better" to "good," while ensuring the context of the word remains intact. Lemmatization maintains linguistic meaning, which makes it more useful for text analysis tasks than stemming, which may truncate words arbitrarily. Reducing the dimensionality of the feature space requires grouping word variations under a comparable basic form. Redundancy is removed, and a more condensed and insightful representation of textual data is produced. By treating terms like "speaking," "speaks," and "spoke" as a single phrase, for example, models are able to concentrate on their common semantic meaning. Lemmatisation hence increases computing efficiency as well as the quality of derived features, especially when working with huge datasets in sentiment analysis. Most studies incorporate lemmatization or stemming in their workflow, but the optimal stage for applying these techniques is after preprocessing and before feature engineering. Implementing them at the right step enhances the effectiveness of sentiment analysis and improves overall results.

Feature Extraction & Feature Representation

In a corpus of texts, words are both distinct and category elements. They cannot be utilised for text processing activities since they are in their raw form. These words must be encoded in some way in order to be mapped to real-valued attributes that machine learning algorithms can easily employ [21]. In a corpus of text, words are converted into numerical features using techniques such as Bag of Words (BoW), which measures word frequency, and TF-IDF, which highlights word importance through weighting. While more straightforward techniques like One-Hot Encoding or Tokenisation translate words to binary vectors or numerical sequences, more sophisticated systems like Word Embeddings (e.g., Word2Vec, GloVe) represent words as dense vectors to capture their semantic associations. These transformations make textual data suitable for processing and analysis by machine learning algorithms. Feature extraction involves deriving meaningful information from raw data to create features suitable for machine learning models. Using word embedding to represent and extract features in a hierarchical manner makes it unique and superior to lexical or syntactic feature extraction [22]. In this work, textual characteristics are extracted from the tweets using methods like Count Vectorization and TF-IDF Vectorization. These methods convert raw text into numerical representations by quantifying the significance of individual words or word combinations (e.g., bigram, trigram, and n-gram approaches). One of the simplest and most efficient representation methods for sentiment analysis on Twitter and natural language analysis is the word N-grams feature model [20].

Feature representation focuses on structuring these extracted features in a way that machine learning algorithms can understand. Sparse vectors generated by vectorization techniques like TF-IDF and Count Vectorizer effectively encode the frequency or relevance of terms, enabling models to capture linguistic and semantic properties. In a dataset, TF-IDF is used to assess a word's importance to a text [23]. This structured representation transforms unprocessed textual input into a sentiment classification-ready format, ensuring that the machine learning models can process the data effectively.

i) TF-IDF Vectorizer

"The **TF-IDF (Term Frequency-Inverse Document Frequency)** vectorizer quantifies the importance of a term within a document relative to its frequency across the entire corpus. It mitigates the influence of common words (e.g., "the", "is") while emphasizing terms more relevant to specific documents." A statistical metric and term-weighting system called TF-IDF gives the bag-of-words model information about the value of words [23]. Two elements are combined in the TF-IDF formula: Inverse Document Frequency (IDF) and Term Frequency (TF) [21]. It is possible to determine the Term Frequency and Inverse Document Frequency by using Equations (1) and (2).

$$TF(t,d) = \frac{Number\ \ of\ times\ \ term\ \ t\ appears\ \ in\ document\ \ d}{Total\ \ number\ \ of\ terms\ \ in\ document\ \ d} \quad (1)$$

$$IDF(t) = log_e\left(\frac{Total\ \ no.of\ \ documents}{no.of\ \ documents\ \ \ with\ word\ \ t\ in\ it}\right) \quad (2)$$

The TF-IDF score is computed as:

$$TF - IDF(t,d) = TF(t,d) * IDF(t) \quad (3)$$

In this case, IDF(t) is the logarithmically scaled inverse percentage of documents containing t, and TF (t, d) is the frequency of term t in document d. TF-IDF is calculated by multiplying a term's TF value from Eq. (1) with its IDF score from Eq. (2), as shown in Eq. (3).

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 1, pp 39-50, January-2026

https://doi.org/10.47001/IRJIET/2026.101006

This technique was configured to generate features using unigrams, bigrams, trigrams, and n-grams (1,3). Sparsity was managed by optimising parameters such as the minimum document frequency thresholds and maximum feature size.

While the Count Vectorizer just assesses term frequency, TF-IDF considers both term frequency and inverse document frequency. Because of this dual consideration, phrases that are less frequent but more informative are assigned bigger weights, whereas high-frequency but less informative terms have less of an impact on the corpus. Consequently, it enhances the quality of feature representation by successfully capturing the significance of phrases for sentiment analysis.

ii) Count Vectorizer

The focus of this most basic text feature approach is solely on word occurrence [21]. The Count Vectorizer transforms textual input into a matrix of word counts to compute the frequency of specific words or n-grams (such as bigrams or trigrams) within the corpus. This method finds patterns in the recurrence of words, which are frequently suggestive of sentiment. For instance, the frequency of terms like "happy" or "sad" can reveal positive or negative sentiments, respectively. This method creates a sparse matrix representation from a set of raw term frequencies. If n-grams are incorporated, it improves its capacity to identify significant sentences or word combinations, as well as relationships between consecutive words. Feature extraction in this study was limited to 500 features and included unigram, bigram, trigram, and n-gram (1,3) features. To guarantee consistency among studies, key settings include n-gram ranges that are constant. Although it is less sophisticated than TF-IDF, Count Vectorizer has its uses in emphasising normal phrases and commonly repeated patterns that make it a useful tool for feature extraction in sentiment analysis.

3.5 Classification Models

For sentiment analysis on Twitter data, this study uses Logistic Regression as a categorisation model. The model uses the sigmoid function to translate the outcome of a linear equation to a probability between 0 and 1, which indicates the likelihood that the text has a positive or negative sentiment. Eq. (4) defines the sigmoid function as follows:

$$\sigma(z) = \frac{1}{1+e^{-z}} \qquad (4)$$

Where z is the weighted sum of input features, given by $z = w^T x + b, where$ weights (w) and bias (b) are learned during training, and x is the vector of input features.

As stated in Eq. (5), the model's goal is to minimise the log-loss (binary cross-entropy) cost function.

$$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\hat{y}^{(i)} + \left(1-y^{(i)}\right)\log\left(1-\hat{y}^{(i)}\right)\right] \qquad (5)$$

The model is optimised via gradient descent, which iteratively updates the weights to minimise the cost function. The trained model predicts sentiment by applying the sigmoid function to the weighted sum of features. If the output is above 0.5, the tweet is classified as positive, otherwise negative. When it comes to binary classification tasks like sentiment analysis, logistic regression is a useful and comprehensible approach. It helps in understanding the relationships between tweet features and their sentiment, providing insights into sentiment-driven patterns in Twitter data.

IV. EXPERIMENTAL SETUP

To evaluate and compare the performance of two feature extraction methods, comprehensive experiments were conducted on the benchmark Twitter dataset, Sentiment140. This dataset, obtained via the Twitter API, is publicly available and widely used for sentiment analysis research. For this study, a subset of 50,000 tweets was selected from the 1 million tweets in the dataset, consisting of 25,000 positive tweets and 25,000 negative tweets. In this procedure, 70% of the data is set aside for model training and 30% is used for testing. The performance of the Count Vectorizer and TF-IDF Vectorizer, using unigram, bigram, trigram, and n-gram features, was evaluated using measures including accuracy, F1-score, precision, and AUC score. The results are summarized in Table I. All necessary preprocessing steps, as outlined in Section 3, were executed using Python Anaconda (Jupyter Notebook). This study provides a comparative analysis of the two feature extraction methods based on these various n-gram models.

V. RESULT & DISCUSSION

5.1 Model Performance

Table1: The comparison of obtained results for Count Vectorizer and TF-IDF Vectorizer

Precision								
	Unigram		Bigram		Trigram		N-gram	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 1, pp 39-50, January-2026

https://doi.org/10.47001/IRJIET/2026.101006

	Data	Data	Data	Data	Data	Data	Data	Data
Count Vectorizer	72.31%	70.59%	59.58%	58.24%	51.48%	50.97%	72%	70.48%
TFIDF Vectorizer	76.91%	72.94%	60.66%	58.96%	51.48%	50.96%	77.81%	73.42%
AUC Score								
	Unigram		Bigram		Trigram		N-gram	
	Training Data	Testing Data	Training Data	Testing Data	Training Data	Testing Data	Training Data	Testing Data
Count Vectorizer	79.87%	77.71%	63.15%	61.46%	51.66%	51.36%	79.68%	77.55%
TFIDF Vectorizer	84.79%	80.52%	64.69%	62.69%	51.66%	51.32%	85.80%	81.25%
F1-Score								
	Unigram		Bigram		Trigram		N-gram	
	Training Data	Testing Data	Training Data	Testing Data	Training Data	Testing Data	Training Data	Testing Data
Count Vectorizer	72.21%	70.47%	55.32%	53.58%	36.95%	36.17%	71.89%	70.36%
TFIDF Vectorizer	76.89%	72.92%	57.16%	54.99%	36.98%	36.17%	77.80%	73.40%
Accuracy								
	Unigram		Bigram		Trigram		N-gram	
	Training Data	Testing Data	Training Data	Testing Data	Training Data	Testing Data	Training Data	Testing Data
Count Vectorizer	56	70.59%	59.58%	58.24%	51.48%	50.97%	72.00%	70.48%
TFIDF Vectorizer	76.91%	72.95%	60.66%	58.97%	51.48%	50.96%	77.82%	73.43%

The performance difference, as shown in Table 1, between unigram, bigram, and trigram models in the sentiment analysis task can be explained by several factors. One key reason for performance degradation is **data sparsity**. Trigrams create significantly more unique word combinations than unigrams or bigrams, resulting in a sparse feature matrix. Since many trigram features are rare in our dataset, this sparsity limits the model's ability to generalise.

Another factor for performance degradation is **overfitting**. Very particular patterns are captured by trigrams, which might lead to the model learning irrelevant features or noise unique to the training set. Unigrams, on the other hand, concentrate on specific words at a time, which reduces their capacity for overfitting and makes it easier for the model to generalise the sentiment signals. Furthermore, the characteristics of tweets also play a role. As tweets are short texts, their sentiment is likely to be expressed through isolated words or sentences such as “great” or “terrible”. Thus, it seems unigrams are sufficient sentiment classifiers to use, while bigrams or trigrams may provide no extra context. In addition, the extra dimensionality from the use of trigrams will increase the computation cost for the model, and perhaps need more data to be able to learn effectively. If the model parameters such as max_iter are not set correctly, the model might miss out on the potential of the deep features created by the bigrams and trigrams. The hyperparameter max_iter, which controls the optimization algorithm's maximum number of iterations for estimating the ideal weights of the logistic regression function, ensures that the algorithm does not run indefinitely and it is particularly helpful for large or complex datasets. You can increase max_iter to give the model extra time to optimise if convergence is not reached within the allotted iterations. The value of max_iter in our model is 500.

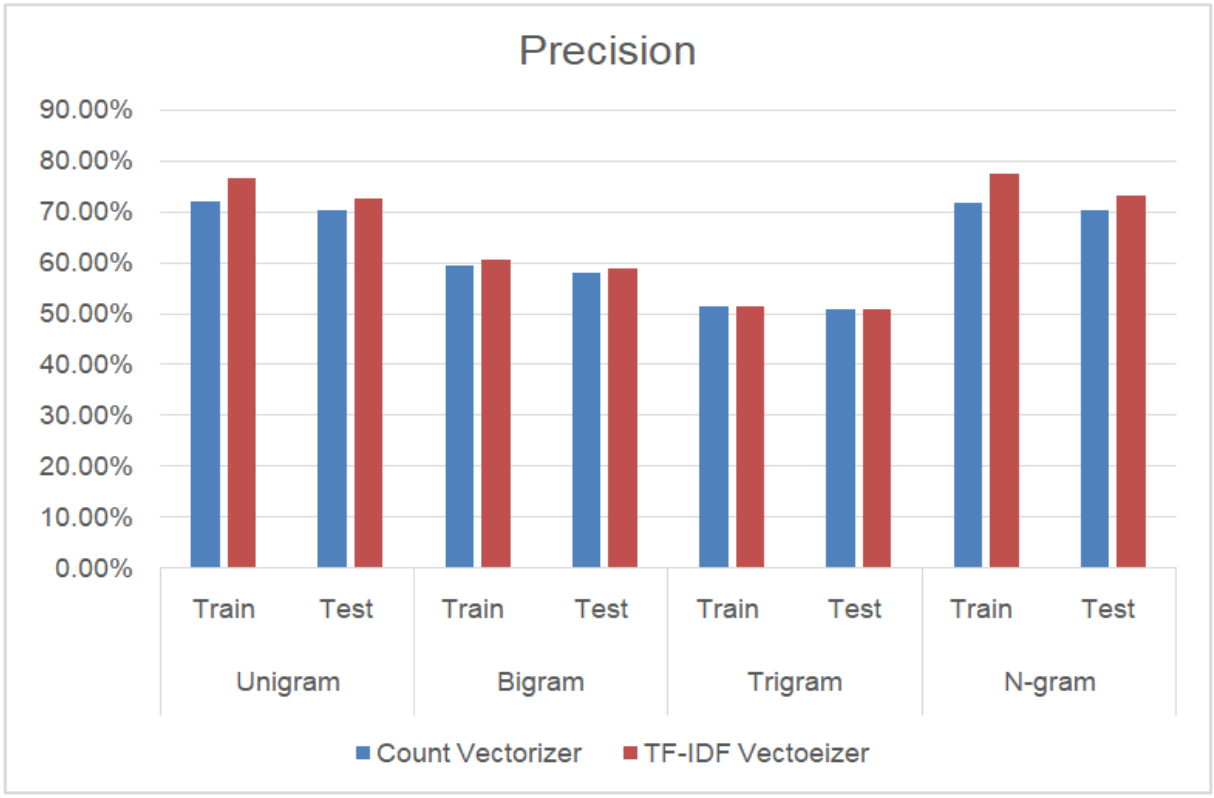
Finally, n-gram performance is affected by how stop words are handled. While most stop words were removed, some common phrases may still introduce irrelevant bigrams or trigrams (e.g., "not very"). Unigrams are less affected by this issue, as they evaluate individual word importance rather than patterns of co-occurrence.

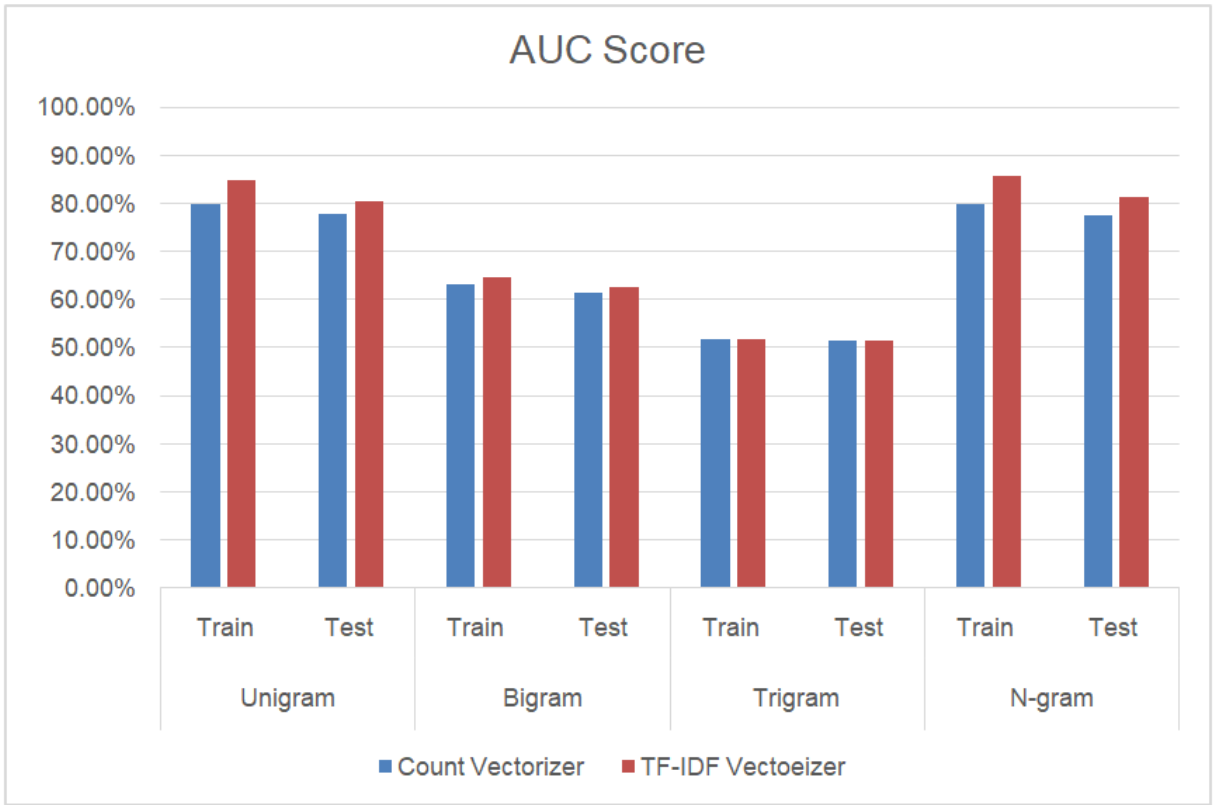
In conclusion, because of their simplicity, decreased vulnerability to overfitting, and resilience to data sparsity, unigrams frequently perform better in short-text sentiment analysis than bigrams and trigrams. To improve performance, you could experiment with combining unigrams, bigrams, and trigrams into a single feature space using *ngram_range = (n,n)*. The *ngram_range* sets the size of word sequences (e.g., unigrams, bigrams) for feature extraction, capturing context and dependencies between words, and improving sentiment analysis performance. In this model, *ngram_range* is set to (1,3). Additionally, applying feature selection techniques or trying more advanced models like Random Forests or gradient-boosted trees may help exploit the contextual patterns in higher-order n-grams.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 1, pp 39-50, January-2026
https://doi.org/10.47001/IRJIET/2026.101006

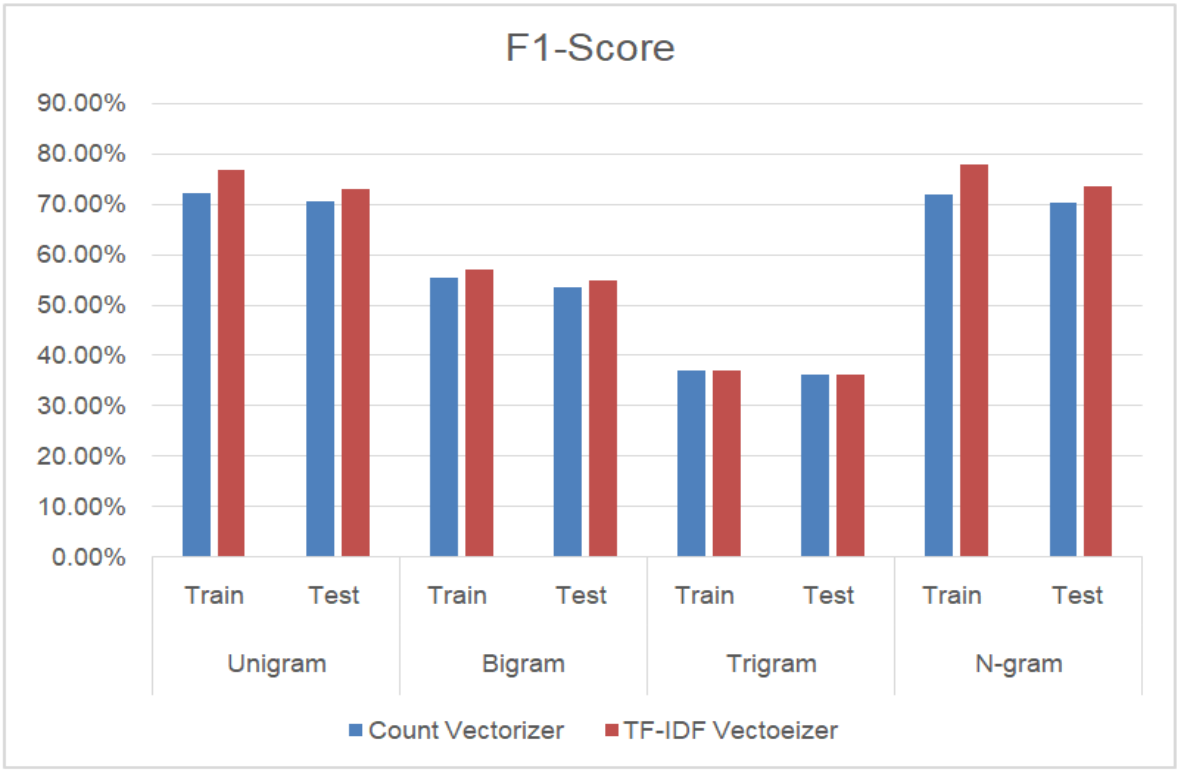
5.2 Visualizations

5.2.1 Comparison Charts





International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 1, pp 39-50, January-2026
https://doi.org/10.47001/IRJIET/2026.101006



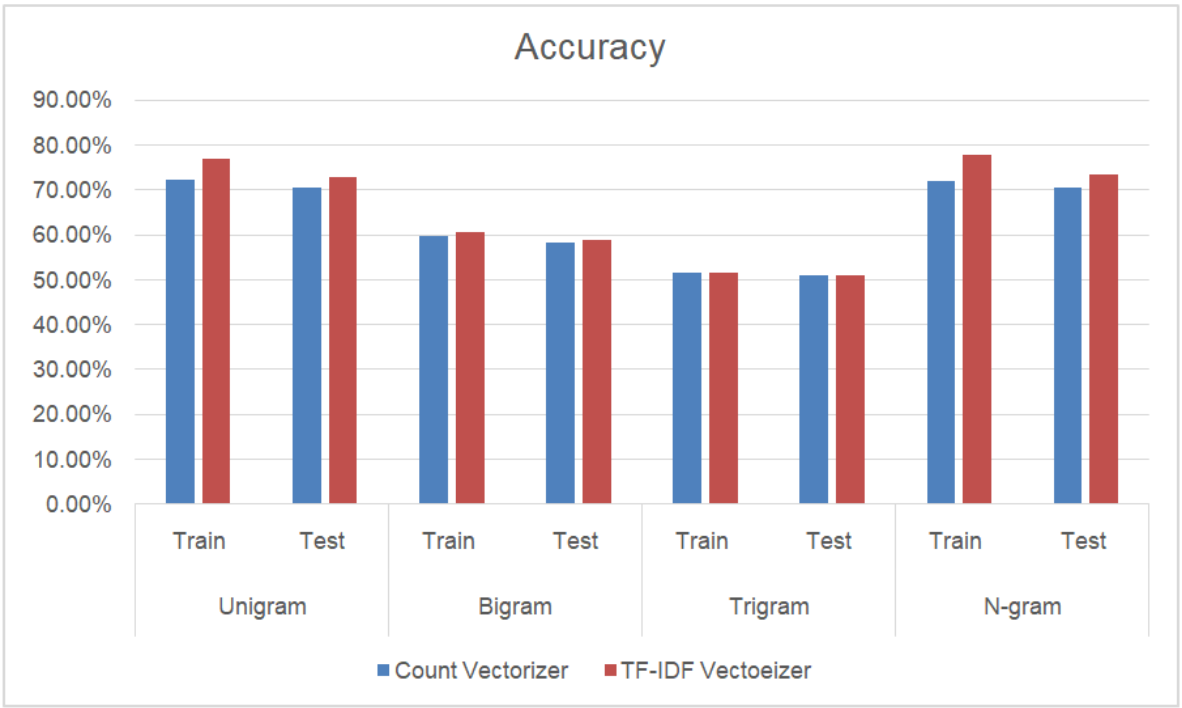


Figure 4: The comparison of Precision (a), AUC Score (b), F1-Score (c), and Accuracy (d) on the Train and Test Dataset using TF-IDF Vectorizer and Count Vectorizer

Figure 4 compares the performance of both techniques across Unigram, Bigram, Trigram, and N-gram configurations for measures such as Precision, F1-Score, Accuracy and, AUC Score. Both methods exhibit comparable results, with TF-IDF slightly outperforming Count Vectorizer, particularly on test sets. Performance decreases with increased n-gram complexity (e.g., Bigram and Trigram) but improves significantly with the N-gram configuration, which captures a broader range of patterns. The consistent results across training and test sets indicate minimal overfitting. Overall, TF-IDF proves to be slightly more effective, particularly when using N-gram features.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 1, pp 39-50, January-2026

https://doi.org/10.47001/IRJIET/2026.101006

5.2.2 Confusion Matrix

A contingency table or confusion matrix is used to assess the suggested system [1]. Figures 5 and 6 illustrate how our system accurately predicted outcomes from the test dataset.

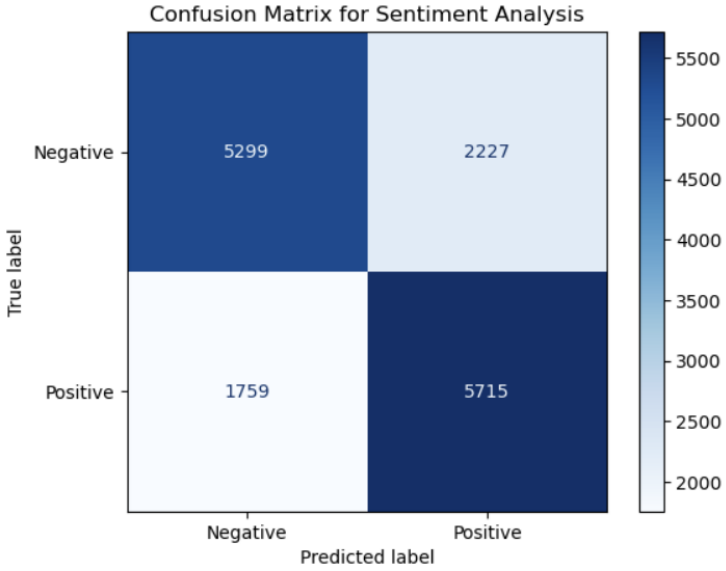


Figure 5: Confusion Matrix Showing Test Data Results with TF-IDF Vectorizer

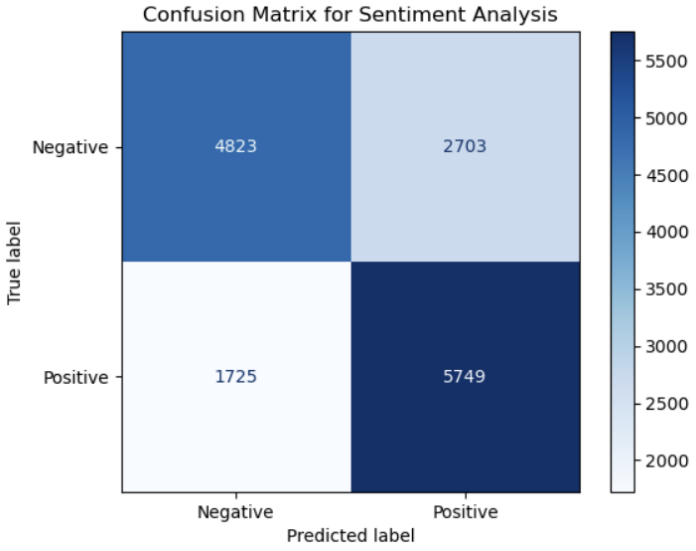


Figure 6: Confusion Matrix Showing Test Data Results with Count Vectorizer

VI. CONCLUSION& FUTURE WORK

The conclusion of this research underscores the critical role of preprocessing and feature engineering in sentiment analysis. Using a benchmark dataset of 50,000 tweets, various preprocessing techniques, including lemmatization, were applied to eliminate redundancy by converting inflected words into their base forms. This step ensured a more consistent and meaningful feature set for analysis. Feature engineering methods such as Count Vectorizer and TF-IDF were applied using unigram, bigram, trigram, and n-gram (1,3) models. The system's ability to capture contextual and semantic relationships will be enhanced by the integration of N-gram modelling. It improves the identification of sentiment-related phrases. Among these methods, TF-IDF outperformed Count Vectorizer by prioritizing less frequent but more meaningful terms, leading to richer feature sets for sentiment classification. The study further demonstrates that combining TF-IDF with n-grams and Logistic Regression yielded the best performance, highlighting the robustness of this approach for sentiment analysis. The integration of lemmatization, n-grams, and vectorization methods provided a well-balanced representation of textual data, enhancing both efficiency and

ISSN (online): 2581-3048
Volume 10, Issue 1, pp 39-50, January-2026
https://doi.org/10.47001/IRJIET/2026.101006

overall accuracy. This research introduces a structured and systematic pipeline for sentiment analysis, addressing key challenges in preprocessing, feature extraction, and model optimization. Outlining a clear sequence of steps ensures consistency in sentiment classification while offering a comparative analysis of different feature engineering methods. Serving as a foundational reference, this work bridges the gap between theoretical concepts and practical implementation, making it a valuable resource for researchers and practitioners in the field.

Future research could focus on hyperparameter tuning to further enhance the performance of the model. Additionally, exploring other classification algorithms like Support Vector Machines and Naïve Bayes could give promising results. Another important area for advancement is tackling challenges like sarcasm detection and context-based sentiment analysis. This research serves as a preliminary step for developing more advanced and precise sentiment analysis models.

REFERENCES

[1] Muhammad Javed and Shahid Kamal, “Normalization of Unstructured and Informal Text in Sentiment Analysis” *International Journal of Advanced Computer Science and Applications (IJACSA)*, 9(10), 2018. http://dx.doi.org/10.14569/IJACSA.2018.091011

[2] Zarisfi Kermani, F., Sadeghi, F. & Eslami, E. solving the twitter sentiment analysis problem based on a machine learning-based approach. *Evol. Intel*.13, 381–398 (2020). https://doi.org/10.1007/s12065-019-00301-x

[3] Abdulfattah Ba Alawi, Ferhat Bozkurt, A hybrid machine learning model for sentiment analysis and satisfaction assessment with Turkish universities using Twitter data, *Decision Analytics Journal*, Volume 11, 2024, 100473, ISSN 2772-6622, https://doi.org/10.1016/j.dajour.2024.100473.

[4] Bello, A., Ng, S.-C., & Leung, M.-F. (2023). A BERT Framework to Sentiment Analysis of Tweets. *Sensors*, 23(1), 506. https://doi.org/10.3390/s23010506

[5] Devarapalli, D., Sri, M.S., Sri, P.K., Charishma, P., Mounika, P.V.N. (2022). Sentiment Analysis of COVID-19 Tweets Using Classification Algorithms. In: Saini, H.S., Sayal, R., Govardhan, A., Buyya, R. (eds) Innovations in Computer Science and Engineering. *Lecture Notes in Networks and Systems*, vol 385. *Springer, Singapore*. https://doi.org/10.1007/978-981-16-8987-1_42

[6] I.Gupta and N. Joshi, "Feature-Based Twitter Sentiment Analysis With Improved Negation Handling," *in IEEE Transactions on Computational Social Systems*, vol. 8, no. 4, pp. 917-927, Aug. 2021, doi: 10.1109/TCSS.2021.3069413.

[7] A.Poornima and K. S. Priya, "A Comparative Sentiment Analysis Of Sentence Embedding Using Machine Learning Techniques," *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India*, 2020, pp. 493-496, doi: 10.1109/ICACCS48705.2020.9074312.

[8] R. Gupta, J. Kumar, H. Agrawal and Kunal, "A Statistical Approach for Sarcasm Detection Using Twitter Data," *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India*, 2020, pp. 633-638, doi: 10.1109/ICICCS48265.2020.9120917.

[9] K. Parmar, N. Limbasiya and M. Dhamecha, "Feature based Composite Approach for Sarcasm Detection using MapReduce," *2018 Second International Conference on Computing Methodologies and Communication (ICCMC), Erode, India*, 2018, pp. 587-591, doi: 10.1109/ICCMC.2018.8488096.

[10] Yafeng Ren, Donghong Ji, Han Ren, Context-augmented convolutional neural networks for twitter sarcasm detection, *Neurocomputing*, Volume 308, 2018, Pages 1-7, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2018.03.047.

[11] Rathan M., Vishwanath R. Hulipalled, K.R. Venugopal, L.M. Patnaik, Consumer insight mining: Aspect based Twitter opinion mining of mobile phone reviews, *Applied Soft Computing*, Volume 68, 2018, Pages 765-773, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2017.07.056.

[12] Dr. Kalpesh H. Wandra, Mehul Barot, Sarcasm Detection in Sentiment Analysis, 2017, *International Journal of Current Engineering and Scientific Research, ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697*, VOLUME-4, ISSUE-9.

[13] K. Lavanya and C. Deisy, "Twitter sentiment analysis using multi-class SVM," *2017 International Conference on Intelligent Computing and Control (I2C2), Coimbatore, India*, 2017, pp. 1-6, doi: 10.1109/I2C2.2017.8321798.

[14] A.Deshwal and S. K. Sharma, "Twitter sentiment analysis using various classification algorithms," *2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India*, 2016, pp. 251-257, doi: 10.1109/ICRITO.2016.7784960.

[15] Ana Carolina E.S. Lima, Leandro Nunes de Castro, Juan M. Corchado, A polarity analysis framework for Twitter messages, *Applied Mathematics and*

ISSN (online): 2581-3048
Volume 10, Issue 1, pp 39-50, January-2026
https://doi.org/10.47001/IRJIET/2026.101006

Computation, Volume 270, 2015, Pages 756-767, ISSN 0096-3003, https://doi.org/10.1016/j.amc.2015.08.059.

[16] Nádia F.F. da Silva, Eduardo R. Hruschka, Estevam R. Hruschka, Tweet sentiment analysis with classifier ensembles, *Decision Support Systems*, Volume 66, 2014, Pages 170-179, ISSN 0167-9236, https://doi.org/10.1016/j.dss.2014.07.003.

[17] M. S. Neethu and R. Rajasree, "Sentiment analysis in twitter using machine learning techniques," *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Tiruchengode, India*, 2013, pp. 1-5, doi: 10.1109/ICCCNT.2013.6726818.

[18] Pak, A., & Paroubek, P. (2010). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, & D. Tapias (Eds.), *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC`10). European Language Resources Association (ELRA).* https://aclanthology.org/L10-1263/

[19] Effrosynidis, D., Symeonidis, S., Arampatzis, A. (2017). A Comparison of Pre-processing Techniques for Twitter Sentiment Analysis. In: Kamps, J., Tsakonas, G., Manolopoulos, Y., Iliadis, L., Karydis, I. (eds) Research and Advanced Technology for Digital Libraries. *TPDL 2017. Lecture Notes in Computer Science ()*, vol 10450. *Springer, Cham*. https://doi.org/10.1007/978-3-319-67008-9_31

[20] Z. Jianqiang, G. Xiaolin and Z. Xuejun, "Deep Convolution Neural Networks for Twitter Sentiment Analysis," *in IEEE Access*, vol. 6, pp. 23253-23260, 2018, doi: 10.1109/ACCESS.2017.2776930.

[21] Nandy, H., Sridhar, R. (2021). A Novel Feature Engineering Approach for Twitter-Based Text Sentiment Analysis. In: Singh, P.K., Noor, A., Kolekar, M.H., Tanwar, S., Bhatnagar, R.K., Khanna, S. (eds) Evolving Technologies for Computing, Communication and Smart World. *Lecture Notes in Electrical Engineering*, vol 694. *Springer, Singapore*. https://doi.org/10.1007/978-981-15-7804-5_23

[22] Akshi Kumar, Kathiravan Srinivasan, Wen-Huang Cheng, Albert Y. Zomaya, Hybrid context enriched deep learning model for fine-grained sentiment analysis in textual and visual semiotic modality social data, *Information Processing & Management*, Volume 57, Issue 1, 2020, 102141, ISSN 0306-4573, https://doi.org/10.1016/j.ipm.2019.102141.

[23] S. E. Saad and J. Yang, "Twitter Sentiment Analysis Based on Ordinal Regression," *in IEEE Access*, vol. 7, pp. 163677-163685, 2019, doi: 10.1109/ACCESS.2019.2952127.

Citation of this Article:

Mansi A. Shah, & Ravi M. Gulati. (2026). Feature Engineering for Sentiment Analysis: Insights from Twitter Data. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 10(1), 39-50. Article DOI https://doi.org/10.47001/IRJIET/2026.101006

****

