National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Bhumika Sardana
Student ID: x22105859@student.ncirl.ie

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name	Bhumika Sardana
Student ID	X22105959@student.ncirl.ie
Programme	Data Analytics
Year:	2023
Module:	Msc Research project
Supervisor:	Vikas Tomer
Submission Due Date:	14/12/2023
Project Title:	Sentimental analysis for Hinenglish-code mixed data using advanced word embeddings
Word Count:	789
Page Count:	23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature	Bhumika
Date	

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Bhumika Sardana
Student ID: x22105859@student.ncirl.ie

1 Introduction

This Configuration Manual lists together all prerequisites needed to duplicate the studies and its effects on a specific setting. A glimpse of the source for Exploratory Data analysis for is done followed by sentiment analysis, data preprocessing and cleaning and vectorization after that all the algorithms are created, and Evaluations is also supplied. After that recommendatory system is build and put together with the necessary hardware components as well as Software applications. The report is organized as follows, with details relating environment configuration provided in Section 2.
Information about data collection is detailed in Section 3. Exploratory Data Analysis is done in Section 4. Sentiment Analysis is included in Section 5. In section 6, the Data Clenaing is described. Section 7 provides details of Tokenisation. Details well about models that were created and tested are provided in Section 8. How the results are calculated and shown is described in Section 9.

2 System Requirements

The specific needs for hardware as well as software to put the research into use are detailed in this section.

2.1 Hardware Requirements

The necessary hardware specs are shown in Figure 1 below. MacOs M1 Chip, macOS 10.15.x (Catalilna) operating system, 8GB RAM, 256GB Storage, 24'' Display.

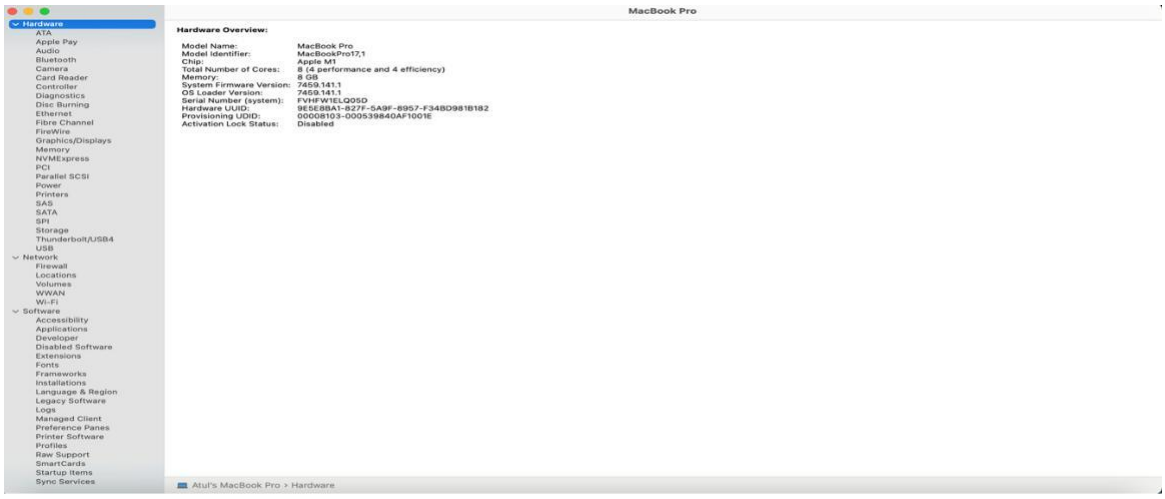

Figure 1: Hardware Requirements

2.2 Software Requirements

- Anaconda 3 (Version 4.8.0)
- Jupyter Notebook (Version 6.0.3)
- Python (Version 3.7.6)

2.3 Code Execution

The code can be run in jupyter notebook. The jupyter notebook comes with Anaconda 3, run the jupyter notebook from startup. This will open jupyter notebook in web browser. The web browser will show the folder structure of the system, move to the folder where the code file is located. Open the code file from the folder and to run the code, go to Kernel menu and Run all cells.

3 Data Gathering

The dataset is collected from https://github.com/l3cube-pune/code-mixed-nlp/tree/main. L3Cube-HingCorpus is the first large-scale real Hindi-English code mixed data in a Roman. It consists of 52.93M sentences and 1.04B tokens, scraped from Twitter. We also present HingBERT, HingMBERT, HingRoBERTa, and HingGPT. The BERT models have been pre-trained on codemixed HingCorpus.

4 Exploratory Data Analysis

Figure 2 includes a list of every Python library necessary to complete the project.

```
!pip install ktrain
#!pip install --upgrade tensorflow
```

```
Requirement already satisfied: ktrain in /usr/local/lib/python3.10/dist-packages (0.39.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from ktrain) (1.2.2)
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from ktrain) (3.7.1)
Requirement already satisfied: pandas>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from ktrain) (1.5.3)
Requirement already satisfied: fastprogress>=0.1.21 in /usr/local/lib/python3.10/dist-packages (from ktrain) (1.0.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from ktrain) (2.31.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from ktrain) (1.3.2)
```

```
#installing all libraries
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import re
from tqdm import tqdm
import warnings
import seaborn as sns
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
import ktrain
from ktrain import text
from transformers import BertTokenizer
import tensorflow as tf
from tensorflow.keras import backend as K
from keras.layers import Dense, Flatten, Input, Dropout, Bidirectional, LSTM, Conv1D, Conv2D, Activation, MaxPooling1D, BatchNormalization
from keras.models import Sequential, Model
import keras
from sklearn.feature_extraction.text import TfidfVectorizer
from transformers import AutoModel, AutoTokenizer
nltk.download('punkt')
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
True
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Figure 2: Necessary Python libraries

The Figure 3 represents the block of code to import data as text file nd enerate a list of sentence from the file.

```
lst = []
with open('/content/drive/MyDrive/HinglishTextSA/code-mixed-nlp-main/code-mixed-nlp-main/L3Cube-HingLID/all.txt') as small_pf:
    tmp_list = []
    for line in small_pf:
        line = line.rstrip("\n")
        if line == "":
            lst.append(tmp_list)
            tmp_list = []
        else:
            tmp_list.extend(line.split())

    if tmp_list:  # add last one
        lst.append(tmp_list)

print(lst[:5])
```

```
[['sanatan', 'HI', '0809', 'HI', 'tiding', 'EN', 'luna', 'EN', 'in', 'EN', 'ham', 'HI', 'bowling', 'EN', 'krte', 'HI', 'hai', 'HI', 'aap',
```

```
lst = [[' '.join(i)] for i in lst]
print(lst[:5])
```

```
[['sanatan HI 0809 HI tiding EN luna EN in EN ham HI bowling EN krte HI hai HI aap HI apne HI stump EN bhjao HI'], ['aap HI ne HI kaha HI
```

Figure 3: Data import

The Figure 4 represents the block of code to convert list of sentences to data frame and printing data information and top 5 rows.

```
df = pd.DataFrame(lst, columns=['Sentences'])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44453 entries, 0 to 44452
Data columns (total 1 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Sentences  44453 non-null  object
dtypes: object(1)
memory usage: 347.4+ KB
```

```
df.head()
```

	Sentences
0	sanatan HI 0809 HI tiding EN luna EN in EN ham...
1	aap HI ne HI kaha HI tha HI apne HI video EN m...
2	gill EN main HI first EN time EN bb13 HI ko HI...
3	ek HI nari HI ko HI duniya HI ke HI samne HI a...
4	jatti7 HI ki HI ho HI gea HI sadi HI stubborn ...

Figure 4: Data Information

In figure 5, the code to check total number of records and drop in case of missing data.

```
print("Total no. of Data points:",len(df))
print("="*50)
print(df[:10])
```

```
Total no. of Data points: 44453
==================================================
                                            Sentence
0  sanatan HI 0809 HI tiding EN luna EN in EN ham..
1  aap HI ne HI kaha HI tha HI apne HI video EN m..
2  gill EN main HI first EN time EN bb13 HI ko HI..
3  ek HI nari HI ko HI duniya HI ke HI samne HI a..
4  jatti7 HI ki HI ho HI gea HI sadi HI stubborn ..
5  when EN i EN send EN this EN i EN mean EN this..
6  sanvee HI arre HI haan HI yaar HI agar HI ek H..
7  phir HI tere HI katora HI khan EN ne HI kyu HI..
8  dwary HI ye HI adhikari HI apne HI power EN ka..
9  aa HI gaya HI na HI apne HI idol EN wale HI fo..
```

```
df.dropna(inplace=True)
df
```

	Sentences
0	sanatan HI 0809 HI tiding EN luna EN in EN ham...
1	aap HI ne HI kaha HI tha HI apne HI video EN m...
2	gill EN main HI first EN time EN bb13 HI ko HI...
3	ek HI nari HI ko HI duniya HI ke HI samne HI a...
4	jatti7 HI ki HI ho HI gea HI sadi HI stubborn ...
...	...
44448	arjuner HI going EN by EN thread EN jaakar HI ...
44449	gill EN twitter EN pe HI mein HI apne HI free ...
44450	sir HI aapne HI to HI sabko HI free EN ki HI a...
44451	999 EN rangnath EN aaj HI apne HI usually EN m...
44452	baat HI sahi HI hai HI but EN ab HI fozol HI k...

44453 rows × 1 columns

Figure 5: Missing Data

The Figure 6, checking train data.

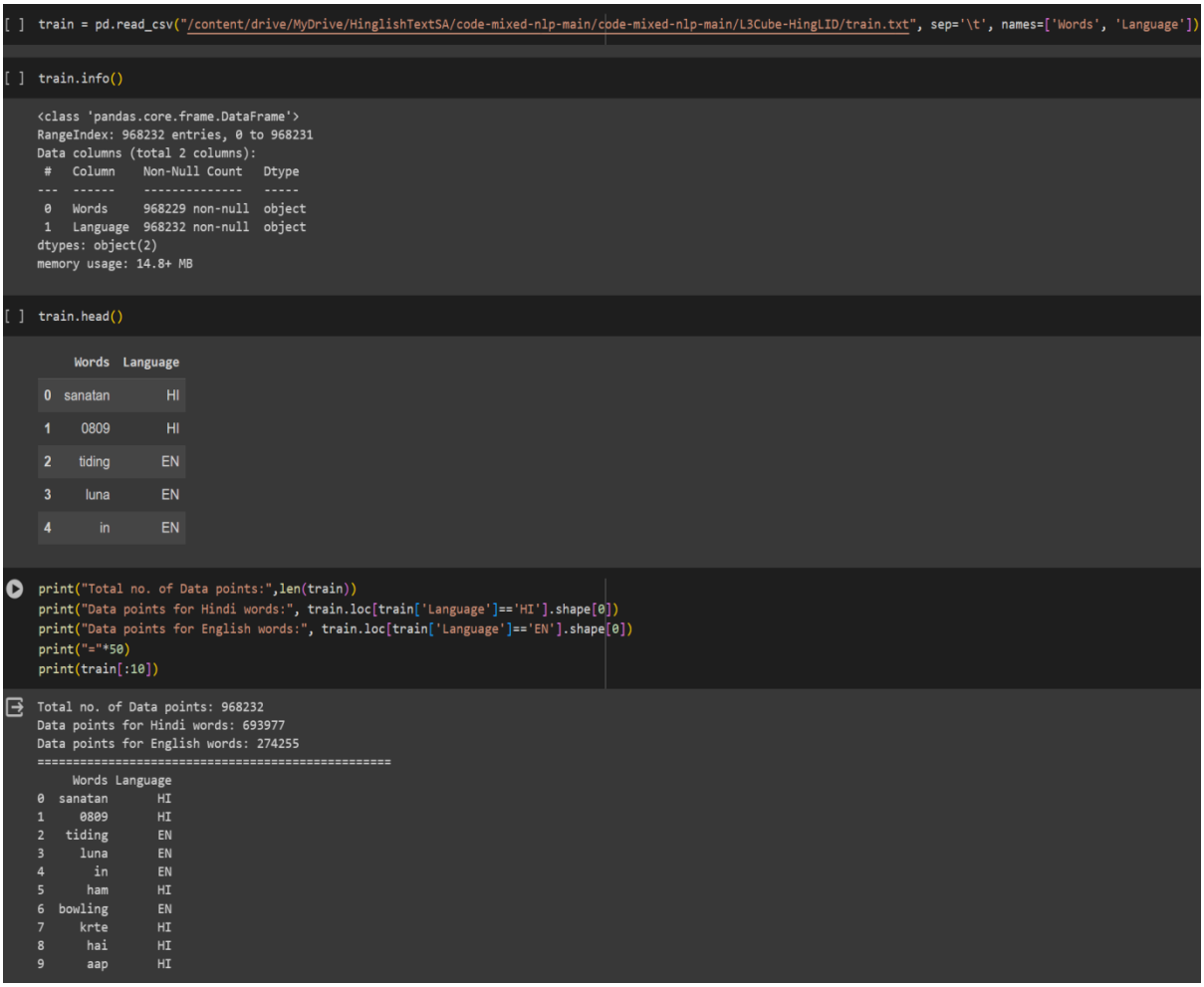

Figure 6: Train data

Figure 7 includes checking test data.

```
[ ] test = pd.read_csv("/content/drive/MyDrive/HinglishTextSA/code-mixed-nlp-main/code-mixed-nlp-main/L3Cube-HingLID/test.txt", sep='\t', names=['Words', 'Language'])

[ ] test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193547 entries, 0 to 193546
Data columns (total 2 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   Words     193546 non-null  object
 1   Language  193547 non-null  object
dtypes: object(2)
memory usage: 3.0+ MB

[ ] test.head()
```

	Words	Language
0	hai	HI
1	behan	HI
2	ki	HI
3	I	EN
4	hai	HI

```
print("Total no. of Data points:",len(test))
print("Data points for Hindi words:", test.loc[test['Language']=='HI'].shape[0])
print("Data points for English words:", test.loc[test['Language']=='EN'].shape[0])
print("="*50)
print(test[:10])

Total no. of Data points: 193547
Data points for Hindi words: 136824
Data points for English words: 56723
==================================================
   Words Language
0    hai       HI
1  behan       HI
2     ki       HI
3      I       EN
4    hai       HI
5   test       EN
6  karna       HI
7     ho       HI
8     ek       HI
9   baar       HI
```

Figure 7: Test data

The Figure 8 code segment to validation data.

```
validation = pd.read_csv("/content/drive/MyDrive/HinglishTextSA/code-mixed-nlp-main/code-mixed-nlp-main/L3Cube-HingLID/validation.txt", sep='\t', names=['Words', 'Language'])

validation.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193718 entries, 0 to 193717
Data columns (total 2 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   Words     193717 non-null  object
 1   Language  193718 non-null  object
dtypes: object(2)
memory usage: 3.0+ MB

validation.head()
```

	Words	Language
0	dahej	HI
1	k	HI
2	liye	HI
3	paisa	HI
4	bcha	HI

```
print("Total no. of Data points:",len(validation))
print("Data points for Hindi words:", validation.loc[validation['Language']=='HI'].shape[0])
print("Data points for English words:", validation.loc[validation['Language']=='EN'].shape[0])
print("="*50)
print(validation[:10])

Total no. of Data points: 193718
Data points for Hindi words: 137575
Data points for English words: 56143
==================================================
   Words Language
0  dahej       HI
1      k       HI
2   liye       HI
3  paisa       HI
4   bcha       HI
5   rhii       HI
6    huu       HI
7   kuch       HI
8    log       HI
9    jab       HI
```

Figure 8: Validation data

As seen in Figure 9, the list of all hindi words and generate wordcloud of them.

```
temp = validation.loc[validation['Language']=='HI']
print(temp['Words'].to_list())
```

```
['dahej', 'k', 'liye', 'paisa', 'bcha', 'rhii', 'huu', 'kuch', 'log', 'jab', 'ki', 'baat', 'karte', 'hai', 'jo', 'ek', 'ko', 'kisi', 'bhi', 'ke', 'sath',
```

```
all_hindi_words=(" ").join(temp['Words'][0:200].to_list())
all_hindi_words
```

```
'dahej k liye paisa bcha rhii huu kuch log jab ki baat karte hai jo ek ko kisi bhi ke sath kar dete hai cahaye shadi shuda kyo na ho taraf ke log sabhi ki
aine apne ummidwar hone ka abhi nahi socha hai abhi tak to hamne karke jeeta hai ka aur aage jeetenge ka me chunav jitana hamari koshish hai phir apne bap
apne ranjhe ko kheer fateh fatejo udaariyan mere wale ke neeche bohut ladkon ne apne kaam ka kiya hai aaj apne us ko kiya jiska se koi lena dena hi nahi t
l sakta hai kya kisi ko loya karke'
```

```
wordcloud = WordCloud(width = 1000, height = 500).generate(all_hindi_words)
plt.figure(figsize=(15,8))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
plt.close()
```

Figure 9: Hindi Wordcloud

As seen in Figure 10, the list of all english words and generate wordcloud of them.

```
temp = validation.loc[validation['Language']=='EN']
print(temp['Words'].to_list())
```

```
['woman', 'respect', 'men', 'women', 'link', 'woman', '1', 'woman', 'respect', '1', 'woman', 'disrespect', 'degrade', 'leval', 'low', 'h', 'job', 'do', 'o
```

```
all_english_words=(" ").join(temp['Words'][0:200].to_list())
all_english_words
```

```
'woman respect men women link woman 1 woman respect 1 woman disrespect degrade leval low h job do or education self respect or sir support seat mla counci
imination stunt that s not fair must check the possibilities 1 evms result 2 medal 3 soldiers planned attacks benefit 4 judge pressure banana battalion ma
s girl badly needs a job degree phr king right castle kr but google kr username janhit bar bar department sides fd prove karma kuye 90 q add kr single mur
30 countdown video twitter afghan tweet famous do to is pakistan dm h jb mn ...'
```

```
wordcloud = WordCloud(width = 1000, height = 500).generate(all_english_words)
plt.figure(figsize=(15,8))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
plt.close()
```

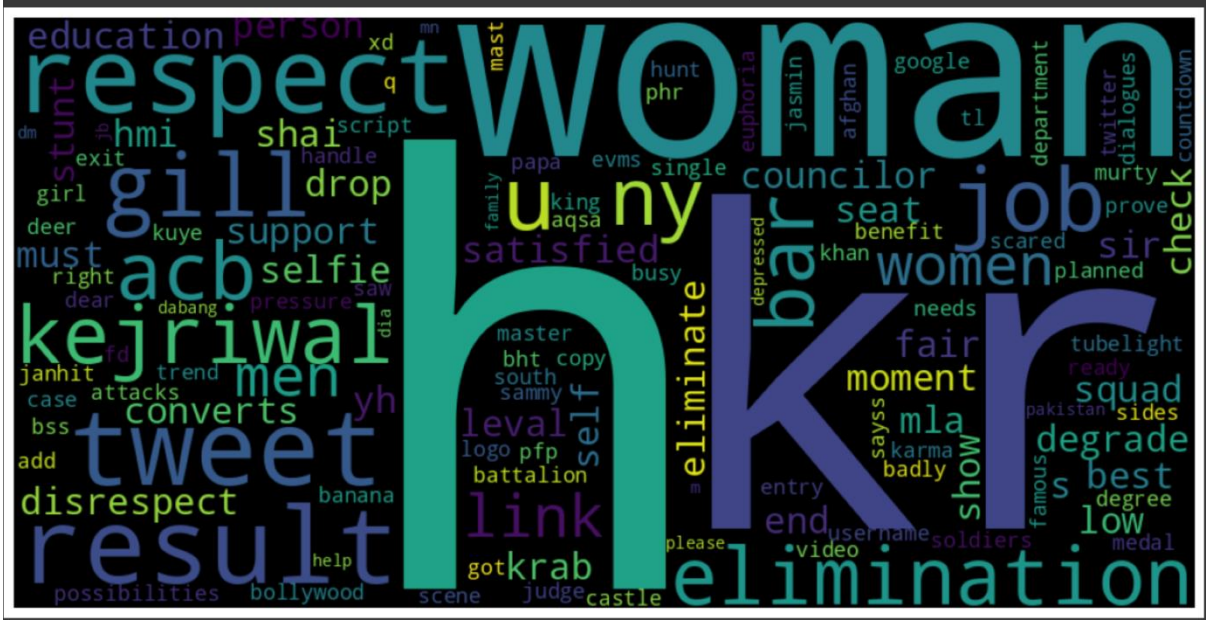

Figure 10: English Wordcloud

5 Sentiment Analysis

The Figure 11, illustrate the sentiment analysis of sentence illustrate the code and to analyze value for positive or negative class based on final sentiment.

```
sa = SentimentIntensityAnalyzer()
```

```
df['Score'] = df['Sentences'].apply(lambda text:sa.polarity_scores(str(text))['compound'])
```

```
s=[]
for compound in df['Score']:
        if compound > 0:
            s.append(1)
        else:
            s.append(0)
df['Score'] = s
```

```
df['Score'].value_counts()
```

```
0    28822
1    15631
Name: Score, dtype: int64
```

Figure 11: Sentiment Analysis

6 Data Cleaning

The Figure 12, illustrate the code to balance the data by applying undersampling.

```
pos = df[df['Score'] == 1]
neg = df[df['Score'] == 0]
neg_sampled = neg.sample(n=len(pos), replace=True)
df = pd.concat([pos, neg_sampled])
df['Score'].value_counts()
```

```
1    15631
0    15631
Name: Score, dtype: int64
```

Figure 12: Class Balancing

The Figure 13, illustrate the function to clear punctuations and contractions of the words in the sentence.

```
def cleanpunc(sentence): #function to clean the word of any punctuation or special characters
    cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
    cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
    return  cleaned
```

```
def decontracted(phrase):
    # This function decontract words like it's to it is.

    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    phrase = re.sub(r"\n", " ", phrase)
    return phrase
```

Figure 13: Clean punctuations and contractions

The Figure 14, illustrate the stopwords.

```
my_file = open("/content/drive/MyDrive/HinglishTextSA/stop_hinglish.txt", "r")

# reading the file
data = my_file.read()

# replacing end of line('/n') with ' ' and
# splitting the text it further when '.' is seen.
stop = data.split("\n")
print(stop)
my_file.close()
```

```
['a', 'aadi', 'aaj', 'aap', 'aapne', 'aata', 'aati', 'aaya', 'aaye', 'ab', 'abbe', 'abbey', 'abe', 'abhi', 'able',
```

```
stop.append('HI')
stop.append('EN')
print(stop)
```

```
['a', 'aadi', 'aaj', 'aap', 'aapne', 'aata', 'aati', 'aaya', 'aaye', 'ab', 'abbe', 'abbey', 'abe', 'abhi', 'able',
```

```
df['Sentences'] = df['Sentences'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
df['Sentences']
```

```
1        video video share link aksar poochti maulana s...
3        nari duniya samne nagan sarir pradarshan sobha...
15                          sp party bade neta humara
18       malik bhut riyaat pmln gira dainge vote mangna...
19       raffle debets karle dango karle chamcho karle ...
                              ...
5533     agr uthane pabandi ge biden kia bap leke aega ...
7865     shukla suar jakar suar idol post comment bheek...
29995    ind tb uncle jagmohan kashmir governor questio...
13253    block level transfer suru kijiye mantri ji shi...
5833           meine sachi bhale twitter pr dekhen called
Name: Sentences, Length: 31262, dtype: object
```

Figure 14: Stopwords

The Figure 15, illustrate basic variable initialized for text cleaning.

```
#Basic variables
i=0
str1=' '
final_string=[]
all_positive_words=[] # store words from +ve reviews here
all_negative_words=[] # store words from -ve reviews here.
s=''
```

```
df
```

	Sentences	Score
1	video video share link aksar poochti maulana s...	1
3	nari duniya samne nagan sarir pradarshan sobha...	1
15	sp party bade neta humara	1
18	malik bhut riyaat pmln gira dainge vote mangna...	1
19	raffle debets karle dango karle chamcho karle ...	1
...	...	...
5533	agr uthane pabandi ge biden kia bap leke aega ...	0
7865	shukla suar jakar suar idol post comment bheek...	0
29995	ind tb uncle jagmohan kashmir governor questio...	0
13253	block level transfer suru kijiye mantri ji shi...	0
5833	meine sachi bhale twitter pr dekhen called	0

31262 rows × 2 columns

Figure 15: Variables

The Figure 16, illustrate the cleaning process checking each word in the sentence.

```
for sent in tqdm(df['Sentences']):
    filtered_sentence=[]
    sent = decontracted(sent)
    for w in sent.split():
        for cleaned_words in cleanpunc(w).split():
            if((cleaned_words.isalpha()) & (len(cleaned_words)>2)):
                if(cleaned_words.lower() not in stop):
                    s=(cleaned_words.lower()).encode('utf8')
                    filtered_sentence.append(s)
                    if (df['Score'].values)[i] == 1:
                            all_positive_words.append(s) #list of all words used to describe positive reviews
                    if (df['Score'].values)[i] == 0:
                            all_negative_words.append(s) #list of all words used to describe negative reviews reviews
                else:
                    continue
            else:
                continue

    str1 = b" ".join(filtered_sentence) #final string of cleaned words
    final_string.append(str1)
    i+=1

df["Sentences"] = final_string
df['Sentences']=df['Sentences'].str.decode("utf-8")

100%|██████████| 31262/31262 [00:11<00:00, 2700.95it/s]
```

Figure 16: Cleaning Process

The Figure 17, illustrate the code to print clean sentence.

```
df["Sentences"]

1        video video share link aksar poochti maulana s...
3        nari duniya samne nagan sarir pradarshan sobha...
15                                  party bade neta humara
18       malik bhut riyaat pmln gira dainge vote mangna...
19       raffle debets karle dango karle chamcho karle ...
                               ...
5533     agr uthane pabandi biden kia bap leke aega tmhare
7865     shukla suar jakar suar idol post comment bheek...
29995    ind uncle jagmohan kashmir governor question u...
13253    block level transfer suru kijiye mantri shiksh...
5833                meine sachi bhale twitter dekhen called
Name: Sentences, Length: 31262, dtype: object
```

Figure 17: Clean sentence

The Figure 18, illustrate wordcloud for positive words.

```
#generating word cloud for all postive words
print("Wordcloud of words present in positive class : \n")
wordcloud = WordCloud(width = 500, height = 500,background_color ='white', min_font_size = 10).generate(b' '.
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

```
Wordcloud of words present in positive class :
```

Figure 18: Wordcloud for Positive words

The Figure 19, illustrate wordcloud for negative words.

```
#generating word cloud for all negative words
print("Wordcloud of words present in negative class : \n")
wordcloud = WordCloud(width = 500, height = 500,background_color ='white', min_font_size = 10).generate(b' '
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

```
Wordcloud of words present in negative class :
```

Figure 19: Wordcloud for Negative words

The Figure 20, the code to split data into training and test set.

```
# Spliting data in test and train data so the models can be trained and tested
n = int(len(df)- (len(df)*0.05))
train= df[0:n]
test = df[n:]
train.shape, test.shape
```

```
((29698, 2), (1564, 2))
```

```
X_train= train['Sentences']
X_test= test['Sentences']
y_train= train['Score']
y_test = test['Score']
```

Figure 20: Train test Split

7 Tokenisation

The Figure 21, illustrate the code for Bert Tokenizer.

```
y_train = keras.utils.to_categorical(y_train, 2)
y_test = keras.utils.to_categorical(y_test, 2)
```

```
from transformers import BertTokenizer, BertForSequenceClassification
tokenizer = BertTokenizer.from_pretrained("ganeshkharad/gk-hinglish-sentiment")
```

```
tokenized_texts = [tokenizer.tokenize(com) for com in X_train]
tokenized_texts = [sent[:len(tokenized_texts)] for sent in tokenized_texts]
trainData = [tokenizer.convert_tokens_to_ids(com) for com in tokenized_texts]
trainData = tf.keras.preprocessing.sequence.pad_sequences(trainData, maxlen=30, truncating='post', padding='post')
trainData.shape
```

```
(29698, 30)
```

```
tokenized_texts = [tokenizer.tokenize(com) for com in X_test]
tokenized_texts = [sent[:len(tokenized_texts)] for sent in tokenized_texts]
testData = [tokenizer.convert_tokens_to_ids(com) for com in tokenized_texts]
testData = tf.keras.preprocessing.sequence.pad_sequences(testData, maxlen=30, truncating='post', padding='post')
testData.shape
```

```
(1564, 30)
```

```
score = pd.DataFrame()
```

Figure 21: Bert Tokenizer

The Figure 22, illustrate the code for TF-IDF Vectorizer

.

```
n=30
Tfidf_vect = TfidfVectorizer(max_features=n)
Tfidf_vect.fit(df['Sentences'])
X_train_Tfidf = Tfidf_vect.transform(X_train).toarray()
X_test_Tfidf = Tfidf_vect.transform(X_test).toarray()
X_train_Tfidf.shape, X_test_Tfidf.shape
```

```
((29698, 30), (1564, 30))
```

Figure 22: TfidfVectorizer

The Figure 23, illustrate the code for MURIL Tokenizer.

```
# Define the model repo
model_name = "google/muril-base-cased"
tokenizer = BertTokenizer.from_pretrained(model_name, do_lower_case=True)
n=7
```

```
x_train_tokenized = tokenizer(X_train.tolist(), return_tensors='pt', max_length=50, truncation=True, padding='max_length')
x_test_tokenized = tokenizer(X_test.tolist(), return_tensors='pt', max_length=50, truncation=True, padding='max_length')
```

```
tokenized_texts = [tokenizer.tokenize(com) for com in X_train]
tokenized_texts = [sent[:len(tokenized_texts)] for sent in tokenized_texts]
x_train_tokenized = [tokenizer.convert_tokens_to_ids(com) for com in tokenized_texts]
x_train_tokenized = tf.keras.preprocessing.sequence.pad_sequences(x_train_tokenized, maxlen=n, truncating='post', padding='post'
x_train_tokenized.shape
```

```
(29698, 7)
```

```
tokenized_texts = [tokenizer.tokenize(com) for com in X_test]
tokenized_texts = [sent[:len(tokenized_texts)] for sent in tokenized_texts]
x_test_tokenized = [tokenizer.convert_tokens_to_ids(com) for com in tokenized_texts]
x_test_tokenized = tf.keras.preprocessing.sequence.pad_sequences(x_test_tokenized, maxlen=n, truncating='post', padding='post')
x_test_tokenized.shape
```

```
(1564, 7)
```

Figure 23: MURIL Tokenizer

8 Machine Learning Models

8.1 LSTM Bert

```
lstm = Sequential()
lstm.add(Input(shape=(30,1)))
lstm.add(LSTM(64, activation='relu'))
lstm.add(Flatten())
lstm.add(Dense(32, activation='relu'))
lstm.add(BatchNormalization())
lstm.add(Dense(16, activation='relu'))
lstm.add(Dense(2, activation='relu'))
lstm.compile(optimizer='sgd', loss="categorical_crossentropy", metrics=["accuracy"])
lstm.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                16896

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 32)                2080

 batch_normalization (BatchN  (None, 32)               128
 ormalization)

 dense_1 (Dense)             (None, 16)                528

 dense_2 (Dense)             (None, 2)                 34

=================================================================
Total params: 19,666
Trainable params: 19,602
Non-trainable params: 64
_________________________________________________________________
```

```
lstm.fit(trainData, y_train )
```

```
929/929 [==============================] - 14s 13ms/step - loss: nan - accuracy: 0.4736
<keras.callbacks.History at 0x7db6f386bc70>
```

Figure 24: Implementation of LSTM Bert

8.2 CNN Bert

```
cnn = Sequential()
cnn.add(Input(shape=(30,1)))
cnn.add(Conv1D(128, 1))
cnn.add(MaxPooling1D(2))
cnn.add(Activation('relu'))
cnn.add(Conv1D(64, 1))
cnn.add(MaxPooling1D(2))
cnn.add(Activation('relu'))
cnn.add(Conv1D(32, 1))
cnn.add(MaxPooling1D(2))
cnn.add(Activation('tanh'))
cnn.add(Flatten())
cnn.add(Dense(2, activation='tanh'))
cnn.compile(optimizer='adam', loss="categorical_crossentropy", metrics=["accuracy"])
cnn.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 30, 128)           256

 max_pooling1d (MaxPooling1D  (None, 15, 128)          0
 )

 activation (Activation)     (None, 15, 128)           0

 conv1d_1 (Conv1D)           (None, 15, 64)            8256

 max_pooling1d_1 (MaxPooling  (None, 7, 64)            0
 1D)

 activation_1 (Activation)   (None, 7, 64)             0

 conv1d_2 (Conv1D)           (None, 7, 32)             2080

 max_pooling1d_2 (MaxPooling  (None, 3, 32)            0
 1D)

 activation_2 (Activation)   (None, 3, 32)             0

 flatten_1 (Flatten)         (None, 96)                0

 dense_3 (Dense)             (None, 2)                 194

=================================================================
Total params: 10,786
Trainable params: 10,786
Non-trainable params: 0
_________________________________________________________________
```

```
cnn.fit(trainData, y_train )
```

```
929/929 [==============================] - 4s 3ms/step - loss: 0.7057 - accuracy: 0.5298
<keras.callbacks.History at 0x7db6f774e2c0>
```

Figure 25: Implementation of CNN Bert

8.3 LSTM TFIDF

```
lstm = Sequential()
lstm.add(Input(shape=(n,1)))
lstm.add(LSTM(12, activation='relu'))
lstm.add(Flatten())
lstm.add(Dense(32, activation='relu'))
lstm.add(BatchNormalization())
lstm.add(Dense(2, activation='softmax'))
lstm.compile(optimizer='adam', loss="categorical_crossentropy", metrics=["accuracy"])
lstm.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)               (None, 12)                672

 flatten_2 (Flatten)         (None, 12)                0

 dense_4 (Dense)             (None, 32)                416

 batch_normalization_1 (Batc  (None, 32)               128
 hNormalization)

 dense_5 (Dense)             (None, 2)                 66

=================================================================
Total params: 1,282
Trainable params: 1,218
Non-trainable params: 64
_________________________________________________________________
```

```
lstm.fit(X_train_Tfidf, y_train )
```

```
929/929 [==============================] - 10s 9ms/step - loss: 0.6873 - accuracy: 0.5306
<keras.callbacks.History at 0x7db6f27e2380>
```

Figure 26: Implementation of LSTM TF-IDF

8.4 CNN TFIDF

```
cnn = Sequential()
cnn.add(Input(shape=(n,1)))
cnn.add(Conv1D(512, kernel_size=1, activation='relu'))
cnn.add(Flatten())
cnn.add(Dense(8, activation='relu'))
cnn.add(BatchNormalization())
cnn.add(Dense(2, activation='softmax'))
cnn.compile(optimizer='Nadam', loss="categorical_crossentropy", metrics=["accuracy"])
cnn.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_3 (Conv1D)           (None, 30, 512)           1024

 flatten_3 (Flatten)         (None, 15360)             0

 dense_6 (Dense)             (None, 8)                 122888

 batch_normalization_2 (Batc  (None, 8)                32
 hNormalization)

 dense_7 (Dense)             (None, 2)                 18

=================================================================
Total params: 123,962
Trainable params: 123,946
Non-trainable params: 16
_________________________________________________________________
```

```
cnn.fit(X_train_Tfidf, y_train )
```

```
929/929 [==============================] - 5s 4ms/step - loss: 0.6619 - accuracy: 0.5482
<keras.callbacks.History at 0x7db6f2d63c70>
```

Figure 27: Implementation of CNN TFIDF

8.5 LSTM Muril

```
lstm = Sequential()
lstm.add(Input(shape=(n,1)))
lstm.add(LSTM(64, activation='relu'))
lstm.add(Flatten())
lstm.add(Dense(32, activation='relu'))
lstm.add(BatchNormalization())
lstm.add(Dense(16, activation='relu'))
lstm.add(Dense(2, activation='relu'))
lstm.compile(optimizer='sgd', loss="categorical_crossentropy", metrics=["accuracy"])
lstm.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 64)                16896

 flatten_4 (Flatten)         (None, 64)                0

 dense_8 (Dense)             (None, 32)                2080

 batch_normalization_3 (Batc  (None, 32)               128
 hNormalization)

 dense_9 (Dense)             (None, 16)                528

 dense_10 (Dense)            (None, 2)                 34

=================================================================
Total params: 19,666
Trainable params: 19,602
Non-trainable params: 64
_________________________________________________________________
```

```
lstm.fit(x_train_tokenized, y_train )
```

```
929/929 [==============================] - 6s 5ms/step - loss: nan - accuracy: 0.4737
<keras.callbacks.History at 0x7db6f29e6c20>
```

Figure 28: Implementation of LSTM Muril

8.6 CNN Muril

```
cnn = Sequential()
cnn.add(Input(shape=(n,1)))
cnn.add(Conv1D(128, 1))
cnn.add(MaxPooling1D(2))
cnn.add(Activation('relu'))
cnn.add(Conv1D(64, 1))
cnn.add(MaxPooling1D(2))
cnn.add(Activation('relu'))
cnn.add(Flatten())
cnn.add(Dense(2, activation='tanh'))
cnn.compile(optimizer='adam', loss="categorical_crossentropy", metrics=["accuracy"])
cnn.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_4 (Conv1D)           (None, 7, 128)            256

 max_pooling1d_3 (MaxPooling  (None, 3, 128)           0
 1D)

 activation_3 (Activation)   (None, 3, 128)            0

 conv1d_5 (Conv1D)           (None, 3, 64)             8256

 max_pooling1d_4 (MaxPooling  (None, 1, 64)            0
 1D)

 activation_4 (Activation)   (None, 1, 64)             0

 flatten_5 (Flatten)         (None, 64)                0

 dense_11 (Dense)            (None, 2)                 130

=================================================================
Total params: 8,642
Trainable params: 8,642
Non-trainable params: 0
_________________________________________________________________
```

```
cnn.fit(x_train_tokenized, y_train )
```

```
929/929 [==============================] - 3s 2ms/step - loss: 0.6932 - accuracy: 0.4736
<keras.callbacks.History at 0x7db6f1cc8970>
```

Figure 29: Implementation of CNN Muri

References

https://github.com/l3cube-pune/code-mixed-nlp/tree/main

Understanding TF-IDF (Term Frequency-Inverse Document Frequency) - GeeksforGeeks

MuRIL | A General Introduction to MuRIL - Analytics Vidhya

ganeshkharad/gk-hinglish-sentiment · Hugging Face

Understanding of LSTM Networks - GeeksforGeeks

Convolutional Neural Network (CNN) in Machine Learning - GeeksforGeeks