Project Report on

Sentiment Analysis for Hinglish Text (Code mix English + Hindi)

for course Natural Language Processing

by

Name	Class	Roll No
KAIF KHAN	**BE3**	**20**
PRATIK NAIK	**BE3**	**34**
NIRAJ PARMAR	**BE3**	**36**

Under the guidance of

Prof. Atul Kachare

DEPARTMENT OF COMPUTER ENGINEERING

SHAH AND ANCHOR KUTCHHI ENGINEERING COLLEGE

CHEMBUR, MUMBAI – 400088.

2021 – 2022

Table of Contents

Abstract

Over recent years, social media has given a huge impact on online sales of products. Currently, companies are acquiring the use of sentiment analysis for better recognition and reviews of the product based on text recognition using sentiment analysis. It is currently focused on English language-based sentiment analysis. We have a goal to open the barriers to English-Hindi (Hinglish) sentiment analysis which comprises both Hindi language and Hindi script in English by means of sentiment analysis algorithms. We first transliterate and translate English to Hindi and then calculate the polarity and the subjectivity.

List of Figures

Chapter 1
Introduction

The social media market has grown exponentially in recent years, data says 63.29 percent of the folk worldwide already have a mobile phone and the number of mobile users in the world is expected to pass the seven billion mark by 2023. As the usage of mobile phones increased the need for social media and people's choice to buy things through it has also increased. So, this is where our project comes into play; we help the customers to get a better view/analysis of the product based on the reviews it gets and studying the sentiments behind it. We are targeting to touch the audience categorized as native language speakers who place their reviews in their native language which narrows the gap in the system.

1.1 Problem statement:

To build a sentiment analysis machine using machine learning based on frequency of words and classification of sentences based on positive, negative or neutral sentences for clear understanding of the message.

1.2 Objective & Scope:

- To achieve the concept of transliteration with sentiment analysis.
- Sentence level sentiment analysis
- To know a user or audience opinion on a target object.
- Analyze text on different levels of detail

1.3 Report Structure

- In Chapter 2, Planning and formulation of the project is given and data sources & its preprocessing.
- In Chapter 3, The system proposed is introduced which will tell the deep specification of the project and will tell how the different modules of the system will work, the flow of the project regarding data flow, control flow and other flow of the system.
- In Chapter 4, we see the implementation of the algorithm of the project and process of model building.
- In Chapter 5, Conclusion and future scope of this project are mentioned.

1.4 System Requirement

1.4.1 Hardware Requirements

- 64-bit Operating System.
- Intel i5 8th Gen and above
- RAM 4 GB

1.4.2 Software Requirements

- Programming Language : Python
- Pandas , NLTK , stanza, googletrans libraries.

1.5 Data Source & Preprocessing

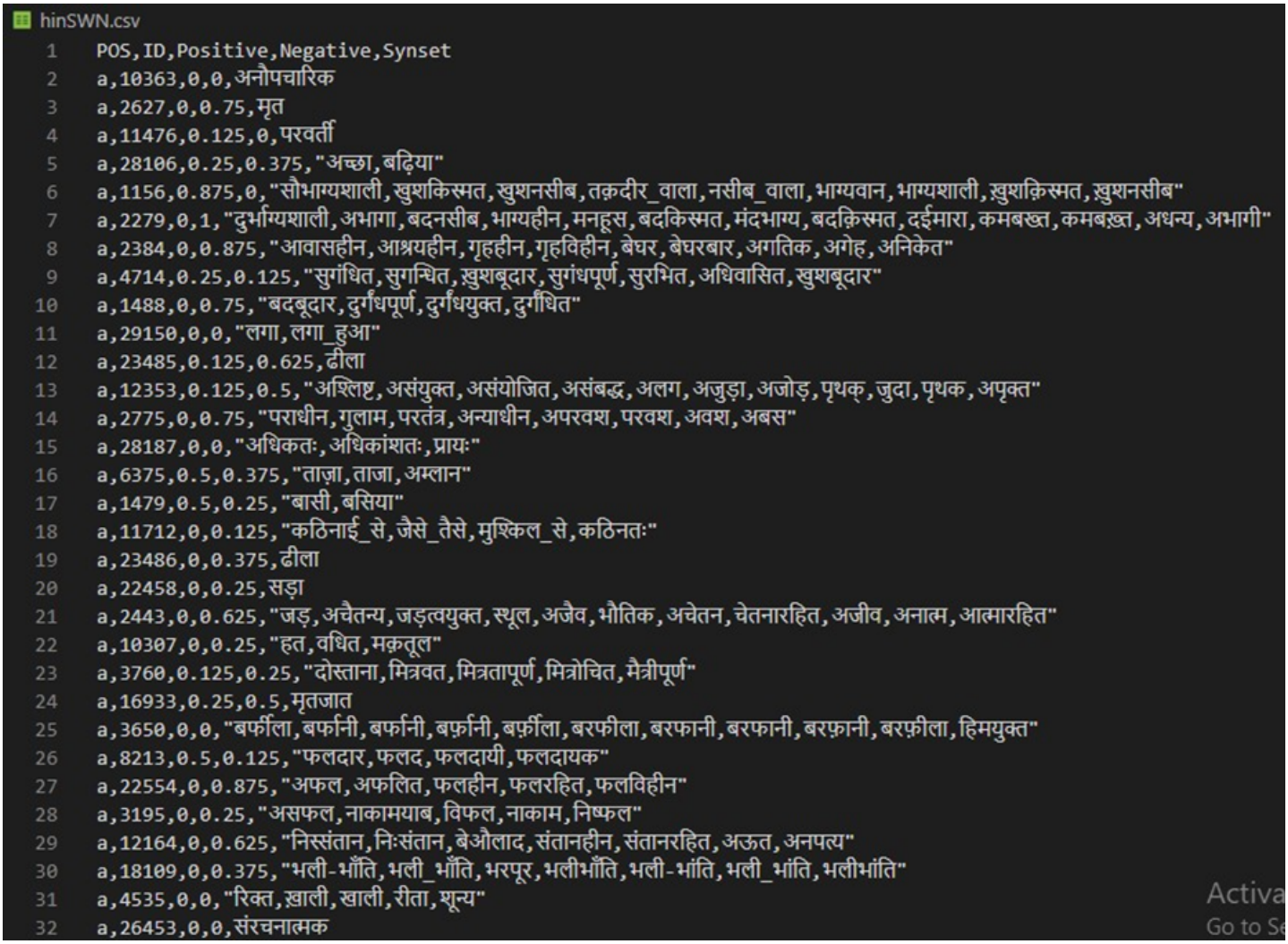
Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset.

When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset. But it is crucial to establish a template for your data cleaning process so you know you are doing it the right way every time.

1.5.1 Data Pre-processing

Data pre-processing simply means cleaning the data. Since every real world data is redundant. Hence Data pre-processing is very important for any machine learning model to give better results. In our project data pre-processing includes removal of unnecessary words i.e. stop words and stemming. It also includes converting all the letters into lowercase letters.

The data pre-processing is done by using nltk library. Also for hindi there is not any library that's why we made a list of hindi words which come frequently and use these words as stopwords.



hinSWN.csv

```
POS,ID,Positive,Negative,Synset
a,10363,0,0,अनौपचारिक
a,2627,0,0.75,मृत
a,11476,0.125,0,परवर्ती
a,28106,0.25,0.375,"अच्छा,बढ़िया"
a,1156,0.875,0,"सौभाग्यशाली,खुशकिस्मत,खुशनसीब,तक़दीर_वाला,नसीब_वाला,भाग्यवान,भाग्यशाली,ख़ुशक़िस्मत,ख़ुशनसीब"
a,2279,0,1,"दुर्भाग्यशाली,अभागा,बदनसीब,भाग्यहीन,मनहूस,बदकिस्मत,मंदभाग्य,बदक़िस्मत,दईमारा,कमबख्त,कमबख़्त,अधन्य,अभागी"
a,2384,0,0.875,"आवासहीन,आश्रयहीन,गृहहीन,गृहविहीन,बेघर,बेघरबार,अगतिक,अगेह,अनिकेत"
a,4714,0.25,0.125,"सुगंधित,सुगन्धित,ख़ुशबूदार,सुगंधपूर्ण,सुरभित,अधिवासित,खुशबूदार"
a,1488,0,0.75,"बदबूदार,दुर्गंधपूर्ण,दुर्गंधयुक्त,दुर्गंधित"
a,29150,0,0,"लगा,लगा_हुआ"
a,23485,0.125,0.625,ढीला
a,12353,0.125,0.5,"अश्लिष्ट,असंयुक्त,असंयोजित,असंबद्ध,अलग,अजुड़ा,अजोड़,पृथक्,जुदा,पृथक,अपृक्त"
a,2775,0,0.75,"पराधीन,गुलाम,परतंत्र,अन्याधीन,अपरवश,परवश,अवश,अबस"
a,28187,0,0,"अधिकतः,अधिकांशतः,प्रायः"
a,6375,0.5,0.375,"ताज़ा,ताजा,अम्लान"
a,1479,0.5,0.25,"बासी,बसिया"
a,11712,0,0.125,"कठिनाई_से,जैसे_तैसे,मुश्किल_से,कठिनतः"
a,23486,0,0.375,ढीला
a,22458,0,0.25,सड़ा
a,2443,0,0.625,"जड़,अचैतन्य,जड़त्वयुक्त,स्थूल,अजैव,भौतिक,अचेतन,चेतनारहित,अजीव,अनात्म,आत्मारहित"
a,10307,0,0.25,"हत,वधित,मक़तूल"
a,3760,0.125,0.25,"दोस्ताना,मित्रवत,मित्रतापूर्ण,मित्रोचित,मैत्रीपूर्ण"
a,16933,0.25,0.5,मृतजात
a,3650,0,0,"बर्फीला,बर्फानी,बर्फानी,बर्फ़ानी,बर्फ़ीला,बरफीला,बरफानी,बरफानी,बरफ़ानी,बरफ़ीला,हिमयुक्त"
a,8213,0.5,0.125,"फलदार,फलद,फलदायी,फलदायक"
a,22554,0,0.875,"अफल,अफलित,फलहीन,फलरहित,फलविहीन"
a,3195,0,0.25,"असफल,नाकामयाब,विफल,नाकाम,निष्फल"
a,12164,0,0.625,"निस्संतान,निःसंतान,बेऔलाद,संतानहीन,संतानरहित,अऊत,अनपत्य"
a,18109,0,0.375,"भली-भाँति,भली_भाँति,भरपूर,भलीभाँति,भली-भांति,भली_भांति,भलीभांति"
a,4535,0,0,"रिक्त,ख़ाली,खाली,रीता,शून्य"
a,26453,0,0,संरचनात्मक
```


Figure 1: HindiSWN

1.6 Stop words:

These are the most commonly occurring words in any language.It should be removed because these are unnecessary and frequent words that carry data that is not relevant. Hence removal of these stop words has proven to be very important.

1.7 Stemming:

Stemming is the process of reducing words to their root word by reducing affixes of a word to get its stem or root even if the root has no dictionary meaning. The hypothesis behind stemming is that words with the same root mostly describe relatively close or similar meanings.

For example- beautiful, beautify, beauty will be converted to beautiful even if it has no meaning in the dictionary. It is done by using the Porter Stemmer algorithm.

Chapter 2

System Design

2.1 Stages of Sentiment Analysis model:

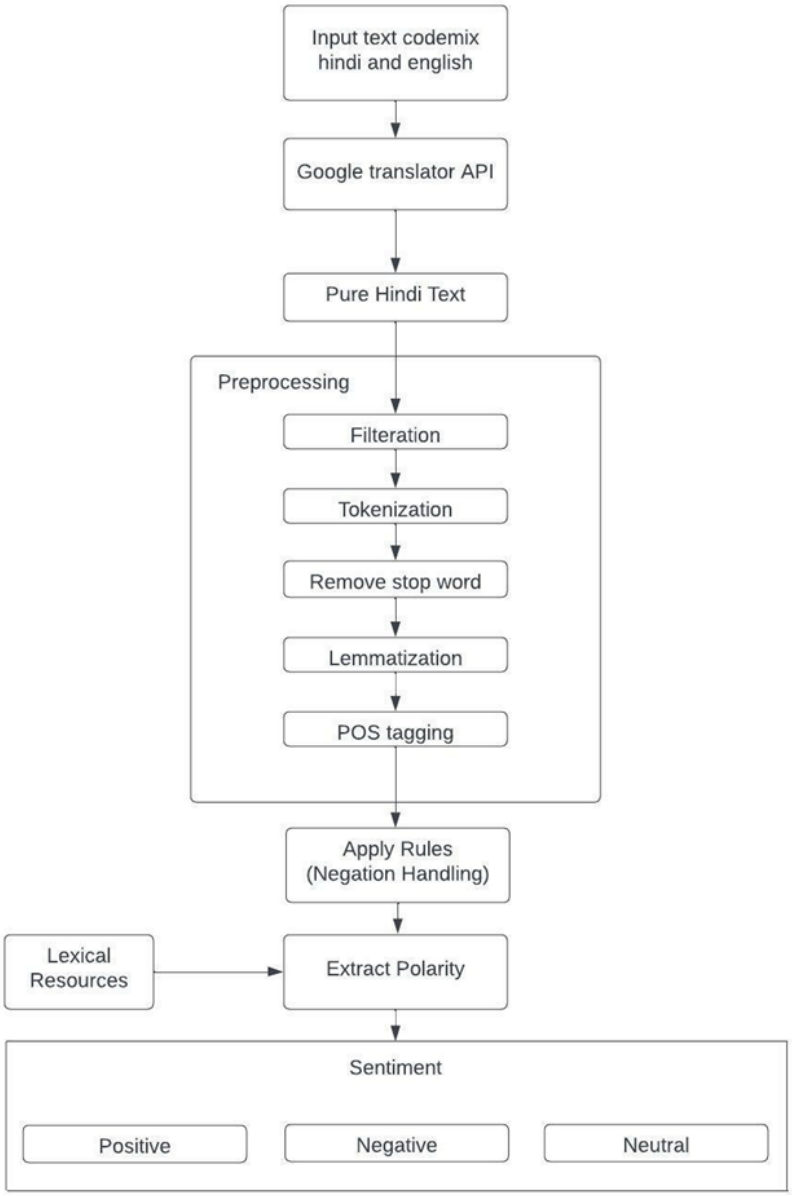


Figure 2: Stages of Sentiment Analysis

2.2 Framework

Many algorithms in machine learning require a numerical representation of objects since such representations facilitate processing and statistical analysis.

2.2.1 NLTK

The Natural Language Toolkit (NLTK) is a Python programming environment for working with human language data in statistical natural language processing (NLP). It includes tokenization, parsing, categorization, stemming, tagging, and semantic reasoning text processing packages. It also comes with a recipe book and a book that describes the ideas behind the core language processing tasks that NLTK provides, as well as graphical examples and sample data sets.

2.2.2 Stanza

Stanza is a Python-based NLP library which contains tools that can be used in a neural pipeline to convert a string containing human language text into lists of sentences and words.

2.3 Rule Based Approach

Rule-based approaches are the oldest approaches to NLP. Why are they still used, you might ask? It's because they are tried and true, and have been proven to work well. Rules applied to text can offer a lot of insight: think of what you can learn about arbitrary text by finding what words are nouns, or what verbs end in -ing, or whether a pattern recognizable as Python code can be identified. Regular expressions and context free grammars are textbook examples of rule-based approaches to NLP.

Chapter 3

Implementation

3.1 Program Code

PREPROCESSING:

```
  #Function to tokenize
  def tokenize(sentence):
     return word_tokenize(sentence)

  def filter(tokenString):
     #Remove numbers
     tokenString = re.sub(r"\d+", '', tokenString)
     #Remove URLS
     tokenString = re.sub(r'(https?|ftp|www)\S+', '', tokenString)
     #Remove punctuations
     exclist = string.punctuation
     table_ = tokenString.maketrans('', '', exclist)
     tokenString = tokenString.translate(table_)
     #Remove extra spaces
     tokenString = re.sub(' +',' ',tokenString).strip()
     #Remove hashtags
     tokenString = ''.join([x + ' ' for x in tokenString.split(" ") if not x.startswith("#")]).strip()
     #Remove emojis
     pattenr =
  re.compile(u'([\U00002600-\U000027BF])|([\U0001f300-\U0001f64F])|([\U0001f680-\U0001f6FF])')
     tokenString = pattenr.sub(r'',tokenString)
     return tokenString

  def script_validation(tokens):
     for word in tokens:
        word = word.strip()
        for ch in word:
           c = ch
           if len(c) == 1:
              if ord(c) not in range(2304,2432):
                 return(0)
     return 1

  def separate_into_dict(tokens):
     return {i:token for i,token in enumerate(tokens)}
```

```
#Function to remove stopwords
def hindiStopwordsRemover(hinDict):
    stopwords = ['मैं', 'मुझको', 'मेरा', 'अपने आप को', 'हमने', 'हमारा', 'अपना', 'हम', 'आप', 'आपका', 'तुम्हारा', 'अपने आप', 'स्वयं',
            'वह', 'इसे, उसके', 'खुद को', 'कि वह', 'उसकी', 'उसका', 'खुद ही', 'यह', 'इसके', 'उन्होने', 'अपने', 'क्या', 'जो', 'किसे',
            'किसको', 'कि', 'ये', 'हूँ', 'होता है', 'रहे', 'थी', 'थे', 'होना', 'गया', 'किया जा रहा है', 'किया है', 'है', 'पडा', 'होने', 'करना',
            'करता है', 'किया', 'रही', 'एक', 'लेकिन', 'अगर', 'या', 'क्यूंकि', 'जैसा', 'जब तक', 'जबकि', 'की', 'पर', 'द्वारा', 'के लिए', 'साथ',
            'के बारे में', 'खिलाफ', 'बीच', 'में', 'के माध्यम से', 'दौरान', 'से पहले', 'के बाद', 'ऊपर', 'नीचे', 'को', 'से', 'तक', 'से नीचे', 'करने में', 'निकल', 'बंद', 'से अधिक',
            'तहत', 'दुबारा', 'आगे', 'फिर', 'एक बार', 'यहाँ', 'वहाँ', 'कब', 'कहाँ', 'क्यों', 'कैसे', 'सारे', 'किसी', 'दोनो', 'प्रत्येक', 'ज्यादा', 'अधिकांश', 'अन्य', 'मैं कुछ', 'ऐसा',
            'मैं कोई', 'मात्र', 'खुद', 'समान', 'इसलिए', 'बहुत', 'सकता', 'जायेंगे', 'जरा', 'चाहिए', 'अभी', 'और', 'कर दिया', 'रखें', 'का', 'हैं', 'इस', 'होता', 'करने', 'ने', 'बनी', 'तो',
            'ही', 'हो','इसका', 'था', 'हुआ', 'वाले', 'बाद', 'लिए', 'सकते', 'इसमें', 'दो', 'वे', 'करते', 'कहा', 'वर्ग', 'कई', 'करें', 'होती', 'अपनी', 'उनके', 'यदि', 'हुई', 'जा', 'कहते',
            'जब', 'होते', 'कोई', 'हुए', 'व', 'जैसे', 'सभी', 'करता', 'उनकी', 'तरह', 'उस', 'आदि', 'इसकी', 'उनका', 'इसी', 'पे', 'तथा', 'भी', 'परंतु', 'इन', 'कम', 'दूर', 'पूरे', 'गये',
            'तुम', 'मै', 'यहां', 'हुये', 'कभी', 'अथवा', 'गयी', 'प्रति', 'जाता', 'इन्हें', 'गई', 'अब', 'जिसमें', 'लिया', 'बड़ा', 'जाती', 'तब', 'उसे', 'जाते', 'लेकर', 'बड़े', 'दूसरे', 'जाने',
            'बाहर', 'स्थान', 'उन्हें', 'गए', 'ऐसे', 'जिससे', 'समय', 'दोनों', 'किए', 'रहती', 'इनके', 'इनका', 'इनकी', 'सकती', 'आज', 'कल', 'जिन्हें', 'जिन्हों', 'तिन्हें', 'तिन्हों', 'किन्हों',
            'किन्हें', 'इत्यादि', 'इन्हों', 'उन्हों', 'बिलकुल', 'निहायत', 'इन्हीं', 'उन्हीं', 'जितना', 'दूसरा', 'कितना', 'साबुत', 'वग़ैरह', 'कौनसा', 'लिये', 'दिया', 'जिसे', 'तिसे', 'काफ़ी', 'पहले',
            'बाला', 'मानो', 'अंदर', 'भीतर', 'पूरा', 'सारा', 'उनको', 'वहीं', 'जहाँ', 'जीधर', 'के', 'एवं', 'कुछ', 'कुल', 'रहा', 'जिस', 'जिन', 'तिस', 'तिन', 'कौन', 'किस', 'संग', 'यही',
            'बही', 'उसी', 'मगर', 'कर', 'मे', 'एस', 'उन', 'सो', 'अत' ]
    newHinDict = {}
    for index,hiToken in hinDict.items():
        if hiToken not in stopwords:
            newHinDict.update({index:hiToken})
    return newHinDict

#Function to lemmatize
def lemmatize_hi(filtered_hin):
    newDict = {}
    keys=list(filtered_hin.keys())
    values=filtered_hin.values()
    string=' '.join(values)
    dochi = nlp_hi(string)
    lemmatized_list = []
    for sent in dochi.sentences:
        for word in sent.words:
            lemmatized_list.append(word.lemma)
    if len(lemmatized_list):
        for i in range(len(values)):
            newDict.update({keys[i]:lemmatized_list[i]})
    return(newDict)
```

```
#Function to tag Part of Speech
def postagger_hi(lemmahin):
    newDict = {}
    keys=list(lemmahin.keys())
    values=lemmahin.values()
    string=' '.join(values)
    dochi = nlp_hi(string)
    pos_list=[]
    for sent in dochi.sentences:
        for word in sent.words:
            pos_list.append(word.text+'/'+word.upos)
    for i in range(len(values)) :
        newDict.update({keys[i]:pos_list[i]})
    return(newDict)

#Function to handle negation
def negation_handling_hin(pos_dict):
    newDict = {}
    exclamation = False
    skip = False
    for index,word in pos_dict.items():
        actualWord = word.split("/")[0]
        pos = word.split("/")[1]
        #Check if the word is नहीं
        if actualWord == "नहीं":
            #Do backward negation
            skip = True
            #Get the list of words backwards
            wordsChange = reversed([(x,y) for x,y in newDict.items()])
            for i,w in wordsChange:
                #Get POS Tag
                p = w.split("/")[1]
                #Exclamation is Alive
                newWord = "!" + w
                #Update in the dictionary
                newDict.update({i : newWord})
                if p == 'ADJ' or p == 'NOUN' or p == 'VERB':
                    break
        elif actualWord == "न":
            #Do forward negation
            exclamation = not exclamation
            skip = True
            #Add the exclamation
            word = '!' + word
            #Check if the word is adjective, noun or a verb
            if pos == 'ADJ' or pos == 'NOUN' or pos == 'VERB':
                #Set exclamation to False
                exclamation = False
        if skip == False:
            newDict.update({index : word})
        else:
            skip = False
```

```
    return newDict

#For Calculating Sentiment
import pandas as pd
import os
hi_SWN =
pd.read_csv("https://raw.githubusercontent.com/harshadrane67/NLP-MIni-Project/main/hinSWN.csv")
length = hi_SWN[hi_SWN.columns[0]].count()

tagsDict = {
    "ADJ" : "a",
    "NOUN" : "n",
    "ADV" : "r",
    "VERB" : "v"
}
Polarity :
ef get_pos_tag(cols):
    return cols[0]
def get_words(cols):
    words = cols[4].split(",")
    return words
def get_positive(cols):
    return cols[2]
def get_negative(cols):
    return cols[3]
def get_objective(cols):
    return 1 - (float(cols[2]) + float(cols[3]))
def get_scores(sentiword):
    res = 0
    wordList = {}
    count = 0
    score = 0.0
    for i in range(length):
        cols = hi_SWN.iloc[i]
        words = get_words(cols)
        pos = get_pos_tag(cols)
        for word in sentiword:
                negate = False
                if word[0] == '!':
                    negate = True
                    actualWord = word[1:]
                else:
                    actualWord = word
                actualWordW,tagWord = actualWord.split("/")[0], actualWord.split("/")[1]
                if tagWord == "ADJ" or tagWord == "ADV" or tagWord == "VERB" or tagWord == "NOUN":
                    tag = tagsDict[tagWord]
                    if actualWordW in words and pos == tag:
                        if not negate:
                            res += float(get_positive(cols)) - float(get_negative(cols))
                        else:
                            res += (float(get_positive(cols)) - float(get_negative(cols))) * -1
                        count = count + 1
```

```
                if actualWordW in wordList.keys():
                    wordList.update({actualWordW : wordList[actualWordW] + float(get_positive(cols)) -
float(get_negative(cols))})
                else:
                    wordList.update({actualWordW : float(get_positive(cols)) - float(get_negative(cols))})
    if len(wordList.keys()) > 0:
        score = res / len(wordList.keys())
    if score > 0:
        return ("Positive",score)
    elif score < 0:
        return ("Negative",score)
    return ("Neutral",score)

def sentiment(hinDict):
    return get_scores(list(hinDict.values()))
from googletrans import Translator
translator = Translator(service_urls=['translate.googleapis.com'])
#input_text = input("Enter Input Text: ")


comments = ["That restraurant is not good. Itna ghatiya khaana to kabhi nahi khaaya","ye khaana kitna taja
hai","mera nam omkar hai"]

#Run task
for comment in comments:
    print("Raw Input:",comment)
    translation = translator.translate(comment, src='en', dest='hi')
    #Filter
    #print("Original:",comment)
    fitered_comment = filter(translation.text)
    print("After Translation:",fitered_comment)
    print("\n")
    #Tokenize
    tokens = tokenize(fitered_comment)
    print("After Tokenization:",tokens)
    print("\n")
    #Removing empty chararcters
    tokensFinal = [tokens[i] for i in range(len(tokens)) if len(tokens[i]) > 1 and tokens[i] != '']
    if script_validation(tokensFinal):
        #Converting tokens to dictionary
        tokDict = separate_into_dict(tokensFinal)
        #print(tokDict)
        #Removing stopwords
        stopRemoved = hindiStopwordsRemover(tokDict)
        print("Removed Stopwords:",stopRemoved)
        print("\n")
        #Lemmatization
        lemmHin = lemmatize_hi(stopRemoved)
        print("Lemmatization:",lemmHin)
        print("\n")
        #POS Tagging
        postag = postagger_hi(lemmHin)
```

```
print("POS Tagging:",postag)
print("\n")
# for key, POSWord in postag.items():
#   print(POSWord.split("/"))
#Negation Handling
negHin = negation_handling_hin(postag)
print("Negation Handling:",negHin)
print("\n")
#Sentiment Score
senti = sentiment(negHin)
print(comment)
print(senti)
print("-----------------------------------------------------\n")
```

3.2 Output : Raw Input: That restraurant is not good. Itna ghatiya khaana to kabhi nahi khaaya
After Translation: वह रेस्टोरेंट अच्छा नहीं है। इतना घटिया खाना तो कभी नहीं खाया

After Tokenization: ['वह', 'रेस्टोरेंट', 'अच्छा', 'नहीं', 'है।', 'इतना', 'घटिया', 'खाना', 'तो', 'कभी', 'नहीं', 'खाया']

Removed Stopwords: {1: 'रेस्टोरेंट', 2: 'अच्छा', 3: 'नहीं', 4: 'है।', 5: 'इतना', 6: 'घटिया', 7: 'खाना', 10: 'नहीं', 11: 'खाया'}

Lemmatization: {1: 'रेस्टोरेंट', 2: 'अच्छा', 3: 'नहीं', 4: 'है', 5: '।', 6: 'इतना', 7: 'घटिया', 10: 'खाना', 11: 'नहीं'}

POS Tagging: {1: 'रेस्टोरेंट/NOUN', 2: 'अच्छा/ADJ', 3: 'नहीं/PART', 4: 'है/AUX', 5: '।/PUNCT', 6: 'इतना/DET', 7: 'घटिया/ADJ', 10: 'खाना/NOUN', 11: 'नहीं/PART'}

Negation Handling: {1: 'रेस्टोरेंट/NOUN', 2: '!अच्छा/ADJ', 4: 'है/AUX', 5: '।/PUNCT', 6: 'इतना/DET', 7: 'घटिया/ADJ', 10: '!खाना/NOUN'}

That restraurant is not good. Itna ghatiya khaana to kabhi nahi khaaya
('Negative', -0.3125)

Raw Input: ye khaana kitna taja hai
After Translation: ये खाना कितना ताजा है

After Tokenization: ['ये', 'खाना', 'कितना', 'ताजा', 'है']

Removed Stopwords: {1: 'खाना', 3: 'ताजा'}

Lemmatization: {1: 'खाना', 3: 'ताजा'}

POS Tagging: {1: 'खाना/NOUN', 3: 'ताजा/ADJ'}

Negation Handling: {1: 'खाना/NOUN', 3: 'ताजा/ADJ'}

ye khaana kitna taja hai
('Positive', 0.0625)

Raw Input: mera nam omkar hai
After Translation: मेरा नाम ओंकार है

After Tokenization: ['मेरा', 'नाम', 'ओंकार', 'है']

Removed Stopwords: {1: 'नाम', 2: 'ओंकार'}

Lemmatization: {1: 'नाम', 2: 'ओंकार'}

POS Tagging: {1: 'नाम/NOUN', 2: 'ओंकार/PROPN'}

Negation Handling: {1: 'नाम/NOUN', 2: 'ओंकार/PROPN'}

mera nam omkar hai
('Neutral', 0.0)

Chapter 4

Conclusion and Future Scope

4.1 Conclusion

NLP gives tokenization, stop word removal, punctuation removal, stemming, and lemmatization, etc process for applying in text to remove words that do not take part in finding sentiment. Then Feature extraction from the speech signal is the second most important step in this field.

4.2 Future Scope

We see a great deal of scope in expanding this system to other regional languages as well and also there is a scope in creating a system based on this to understand the sentiments of people posting reviews in regional languages.

References

1. Sharma, S., Srinivas, P. Y. K. L., & Balabantaray, R. C. (2015, December). Sentiment analysis of code-mix script. In 2015 international conference on computing and network communications (CoCoNet) (pp. 530-534). IEEE
2. Shelke, N., Deshpande, S., & Thakare, V. (2017). Domain independent approach for aspect oriented sentiment analysis for product reviews. In Proceedings of the 5th international conference on frontiers in intelligent computing: Theory and applications (pp. 651-659). Springer, Singapore.
3. Tadesse, M. M., Lin, H., Xu, B., & Yang, L. (2019). Detection of Depression-Related Posts in Reddit Social Media Forum. IEEE Access, 7, 44883-44893.
4. Hajar, M. (2016). Using YouTube comments for text-based emotion recognition. Procedia Computer Science, 83, 292-299.
5. Kaur, H., Mangat, V., & Krail, N. (2017). Dictionary based sentiment analysis of hinglish text. International Journal of Advanced Research in Computer Science, 8(5).
6. Ravi, K., & Ravi, V. (2016, March). Sentiment classification of Hinglish text. In 2016 3rd International Conference on Recent Advances in Information Technology (RAIT) (pp. 641-645). IEEE