Import pandas as pd

data=pd.read\_csv("https://raw.githubusercontent.com/Ankit152/IMDB-sentiment-analysis/master/IMDB-Dataset.csv")

Run this command on your terminal

pip install ipykernel, pandas

import string

exclude=string.punctuation

def remove\_punc(text):

for char in exclude:

text=text.replace(char,"")

return text

text2="my name is s@nny sa##vita!!!!!!!!!"

remove\_punc(text2)

def remove\_punc1(text):

return text.translate(str.maketrans("","",exclude))

data["review"]=data["review"].apply(remove\_punc)

<https://github.com/rishabhverma17/sms_slang_translator/blob/master/slang.txt>

text="FYI this is not true"

text2="LAMO the class was so funny"

text3="i want report ASAP"

chat\_words={

" AFAIK":"As Far As I Know",

"AFK": "Away From Keyboard",

"ASAP":"As Soon As Possible",

"BTW":"By The Way",

"B4":"Before",

"LAMO":"Laugh My A.. Off",

"FYI":"For your information"

}

def chat\_conversion(text):

new\_text=[]

for w in text.split():

if w.upper() in chat\_words:

new\_text.append(chat\_words[w.upper()])

else:

new\_text.append(w)

return " ".join(new\_text)

Pip install textblob

Import textblob

txtblob=TextBlob(text)

txtblob.correct().string

Pip install nltk

from nltk.corpus import stopwords

nltk.download('stopwords')

stopwords.words("english")

def remove\_stopwords(text):

new\_text=[]

for word in text.split():

if word in stopwords.words("english"):

new\_text.append("")

else:

new\_text.append(word.strip())

return " ".join(new\_text).replace(" ","")

Pip install emoji

<https://pypi.org/project/emoji/>

def remove\_emoji(text):

clean\_text=emoji.demojize(text)

return clean\_text

remove\_emoji(original\_text)

import re

def remove\_emojis\_manually(text):

emoji\_pattern = re.compile("["

u"\U0001F600-\U0001F64F" # emoticons

u"\U0001F300-\U0001F5FF" # symbols & pictographs

u"\U0001F680-\U0001F6FF" # transport & map symbols

u"\U0001F700-\U0001F77F" # alchemical symbols

u"\U0001F780-\U0001F7FF" # Geometric Shapes Extended

u"\U0001F800-\U0001F8FF" # Supplemental Arrows-C

u"\U0001F900-\U0001F9FF" # Supplemental Symbols and Pictographs

u"\U0001FA00-\U0001FA6F" # Chess Symbols

u"\U0001FA70-\U0001FAFF" # Symbols and Pictographs Extended-A

u"\U00002702-\U000027B0" # Dingbats

u"\U000024C2-\U0001F251"

"]+", flags=re.UNICODE)

clean\_text = emoji\_pattern.sub(r'', text)

return clean\_text

pip install spacy

python -m spacy download en\_core\_web\_sm

pip install ntlk

nltk.download(“alll”)

Assignment⇒<https://www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter>

from nltk.stem import WordNetLemmatizer

def lammatization(text):

words=text.split()

lemmetizer=WordNetLemmatizer()

lemetized\_word=[lemmetizer.lemmatize(word) for word in words]

return lemetized\_word

========================================================================

04/02/2024

Import pandas as pd

pd.DataFrame({"text":["people watch ineuron","ineuron watch ineuron","people write comment","ineuron write comment"],"output":[1,1,0,0]})

Link of Document matrix(BOW):<https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html>

pip install scikit-learn

from sklearn.feature\_extraction.text import CountVectorizer

BOW=CountVectorizer()

document\_matrix=BOW.fit\_transform(data["text"])

BOW.vocabulary\_

document\_matrix[0].toarray()

bigram=CountVectorizer(ngram\_range=(2,2))

bigramvocab=bigram.fit\_transform(data["text"])

bigram.vocabulary\_

mix=CountVectorizer(ngram\_range=(1,2))

mix\_vocab=mix.fit\_transform(data["text"])

mix.vocabulary\_

<https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html>

from sklearn.feature\_extraction.text import TfidfVectorizer

tfidf=TfidfVectorizer()

tfidf.fit\_transform(data["text"]).toarray()

tfidf.get\_feature\_names\_out()

tfidf.idf\_

Assignment⇒<https://www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter>

1. You need to do one hot encoding on top of text feature
2. You need to perform BOW from the text column
3. Here you can try N-grams also like 2-gram, 3-gram,4-gram
4. At the end you have to perform the tf-idf

Write 5 advantage and 5 disadvantage of it(you can write it down from anywhere google chat gpt)

OHE vs BOW vs N-grams vs TF-IDF

You can submit the assignment through this google form:

<https://forms.gle/LTRvpoeMFky4VgtC6>

**Special note:** share the github link only.

The timeline is till friday 5PM IST

Next sat and sun:

This is the agenda of the next class: Embedding(word2vec) and transformer architecture in detail

Will start from hugging face and project:

===============================================================

11/02/2024

import pandas as pd

import numpy as np

import os

import gensim

import nltk

nltk.download("all")

from nltk import sent\_tokenize

from gensim.utils import simple\_preprocess

for filename in os.listdir("/content/data"):

file\_path=os.path.join("/content/data",filename)

with open(file\_path,encoding='unicode\_escape') as f:

corpus=f.read()

raw\_sent=sent\_tokenize(corpus)

for sent in raw\_sent:

story.append(simple\_preprocess(sent))

model=gensim.models.Word2vec(window=10,min\_count=5,vector=150)

==============================================================

**Assignment for this week:**

Data⇒<https://www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter>

Generating:

1.Count or frequency based method:

2. Prediction based method(word2vec)

Create a model: naive bayes

<https://scikit-learn.org/stable/modules/naive_bayes.html>

If you are familiar then: You can use rnn/lstm/gru

Then figure out which embedding technique is working well for you

You can submit the assignment through this google form:

<https://forms.gle/LTRvpoeMFky4VgtC6>

**Special note:** share the github link only.