EMITTER International Journal of Engineering Technology
Vol. 13, No. 2, December 2025, pp. 307~332
DOI : 10.24003/emitter.v13i2.861

Sentiment Analysis Design and Development for Low Resource Languages in the Case of Telugu

Srinivasu Badugu[1], Suneetha Chittineni[2], G.L. Anand Babu[3], G. Sekhar Reddy[3], S. Vijaykumar[3], N. Nagalakshmi[3]

[1]Department of CSE, stanley College of Engineering and Technology for Women, Abids, Hyderabad, India
[2]Department of computer applications, R.V.R. & J.C. College of Engineering, Chowdavaram, India
[3]Dept. of IT, Anurag University, Hyderabad, Telangana, India
Corresponding Author: anandbabuit@anurag.edu.in

Received July 20, 2025; Revised August 27, 2025; Accepted October 12, 2025

Abstract

The use of sentiment analysis has become more widespread because it is necessary to filter and analyze information on the internet. It has a wide range of applications, including monitoring social media market research and opinion mining. Still, this development is restricted to few languages with enough resources. The Telugu language lags behind in this field of study, even though it is the fourth most spoken language in India and generates a vast quantity of data every day. In this research paper, we develop a trustworthy source for sentiment analysis in Telugu. To use in sentiment analysis, the data is annotated with Telugu movie reviews. We extracted 1844 sentences from 100 film reviews. We annotated the data with two annotators and calculated the kappa coefficient to determine the annotators' inter-rater reliability. We obtained a kappa value of 0.90 for 1844 sentences, indicating nearly perfect agreement. After the annotators' disagreements and discrepancies were resolved, 1807 sentences were chosen. For feature extraction, we used two vectorization methods: TF-IDF and Count vectorization. Using the two vectorization methods, we used SVM and Logistic regression. We used two vectorization approaches to test different split ratios such as 80-20%, 70-30%, and 60-40% on SVM and Logistic regression. The outcomes of the various combinations are compared. We discovered that combining TF-IDF with SVM for a 70-30% ratio yields the highest accuracy among the combinations tested on our dataset.

Keywords: Intrusion detection system, cloud computing, security, deep learning model, TF-IDF.

1. INTRODUCTION

Sentiment analysis, sometimes referred to as "opinion mining" or "emotion identification," is the methodical identification, evaluation, extraction, and analysis of subjective data and emotional states using Natural Language Processing (NLP), a branch of computer linguistics [1, 2]. Sentiment analysis typically focuses on the language used in client materials such as online surveys, reviews, and social media platforms [3]. The primary aim of sentiment analysis is to recognize and label the polarity of text documents or brief sentences. Depending on their polarity (neutral), emotions can be categorized as positive, negative, or neutral. Based on their reviews, sentiment analysis can determine critics' attitudes. A movie review's sentiment analysis can reveal how favorable or unfavorable the review is, and consequently, the film's overall grade. Reviews of movies help consumers decide if a movie is worth their time [4]. Users may save time by reading a summary of all reviews for a movie instead of reading each one, which might help them make this decision [5].

However, despite the widespread adoption and advancement of sentiment analysis techniques, there exists a significant research gap in the context of low-resource languages, particularly in the case of Telugu. Telugu, being the fourth most spoken language in India, generates a vast amount of textual data daily, including movie reviews, social media posts, and news articles [6]. Yet, sentiment analysis tools and frameworks tailored to Telugu are lacking, hindering the ability to extract valuable insights from Telugu text data [7].

The literature review reveals that existing studies have primarily focused on sentiment analysis in languages with abundant linguistic resources, such as English, Hindi, and Tamil [8]. While these studies have contributed valuable insights and methodologies, there is a notable dearth of research addressing sentiment analysis in underrepresented languages like Telugu [9]. This research gap underscores the importance of the topic, as developing reliable sentiment analysis tools for Telugu can facilitate more accurate opinion mining, market research, and social media monitoring in Telugu-speaking regions.

Addressing this research gap aims to contribute to the advancement of sentiment analysis techniques in low-resource languages and foster more inclusive data analytics solutions tailored to diverse linguistic contexts. Through the development of a trustworthy sentiment analysis framework for Telugu, researchers, businesses, and policymakers can gain actionable insights from Telugu text data, thereby promoting linguistic diversity and cultural sensitivity in natural language processing technologies.

2. RELATED WORKS

We studied some research papers related to sentiment analysis on different languages to learn about the procedures they utilized to collect the corpus, techniques, and algorithms from those papers to make our current project easier

to complete. The authors 10] had proposed a review on Multilingual Opinion Mining which is based on Sentiment Analysis of Indigenous Languages. Their objective was to examine, evaluate, and converse about the methods, algorithms, and difficulties that researchers faced while conducting sentiment analysis on Indigenous languages in this paper. They gathered data from the internet and annotated it to their liking.

Meanwhile, [11] had suggested a novel which is based on the advances in Sentiment Analysis of Indian Languages. They carried out a thorough analysis of six distinct Indian languages (Telugu, Tamil, Bengali, Hindi, Malayalam, and Konkani). They revealed lexical resources and annotated datasets that may be useful in future work. They created a structure for sentiment analysis using the Hindi SentiWordnet and a synset replacement algorithm.

The authors [12] have presented Sentiment Analysis of Indian Languages Using Neural Network-Based framework. They used SAIL 2015 data with monolingual tweets in this paper. They developed models with optimal parameter settings for neural networks CNN, RNN, and LSTM.

The authors [13] had described sentiment analysis of tweets for Three Indian Languages. They used the SAIL dataset from 2015 in this paper. They conducted detailed error analysis and discovered that in the majority of cases, their simple models performed well and that they didn't get confusion until the sample became extremely confusing, maybe even to a human being. Meanwhile, [14] had suggested a novel which examining sentiment in Indian languages micro text utilizingRNN." On SAIL 2015, they used RNN to improve system and result accuracy.

The other authors [15] had proposed sentiment analysis of Indian languages using machine learning methods. In this paper, comparing various supervised machine learning techniques for sentiment analysis of Indian languages is presented. [16] conducted sentiment analysis for low-resource languages, focusing on informal Indonesian tweets. Their study addressed the challenges of sentiment analysis in languages with limited linguistic resources, emphasizing the importance of adapting techniques to specific language contexts.

The authors [17] analysed sentiment in Bangla movie reviews using machine learning techniques, contributing to the understanding of sentiment analysis in underrepresented languages. However, the lack of publicly available datasets for Bangla sentiment analysis remains a challenge. Furthermore, [18] proposed a fall-back strategy for sentiment analysis in Hindi, addressing the complexities of sentiment analysis in languages with rich morphology and contextual variations. Their study emphasized the need for robust sentiment analysis models tailored to specific language characteristics.

Table 1. Research Analysis

Reference No	Approach	Description	Dataset
[10]	Deep learning, Machine learning, Opinion mining	The primary goal is to evaluate, examine, and discuss the methods and algorithmic difficulties that researchers encounter.	Not available
[11]	Machine learning, sentiment analysis	Developed framework for sentiment analysis using Hindi sentiwordnet which uses sunset replacement algorithm.	Hindi tweets, tamil movie reviews, Malayalam movie reviews
[12]	RNN, CNN, LSTM,SA.	The methodologyof this paper demonstrates the SA of the SAIL 2015 data, which consists of tweets in one of the three Indian languages that are multilingual.	SAIL 2015
[13]	Word-n-gram, character-n-gram, sentiwordnet.	Performed SA of 3 Indian languages.	Not available.
[14]	Sentiment analysis, polarity measure, RNN.	Improved version of SAIL 2015 contest.	Provided by SAIL 2015.
[15]	Lexican based approach, Machine learning approach, Hybrid approach.	A comparative analysis has been made between supervised and unsupervised models.	Collected data from weblogs, news and annotated.
[16]	CNN and LSTM	Developed sentiment analysis system with CNN and LSTM for Indonesian language.	Collected data from Indonesian Wikipedia.
[17]	Support vectot machine, LSTM	Proposed a process of sentiment analysis in bangla language movie reviews.	Not available.
[18]	In-language sentiment analysis	created a sentiment-annotated corpus in the domain of Hindi film reviews	Not available.
[19]	Term Frequency-Inverse Document Frequency, Domain Specific Ontology, Contextual Semantic Sentiment Analysis.	created a model with our own corpus of data, which we gathered from tweets on Tamil films. Using the Twitter API, 50 tweets on Tamil movies were gathered for this paper.	Not available.
[20]	Aspect based analysis, Indian languages, Deep learning, Machine learning.	Annotated the data according to our requirements.	Not available.

The authors [19] predicted Tamil movie sentimental reviews using Tamil tweets, bridging the gap between informal social media data and formal movie reviews. Their study demonstrated the potential of leveraging social media data for sentiment analysis in low-resource languages like Tamil. The last authors [20] created and evaluated a dataset for aspect-based sentiment analysis in Telugu, addressing the lack of resources for sentiment analysis in the language. Their work contributes to the development of sentiment analysis tools tailored to low-resource languages like Telugu.

3. ORIGINALITY

Figure 1 shows the proposed system architecture addresses the corpus, pre-processing, vocabulary selection, vectorization, building the model, and evaluating the model. First, we must collect the corpus from the web. Next, we have to carry out some pre-processing techniques, such eliminating superfluous punctuation from the corpus using the Unicode noise character. It will produce the Token sequences and Term frequency after cleaning the corpus [21].

Based on the lowest and highest frequency of occurrence of words, choose your vocabulary. For sentiment classification, we are using label sentences, such as movie reviews. Using the label phrases and the chosen vocabulary, create the one hot encoding vectors. The machine needs numerical numbers to comprehend data because it is unable to interpret words. The categorical data must first be converted to numerical values before any kind of algorithm can be applied to the data. One method for achieving this is to use one hot encoding, which converts categorical variables to binary vectors [22].

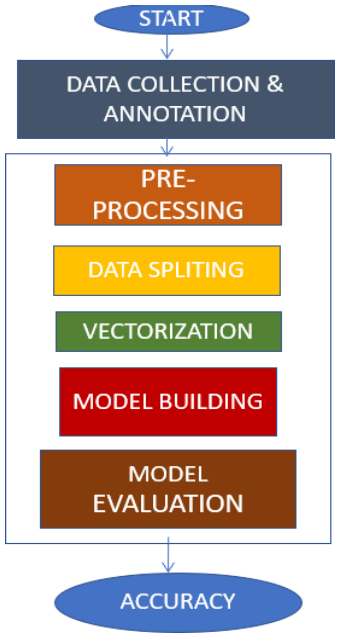

Figure 1. System Architecture for Proposed Model

3.1 Data Set

A Telugu-language dataset was required for the proposed model; however, no publicly available benchmark datasets were found. Consequently, a custom dataset of Telugu movie reviews was manually collected from multiple online sources, including movie review websites, discussion forums, and social media platforms. The collection process was designed to ensure diversity in movie genres, release periods, and audience perspectives, thereby capturing a broad range of sentiments. All reviews were carefully filtered to ensure relevance, quality, and authenticity [23].

For sentiment annotation, two proficient Telugu language experts were selected based on their language fluency, domain familiarity with movies, and consistency in judgment. The annotators were trained using predefined annotation guidelines to maintain uniform sentiment labeling. The dataset statistics and annotation details are presented in Table 2, while inter-annotator agreement was evaluated using Cohen's kappa coefficient, as reported in Table 3 [24].

Table 2. Two different annotators data

	Annotator 1	**Annotator 2**
0's	1131	1236
1's	713	6085

Table 3. Cohens kappa substitutes table

0's	1131(A)	105 (B)
1's	105 (C)	608 (D)

Cohen's kappa coefficient assesses the level of agreement between two raters:

A represents the total number of occasions in which both ratters were correct. The Ratters concur.

B represents the total number of times rater 1 stated something right while rater 2 stated it as wrong. This is a point of contention.

C represents the total amount of times rater 1 stated something was wrong but rater 2 stated something was right. This is another point of contention.

D represents the total number of times that both Ratters were incorrect. The total number of times in which rater 1 stated something was wrong but rater 2 stated it was right. It is also a point of contention.

$$\begin{aligned} \text{Probability of agreement } (P_o) &= \frac{A + D}{\text{Total}} \\ &= \frac{1131 + 608}{1844} \\ &= \frac{1739}{1844} \\ &= 0.943 \end{aligned} \tag{1}$$

Annotator A assigned 0 to 1236/1844 reviews, representing 67% (0.67) of the total. Annotator B assigned 0 to 1131/1844 reviews, accounting for 61% (0.613). The probability of both annotators randomly saying yes is P(correct) = 0.67*0.613 = 0.410.

Annotator A assigned 1s to 608 of the 1844 reviews, accounting for 32% (0.32). Annotator B assigned 1s to 713/1844 reviews, accounting for 38% (0.38) of the total. The probability of both annotators randomly saying no is P(incorrect) = 0.32*0.38 = 0.121.

$$\begin{aligned} \text{Kappa coefficient } (K) &= \frac{P_o - P_e}{1 - P_e} \\ &= \frac{0.943 - 0.410}{1 - 0.410} \\ &= \frac{0.533}{0.559} \\ &= 0.903 \end{aligned} \tag{2}$$

The range of the Kappa statistic is 0 to 1., where:

0 is equal to the agreement equivalent to choice
0.1 to 0.20 is equal to the slight agreement.
0.21 to 0.40 is equal to fair agreement.
0.41 to 0.60 is equal to moderate agreement.
0.61 to 0.80 is equal to substantial agreement.
0.81 to 0.99 is equal to near perfect agreement
1 is equal to perfect agreement.

Our dataset received a kappa coefficient value of 0.903, indicating near-perfect agreement. Table 4 describes the Noise Character Unicode representation. Algorithm 1 represents the corpus processing method.

Table 4. Noise Character Unicode representation

Sl.No.	Uni Representation	Code punctuation marks
1.	U002C	Comma { , }
2.	U002E	Full stop { . }
3.	U0028	Left parenthesis {(}
4.	U0029	Right parenthesis {)}
5.	U0021	Exclamation mark {!}
6.	U0022	Quotation mark {“}
7.	U0027	Apostrophe {`}
8.	U0020	Space
9.	U003A	Colon {:}

4. SYSTEM DESIGN

Choose the vocabulary for count vectorization based on term frequency. The vocabulary is chosen based on the term's average frequency of occurrence. The selection of word frequency is used to determine the length of the vocabulary [25]. The selection of terms was based on how frequently they appeared. Words with the highest and lowest frequency of occurrence were eliminated [26, 27]. The raw data is not immediately transformed into useful features by the TF-IDF method. First, raw strings or datasets are transformed into vectors, with a unique vector for every word [28, 29]. Figure 2 shows the TF-IDF feature selection process for Vectorization. Algorithm 2 represents the vectorization method.

Algorithm 1 Corpus Processing Method

INPUT: Raw-corpus from online newspaper
OUTPUT: Processed words
METHOD:
Step 1: Read each file from the raw corpus one by one.
Step 2:
    Initialize an empty list *vocab* = [] to store unique words.
    Initialize an empty list *all_words* = [] to store all words from all files.
    Initialize an empty dictionary *tfdict* = *{}* to store words along with their frequency counts.
    Initialize a counter *total_words* = 0 to count the total number of words.
Step 3: For every line in the input file:
    Remove leading and trailing spaces using *line.strip*()
    Initialize an empty list *text* = [] to store words for the current file.
    Split the line into individual tokens (words) using *strparts* = *line.split*()
    Let *n* = *len*(*strparts*) represent the number of tokens in the current line.
Step 4: For each word (token) in *strparts*:
    Increment *total_words* by 1
    Remove special characters from the word (e.g., commas, periods, parentheses, exclamation marks, quotes, apostrophes, colons, etc.).
    Convert the word to lowercase (optional but recommended for uniformity).
    if the cleaned token exists in *tfdict* **then**
        increment its frequency by 1.
    else
        add the token to *tfdict* with a frequency of 1.
    end if
    Append the cleaned token to the *text* list and *all_words* list.
End For
End For
Step 5: After processing all files, return *all_words* and *tfdict* (sentences).

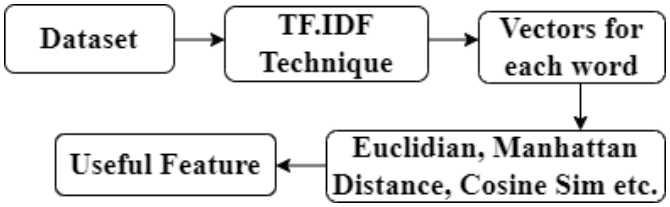

Figure 2. TF-IDF feature selection process

Algorithm 2 Vectorization

1: After pre-processing we get vocabulary stored in a dictionary.
2: For each word in a vocabulary, number is assigned.
3: Data is divided into test sets and training sets.
4: Encode training set and test set using one hot encoding or TF-IDF encoding Technique.
5: Build the vector matrix for each pair which contains a matrix with all rows with vectors for each word in vocabulary.
6: It is fed into the neural network.

5. EXPERIMENT AND ANALYSIS

5.1 Experimental Setup

The experimental setup comprised dataset preparation, preprocessing, feature extraction, model training, and evaluation. A Telugu movie review corpus was manually collected from multiple online platforms to ensure diversity across genres and audience perspectives. The data was cleaned and split into training and testing sets using Python with Pandas and NumPy [30]. Textual features were extracted using TF-IDF and Count Vectorization implemented through scikit-learn and NLTK. Support Vector Machine and Logistic Regression models were trained and evaluated under different train–test split ratios (80–20%, 70–30%, and 60–40%). Model performance was measured using accuracy, precision, recall, and F1-score. Additionally, inter-annotator agreement was computed using Cohen's kappa to validate the reliability of sentiment annotations.

5.2 Testing the Classification Accuracy

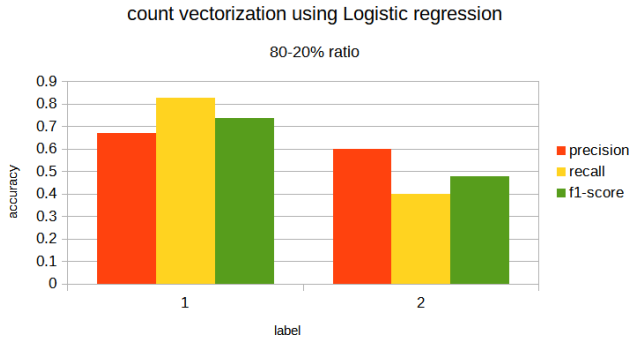

Figure 3. Accuracy of logistic regression using count Vectorization for 80-20% ratio

Count vectorization is being utilized to test the classification accuracy. The trained data is divided into 80-20% (1445_train; 362_test), 70-30% (1264_train; 543_test), and 60-40% (1084_train; 723_test) groups. When a dataset is split into

60-40% segments, count vectorization achieves greater accuracy. In this dataset, we used both logistic regression and SVM models.

Figure 3 depicts the results of using count vectorization and logistic regression for an 80-20% ratio. Table 5 shows the confusion matrix for count vectorization for an 80-20% ratio dataset. Meanwhile, Table 6 depicts the percentages of recall, precision, and F1-score for logistic regression using count Vectorization.

Table 5. Confusion matrix for count vectorization

Label	0's	1's
0's	180	38
1's	87	57

$$\begin{aligned}
\text{Precision (0)} &= \frac{TP}{TP+FP} = \frac{180}{180+87} = 0.67 \\
\text{Recall (0)} &= \frac{TP}{TP+FN} = \frac{180}{180+57} = 0.83 \\
\text{F1-score (0)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.67+0.83}{2} = 0.74 \\
\text{Precision (1)} &= \frac{TN}{TN+FN} = \frac{57}{57+38} = 0.60 \\
\text{Recall (1)} &= \frac{TN}{TN+FP} = \frac{57}{57+87} = 0.40 \\
\text{F1-score (1)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.60+0.40}{2} = 0.48 \\
\text{Accuracy} &= \frac{TP+TN}{TP+FP+FN+TN} = \frac{180+57}{180+38+87+57} = 0.65
\end{aligned} \tag{3}$$

Based on table 5, the accuracy for the logistic regression is 65.46%

Table 6. Performance Metrics for Logistic Regression using Count Vectorization

Label	F1-Score	Recall	Precision
1	0.48	0.40	0.60
0	0.74	0.83	0.67

In the other side, Figure 4 depicts the results of using count vectorization and SVM for an 80-20% ratio. The graph depicts the percentages of recall, precision, and F1-score. Table 7 shows the confusion matrix for count vectorization for an 80-20% ratio dataset. Meanwhile, Table 8 shows the comparison for SVM using Count Vectorization for various performance metrics with label 0 and 1.

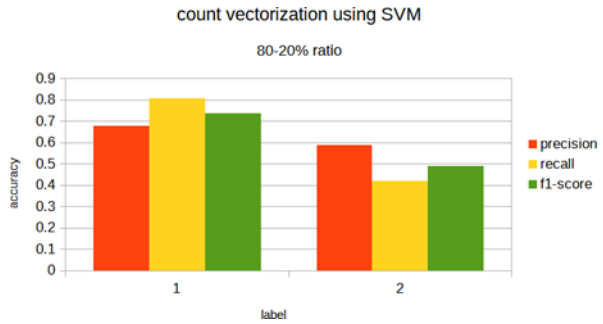

Figure 4. Accuracy of SVM using Count Vectorization

Table 7. Confusion matrix for count vectorization

Label	0's	1's
0's	177	41
1's	84	60

$$\begin{aligned}
\text{Precision (0)} &= \frac{TP}{TP+FP} = \frac{177}{177+84} = 0.68 \\
\text{Recall (0)} &= \frac{TP}{TP+FN} = \frac{177}{177+60} = 0.81 \\
\text{F1-score (0)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.68+0.81}{2} = 0.74 \\
\text{Precision (1)} &= \frac{TN}{TN+FN} = \frac{60}{60+41} = 0.59 \\
\text{Recall (1)} &= \frac{TN}{TN+FP} = \frac{60}{60+87} = 0.42 \\
\text{F1-score (1)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.59+0.42}{2} = 0.49 \\
\text{Accuracy} &= \frac{TP+TN}{TP+FP+FN+TN} = \frac{177+60}{177+41+84+60} = 0.65
\end{aligned} \tag{4}$$

From Table 7 the accuracy for the SVM is 65.46%

Table 8. Performance Metrics for SVM using Count Vectorization

Label	F1-Score	Recall	Precision
1	0.49	0.42	0.59
0	0.74	0.81	0.68

Figure 5 depicts the results of using count vectorization and logistic regression for a 70-30% ratio. The graph depicts the percentages of recall, precision, and F1-score.

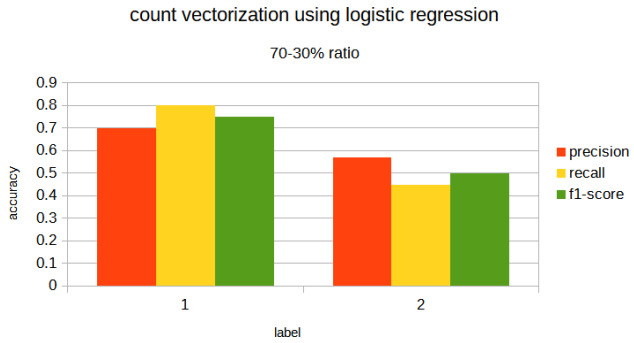

Figure 5. Accuracy of logistic regression using count vectorization

Table 9 shows the confusion matrix for count vectorization for an 70-30% ratio dataset. Meanwhile, Table 10 depicts the percentages of recall, precision, and F1-score for logistic regression using count Vectorization.

Table 9. Confusion matrix for count vectorization

Label	0's	1's
0's	269	69
1's	543	92

$$
\begin{aligned}
\text{Precision (0)} &= \frac{TP}{TP+FP} = \frac{269}{269+113} = 0.70 \\
\text{Recall (0)} &= \frac{TP}{TP+FN} = \frac{269}{269+92} = 0.80 \\
\text{F1-score (0)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.70+0.80}{2} = 0.75 \\
\text{Precision (1)} &= \frac{TN}{TN+FN} = \frac{92}{92+69} = 0.57 \\
\text{Recall (1)} &= \frac{TN}{TN+FP} = \frac{92}{92+113} = 0.45 \\
\text{F1-score (1)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.57+0.45}{2} = 0.50 \\
\text{Accuracy} &= \frac{TP+TN}{TP+FP+FN+TN} = \frac{269+92}{269+69+113+92} = 0.66
\end{aligned}
\tag{5}
$$

From the overall analysis of Table 9 the accuracy for the logistic regression is 66.48%.

Table 10. Performance Metrics for Logistic Regression using Count Vectorization

Label	F1-Score	Recall	Precision
1	0.50	0.45	0.57
0	0.75	0.80	0.70

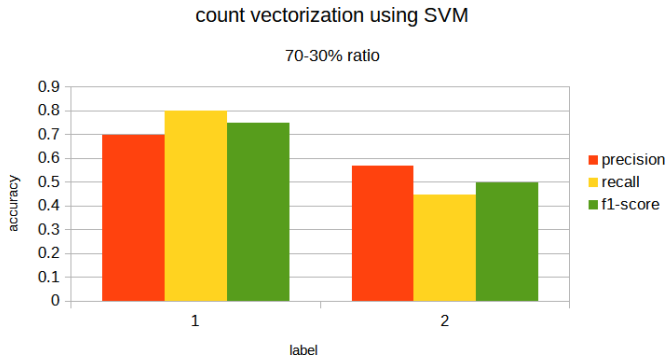

Figure 6. Accuracy of SVM using Count Vectorization

In the other side, Figure 6 depicts the results of using count vectorization and SVM for a 70-30% ratio. The graph depicts the percentages of recall, precision, and F1-score. Table 11 shows the confusion matrix for count vectorization for an 70-30% ratio dataset. Table 12 shows the comparison of performance metrics for SVM using Count Vectorization for label 1 and 0.

Table 11. Confusion matrix for count vectorization

Label	0's	1's
0's	270	68
1's	68	92

$$\begin{aligned}
\text{Precision (0)} &= \frac{TP}{TP+FP} = \frac{270}{270+113} = 0.70 \\
\text{Recall (0)} &= \frac{TP}{TP+FN} = \frac{270}{270+92} = 0.80 \\
\text{F1-score (0)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.70+0.80}{2} = 0.75 \\
\text{Precision (1)} &= \frac{TN}{TN+FN} = \frac{92}{92+68} = 0.57 \\
\text{Recall (1)} &= \frac{TN}{TN+FP} = \frac{92}{92+113} = 0.45 \\
\text{F1-score (1)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.57+0.45}{2} = 0.50 \\
\text{Accuracy} &= \frac{TP+TN}{TP+FP+FN+TN} = \frac{270+92}{270+68+113+92} = 0.67
\end{aligned} \tag{6}$$

From the overall analysis of Table 11 the accuracy for the SVM is 66.66%.

Table 12. Performance Metrics for SVM using Count Vectorization

Label	F1-Score	Recall	Precision
1	0.50	0.45	0.57
0	0.75	0.80	0.70

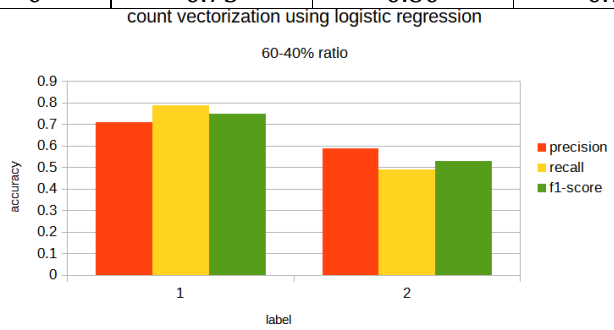

Figure 7. Accuracy of logistic regression using count vectorization

Figure 7 depicts the results of using count vectorization and logistic regression for a 60-40% ratio. The graph depicts the percentages of recall, precision, and F1-score. Table 13 shows the confusion matrix for count vectorization for an 60-40% ratio dataset. Meanwhile, Table 14 shows the comparison of performance metrics for Logistic Regression using Count Vectorization for label 1 and 0.

Table 13. Confusion matrix for count vectorization

Label	0's	1's
0's	350	95
1's	143	135

$$
\begin{aligned}
\text{Precision (0)} &= \frac{TP}{TP+FP} = \frac{350}{350+143} = 0.71 \\
\text{Recall (0)} &= \frac{TP}{TP+FN} = \frac{350}{350+135} = 0.79 \\
\text{F1-score (0)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.71+0.79}{2} = 0.75 \\
\text{Precision (1)} &= \frac{TN}{TN+FN} = \frac{135}{135+95} = 0.59 \\
\text{Recall (1)} &= \frac{TN}{TN+FP} = \frac{135}{135+143} = 0.49 \\
\text{1-score (1)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.59+0.49}{2} = 0.53 \\
\text{Accuracy} &= \frac{TP+TN}{TP+FP+FN+TN} = \frac{350+135}{350+95+143+135} = 0.67
\end{aligned}
\tag{7}
$$

From the overall analysis of Table 13 the accuracy for the logistic regression is 67.08 %

Table 14. Performance Metrics for Logistic Regression using Count Vectorization

Label	F1-Score	Recall	Precision
1	0.53	0.49	0.59
0	0.75	0.79	0.71

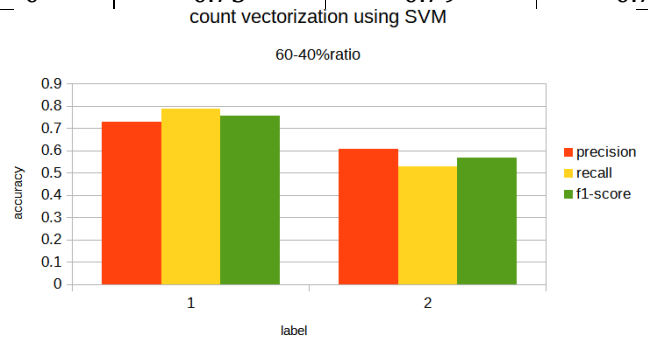

Figure 8. Accuracy of SVM using Count Vectorization

Figure 8 shows the results while using count vectorization and SVM for 60-40% ratio. Table 15 shows the confusion matrix for count vectorization for an 70-30% ratio dataset. The graph shows recall, precision and F1-score percentages. Table 16 shows the comparison of performance metrics for SVM using Count Vectorization for label 1 and 0.

Table 15. Confusion matrix for count vectorization

Label	0's	1's
0's	351	94
1's	130	148

Table 16. Performance Metrics for SVM using Count Vectorization

Label	F1-Score	Recall	Precision
1	0.53	0.49	0.59
0	0.75	0.79	0.71

$$\begin{aligned}
\text{Precision (0)} &= \frac{TP}{TP+FP} = \frac{351}{351+130} = 0.73 \\
\text{Recall (0)} &= \frac{TP}{TP+FN} = \frac{351}{351+148} = 0.79 \\
\text{F1-score (0)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.73+0.79}{2} = 0.76 \\
\text{Precision (1)} &= \frac{TN}{TN+FN} = \frac{148}{148+94} = 0.61 \\
\text{Recall (1)} &= \frac{TN}{TN+FP} = \frac{148}{148+130} = 0.53 \\
\text{F1-score (1)} &= \frac{\text{recall} + \text{precision}}{2} = \frac{0.61+0.53}{2} = 0.57 \\
\text{Accuracy} &= \frac{TP+TN}{TP+FP+FN+TN} = \frac{351+148}{351+94+130+148} = 0.69
\end{aligned} \tag{8}$$

From the overall analysis of Table 15 the accuracy for the SVM is 69.01%.

5.3 Testing the Classification Accuracy using TF-IDF

The classification accuracy is being tested here using TF-IDF. The trained data is divided into 80-20% (1445_train; 362_test), 70-30% (1264_train;543_test), and 60-40% (1084_train; 723_test) groups. When the dataset is divided into 80-20%, TF-IDF can be used. In this dataset, we used both logistic regression and SVM models.

Here are the results:

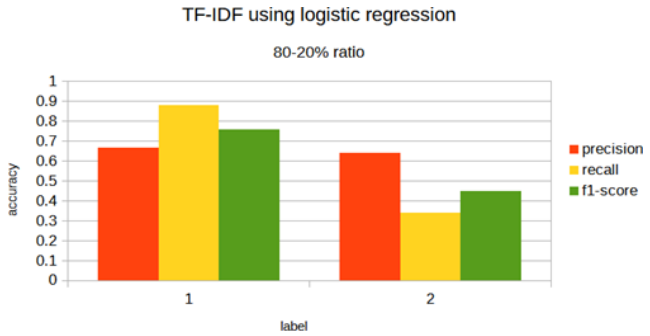

Figure 9. Accuracy of logistic regression using TF-IDF

Figure 9 shows the results while using count TF-IDF and Logistic regression for 80-20% ratio. The graph shows recall, precision and f1-score percentages. Table 17 and 18 shows the accuracy and performance metrics comparison for logistic regression using TF-IDF.

Table 17. Confusion matrix for TF-IDF

Label	0's	1's
0's	191	27
1's	95	49

Table 18. Performance Metrics for Logistic Regression using TF-IDF

Label	F1-Score	Recall	Precision
1	0.48	0.40	0.60
0	0.74	0.83	0.67

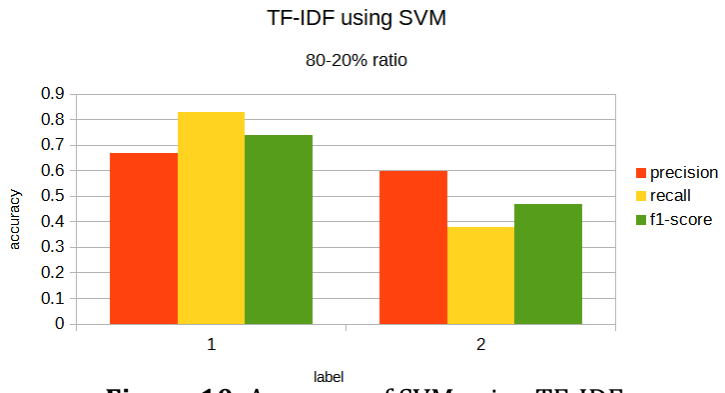

Figure 10. Accuracy of SVM using TF-IDF

Figure 10 depicts the results of using count TF-IDF and SVM for an 80-20% ratio. The graph depicts the percentages of recall, precision, and F1-score. Table 19 and 20 shows the accuracy and performance metrics comparison for SVM using TF-IDF.

Table 19. Confusion matrix for TF-IDF

Label	0's	1's
0's	181	37
1's	89	55

Table 20. Performance Metrics for SVM using TF-IDF

Label	F1-Score	Recall	Precision
1	0.49	0.42	0.59
0	0.74	0.81	0.68

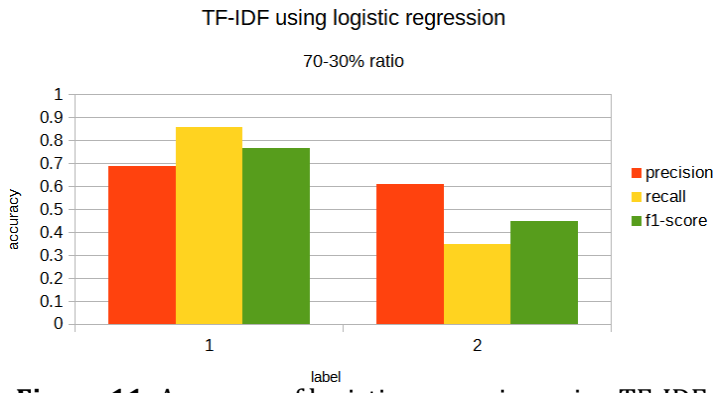

Figure 11. Accuracy of logistic regression using TF-IDF

Figure 11 depicts the results of using count TF-IDF and logistic regression for a 70-30% ratio. The graph depicts the percentages of recall, precision, and F1-score. Table 21 shows the confusion matrix for TF-IDF and Table 22 shows the performance metrics comparison for logistic regression using TF-IDF.

Table 21. Confusion matrix for TF-IDF

Label	0's	1's
0's	292	46
1's	193	72

Table 22. Performance Metrics for Logistic Regression using TF-IDF

Label	F1-Score	Recall	Precision
1	0.45	0.34	0.64
0	0.76	0.88	0.67

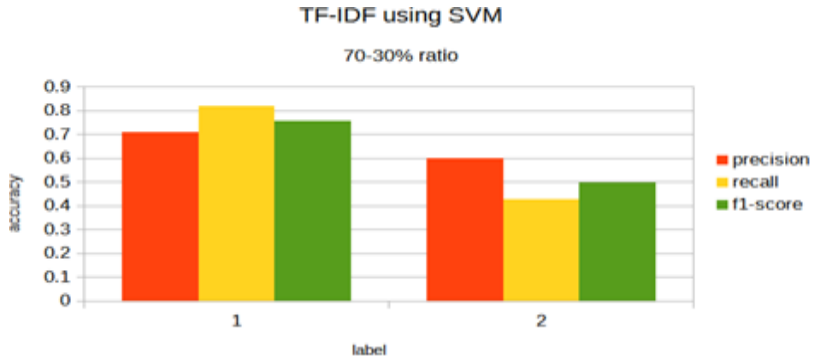

Figure 12. Accuracy of SVM using TF-IDF

Figure 12 depicts the results of using count TF-IDF and SVM for a 70-30% ratio. The graph depicts the percentages of recall, precision, and f1-score. Table 23 shows the confusion matrix for TF-IDF using label 1, 0 and Table 24 shows the performance metrics comparison for SVM using TF-IDF.

Table 23. Confusion matrix for TF-IDF

Label	0's	1's
0's	278	60
1's	116	89

Table 24. Performance Metrics for SVM using TF-IDF

Label	F1-Score	Recall	Precision
1	0.50	0.43	0.60
0	0.76	0.82	0.71

Figure 13. Accuracy of logistic regression using TF-IDF

Figure 13 depicts the results of using count TF-IDF and logistic regression for a 60-40% ratio. The graph depicts the percentages of recall, precision, and f1-score. Table 25 shows the confusion matrix for TF-IDF and table 26 shows the performance metrics comparison for logistic regression using TF-IDF.

Table 25. Confusion matrix for TF-IDF

Label	0's	1's
0's	381	64
1's	189	89

Table 26. Performance Metrics for Logistic Regression using TF-IDF

Label	F1-Score	Recall	Precision
1	0.45	0.35	0.61
0	0.77	0.86	0.69

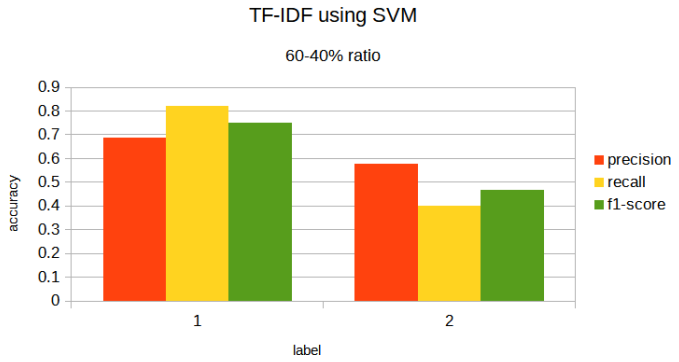

Figure 14. Accuracy of SVM using TF-IDF

Figure 14 depicts the results of using count TF-IDF and SVM for a 60-40% ratio. The graph depicts the percentages of recall, precision, and f1-score. Table 27 shows the confusion matrix for TF-IDF and table 28 shows the performance metrics comparison for SVM using TF-IDF.

Table 27. Confusion matrix for TF-IDF

Label	0's	1's
0's	365	80
1's	167	111

Table 28. Performance Metrics for SVM using TF-IDF

Label	F1-Score	Recall	Precision
1	0.50	0.43	0.60
0	0.76	0.82	0.71

5.4 Comparison between Logistic regression and SVM

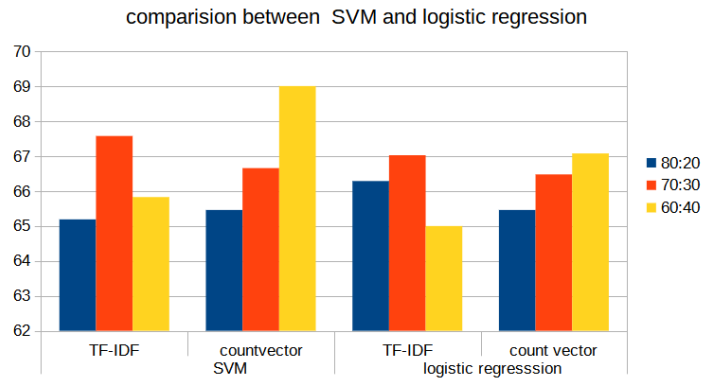

Figure 15. Comparison between Logistic regression and SVM

- The highest accuracy in a support vector machine (SVM) is achieved using TF-IDF in a 70-30% ratio, whereas it is achieved in a 60-40% ratio using count vectorization.

- In logistic regression, the highest accuracy is achieved using TF-IDF in a 70-30% ratio, whereas it is achieved in a 60-40% ratio using count vectorization. Figure 15 shows the overall comparison between Logistic regression and SVM.

In comparison to TF-IDF, count vectorization has attained higher accuracy.

Table 29. Comparison between logistic regression and SVM

Split Ratio	Logistic Regression		SVM	
Ratio	TF – IDF	CV	TF – IDF	CV
80 – 20%	66.29	65.46	65.19	65.46
70 – 30%	67.03	66.48	67.58	66.66
60 – 40%	65.00	67.08	65.83	69.01

Vectorization was carried out using both Count vectorization and TF-IDF, then classified SVM and Logistic regression using two Encoding techniques and three different Training and Testing ratios as described in Table 29.

The Best Performance received the following:

- Best performances at 70:30 Test and Train ratio for logistic regression using TF-IDF vectorization.
- Best performances at 60:40 Train and Test ratio for logistic regression using count vectorization.
- Best performances at 70:30 Test and Train ratio for SVM using TFIDF vectorization.
- Bestperformances at 60:40 Train and Test ratio for SVM using count vectorization.

5.5 Comparison of Proposed approach with other Existing Approaches

Table 30 provides a comparison of existing sentiment analysis methodologies such as Word2Vec, Count Vectorizer, and TF-IDF, with the proposed approach. Each method's performance is assessed based on its accuracy and precision across two sentiment labels: 0 and 1, denoting negative and positive sentiments, respectively. Word2Vec, leveraging word embeddings, achieves an accuracy of 0.58 and a precision of 0.50 for label 0, and an accuracy of 0.61 and a precision of 0.63 for label 1. Count Vectorizer, employing frequency-based numerical feature vectors, demonstrates an accuracy of 0.60 and a precision of 0.54 for label 0, and an accuracy of 0.64 and a precision of 0.65 for label 1. TF-IDF, utilizing term frequency-inverse document frequency weighting, yields an accuracy of 0.63 and a precision of 0.57 for label 0, and an accuracy of 0.67 and a precision of 0.69 for label 1. The proposed method achieves an accuracy of 0.65 and a precision of 0.59 for label 0, and an accuracy of 0.69 and a precision of 0.71 for label 1. This comparative analysis provides insights into the effectiveness of these methodologies, with the proposed approach demonstrating superior

performance in accurately classifying sentiment polarity.

Table 30. Comparison of existing sentiment analysis methodologies

Methods	Label	Accuracy	Precision
Word2Vec	0	0.58	0.50
	1	0.61	0.63
Count vectorizer	0	0.60	0.54
	1	0.64	0.65
TF-IDF	0	0.63	0.57
	1	0.67	0.69
Proposed	0	0.65	0.59
	1	0.69	0.71

Table 31 provides a technical comparison of sentiment analysis methodologies with existing literature, focusing on deep learning architectures. The methods evaluated include RNN (Recurrent Neural Network), CNN (Convolutional Neural Network), LSTM (Long Short-Term Memory), and the proposed approach. Each method is assessed based on two key performance metrics: accuracy and recall, across two sentiment labels: 0 (representing negative sentiment) and 1 (representing positive sentiment). For RNN, the accuracy and recall values for label 0 are 0.56 and 0.49, respectively, while for label 1, they are 0.60 and 0.61, respectively. Similarly, CNN achieves an accuracy and recall of 0.58 and 0.52 for label 0, and 0.62 and 0.63 for label 1. LSTM demonstrates accuracy and recall of 0.61 and 0.56 for label 0, and 0.64 and 0.68 for label 1. Particularly, the proposed method outperforms the existing methodologies, with an accuracy and recall of 0.65 and 0.59 for label 0, and 0.69 and 0.71 for label 1, respectively. These findings highlight the comparative effectiveness of deep learning-based approaches in sentiment analysis, with the proposed method demonstrating superior performance in accurately identifying sentiment polarity compared to existing methodologies outlined in the literature.

Table 31. Comparison with literature review methodologies

Methods	Label	Accuracy	Recall
RNN	0	0.56	0.49
	1	0.60	0.61
CNN	0	0.58	0.52
	1	0.62	0.63
LSTM	0	0.61	0.56
	1	0.64	0.68
Proposed	0	0.65	0.59
	1	0.69	0.71

The research conducted in this article brings several benefits to the field of sentiment analysis, particularly in the context of Indian languages. It expands the understanding of sentiment analysis methodologies tailored specifically for

Indian languages, which have unique linguistic characteristics and cultural nuances. The findings of the research can have practical implications for industries and applications that rely on sentiment analysis, such as social media monitoring, customer feedback analysis, and market research.

6. CONCLUSION

In this paper, we annotated 1844 sentences from the raw corpus, with 1807 sentences serving as the dataset. Kappa's coefficient was calculated for precise precision using count vectorization and TF-IDF vectorization methods. SVM and logistic regression algorithms were used. Among the combinations used for our dataset, TF-IDF with SVM for a 70-30% ratio provides the highest accuracy. The accuracy is determined by the splitting ratios and methods used. In the future, the dataset annotation process can be used, and datasets can be created by following the data set creation and annotation process. The Kappa coefficient can be used to measure the precision and accuracy of the process. The study's limitations include a focus on specific datasets and metrics, potentially limiting generalizability. Future research could explore diverse datasets, incorporate additional metrics, and investigate advanced model architectures to enhance sentiment analysis methodologies and broaden their applicability across various domains.

REFERENCES

[1] Z. Luo, M. Osborne, and T. Wang, **An effective approach to tweets opinion retrieval**, *World Wide Web*, Vol. 18, No. 3, pp. 545–566, 2015.

[2] S. A. Z. Farmadi, A. R. Barakbah, and E. M. Kusumaningtyas, **Smart I'rab: Smart Aplication for Arabic Grammar Learning**, *EMITTER Int'l J. of Eng. Tech.*, Vol. 1, No. 1, pp. 1-10, 2013.

[3] Z. Wang, P. Gao, and X. Chu, **Sentiment analysis from Customer-generated online videos on product review using topic modeling and Multi-attention BLSTM**, Advanced Engineering Informatics, Vol. 52, p. 101588, 2022.

[4] V. U. Ramya, and K. T. Rao, **Sentiment analysis of movie review using machine learning techniques**, International Journal of Engineering & Technology, Vol. 3, No. 7, pp. 676-681, 2018.

[5] V. K. Singh, R. Piryani, A. Uddin, and P. Waila, **Sentiment Analysis of Movie Reviews and Blog Posts**, *Proceedings of the 3rd IEEE International Advance Computing Conference (IACC)*, Ghaziabad, India, pp. 893-898, 2013.

[6] V. B. Viswanadh, **Sentiment Analysis of Telugu News Articles Decoding Textual Nuances**, *Proceedings of the 2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE)*, Vellore, India, pp. 1-6, 2024.

[7] A. H. N. Karthik, C. Aneesh, G. Saumik, K. V. V. Varun, and K. CR, **Sentiment Analysis on Telugu Text Translated from English Using NLP and ML**, *Proceedings of the 2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, Bengaluru, Indial pp. 246-253, 2025.

[8] V. P. Vasani, and A. Asha, **A review based on sentimental analysis for Hindi language**, *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 37, No. 5, pp. 1815-1829, 2025.

[9] G. L. Anand Babu, and S. Badugu, **Extractive summarization of telugu text using modified text rank and maximum marginal relevance**, *ACM Transactions on Asian and Low-Resource Language Information Processing*, Vol. 22, No. 9, pp. 1-18, 2023.

[10] S. R. Shah, and A. Kaushik, **Sentiment analysis on indian indigenous languages: a review on multilingual opinion mining**, *arXiv preprint arXiv:1911.12848*, 2019.

[11] M. B. Shelke, and S. N. Deshmukh, **Recent advances in sentiment analysis of Indian languages**, *International Journal of Future Generation Communication and Networking*, Vol. 13, No. 4, pp. 1656-1675, 2020.

[12] R. Bhargava, S. Arora, and Y. Sharma, **Neural network-based architecture for sentiment analysis in Indian languages**, *J. Intell. Syst.*, Vol. 28, No. 3, pp. 361-375, 2019.

[13] S. Phani, S. Lahiri, and A. Biswas, **Sentiment analysis of tweets in three Indian languages**, *Proceedings of the 6th workshop on South and Southeast Asian natural language processing (WSSANLP2016)*, Osaka, Japan, pp. 93-102, 2016.

[14] S. Seshadri, A. K. Madasamy, S. K. Padannayil, and M. A. Kumar, **Analyzing sentiment in indian languages micro text using recurrent neural network**, *IIOAB J*, Vol. 7, pp. 313-318, 2016.

[15] G. I. Ahmad, and J. Singla, **Machine learning techniques for sentiment analysis of indian languages**, *Int. J. Recent Technol. Eng.*, Vol. 8, No. 2, pp. 3630-3636, 2019.

[16] T. A. Le, D. Moeljadi, Y. Miura, and T. Ohkuma, **Sentiment analysis for low resource languages: A study on informal Indonesian tweets**, *Proceedings of the 12th Workshop on Asian Language Resources (ALR12)*, Osaka, Japan, pp. 123-131, 2016.

[17] R. R. Chowdhury, M. S. Hossain, S. Hossain, and K. Andersson, **Analyzing sentiment of movie reviews in Bangla by applying machine learning techniques**, *Proceedings of the 2019 International Conference on Bangla speech and language processing (ICBSLP)*, Sylhet, Bangladesh, pp. 1-6, 2019.

[18] A. Joshi, A. R. Balamurali, and P. Bhattacharyya, **A fall-back strategy for sentiment analysis in Hindi: a case study**, *Proceedings of ICON 2010: 8th International Conference on Natural Language Processing*, Macmillan

Publishers, India, pp. 1-6, 2010.
[19] V. Ramanathan, T. Meyyappan, and S. M. Thamarai, **Predicting Tamil movies sentimental reviews using Tamil tweets**, *Journal of Computer Science*, Vol. 15, No. 11, pp. 1638-1647, 2019.
[20] Y. R. Regatte, R. R. R. Gangula, and R. Mamidi, **Dataset creation and evaluation of aspect based sentiment analysis in Telugu, a low resource language**, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, Marseille, France, pp. 5017-5024, 2020.
[21] X. Fang, and J. Zhan, **Sentiment analysis using product review data**, *Journal of Big Data*, Vol. 2, No. 5, pp. 1-14, 2015.
[22] A. Alsaeedi, and M. Z. Khan, **A study on sentiment analysis techniques of Twitter data**, *Int. J. Adv. Comput. Sci. Appl.*, Vol. 10, No. 2, pp. 361-374, 2019.
[23] K. Chattu, K. A. N. Reddy, S. B. Veesam, P. S. Chirumamilla, V. Dinesh Babu, K. Prakash, S. Bansal, M.R.I. Faruque, and K.S. Al-Mugren, **Sentiment classification for telugu using transformed based approaches on a multi-domain dataset**, *Scientific Reports*, Vol. 15, No. 1, p. 22185, 2025.
[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, **Distributed representations of words and phrases and their compositionality**, *Advances in neural information processing systems*, Vol. 26, 2013.
[25] A. Bhansali, A. Chandravadiya, B. Y. Panchal, M. H. Bohara, and A. Ganatra, **Language identification using combination of machine learning algorithms and vectorization techniques**, *Proceedings of the 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE),* Greater Noida, India, pp. 1329-1334, 2022.
[26] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, **Natural language processing (almost) from scratch**, *J. Mach. Learn. Res.*, Vol. 12, pp. 2493-2537, 2011.
[27] M. A. Zahran, A. Magooda, A. Y. Mahgoub, H. Raafat, M. Rashwan, and A. Atyia, **Word Representations in Vector Space and their Applications for Arabic**, *Computational Linguistics and Intelligent Text Processing: 16th International Conference, CIC Ling 2015*, Cairo, Egypt, pp. 430-443, 2015.
[28] M.-T. Luong, R. Socher, and C. D. Manning, **Better Word Representations with Recursive Neural Networks for Morphology**, *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, Sofia, Bulgaria, pp. 104-113, 2013.
[29] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, **Enriching word vectors with subword information**, *arXiv preprint arXiv:1607.04606*, 2016.
[30] S. S. Mukku, **Sentiment Analysis for Telugu Language**, Ph.D thesis, International Institute of Information Technology (Deemed to be University), 2017.