A Mini Project Report On

SENTIMENT ANALYSIS ON HINDI REVIEWS

Submitted in Fulfilment of the Requirements for the Award of Degree

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By:

DUDIGANI SAI PRANATHI (19015A0403)

SANGALA RAGHU VAMSI (18011A0444)

KUMMARI BHARATH (18011A0422)

JAIRAM ABHISHEK (19015A0405)

Under the esteemed guidance of

DR. L. PRATAP REDDY

Professor in ECE Department

Department Of Electronics and Communication Engineering

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

College of Engineering Hyderabad

Autonomous

(Kukatpally-Hyderabad-500085)

2021-22

Presented to the Faculty Of

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY-COLLEGE OF ENGINEERING HYDERABAD-500085

In Partial Fulfilment of the Requirements for the Degree of

ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE BY THE SUPERVISOR

This is to certify that the project entitled "**SENTIMENT ANALYSIS ON HINDI REVIEWS**" being submitted by

DUDIGANI SAI PRANATHI (19015A0403)

SANGALA RAGHU VAMSI (18011A0444)

KUMMARI BHARATH (18011A0422)

JAIRAM ABHISHEK (19015A0405)

in partial fulfilment of the requirements for the award of degree in Bachelors of Technology in Electronics and Communication Engineering at the Jawaharlal Nehru Technological University during the academic year 2021-22 is a bonafide work carried out under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or institute for the award of any degree or diploma.

Project Supervisor

DR. L. PRATAP REDDY

(Professor)

Presented to the Faculty Of

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY COLLEGE OF ENGINEERING HYDERABAD-500085

In Partial Fulfilment of the Requirements for the Degree of

ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE BY THE HEAD OF THE DEPARTMENT

This is to certify that the project report entitled **"SENTIMENT ANALYSIS ON HINDI REVIEWS"**
Submitted by

DUDIGANI SAI PRANATHI (19015A0403)

SANGALA RAGHU VAMSI (18011A0444)

KUMMARI BHARATH (18011A0422)

JAIRAM ABHISHEK (19015A0405)

In partial fulfilment of the requirements for the award of degree in Bachelor of Technology in Electronics and Communication Engineering at the Jawaharlal Nehru Technological University during the academic year 2021-22 is a bonafide work carried out by them.

HEAD OF THE DEPARTMENT

DR. K. ANITHA SHEELA

(Professor of ECE department JNTUH-CEH)

Presented to the Faculty Of

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY COLLEGE OF ENGINEERING HYDERABAD-500085

In Partial Fulfilment of the Requirements for the Degree of

ELECTRONICS AND COMMUNICATION ENGINEERING

DECLARATION OF THE CANDIDATE

We hereby declare that the project report title "**SENTIMENT ANALYSIS ON HINDI REVIEWS**" is a bonafide record work done and submitted under the esteemed guidance of **Dr. L. PRATAP REDDY**, Professor, Department of ECE, JNTUH-CEH, in partial fulfilment of the requirements for Mini project in Electronics and Communication Engineering at the Jawaharlal Nehru Technological University during the academic year 2021-22 is a bonafide work carried out by us and the results kept in the minor project have not been reproduced. The results have not been submitted in any other institute of university for the award of degree or diploma.

By:

DUDIGANI SAI PRANATHI (19015A0403)

SANGALA RAGHU VAMSI (18011A0444)

KUMMARI BHARATH (18011A0422)

JAIRAM ABHISHEK (19015A0405)

ACKNOWLEDGEMENT

This project titled "**SENTIMENT ANALYSIS ON HINDI REVIEWS**" was carried out by us. We are grateful to **Dr. L. PRATAP REDDY,** Professor and **Dr. K. ANITHA SHEELA**, Professor and Head of the Department of Electronics and Communication Engineering, JNTUH College of Engineering, for their guidance while pursuing this project.

We take this precious opportunity to acknowledge our internal project guide **Dr. L. PRATAP REDDY**, Professor of Electronics and Communication Engineering, JNTUH College of Engineering, Hyderabad for his timely advice, effective guidance and encouragement throughout the completion of our mini-project work.

We also owe a deep respect of gratitude to our parents and friends for their cheerful encouragement and valuable suggestions, without whom this work would not have been completed in this stipulated time.

We would like to articulate our heartfelt gratitude to the authorities of JNTU for their help throughout our project work. A few lines of acknowledgement do not fully express our gratitude and appreciation for all those who guided and supported us throughout this project. Last but not in the least, we acknowledge the help received from many journals and websites.

Finally, we thank one and all who helped us directly or indirectly throughout our project work.

By:

DUDIGANI SAI PRANATHI (19015A0403)

SANGALA RAGHU VAMSI (18011A0444)

KUMMARI BHARATH (18011A0422)

JAIRAM ABHISHEK (19015A0405)

ABSTRACT

Sentiment analysis is the natural language processing task which helps to identify and categorize opinions expressed in a piece of text, determine the reviewer's point of view on a particular topic. It combines the techniques of computational linguistics and Information Retrieval. Most of the research on Sentiment Analysis has been focussed on English language and very less attention is paid in direction of sentiment analysis in Indian languages. Increasing user-generated content in Hindi on the internet has motivated us to perform sentiment analysis research on movie reviews in Hindi.

In this project, sentiment analysis is performed on Hindi movie reviews. Sentiment analysis is a text classification problem in which text is assigned as positive, negative or neutral depending on sentiment which it strongly forces. In this two different classification approaches: Resource based classification using HindiSentiWordNet (H-SWN) and In-language classification are discussed.

CONTENTS

1 INTRODUCTION

Sentiments are hidden behind online comments on social media of all kinds. In the last decade there is a rise of social media such as blogs and social networks which has fuelled the interest in sentiment analysis. Online opinion has turned into a kind of virtual currency with the proliferation of reviews, ratings, recommendations and other forms of online expression, for businesses that are looking to market their products, identify new opportunities and manage their reputations. In order to automate the process of filtering out the noise, understanding the conversations, identifying the relevant content and following appropriate actions, many are now looking to the field of sentiment analysis. Therefore instead of navigating the comments thoroughly, a more efficient method to extract the sentiments behind texts is needed

Hindi is fourth highest speaking language in the world. It is spoken nearly by 425 million people as first language and 120 million people as a second language. Lots of websites, blogs and tweets now a days support Hindi language and some of them use Hindi as a primary language as well. But most of the research on Sentiment Analysis has been focused on English language and very less attention is paid in direction of sentiment analysis in Hindi language.

Sentiment Analysis in Hindi is very challenging due to the following reasons:

1. Hindi is a resource scarce language which causes problems in collection and generation of datasets. Also there are not efficient parsers and taggers for this language.
2. Hindi is morphologically rich and a free order language as compared to English language. It means there is no specific arrangement of words in Hindi language i.e. subject, object and verb come in any order whereas English is fixed word order language i.e. subject is always followed by a verb and then followed by an object. Word order is important for determining the polarity of a given text.
3. Unavailability of well annotated standard corpora.
4. Limited resources are available for it like Hindi-SentiWordNet (H-SWN). It consists of limited number of adjectives and adverbs.

Applications of Sentiment Analysis are endless.

1. Sentiment analysis has been used by e-commerce companies for customer satisfaction. You can estimate how happy customers are by estimating the ratio b/w positive and negative reviews
2. Sentiment Analysis is used every day in social media, surveys, feedbacks to identify the needs of the people
3. To identify the detractors and promoters

2 RELATED WORK

Techniques for Sentiment analysis have been developing in recent years. In general, sentiment analysis includes both the processes of feature extraction and sentiment - classification [1]. Very few research work has been done related to sentiment analysis in Hindi. The earliest of them was by Aditya Joshi et al [5]. They proposed that a fall-back strategy could be adopted for doing sentiment analysis for a new language. They suggested that we could first of all train a sentiment classifier on In-language labelled corpus and use it to classify a new document. In case of In-language data not being available, rough machine translation could be applied to convert the document into a resource rich language like English. Hence, the polarity of translated document could be predicted using a classifier for English, assuming polarity is not lost in translation.

An important contribution to Hindi Polarity Classification was done by Bakliwal et al [3]. Their major contribution was that they created a resource for Hindi by using Hindi WordNet to retrieve synonyms and antonyms of a given word in Hindi for which they knew the polarity and then assigned the similar polarity to synonyms and opposite polarity to antonyms. They also developed an annoted corpora of Hindi Product Reviews.

An efficient approach was developed by Namita mittal et al. [6] based on negation and discourse relation for identifying the sentiments from Hindi content. The annotated corpus for Hindi language was developed and existing Hindi-SentiWordNet (H-SWN) was improved by incorporating more opinion words into it. They also devised the rules for handling negation and discourse that affect the sentiments expressed in the review.

Another important work was done by Piyush Arora [2]. He proposed a technique to build a subjective lexicon given a pre-annotated seed list for a language and its WordNet representing the network/connectivity of words using synonyms and antonyms relations. One of the salient features of his approach is that his technique could be applied for any language which has the WordNet available.

Namam Bansal et. al [4] proposed a semi-supervised approach to train a Deep Belief Network on a small percentage of labelled data and assign polarity to unlabelled data. They used semi-supervised learning because supervised polarity classification systems are domain-specific and hence systems trained on one dataset typically perform much worse on a different dataset. They also stated that annotating a large amount of data could be an expensive process.

A novel approach was proposed by Richa Sharma et al. [7] in which they developed a Hindi language opinion mining system to classify reviews as positive, negative or neutral. They also handled negation in their proposed system. Instead of using Wordnet, they developed their own Hindi dictionary to determine the polarity of Hindi reviews. In this project we have analysed and implemented two research papers namely, Sentiment Analysis

in Hindi by Naman Bansal et. al [4] and sentiment Analysis for Hindi Language by Piyush Arora [2]. The accuracy and f-measure obtained by following the models implemented in the above mentioned papers are compared.

3 WORK DONE

In this section, we have proposed the details of dataset and models used for reviews pre-processing, algorithms used for feature set generation and various classifiers used.

3.1 Dataset Used

We have used 250 sentences (125 positive and 125 negative) of movie reviews available from IIT Bombay for research purposes [4]. In addition to this, we have manually collected and labelled around 750 sentences of movie reviews (375 positive and 375 negative reviews) from Hindi review site (jagran.com). In total, we have a dataset of 1000 movie reviews.

3.2 Data Pre-processing

We have removed all those words from each review which do not contribute to the accuracy of classification. These words are basically as follows:

1. Punctuations: They are symbols such as ”,”,”—”,”!” etc. which are used in writing to separate sentences and their elements and to clarify the meaning.
2. Numbers: Numbers do not contribute to the accuracy and has no meaning in sentiment analysis. So they are removed during pre-processing.
3. Stop words: The natural language words which have very little meaning such as articles, pronouns and prepositions are denoted as stop words.
4. One length words and etc.

3.3 Feature Extraction

Once the pre-processing of data is done, we compute the feature matrix using the TF-IDF and unigram model.

3.3.1 TF-IDF algorithm

In this project, a function named TfidfVectorizer() available in scikit-learn library [8] is used to convert a collection of raw documents to a matrix of TF-IDF features. The goal of using TF-IDF instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus.

3.3.2 Unigram model

Unigram model considers each word at a time. It doesn't take word ordering into account, so the order doesn't make a difference in how words are tagged or split up. In this model, we create a lexicon containing all the words that occur in any review of our dataset. Lexicons are the set of combined word of all the positive and negative reviews. We consider only those words in the lexicon which have frequency count in a specific range in order to eliminate those words that do not contribute much in sentiment classification. We generate a feature matrix of size m*n (where m= number of reviews in our dataset and n= number of words in the lexicon). For each element of the matrix, if that lexicon word occurs in the review, the element is assigned frequency count of that word in the review.

Algorithm 1 Feature matrix generation using unigram model

Input: We have a list of reviews, R which contain positive reviews and negative reviews in string format. R = {p1, p2....pm1 , n1, n2....nm2 }, where m1 = number of positive reviews and m2 = number of negative reviews

Output: A feature set F, which is list of features for each review. F= {f1, f2...fm1 , fm1+1...fm1+m2 } size of feature set F : (m1 + m2) ∗ n where n = the number of features in a single review that is equal to the length of lexicons set

```
BEGIN
Create a set of lexicons L.
For each review ri in R:
        tokenized words = word tokenize(ri)
        words = preprocessing(tokenized words)
        L+=words
ENDFOR
For each review ri in R:
F = list along with features and labels
features = list of zeros, size equal to length of lexicons set L
        For each word w in ri
                If w exist in L
                index = L.index(w)
                features[index] += 1
        If it belongs to positive review
                F.append([features,1])
        Else
                F.append([features,0])
ENDFOR
```

Shuffle the Feature set F

END

3.4 Approaches Used

The pre-processed movie reviews are classified using two methods:

1. Resource based classification using Hindi-SentiWordNet(H-SWN)
2. In-language classification through various classifiers like Deep Belief Network, Neural Network, SVM etc.

3.4.1 Resource Based Classification

A simple approach to predict the sentiment of a review is to use the prior polarity of terms present in it. In order to find the polarity, a lexical resource is required. In this method of classification, we use Hindi SentiWordNet(H-SWN) as the resource for developing majority based sentiment classifier.

Each word present in the H-SWN has a positive and a negative sentiment score. Based on the maximum of the scores, a polarity is assigned to each word in a review. The polarity which covers the maximum number of words in a review is predicted as the sentiment of that review.

Algorithm 2 Resource based classification using Hindi SentiWordNet

Input:

1) A list of reviews, R which contain positive reviews and negative reviews in string format. R = {p1, p2....pm1 , n1, n2....nm2 }, where m1 = number of positive reviews and m2 = number of negative reviews

2) Hindi SentiWordNet dictionary dict which contain a tuple for every word having its positive polarity score and negative polarity score.

Output:

A list P containing the polarity of reviews

BEGIN

Make a list of polarity of reviews P=[]

For each review ri in R

    Apply preprocessing on ri

    Make a list of votes v=[]

    Initialize two variables x1=0, x2=0

    For each word w in ri

        if w exists in dict

            pos score, neg score = dict[w]

            if pos score > neg score

```
                    v.append(1)
                    x1+=pos score
                else
                    v.append(0)
                    x2+=neg score
                else
                    ignore the word
        ENDFOR
        x = number of ones in the list
        y = number of zeros in list
        if x > y
            sense = 1 (here 1 denotes positive)
        else if y > x
            sense = 0 (here 0 denotes negative)
        else
            if x1 > x2
                sense = 1
            else
                sense = 0
        P.append(sense)
ENDFOR
END
```

3.4.2 In-language Classification

This approach is based on training the classifiers on the same language as the text. It relies heavily on availability of resources in the same language to analyse the sentiment. Thus all training text, testing text are in Hindi language. The feature representation (term frequency or TF-IDF) can be varied to see the effect on In language classification on Hindi reviews. In this approach, we use a variety of seven classifiers to train and test the data. We know TF-IDF can be a better way of feature matrix generation as it reduces effect of very frequent words in document but do not contribute much to the relevance of text.

3.5 Classification

This section contains different classification models that we have used to p redict the polarity of movie reviews in Hindi.

3.5.1 Logistic Regression

Logistic Regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. Logistic regression is a linear method, but the predictions are transformed using the logistic function. It is a statistical method for analysing a dataset in which there are one or more independent variables that determine an outcome.

3.5.2 Naive Bayes

Naive Bayes is a classification technique based on Bayes' Theorem with an assumption of independence among predictors [iii]. It assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes based classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem.

3.5.3 Support Vector Machine

Support Vector Machine (SVM) also known as Support Vector Networks [vi] is a supervised learning model which contains various learning algorithms to analyse classification and regression problems. It is also known as binary classifier which attempts to find a hyperplane that can separate two class of data by the largest margin. Given a set of points of two types in N- dimensional place, SVM generates a (N-1) dimensional hyperplane to separate those points into two groups. SVMs can be used to solve various real life problems like classification of images, recognizing hand written characters, categorizing text and hypertext, etc.

3.5.4 Stochastic Gradient Descent

To improve the performance of SVM, Stochastic Gradient Descent (SGD) technique is used. It is a simple yet very efficient approach to discriminative learning of linear classifiers such as Support Vector Machines and Logistic Regression under convex loss functions. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention recently in the context of large-scale learning. The advantages of Stochastic Gradient Descent are efficiency, ease of implementation and there are a lot of opportunities for code tuning. One drawback of Stochastic Gradient Descent include SGD is it is sensitive to feature scaling.

3.5.5 Decision Tree

Decision Trees (DTs) [i] are a non-parametric supervised learning method used for classification and regression. A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute and each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules. The goal is to create a

model that predicts the value of a target variable by learning simple classification rules inferred from the data features.

3.5.6 Voting Classifier

Voting classifier is a kind of Ensemble method. The goal of Ensemble methods is to combine the predictions of several base estimators with a given learning algorithm in order to improve the accuracy and robustness of the classifier [vii]. The idea behind the Voting Classifier is to combine conceptually different machine learning classifiers and use a majority vote or the average predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well performing model in order to balance out their individual weaknesses.

3.5.7 Deep Neural Network

Deep Neural Network (DNN) [ii] consists of systems of interconnected "neurons" capable of computing or processing values from inputs. They are designed to recognize patterns. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated. The network is formed by connecting the output of certain neurons to the input of other neurons. This forms a directed and weighted graph, where the neurons are the nodes and the connection between the neurons are weighted directed edges

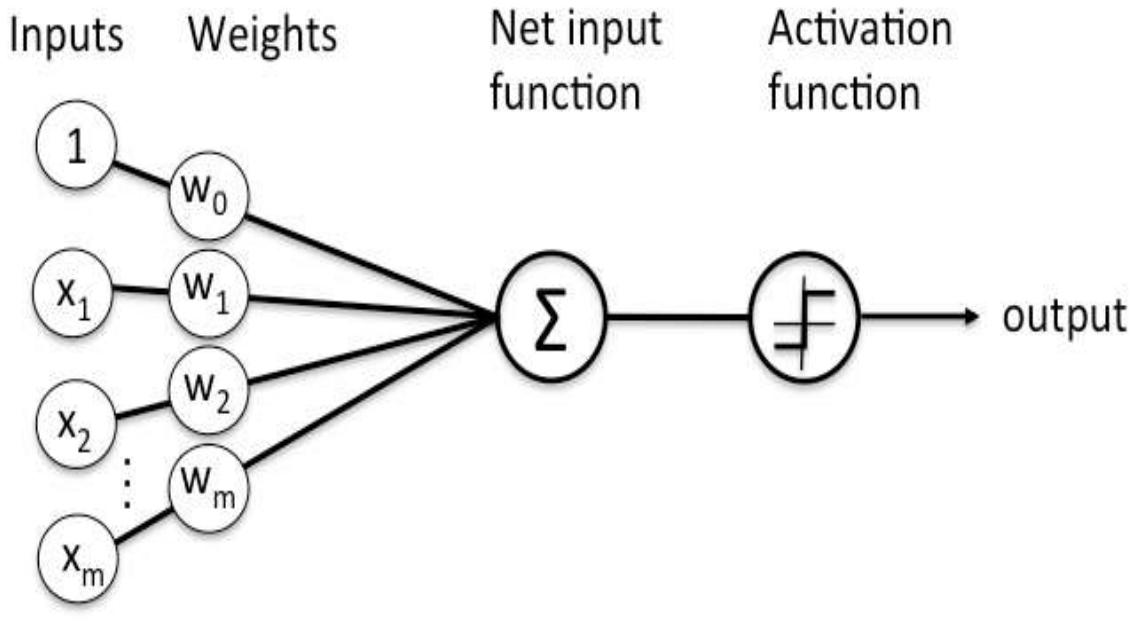


Fig1: How a single node looks like in neural network

4 EXPERIMENTAL SETUP

In our project, we have used the dataset of 1000 movie reviews in Hindi. They are manually labelled into two classes- positive and negative. Then we have generated a feature matrix using TF-IDF and unigram models. The obtained feature matrix is fed into different classifiers. We have used following classifiers in our project: Naive Bayes, Logistic Regression, Support Vector Machine, Decision Trees, Neural Network and Stochastic Gradient Descent.

Apart from predicting classes using feature matrix, we have also used Resource based Classification Technique using Hindi SentiWordNet (H-SWN).

All the results are reported for 10-fold cross validation. But in case of deep belief network and deep neural network, we have used 80-20 percent split. We just calculated accuracy by comparing actual class label of reviews against the predicted ones. We also compare accuracy obtained using various classifiers.

	Actual Value		
Predicted values		Positives	negatives
	positives	True Positive (t_p)	False Positive (f_p)
	negatives	False Negative (f_n)	True Negative (t_n)

Table 1: Confusion matrix

We consider the following evaluation measures in order to compute the overall performance of the system.

1. Precision**:** Precision is defined as portion of true positive predicted instances among all positive predicted instances.

$$\text{Precision} = \frac{tp}{tp + fp}$$

2. Recall: Recall is calculated as portion of true positive predicted instances against all actual positive instances.

$$\text{Recall} = \frac{tp}{tp + fn}$$

3. Accuracy: Accuracy basically is the portion of true predicted instances against all predicted instances.

$$\text{Accuracy} = \frac{(tp + tn)}{(tp + tn + fp + fn)}$$

4. F-measure: F-measure is the combination of Precision and Recall and is calculated as:

$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Recall} + \text{Precision}}$$

4.1 Project Code:

4.1.1 Resource Based Semantic analysis using Hindi-Sentiwordnet

```
import pandas as pd
import codecs
from nltk.tokenize import word_tokenize
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
import re
data = pd.read_csv("HindiSentiWordnet.txt", delimiter=' ')
fields = ['POS_TAG', 'ID', 'POS', 'NEG', 'LIST_OF_WORDS']
words_dict = {}
for i in data.index:
    words = data[fields[4]][i].split(',')
    for word in words:
        words_dict[word] = (data[fields[0]][i], data[fields[2]][i], data[fields[3]][i])
def sentiment(text):  //This function determines sentiment of text.
    words = word_tokenize(text)
    votes = []
    pos_polarity = 0
    neg_polarity = 0
    allowed_words = ['a','v','r','n']    //adverbs, nouns, adjective, verb are only used
    for word in words:
    //if word in dictionary, it picks up the positive and negative score of the word
        if word in words_dict:
            pos_tag, pos, neg = words_dict[word]
            #print(word, pos_tag, pos, neg)
            if pos_tag in allowed_words:
```

```
            if pos > neg:
                pos_polarity += pos
                votes.append(1)
            elif neg > pos:
                neg_polarity += neg
                votes.append(0)
    //calculating the no. of positive and negative words in total in a review to give class labels
    pos_votes = votes.count(1)
    neg_votes = votes.count(0)
    if pos_votes > neg_votes:
        return 1
    elif neg_votes > pos_votes:
        return 0
    else:
        if pos_polarity < neg_polarity:
            return 0
        else:
            return 1
pred_y = []
actual_y = []
//to calculate accuracy
pos_reviews = codecs.open("pos_hindi.txt", "r", encoding='utf-8', errors='ignore').read()
for line in pos_reviews.split('$'):
    data = line.strip('\n')
    if data:
        pred_y.append(sentiment(data))
        actual_y.append(1)
#print(accuracy_score(actual_y, pred_y) * 100)
#print(len(actual_y))
neg_reviews = codecs.open("neg_hindi.txt", "r", encoding='utf-8', errors='ignore').read()
for line in neg_reviews.split('$'):
    data=line.strip('\n')
```

```
    if data:
        pred_y.append(sentiment(data))
        actual_y.append(0)
print(len(actual_y))
print(accuracy_score(actual_y, pred_y) * 100)
print('F-measure:  ',f1_score(actual_y,pred_y))

if __name__ == '__main__':
    print(sentiment("संदर्भ और स्थितियां बदल गई हैं, लेकिन सोच और समझ में अधिक बदलाव नहीं आया"))
```

4.1.2 To fit different type of classifiers on the featureset generated through tf and tfidf

```
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from UnigramTfFeatureGeneration import create_feature_set_and_labels
from UnigramTfifdFeaturesetGeneration import get_features
def begin_test(train_x, test_x, train_y, test_y):
    x = train_x + test_x
    y = train_y + test_y
    clf1 = LinearRegression()
    clf2 = LogisticRegression()
    clf3 = SGDClassifier()
    clf4 = SVC()
    clf5 = KNeighborsClassifier()
```

```
    clf6 = MLPClassifier()
    clf7 = DecisionTreeClassifier()
    clf8 = MultinomialNB()
    eclf = VotingClassifier(estimators=[('logr', clf2), ('sgd', clf3), ('svm', clf4), ('kn', clf5), ('nn', clf6),
('dt', clf7)],voting='hard')
    for label, clf in zip(['LogisticRegressionClassifier', 'SGDClassifierClassifier',
'SVCClassifier', 'NearestNeighbourClassifier', 'NeuralNetworkClassifier',
'DecisionTreeClassifier','MultinomialNB', 'EnsembleClassifier'],[clf2, clf3, clf4, clf5, clf6, clf7,
clf8, eclf]):
        scores = cross_val_score(clf, x, y, cv=10, scoring='accuracy')
        f_measure = cross_val_score(clf, x, y, cv=10, scoring='f1')
        print(label, "Accuracy: ", scores.mean(), "+/- ", scores.std())
        print(label, "F-measure: ", f_measure.mean())
        scores = cross_val_score(clf, x, y, cv=10, scoring='accuracy')
        f_measure = cross_val_score(clf, x, y, cv=10, scoring='f1')
        print("Accuracy: ", scores.mean(), "+/- ", scores.std())
        print("F-measure: ", f_measure.mean())
def test_by_tf():
    train_x, train_y, test_x, test_y = create_feature_set_and_labels \
        ('pos_hindi.txt', 'neg_hindi.txt')
    begin_test(train_x, test_x, train_y, test_y)
def test_by_tfidf():
    train_x, train_y, test_x, test_y = get_features()
    begin_test(train_x, test_x, train_y, test_y)
if __name__ == '__main__':
    print("="*10)
    print("Unigram+Tf Accuracies")
    test_by_tf()
    print("=" * 10)
    print("Unigram+Tfidf Accuracies")
    test_by_tfidf()
```

4.1.3 Unigram-tf Feature Generation

```
import random
from collections import Counter
import numpy as np
from googletrans import Translator
from nltk.tokenize import word_tokenize
import codecs
from dbn_outside.dbn.tensorflow import SupervisedDBNClassification
hm_lines = 5000000
translator = Translator()
stopwords = codecs.open("hindi_stopwords.txt", "r", encoding='utf-8',
errors='ignore').read().split('\n')
//Creating a set of lexicons which is a kind of dictionary of words.
def create_lexicon(pos, neg):
    lexicon = []
    for file_name in [pos, neg]:
        with codecs.open(file_name, 'r',encoding='utf-8',errors='ignore') as f:
            contents = f.read()
            for line in contents.split('$'):
                data = line.strip('\n')
                if data:
                    all_words = word_tokenize(data)
                    lexicon += list(all_words)
    lexicons = []
    for word in lexicon:
        if not word in stopwords:
            lexicons.append(word)
    word_counts = Counter(lexicons)  // it will return kind of dictionary
    l2 = []
    for word in word_counts:
        if 60 > word_counts[word]:
            l2.append(word)
    return l2
```

```
def sample_handling(sample, lexicon, classification):
    featureset = []
    with codecs.open(sample, 'r', encoding="utf8",errors='ignore') as f:
        contents = f.read()
        for line in contents.split('$'):
            data = line.strip('\n')
            if data:
                all_words = word_tokenize(data)
                all_words_new = []
                for word in all_words:
                    if not word in stopwords:
                        all_words_new.append(word)
                features = np.zeros(len(lexicon))
                for word in all_words_new:
                    if word in lexicon:
                        idx = lexicon.index(word)
                        features[idx] = 1
                features = list(features)
                featureset.append([features, classification])
    return featureset
def create_feature_set_and_labels(pos, neg, test_size=0.2):
    lexicon = create_lexicon(pos, neg)
    features = []
    features += sample_handling(pos, lexicon, 1)
    features += sample_handling(neg, lexicon, 0)
    random.shuffle(features)
    features = np.array(features)
    training_size = int((1 - test_size) * len(features))
    x_train = list(features[:, 0][:training_size])  // taking features array upto training_size
    y_train = list(features[:, 1][:training_size])  // taking labels upto training_size
    x_test = list(features[:, 0][training_size:])
    y_test = list(features[:, 1][training_size:])
```

```
    return x_train, y_train, x_test, y_test
```

4.1.4 Unigram TF-IDF Feature Generation

```
import codecs
from operator import neg
import random
from turtle import onclick
import numpy as np
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
stopwords = codecs.open("hindi_stopwords.txt", "r", encoding='utf-8',
errors='ignore').read().split('\n')
def get_features(type='dbn'):
    pos_reviews = codecs.open("pos_hindi.txt", "r", encoding='utf-8', errors='ignore').read()
    neg_reviews = codecs.open("neg_hindi.txt", "r", encoding='utf-8', errors='ignore').read()
    documents = []
    // here j for adjectives, r is adverb , v is verb
    allowed_word_types = ["J"]
    pos_count = 0
    for i in pos_reviews.split('$'):
        data = i.strip('\n')
        if data:
            documents.append(data)
            pos_count += 1
    neg_count = 0
    for i in neg_reviews.split('$'):
        data = i.strip('\n')
        if data:
            documents.append(data)
            neg_count += 1
    vectorizer = TfidfVectorizer(lowercase=False, analyzer=word_tokenize)
    tfidf = vectorizer.fit_transform(documents)
```

```
    featureset = tfidf.toarray()
        #print(vectorizer.get_feature_names())
    positive_feature_set = featureset[0:pos_count]
    negative_set = featureset[pos_count:]
    finalset = []
    for i in positive_feature_set:
        if type == 'simple':
            finalset.append([i, [1, 0]])
        else:
            finalset.append([i, 1])
    for i in negative_set:
        if type == 'simple':
            finalset.append([i, [0, 1]])
        else:
            finalset.append([i, 0])
    print(finalset)
    finalset = np.array(finalset)
    random.shuffle(finalset)
    test_size = 0.2
    testing_size = int((1 - test_size) * len(finalset))
    x_train = list(finalset[:, 0][:testing_size])  // taking features array upto testing_size
    y_train = list(finalset[:, 1][:testing_size])
    x_test = list(finalset[:, 0][testing_size:])
    y_test = list(finalset[:, 1][testing_size:])
    return x_train, y_train, x_test, y_test
get_features()
```

5 RESULT ANALYSIS

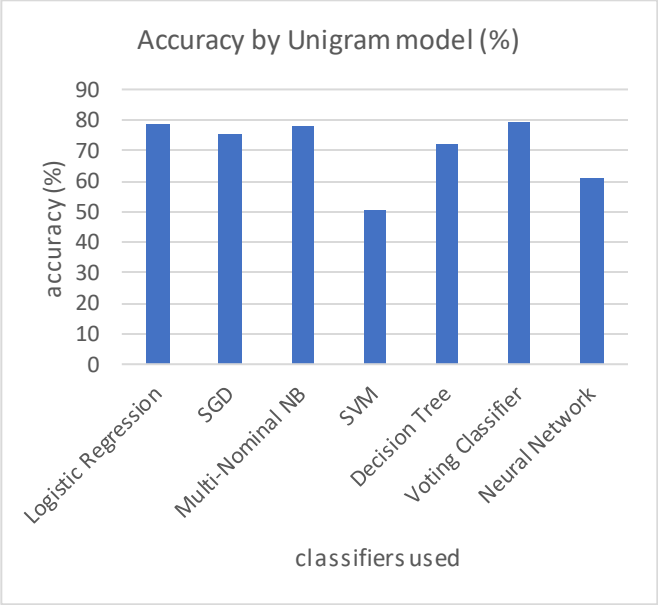
In this section we discuss the results obtained from the various runs of the experiments conducted.

The resource based classifier produces only 53.51% of accuracy which is not desirable.

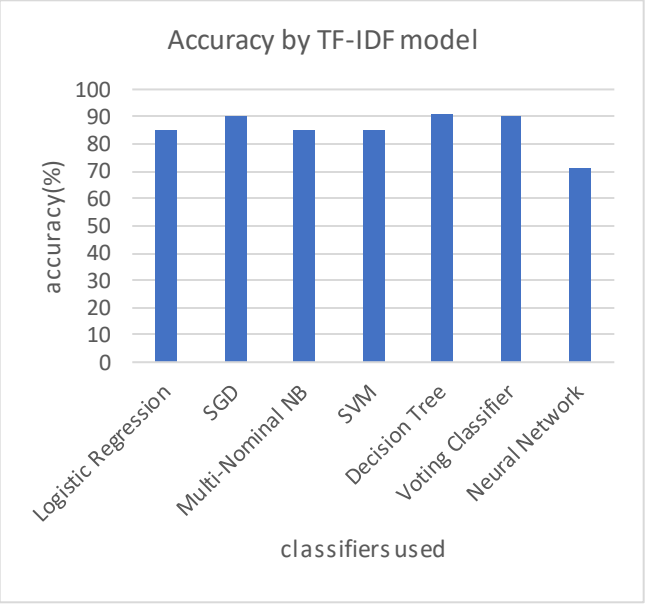
Below results are for the In-language classification.

Classifiers	Unigram(%)	TF-IDF(%)
Logistic Regression	78.98	85.24
Stochastic Gradient Descent	75.46	90.05
Multi-Nominal Naïve Bayes	77.8	85.14
Support Vector Machine	50.4	85.24
Decision tree	72.08	**90.85**
Voting classifier	**79.09**	89.94
Neural network	61.05	70.99

Table 2: Accuracy obtained using Unigram model and TF-IDF model by different classifiers.



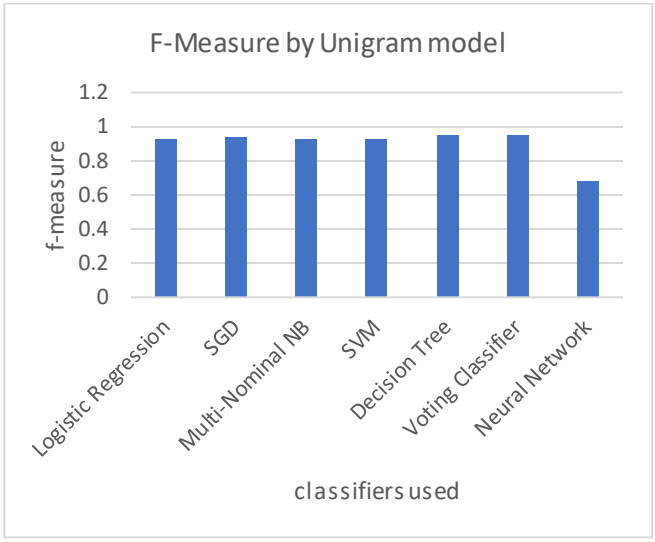
Graph 1: Accuracy obtained using unigram model by various classifiers.



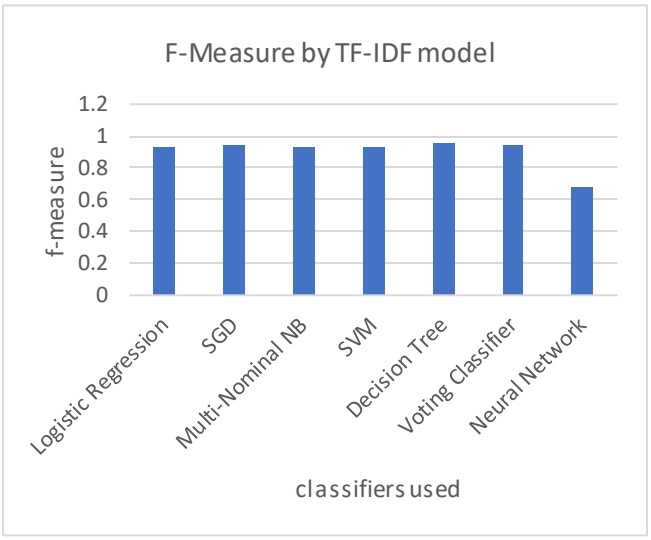
Graph 2: Accuracy obtained using TF-IDF model by various classifiers.

Classifiers	Unigram	TF-IDF
Logistic Regression	**0.7853**	0.9303
Stochastic Gradient Descent	0.7661	0.9433
Multi-Nominal Naive Bayes	0.7848	0.9298
Support vector machine	0.6702	0.9278
Decision tree	0.72	**0.9533**
Voting classifier	0.7628	0.9491
Neural network	0.58	0.68

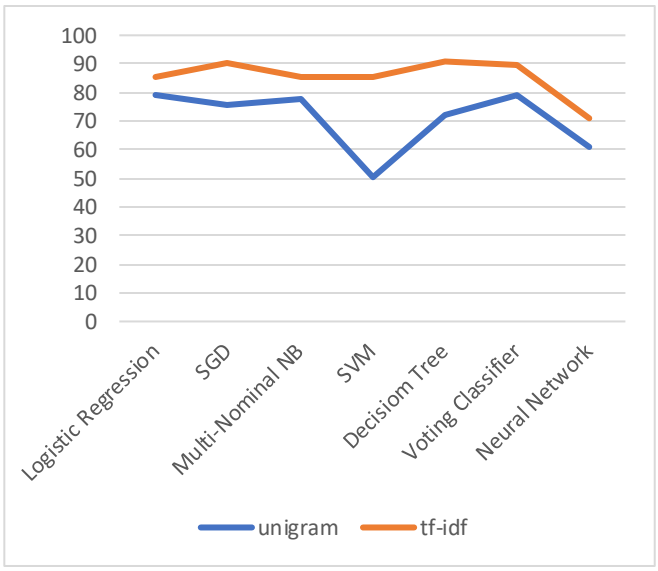
Table 3: f-measure obtained using Unigram model and TF-IDF model by different classifiers



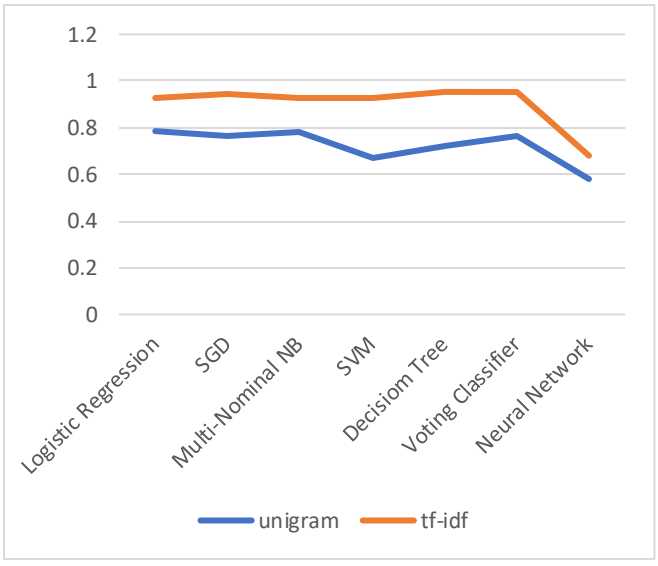
Graph 4: f-measure obtained using unigram model by various classifiers.



Graph 5: f-measure obtained using TF-IDF model by various classifiers.



Graph 3: Comparison between accuracies obtained by two models.



Graph 6: Comparison between f-measure obtained by two models.

6 CONCLUSION AND FUTURE WORK

In this project, we proposed two approaches . The first approach involved using a majority-based classifier for Hindi SentiWordNet. In the second approach a classifier model is constructed for Hindi using a training corpus in Hindi. The result proved that the second approach is better among the others. This means that best result can be achieved with an annotated corpus in the same language of analysis. Also, among the 2 models, unigram & TF-IDF used in the second approach, TF-IDF proves to generate a better result than the other.

Table 2 presents the values of classification accuracy for the experiment. It shows that the accuracy for TF-IDF representation is high as compared to the other representation in each set of the experiment. It also shows that decision tree gives best accuracy of 90.85% in case of TF-IDF representation, and in case of unigram representation we have achieved 79% accuracy through voting classifier. Table 2 presents the values of F-measure achieved by different classifiers. F-measure is a measure of how complete and accurate results are. We can see from the table that decision tree gives best F-measure of 0.9533 in case of TF-IDF representation, and in case of unigram representation we have achieved 0.7853 F-measure through logistic regression.

In future, the work on resource-based sentiment analysis can be extended to include Word Sense Disambiguation(WSD) so that a specific sense of word can be looked up in the H-SWN. Since Hindi SentiWordNet covers only limited number of words at present, the work can be extended to cover larger number of words by improving H-SWN. This will help in achieving better accuracy. Further this approach can also be extended to handle negation rules which is not supported by present models.

WEB RESOURCES

[i] "Decision Tree" https://en.wikipedia.org/wiki/Decision_tree (accessed Feb. 1,2022).

[ii] "Deep Neural Network" https://deeplearning4j.konduit.ai/multi-project/tutorials/quickstart (accessed Feb. 1,2022).

[iii] "Naive Bayes" https://en.wikipedia.org/wiki/Naive_Bayes_classifier (accessed Feb. 1,2022).

[iv] "RBM Wikipedia" https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine (accessed Feb. 1,2022).

[v] "Sentiment Analysis Wikipedia" https://en.wikipedia.org/wiki/Sentimentanalysis (accessed Jan. 29,2022).

[vi] "Support Vector Machine" https://en.wikipedia.org/wiki/Support-vector_machine (accessed Jan. 31,2022).

[vii] "Voting Classifier" https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html (accessed Feb. 1,2022).

REFERENCES

[1] MEDHAT, WALAA. AHMED HASSAN, and HODA KORASHY. "Sentiment analysis algorithms and applications: A survey." *Ain Shams engineering journal 5.4* (2014), pp. 1093-1113.

[2] ARORA P. "Sentiment analysis for hindi language." *MS by Research in Computer Science* (2013).

[3] BAKLIWAL A, ARORA P, and VARMA V. "Hindi subjective lexicon: A lexical resource for hindi polarity classification." In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC)* (2012), pp.1189–1196.

[4] BANSAL N, AHMED U.Z, and MUKHERJEE A. "Sentiment analysis in hindi." *Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India* (2013), pp. 1-10.

[5] JOSHI A, BALAMURALI A, and BHATTACHARYYA P. . "A fall-back strategy for sentiment analysis in hindi: a case study." *Proceedings of the 8th ICON* (2010).

[6] MITTAL N, AGARWAL B, VHOUHAN G, PAREEK P and BANIA N. "Discourse based sentiment analysis for hindi reviews." In *International Conference on Pattern Recognition and Machine Intelligence* (2013), Springer, pp.720-725.

[7] SHARMA R, NIGAM S and JAIN R. "Polarity detection movie reviews in hindi language." *arXiv Preprint arXiv:1409.3942* (2014).

[8] PEDREGOSA F, VAROQUAUX G, GRAMFORT A, MICHEL V, THIRION B, GRISEL O, BLONDEL M, PRETTENNHOFER P, WEISS R, DUBOURG V, VANDERPLAS J, PASSOS A, COURNAPEAU D, BRUCHER M, PERROT M and DUCHESNAY E. "Scikit-learn: Machine learning in Python." *Journal of Machine Learning Research 12* (2011), pp. 2825-2830.