```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
import re
import nltk
```

```
# impurities that our text can have
# max , MAX  , MaX, maxxxxxxxxxxx
```

```
!pip install requests
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests) (2025.4.26)
```

```
# Download resources if not already available
def clean_text(text):
    text = text.lower()   #  Lowercase conversion
    text = re.sub(r'http\S+|www\S+|https\S+', '', text) # URL removal
    text = re.sub(r'<.*?>', '', text) # HTML tag removal
    text = re.sub(r'[^a-z\s]', '', text) # Special character removal
    text = re.sub(r'\s+', ' ', text).strip() # Extra whitespace removal   #strip to remove forward or backward spcaes
    text = re.sub(r'\S+@\S+\.\S+', '', text) # email removal
    text = re.sub(r'[^\w\s]', '', text) # punctuations
    text = re.sub(r'(.)\1{2,}', r'\1', text) # Repeated Characters / Elongated Words
    text = re.sub(r'#', '', text) # Hashtags (like from Twitter/Instagram)
    text= re.sub(r'@\w+', '', text) # Mention username
    text= re.sub(r'[^\x00-\x7F]+', '', text) # non ASCII characters (emojis , foreign characters)
    return text
```

```
dirty_text = """
<Review>OMG!!! 😱 I Loooove the NEW Phone Max 13 Pro+++... it's amaaaazinggg!!! 😍😍😍
Visit https://www.phone-world.com/product?id=123 or check www.tech-deals.net 😎📱
Seriously, best purchase ever (even though it cost me $1,299.99)!!!
Customer support was gr8! Contacted them via email: support@phone-world.com & got a reply within 2 hrs.
Follow them @PhoneWorldOfficial for #deals and more info!!
P.S. -- I had to return the case [wrong size] but the process was smooth.
    TOO     MANY     SPACES here     lol 😂😂
</Review>
"""
```

```
cleaned= clean_text(dirty_text)
```

```
cleaned
```

```
'omg i love the new phone max pro its amazing visit or check seriously best purchase ever even though it cost me customer suppo
rt was gr contacted them via email supportphoneworldcom got a reply within hrs follow them phoneworldofficial for deals and mor
e info ps i had to return the case wrong size but the process was smooth too many spaces here lol'
```

```
# LEMMITIZING USING SPACY
!pip  install spacy
!python - m spacy download en_core_web_sm
```

```
Requirement already satisfied: spacy in /usr/local/lib/python3.11/dist-packages (3.8.7)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.0.13)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.11)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.0.10)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.11/dist-packages (from spacy) (8.3.6)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.5.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.4.1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.16.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (4.67.1)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.2)
```

```
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.32.3)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.1
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (24.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.5.0)
Requirement already satisfied: language-data>=1.2 in /usr/local/lib/python3.11/dist-packages (from langcodes<4.0.0,>=3.2.0->spac
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spac
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spac
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3.4->sp
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (8.2.1)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (13.9.
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0-
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0->s
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->spacy) (3.0.2)
Requirement already satisfied: marisa-trie>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from language-data>=1.2->langcodes
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0.0
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0-
Python 3.11.12 (main, Apr  9 2025, 08:55:54) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>

KeyboardInterrupt
>>>
KeyboardInterrupt
>>> ^C
```

```
import spacy
# lemmatize using spacy
def clean_text_spacy(text):
    nlp= spacy.load("en_core_web_sm")
    doc= nlp(text)
    word=[]
    for token in doc:
        if not token.is_stop and not token.is_punct:
            word.append(token.lemma_)
    return word
```

```
clean_text_spacy(cleaned)
```

```
['omg',
 'love',
 'new',
 'phone',
 'max',
 'pro',
 'amazing',
 'visit',
 'check',
 'seriously',
 'good',
 'purchase',
 'cost',
 'customer',
 'support',
 'gr',
 'contact',
 'email',
 'supportphoneworldcom',
 'get',
 'reply',
 'hrs',
 'follow',
 'phoneworldofficial',
 'deal',
 'info',
 'ps',
 'return',
 'case',
 'wrong',
 'size',
```

```
 'process',
 'smooth',
 'space',
 'lol']
```

word cloud

```
text="""
These headphones are amazing! Great sound quality and super comfortable. I use them at the gym daily
            Battery life is fantastic—lasts me for days! Also love the sleek design and fast pairing
The noise cancellation is top-notch. Blocks out everything when I'm working.
            Very lightweight and doesn’t hurt my ears after long use. Excellent for video calls.
I love how quick they connect to my phone. The bass is powerful too.
            Perfect for travel. They fold nicely and the carrying case is a great addition.
Great product at a reasonable price. Feels like a premium experience.
            Customer service was excellent when I had a question about charging. Very responsive
Best headphones I’ve owned. Better than some high-end brands.
            Looks stylish and professional. I wear them to work every day."""
```

```
cleaned_1= clean_text(text)
cleaned_1= clean_text_spacy(cleaned_1)
```

```
cleaned_1
```

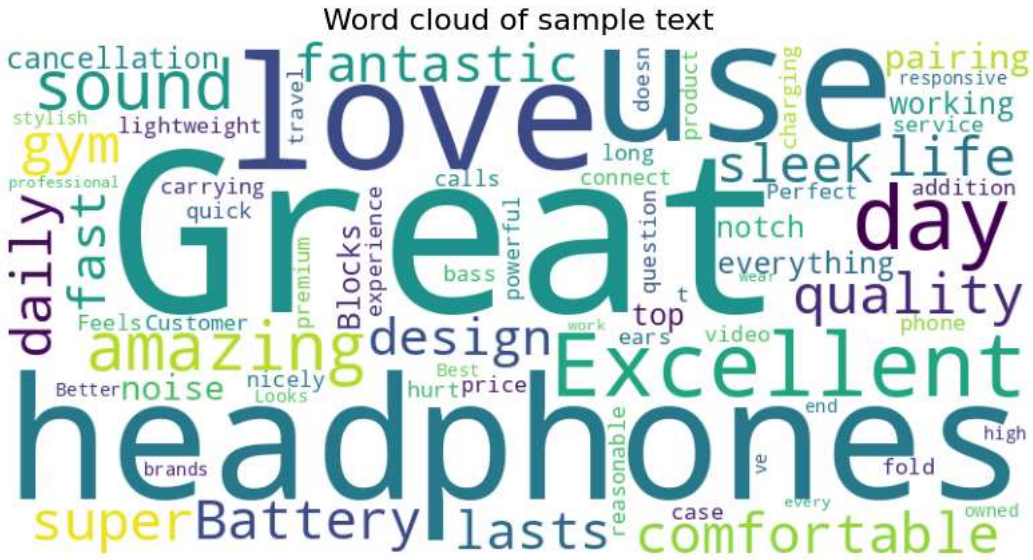
```
 'fast',
 'pair',
 'noise',
 'cancellation',
 'topnotch',
 'block',
 'm',
 'work',
 'lightweight',
 'not',
 'hurt',
 'ear',
 'long',
 'use',
 'excellent',
 'video',
 'call',
 'love',
 'quick',
 'connect',
 'phone',
 'bass',
 'powerful',
 'perfect',
 'travel',
 'fold',
 'nicely',
 'carrying',
 'case',
 'great',
 'addition',
 'great',
 'product',
 'reasonable',
 'price',
 'feel',
 'like',
```

```
 'professional',
 'wear',
 'work',
 'day']
```

```
Start coding or generate with AI.
```

```
from wordcloud import WordCloud
```

```
wordcloud= WordCloud(width=800, height=400, background_color="white").generate(text)

plt.figure(figsize=(10,5))
plt.imshow(wordcloud)
plt.axis("off")
plt.title("Word cloud of sample text", fontsize=16)
plt.show()
```




```
def show_wordcloud(text):
    try:
        wordcloud= WordCloud(width=800, height=400, background_color="white").generate(text)
        plt.figure(figsize=(10,5))
        plt.imshow(wordcloud)
        plt.axis("off")
        return plt
    except Exception as e:
        return f"Erroe generating word cloud:m {e}"
```

```
show_wordcloud(text)
```

```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.11/dist-packages/matplotlib/pyplot.py'>
```

N-GRAM ANALYSIS

```
from nltk.util import ngrams
from collections import Counter
import plotly.graph_objects as go
```

```
# create an bigram
biagrams= list(ngrams(cleaned_1, 3))
Counter(biagrams)
biagram_counts= Counter(biagrams).most_common(15)
```

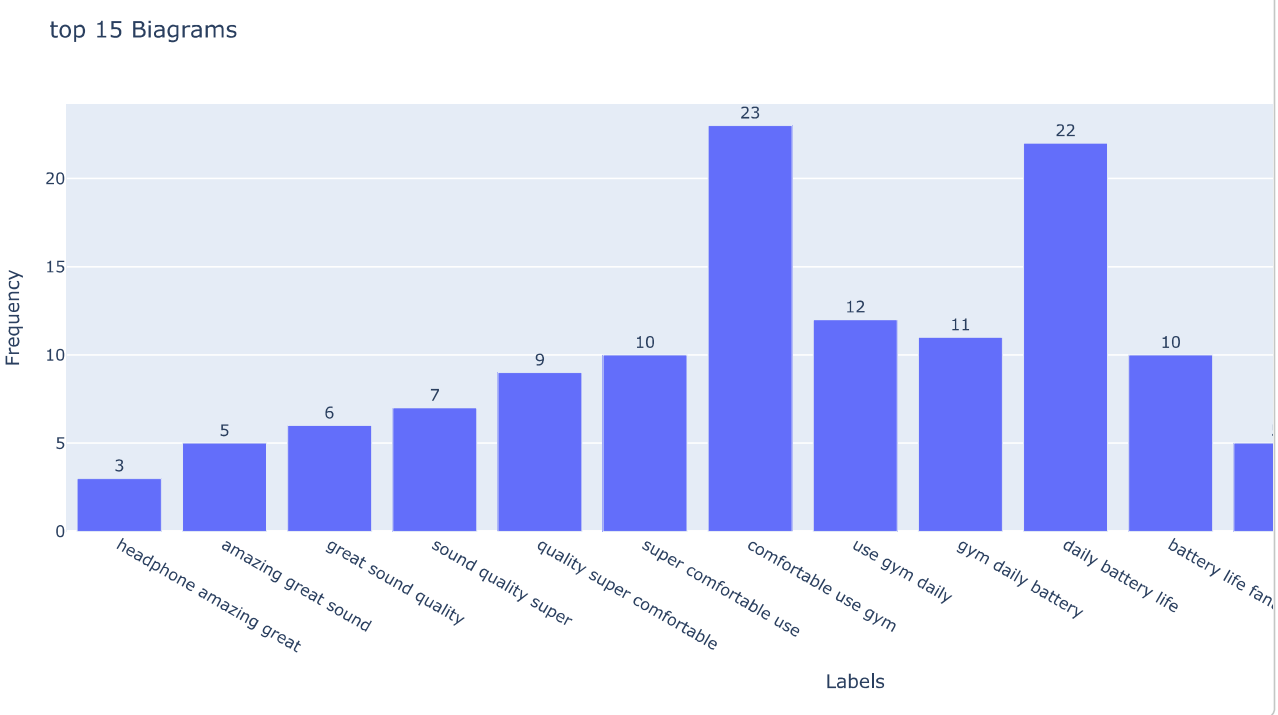
```
labels=[]
counts=[]


for biagram , count in biagram_counts:
    labels.append(" ".join(biagram))
    counts.append(count)
```

```
labels
```

```
['headphone amazing great',
 'amazing great sound',
 'great sound quality',
 'sound quality super',
 'quality super comfortable',
 'super comfortable use',
 'comfortable use gym',
 'use gym daily',
 'gym daily battery',
 'daily battery life',
 'battery life fantasticlast',
 'life fantasticlast day',
 'fantasticlast day love',
 'day love sleek',
 'love sleek design']
```

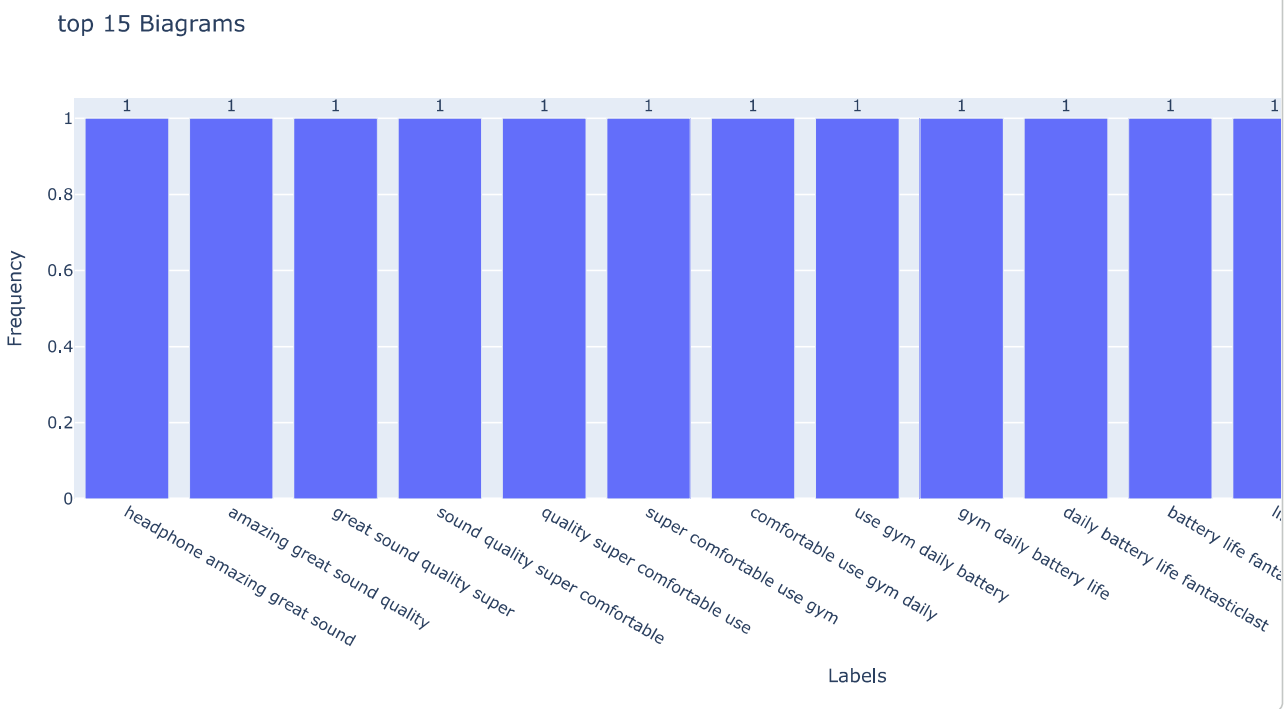
```
counts=[3,5,6,7,9,10,23,12,11,22,10,5,13,14,15]
```

```
# plotly bar chart
fig= go.Figure(data=
               [go.Bar(
                   x=labels,
                   y=counts,
                   text=counts,
                   textposition="outside")])




# update layout
fig.update_layout(height=550,
                  title="top 15 Biagrams",
                  xaxis_title= "Labels",
                  yaxis_title="Frequency")


fig.show()
```




```
# Creating function for N-gram analysis
def plot_top_ngrams_bar_chart(tokens,gram_n=4, top_n=15):
    try:
        ngram = list(ngrams(tokens, gram_n))
        ngram_counts= Counter(ngram).most_common(top_n)

        if not ngram_counts:
            raise ValueError("No n-gramsfound in the given  token list")

        labels=[]
        counts=[]
        for biagram , count in ngram_counts :
            labels.append(" ".join(biagram))
            counts.append(count)

        # plotly bar chart
        fig= go.Figure(data=
                       [go.Bar(
                           x=labels,
                           y=counts,
                           text=counts,
                           textposition="outside")])




        # update layout
        fig.update_layout(height=550,
                          title="top 15 Biagrams",
                          xaxis_title= "Labels",
                          yaxis_title="Frequency")


        fig.show()
    except Exception as e:
        print(f"An Error Occured: {e}")
```

```
plot_top_ngrams_bar_chart(cleaned_1)
```



EMOTION DETECTION.

```
!pip install transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.52.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from transformers) (3.18.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.32
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.21.1)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.5.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0.30.0->t
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>
Requirement already satisfied: hf-xet<2.0.0,>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0.30.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->transformers)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2.4.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2025
```

```
from transformers import pipeline
```

```
model_name= "nateraw/bert-base-uncased-emotion"

emotion_classifier= pipeline("text-classification", model=model_name, tokenizer=model_name, top_k=3)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:


The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.

config.json: 100%                                                 768/768 [00:00<00:00, 16.9kB/s]

pytorch_model.bin: 100%                                           438M/438M [00:04<00:00, 142MB/s]

model.safetensors: 100%                                           438M/438M [00:05<00:00, 81.8MB/s]

tokenizer_config.json: 100%                                       252/252 [00:00<00:00, 3.64kB/s]

vocab.txt: 100%                                                   232k/232k [00:00<00:00, 2.26MB/s]

special_tokens_map.json: 100%                                     112/112 [00:00<00:00, 1.80kB/s]

Device set to use cpu
```

```
text="I love India, I am proud Indian"
results= emotion_classifier(text)
results[0]
```

```
[{'label': 'joy', 'score': 0.9940181970596313},
 {'label': 'love', 'score': 0.0026790183037519455},
 {'label': 'anger', 'score': 0.0015130413230508566}]
```

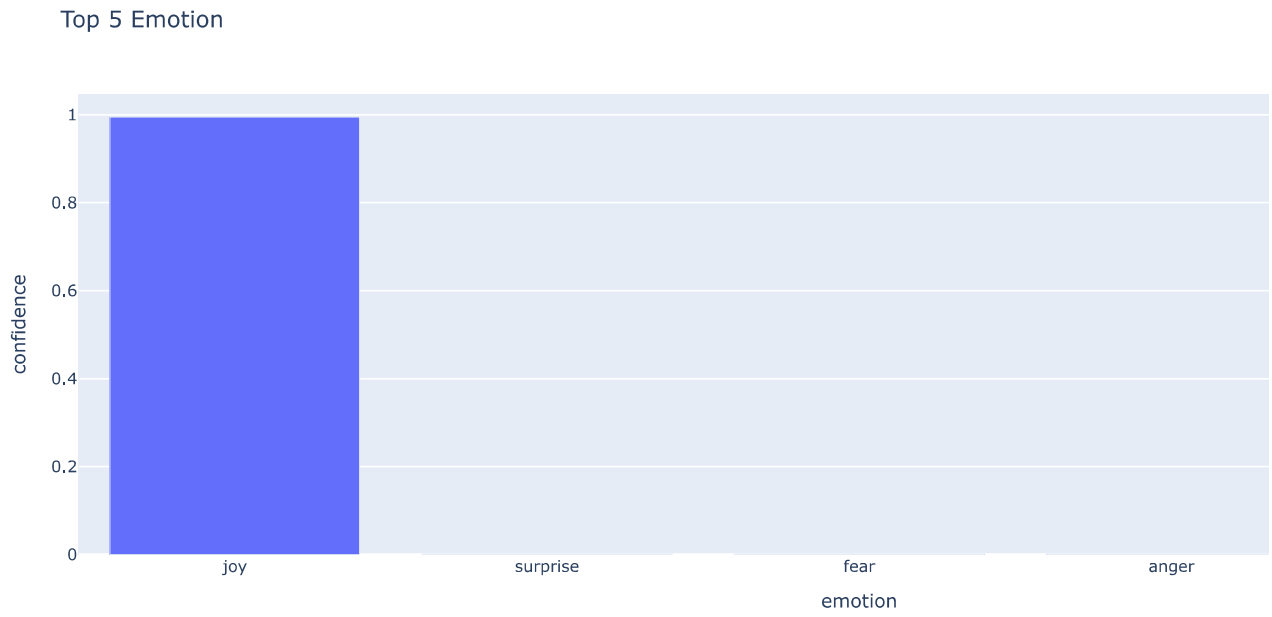
```
import plotly.express as px
# Creating function for emotion detection.
model_name= "nateraw/bert-base-uncased-emotion"
emotion_classifier= pipeline("text-classification", model=model_name, tokenizer=model_name, top_k=5)

def detect_emotion(text):
    try:
        # Performed emotion classification.
        results= emotion_classifier(text)
        emotion_data=[{"emotion":res["label"], "confidence":res["score"]} for res in results[0]]
        df= pd.DataFrame(emotion_data)

        # Highest Scoring emotion
        max_emotion_row= df.loc[df["confidence"].idxmax()]

        #plotly bar chart
        fig = px.bar(df, x="emotion", y="confidence", color="emotion", title=f"Top 5 Emotion")
        fig.update_layout(showlegend=False)

        #return full result
        return{
            "text":text,
            "detected_emotion": max_emotion_row["emotion"],
            "confidence":max_emotion_row["confidence"],
            "emotion_table": df,
            "plot":fig
        }
    except Exception  as e:
        return {"error": str(e)}
```

```
Device set to use cpu
```

```
# results
result1=detect_emotion("I am so excited  to start my new job")
result2= detect_emotion("I am feeling hopless and tired  of everything")

# extracting information from results
if "error" not in result1:
    print(f"Text: {result1['text']}")
    print("Emotion Table")
    print(result1["emotion_table"])
    print(f"Detected Emotion: {result1['detected_emotion']} with confidence: {result1['confidence']}")
    result1["plot"].show()
```

```
if "error" not in result1:
    print(f"Text: {result2['text']}")
    print("Emotion Table")
    print(result2["emotion_table"])
    print(f"Detected Emotion: {result2['detected_emotion']} with confidence: {result2['confidence']}")
    result1["plot"].show()
```

```
Text: I am so excited  to start my new job
Emotion Table
    emotion  confidence
0       joy    0.994770
1  surprise    0.001426
2      fear    0.001117
3     anger    0.001114
4      love    0.000831
Detected Emotion: joy with confidence: 0.9947699904441833
```




```
Text: I am feeling hopless and tired  of everything
Emotion Table
    emotion  confidence
0   sadness    0.987383
1      fear    0.005376
2     anger    0.003178
3       joy    0.002219
4  surprise    0.001269
Detected Emotion: sadness with confidence: 0.9873833060264587
```


```
text= """
There are moments in life when love feels like the most beautiful and tender thing in the world. When I look into your eyes, it

It’s the quiet moments, the stillness in your touch, that speak volumes. I can’t help but think of how far we’ve come, how many

But then, there are these moments, too, when the sadness creeps in. It’s as if I can feel the weight of time, the fleeting natu

Sometimes, the love we share feels so strong, so overwhelming, that it brings tears to my eyes. Tears not just because I am dee

But I won’t let that sadness define us. No, it only makes me hold on to you tighter, cherish every laugh, every touch, every sh
"""
```

```
text
```

```
'\nThere are moments in life when love feels like the most beautiful and tender thing in the world. When I look into your eyes,
it’s as if time stops for just a fleeting second, and in that moment, I see everything — all the joy, all the dreams we’ve shar
ed, and the quiet, beautiful promise of forever. I’ve never known a love like this, so pure, so real, and yet, there’s this dee
p ache that resides in my chest, a silent companion that reminds me that love, no matter how strong, is not immune to the fragi
lity of life.\n\nIt’s the quiet moments, the stillness in your touch, that speak volumes. I can’t help but think of how far w
e’ve come, how many nights we’ve spent talking about everything and nothing, laughing until our stomachs hurt, and holding each
```

```
result1= detect_emotion(text)
```

```
Token indices sequence length is longer than the specified maximum sequence length for this model (644 > 512). Running this sequ
```

```
Start coding or generate with AI.
```

```
# extracting information from results
if "error" not in result1:
    print(f"Text: {result1['text']}")
    print("Emotion Table")
    print(result1["emotion_table"])
    print(f"Detected Emotion: {result1['detected_emotion']} with confidence: {result1['confidence']}")
    result1["plot"].show()
else:
    print(result1)
```

```
{'error': 'The size of tensor a (644) must match the size of tensor b (512) at non-singleton dimension 1'}
```

CREATING CHUNKS.

```
nlp = spacy.load("en_core_web_sm")
doc= nlp(text)
```

```
for i in doc.sents:
    print(i)
    print("***")
```

```
There are moments in life when love feels like the most beautiful and tender thing in the world.
***
When I look into your eyes, it’s as if time stops for just a fleeting second, and in that moment, I see everything — all the joy
***
I’ve never known a love like this, so pure, so real, and yet, there’s this deep ache that resides in my chest, a silent companio


***
It’s the quiet moments, the stillness in your touch, that speak volumes.
***
I can’t help but think of how far we’ve come, how many nights we’ve spent talking about everything and nothing, laughing until o
***
There’s such beauty in the way our hearts have become so intertwined, so inseparable.
***
Every part of me loves you, not just in the bright moments, but even in the quiet, in the moments where words are unnecessary, w


***
But then, there are these moments, too, when the sadness creeps in.
***
It’s as if I can feel the weight of time, the fleeting nature of this precious thing we call life, pulling at my heart.
***
The thought of losing you, of one day having to walk this earth without the warmth of your hand in mine, fills me with a sadness
***
It’s not a sadness born of anything we’ve done, but rather the simple realization that nothing, no matter how perfect it seems,
***
And I wish, with all my heart, that I could hold on to this moment, this love, forever.
***
I wish that we could grow old together, not with fear or regret, but with the quiet knowing that every single day was enough bec


***
Sometimes, the love we share feels so strong, so overwhelming, that it brings tears to my eyes.
***
Tears not just because I am deeply moved by how much I care for you, but because I also know that love, as beautiful as it is, a
***
It’s a love that feels eternal in the moment, but the sadness reminds me that nothing stays the same forever, and that thought b
```

```
***
But I won’t let that sadness define us.
***
No, it only makes me hold on to you tighter, cherish every laugh, every touch, every shared silence.
***
Our love is a beautiful story, and even if it’s only ours for a time, it will be enough.
***
It will be more than enough because love, true love, is not measured by how long we have it, but by how deeply we live it.
***
And in every moment we share, there is nothing but love — love that fills me, that makes me whole, even as the sadness reminds m

***
```

```
for i in doc.sents:
    print(i.text)
    print("***")
```

```
There are moments in life when love feels like the most beautiful and tender thing in the world.
***
When I look into your eyes, it’s as if time stops for just a fleeting second, and in that moment, I see everything — all the joy
***
I’ve never known a love like this, so pure, so real, and yet, there’s this deep ache that resides in my chest, a silent companio


***
It’s the quiet moments, the stillness in your touch, that speak volumes.
***
I can’t help but think of how far we’ve come, how many nights we’ve spent talking about everything and nothing, laughing until o
***
There’s such beauty in the way our hearts have become so intertwined, so inseparable.
***
Every part of me loves you, not just in the bright moments, but even in the quiet, in the moments where words are unnecessary, w


***
But then, there are these moments, too, when the sadness creeps in.
***
It’s as if I can feel the weight of time, the fleeting nature of this precious thing we call life, pulling at my heart.
***
The thought of losing you, of one day having to walk this earth without the warmth of your hand in mine, fills me with a sadness
***
It’s not a sadness born of anything we’ve done, but rather the simple realization that nothing, no matter how perfect it seems,
***
And I wish, with all my heart, that I could hold on to this moment, this love, forever.
***
I wish that we could grow old together, not with fear or regret, but with the quiet knowing that every single day was enough bec


***
Sometimes, the love we share feels so strong, so overwhelming, that it brings tears to my eyes.
***
Tears not just because I am deeply moved by how much I care for you, but because I also know that love, as beautiful as it is, a
***
It’s a love that feels eternal in the moment, but the sadness reminds me that nothing stays the same forever, and that thought b


***
But I won’t let that sadness define us.
***
No, it only makes me hold on to you tighter, cherish every laugh, every touch, every shared silence.
***
Our love is a beautiful story, and even if it’s only ours for a time, it will be enough.
***
It will be more than enough because love, true love, is not measured by how long we have it, but by how deeply we live it.
***
And in every moment we share, there is nothing but love — love that fills me, that makes me whole, even as the sadness reminds m

***
```

```
# chunks out of sentence.
# storage to store chunks
# loop ----> chunks
# chunk---> dont cross extreme mark.  -----> chuunk = 0
# loop--> len(sentence) < benchmark
# chuunk+sentence  ----> chunk= n tokens
# chunks saturate.
```

```
# CREATING CHUNKS.
chunks=[]
current_chunk=""

for sent in doc.sents:
    sentence= sent.text.strip()
    if len(current_chunk)+len(sentence)<=500:
        current_chunk+=" "+sentence
    else:
        chunks.append(current_chunk.strip())
        current_chunk=sentence

if current_chunk:
    chunks.append(current_chunk.strip())
```

```
chunks[5]
```

```
# CREATING FUCNTIONS FOR THE CHUNKS
def split_into_chunks_spacy(text , max_length=500):
    nlp= spacy.load("en_core_web_sm")
    doc= nlp(text)
    chunks=[]
    current_chunk=""

    for sent in doc.sents:
        sentence= sent.text.strip()
        if len(current_chunk)+len(sentence)<=500:
            current_chunk+=" "+sentence
        else:
            chunks.append(current_chunk.strip())
            current_chunk=sentence

    if current_chunk:
        chunks.append(current_chunk.strip())
    return chunks
```

```
split_into_chunks_spacy(text)[3]
```

```
result= emotion_classifier("I love India , I am an proud Indian")
emotions= result[0]
```

```
emotions
```

```
result= emotion_classifier("my fried was weeping , he is going through tough times")
emotions_1= result[0]
```

```
emotions_1
```

```
result= emotion_classifier("my fried  is getting married , he is excited for it")
emotions_2= result[0]
```

```
emotions_2
```

```
emotionss={}
emotion_count={}
from collections import defaultdict
emotion_count= defaultdict(int)
```

```
emotionss
```

```
for i in emotions:
    label = i["label"]
    score= i["score"]
    emotionss[label]= emotionss.get(label ,0)+score
    emotion_count[label]+=1
```

```
emotionss
```

```
emotion_count
```

```
for i in emotions_1:
    label = i["label"]
    score= i["score"]
    emotionss[label]= emotionss.get(label ,0)+score
    emotion_count[label]+=1
```

```
emotionss
```

```
emotion_count
```

```
for i in emotions_2:
    label = i["label"]
    score= i["score"]
    emotionss[label]= emotionss.get(label ,0)+score
    emotion_count[label]+=1
```

```
emotionss.items()
```

```
emotion_count
```

```
dict(emotion_count)
```

```
# CREATING FUNCTION FOR EMOTIONS DETECTION
model_name= "nateraw/bert-base-uncased-emotion"
emotion_classifier= pipeline("text-classification", model=model_name, tokenizer=model_name, top_k=None)
from collections import defaultdict

def detect_emotions(text):
    chunks= split_into_chunks_spacy(text)
    emotion_totals={}
    emotion_count= {}
    emotion_count= defaultdict(int)

    for chunk in chunks:
        results= emotion_classifier(chunk)[0]
        for result in results:
            label= result["label"]
            score= result["score"]
            emotion_totals[label]= emotion_totals.get(label, 0)+score
            emotion_count[label]+=1

    emotion_counts= dict(emotion_count)

    emotion_averages= {label:emotion_totals[label]/emotion_counts[label] for label in emotion_totals}
    sorted_emotions= sorted(emotion_averages.items(), key= lambda x:x[1], reverse= True)
    top_5= sorted_emotions[:5]
    df= pd.DataFrame(top_5, columns=["Emotion", "Score"])
    return df
```

```
Device set to use cpu
```

```
detect_emotions("I am happy , as I a playing the sports that I lovr the most")
```

```
emotionss
```

```
emotion_count= dict(emotion_count)
emotion_count
```

```
detect_emotions
```

```
abc= {}
for label in emotionss:
```

```
    print(label)
    print(emotionss[label])
    print(emotion_count[label])
    print(emotionss[label]/emotion_count[label])
    abc[label]=emotionss[label]/emotion_count[label]
    print("***")
```

```
abcd= sorted(abc.items(), key= lambda x:x[1], reverse=True)
```

```
pd.DataFrame(abcd[:5], columns=["Emotions", "Score"])
```

```
result = detect_emotions("I am happy , as I a playing the sports that I lovr the most")
```

```
max_index= result["Score"].idxmax()
emotion= result.loc[max_index, "Emotion"]
score=  result.loc[max_index, "Score"]
print(f"Predicted Emotion :- {emotion}, with {score*100}% confidence")
fig= px.bar(result , x="Emotion", y="Score", color="Emotion")
fig.show()
```

```
score
```

SENTIMENTAL ANALYSIS

```
model_name= "cardiffnlp/twitter-roberta-base-sentiment"
sentiment_classifier= pipeline("sentiment-analysis", model= model_name,  tokenizer= model_name , return_all_scores=True)
```

```
config.json: 100%                                                   747/747 [00:00<00:00, 21.7kB/s]
pytorch_model.bin: 100%                                             499M/499M [00:17<00:00, 29.6MB/s]
model.safetensors: 100%                                             499M/499M [00:14<00:00, 26.4MB/s]
vocab.json: 100%                                                    899k/899k [00:00<00:00, 11.1MB/s]
merges.txt: 100%                                                    456k/456k [00:00<00:00, 4.25MB/s]
special_tokens_map.json: 100%                                       150/150 [00:00<00:00, 2.59kB/s]
Device set to use cpu
/usr/local/lib/python3.11/dist-packages/transformers/pipelines/text_classification.py:106: UserWarning:

`return_all_scores` is now deprecated,  if want a similar functionality use `top_k=None` instead of `return_all_scores=True` or
```

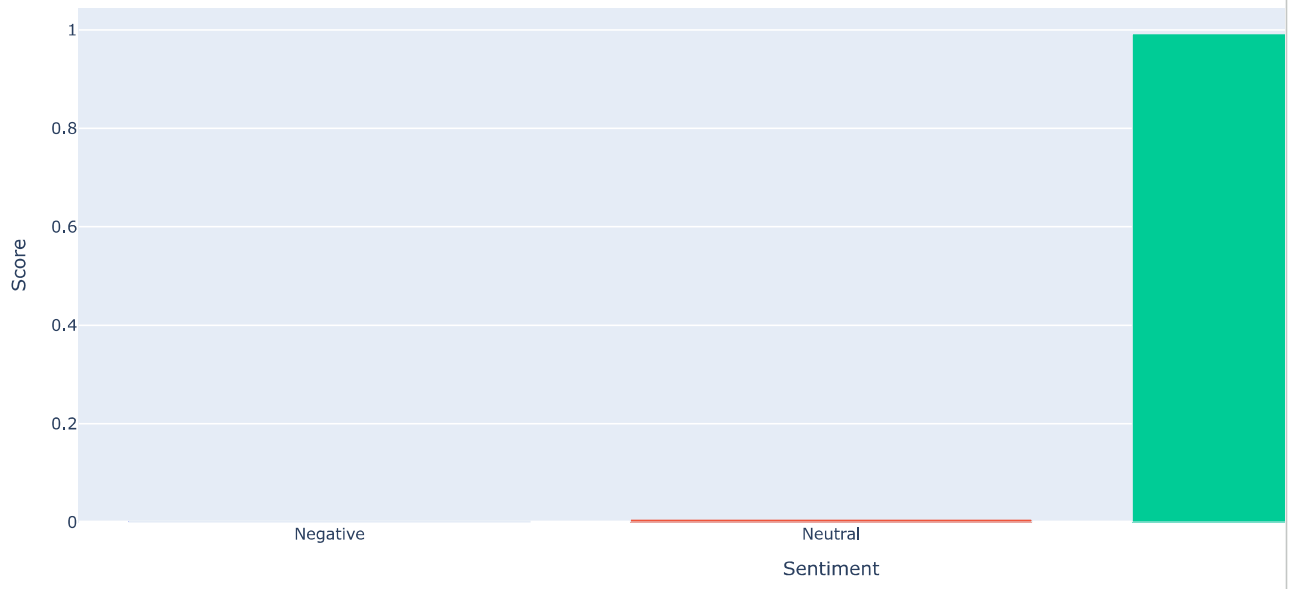
```
sentiment_classifier("I love RCB")
```

```
# CREAING FUNCTION FOR SNETIMENT ANALYSIS
def detect_overall_sentiment_avg(text):
    try:
        sentiment_labels= {
            "LABEL_0":"Negative",
            "LABEL_1":"Neutral",
            "LABEL_2":"Positive"
        }
        chunks= split_into_chunks_spacy(text)
        score_total = {"Negative":0.0, "Neutral":0.0,"Positive":0.0}
        chunk_count= len(chunks)

        for chunk in chunks:
            results= sentiment_classifier(chunk)[0]
            for res in results:
                label= sentiment_labels[res["label"]]
                score_total[label]+=res["score"]

        avg_score= {}
        for label in score_total:
           avg_score[label]=score_total[label]/chunk_count
        overall_sentiment= max(avg_score, key= avg_score.get)
        return{
           "overall_sentiment":overall_sentiment,
           "average_scores":avg_score
```

```
            }

    except Exception as e:
        return {f"error {e}"}
```

```
result= detect_overall_sentiment_avg("I am so excited to start my new job")
if "error" in result:
    print(f"Error: {result['error']}")
else:
    print(f"Overall Sentiment :- {result['overall_sentiment']}, with score {max(result['average_scores'].values())}")
    print("Average Scores:", pd.DataFrame(result['average_scores'].items(), columns=['Sentiment','Score']))
    df= pd.DataFrame(result["average_scores"].items(), columns=["Sentiment","Score"])
    fig= px.bar(df, x="Sentiment", y="Score", color="Sentiment")
    fig.show()
```

```
Overall Sentiment :- Positive, with score 0.9918146729469299
Average Scores:   Sentiment     Score
0  Negative  0.001962
1   Neutral  0.006223
2  Positive  0.991815
```



TONE OF SPEECH DETECTION

```
# LOAD MY MODEL
classifier= pipeline("zero-shot-classification",model="facebook/bart-large-mnli")


labels = [
    "factual",
    "opinion",
    "question",
    "command",
    "emotion",
    "personal experience",
    "suggestion",
    "story",
    "prediction",
    "warning",
    "instruction",
    "definition",
    "narrative",
    "news",
    "argument"
```

```
]


def classify_custom(text):
  result = classifier(text, candidate_labels=labels)
  return{
      "text":text,
      "Predicted_Category":result["labels"][0],
      "score":result["scores"][0],
      "all_categories":list(zip(result["labels"], result["scores"]))

  }
```

```
Device set to use cpu
```

```
Start coding or generate with AI.
```

```
example_texts = [
"India got independence in 1947.",
"I think the new update is terrible.",
"What time does the train arrive?",
"Please restart your computer.",
"I'm so excited about the concert!",
"Last year, I traveled to Japan alone.",
"You should drink more water daily.",
"The Earth revolves around the Sun.",
"Be careful while crossing the road.",
"They argued about politics for hours.",
"Once upon a time, a lion lived in a forest.",
"Photosynthesis is the process by which plants make food."
]
```

```
for text in example_texts:
  output= classify_custom(text)
  print(f"Text; {output['text']}")
  print(f"Predicted  : {output['Predicted_Category']}, score : {output['score']}")
  print("Top 3 Categories.")
  for label , score in output["all_categories"][1:4]:
    print(f"Label :- {label}, Score:- {score}")
  print( )
```

```
Text; India got independence in 1947.
Predicted  : news, score : 0.12362156063318253
Top 3 Categories.
Label :- question, Score:- 0.12255702912807465
Label :- factual, Score:- 0.11816497892141342
Label :- argument, Score:- 0.0854959487915039

Text; I think the new update is terrible.
Predicted  : opinion, score : 0.5713661909103394
Top 3 Categories.
Label :- news, Score:- 0.19079820811748505
Label :- warning, Score:- 0.052877359092235565
Label :- question, Score:- 0.041321542114019394

Text; What time does the train arrive?
Predicted  : question, score : 0.4811919033527374
Top 3 Categories.
Label :- prediction, Score:- 0.13178861141204834
Label :- suggestion, Score:- 0.11841640621423721
Label :- instruction, Score:- 0.07243102788925171

Text; Please restart your computer.
Predicted  : instruction, score : 0.2623298764228821
Top 3 Categories.
Label :- warning, Score:- 0.2569589912891388
Label :- suggestion, Score:- 0.20582665503025055
Label :- command, Score:- 0.11124356091022491

Text; I'm so excited about the concert!
Predicted  : emotion, score : 0.5371499061584473
Top 3 Categories.
Label :- news, Score:- 0.10945979505777359
Label :- opinion, Score:- 0.06264781206846237
Label :- question, Score:- 0.04097164794802666

Text; Last year, I traveled to Japan alone.
```

```
Predicted  : personal experience, score : 0.5674536228179932
Top 3 Categories.
Label :- question, Score:- 0.06686950474977493
Label :- suggestion, Score:- 0.046591825783252716
Label :- factual, Score:- 0.04639673978090286

Text; You should drink more water daily.
Predicted  : suggestion, score : 0.33540526032447815
Top 3 Categories.
Label :- instruction, Score:- 0.1504247933626175
Label :- command, Score:- 0.12078306823968887
Label :- factual, Score:- 0.07930021733045578

Text; The Earth revolves around the Sun.
Predicted  : factual, score : 0.6235148310661316
Top 3 Categories.
Label :- definition, Score:- 0.07045373320579529
Label :- question, Score:- 0.05960961431264877
Label :- warning, Score:- 0.04448428004980087

Text; Be careful while crossing the road.
```

```
# Ronaldo was injured in first half. desite that he retured in the second half and scored an goal..

# find summary of your text , then  classify the tone of speech.

# sliding window
# 500 tokens , 100 token overlap
# intotal 2500 tokens
# chunk1 - [0,500]
# chunk2 - [100,600]
# chunk3 - [200,700]

# sentences. ---> spacy  (done)  list of setences
# chunks ----> overlap chunks



# list = [s1,s2,s3,s4,s5,s6,s7,s8,s9,s10]  len=10


def chunk_text(sentences, chunk_size=5, overlap=2):
  chunks=[]
  start=0
  while start < len(sentences):
    end = start+chunk_size
    chunk= sentences[start:end]
    chunks.append(" ".chunk)

    if end>=len(sentences):
      break

    start+=chunk_size-overlap
    return chunks


def classify_customs(text):
  nlp = spacy.load("en_core_web_sm")
  doc= nlp(text)
  sentences=[sent.text.strip() for sents in doc.sents]
  if len(sentences)<=5:
    result= classifier(text, candidate_labels=labels)
    return{
      "text":text,
      "Predicted_Category":result["labels"][0],
      "score":result["scores"][0],
      "all_categories":list(zip(result["labels"], result["scores"]))
     }
  else:
    chunks = chunk_text(sentences, chunk_size=5, overlap=2)

    # forming dictionary to calculate aggregrates
    aggregated_scores = {label: 0.0 for label in labels }

    for chunk in chunks:
      result = classifier(chunk, candidate_label=labels)
      for label , score in zip(result['labels'], result['score']):
        aggregated_scores[label]+=score
```

```
    # calculate aggregrates
    num_chunks= len(chunks)
    average_scores= {label:score/num_chunks for label , score in aggregated.scores.items()}

    # Sort Top Category
    sorted_category = sorted(average_scores.items(), key= lambda x:x[1], reverse=True)

    return{
      "text":text,
      "Predicted_Category":sorted_category[0][0],
      "score":sorted_category[0][1],
      "all_categories": sorted_category
    }
```

```
dic= {"a":10,"b":20 ,"c":8, "d":23}
```

```
sorted(dic.items(),key= lambda x:x[1], reverse=True)[0][1]
```

```
23
```

```
text= """
Climate Change and Technology: An Informative Report

Climate change represents one of the most pressing global challenges of the 21st century. Driven largely by the emission of gre

Historically, industrial revolutions and technological advancements have been linked to increased carbon emissions and environm

One of the most prominent ways technology is addressing climate change is through the development of renewable energy. Solar pa

Agriculture is a significant contributor to greenhouse gas emissions, particularly methane and nitrous oxide. Technology is hel

Another technological response to climate change involves carbon capture and storage (CCS). These systems extract CO2 from the

The transportation sector is undergoing a technological revolution in response to climate concerns. Electric vehicles (EVs), pc

From renewable energy to smart agriculture, green transportation to carbon capture, technology is playing a pivotal role in sha
```

```
len(text)
```

```
4238
```

```
classify_custom(text)
```

```
{'text': '\nClimate Change and Technology: An Informative Report\n\nClimate change represents one of the most pressing global challenges of the 21st century. Driven largely by the emission of greenhouse gases through industrial activities, deforestation, and fossil fuel consumption, the Earth’s temperature is rising at an unprecedented rate. This warming is leading to erratic weather patterns, rising sea levels, and a surge in climate-related natural disasters. In the face of such challenges, technology emerges as both a contributor to the problem and a powerful tool for the solution. This report explores how technology is being harnessed to combat climate change, reduce emissions, adapt to new environmental realities, and build a sustainable future.\n\nHistorically, industrial revolutions and technological advancements have been linked to increased carbon emissions and environmental degradation. The combustion engine, coal-based power generation, and mass production all significantly contributed to the current levels of atmospheric CO2. Technological development, when pursued without consideration for ecological balance, has caused deforestation, pollution, and over-reliance on non-renewable resources. However, recognizing this history is essential to better redirect innovation toward green and sustainable alternatives.\n\nOne of the most prominent ways technology is addressing climate change is through the development of renewable energy. Solar panels, wind turbines, and hydroelectric systems now provide cleaner alternatives to coal and oil. Technological advances have made these sources more efficient, cost-effective, and scalable. Smart grids powered by machine learning optimize electricity distribution, reduce waste, and integrate renewable energy into national grids more seamlessly. Battery storage technology has also seen rapid innovation, allowing excess energy to be stored and used when renewable sources are not generating power.\n\nAgriculture is a significant contributor to greenhouse gas emissions, particularly methane and nitrous oxide. Technology is helping transform this sector through smart agriculture, which uses sensors, satellite data, and AI to monitor crop health, optimize irrigation, and reduce fertilizer use. Vertical farming and lab-grown meat are examples of how technology is creating sustainable food systems that require less land and emit fewer greenhouse gases. These innovations aim to increase food production efficiency while minimizing environmental impact.\n\nAnother technological response to climate change involves carbon capture and storage (CCS). These systems extract CO2 from the air or directly from industrial emissions and store it underground or repurpose it for industrial use. While CCS is still in its early stages, it holds promise as a means to directly remove carbon from the atmosphere. Geoengineering, including solar radiation management and ocean fertilization, is more controversial but continues to be researched as a potential last resort. Ethical considerations and long-term risks must be thoroughly evaluated before large-scale deployment.\n\nThe transportation sector is undergoing a technological revolution in response to climate concerns. Electric vehicles (EVs), powered by renewable electricity, are replacing traditional gasoline and diesel vehicles. Advancements in battery technology have significantly extended the range and reduced the cost of EVs. Additionally, public transportation systems are integrating clean energy buses and trains. Innovations in mobility-as-a-service (MaaS), carpooling platforms, and smart traffic systems also contribute to reduced emissions and improved urban planning.\n\nFrom renewable energy to smart agriculture, green transportation to carbon capture, technology is playing a pivotal role in shaping a more sustainable and resilient future. However, innovation must be guided by ethical considerations,
```

```
inclusivity, and sustainability to ensure that technological progress does not come at the cost of environmental degradation.
With global collaboration, policy support, and continued investment, technology can be harnessed to not only address climate
change but to build a healthier planet for future generations.',
 'Predicted_Category': 'command',
 'score': 0.14209191501140594,
 'all_categories': [('command', 0.14209191501140594),
  ('suggestion', 0.11093473434448242),
  ('opinion', 0.10708387941122055),
  ('definition', 0.08572722226381302),
  ('narrative', 0.07826205343008041),
  ('argument', 0.07711680233478546),
  ('instruction', 0.0676606297492981),
  ('emotion', 0.0653451681137085),
  ('story', 0.06236187741160393),
  ('news', 0.055850863456726074),
  ('prediction', 0.03636352717876434),
  ('factual', 0.03593968227505684),
  ('warning', 0.034615498036146164),
  ('question', 0.03126619756221771),
  ('personal experience', 0.009379901923239231)]}
```

```
# SUMMARY GENERATION.
```

```
summarizer= pipeline("summarization",model="facebook/bart-large-cnn")
```

```
config.json: 100%                                                  1.58k/1.58k [00:00<00:00, 102kB/s]
model.safetensors: 100%                                            1.63G/1.63G [01:02<00:00, 40.0MB/s]
generation_config.json: 100%                                       363/363 [00:00<00:00, 16.7kB/s]
vocab.json: 100%                                                   899k/899k [00:00<00:00, 7.13MB/s]
merges.txt: 100%                                                   456k/456k [00:00<00:00, 14.6MB/s]
tokenizer.json: 100%                                               1.36M/1.36M [00:00<00:00, 18.8MB/s]
Device set to use cpu
```

```
short_text = """
Artificial intelligence is transforming industries worldwide. From healthcare to finance, AI technologies are improving efficie
"""
```

```
short_text
```

```
'\nArtificial intelligence is transforming industries worldwide. From healthcare to finance, AI technologies are improving effi
ciency and enabling smarter decisions. Businesses that adopt AI gain a competitive advantage by automating routine tasks and un
covering valuable insights from data. However, ethical considerations and responsible AI development remain critical to ensure
```

```
len(short_text)
```

```
418
```

```
max_summary_length = int(len(short_text)*0.8)
min_summary_length= int(len(short_text)*0.3)
```

```
summary= summarizer(short_text, max_length=max_summary_length, min_length=min_summary_length, do_sample=False)
```

```
Your max_length is set to 334, but your input_length is only 68. Since this is a summarization task, where outputs shorter than
```

```
summary[0]["summary_text"]
```

```
'Artificial intelligence is transforming industries worldwide. From healthcare to finance, AI technologies are improving effici
ency and enabling smarter decisions. Businesses that adopt AI gain a competitive advantage by automating routine tasks and unco
vering valuable insights from data. However, ethical considerations and responsible AI development remain critical to ensure te
chnology benefits all people fairly. For more information, visit artificialintelligence.com. For confidential support call the
```

```
print('Artificial intelligence is transforming industries worldwide. From healthcare to finance, AI technologies are improving
```

```
Artificial intelligence is transforming industries worldwide. From healthcare to finance, AI technologies are improving efficien
 Businesses that adopt AI gain a competitive advantage by automating routine tasks.
```

```
# chunks---> 500
# chunks --> summary  ---> 500 summaries (DONE)  -- 10,000-- 20
# summarize the summary of chunks again.
```

```
def summarize_large_texts(text):
  summarizer= pipeline("summarization",model="facebook/bart-large-cnn")
  chunks= split_into_chunks_spacy(text, max_length=500)

  # summarize each chunk
  chunk_summaries=[]
  for chunk in chunks:
    input_length= len(chunk.split())
    max_summary_length = int(input_length*0.8)
    min_summary_length= int(input_length*0.3)
    summary = summarizer(chunk, max_length=max_summary_length, min_length=min_summary_length, do_sample=False)[0]["summary_text
    chunk_summaries.append(summary)

  # Forming text out of all the summaries in the variable chunk_summaries
  combined_summary_text= " ".join(chunk_summaries)

  # finding summary of combined summary text.
  input_length= len(combined_summary_text.split())
  max_summary_length = int(input_length*0.8)
  min_summary_length= int(input_length*0.3)
  final_summary = summarizer(chunk, max_length=max_summary_length, min_length=min_summary_length, do_sample=False)[0]["summary_

  return final_summary
```

```
 summarize_large_texts(short_text)
```

```
Device set to use cpu
'Artificial intelligence is transforming industries worldwide. Businesses that adopt AI gain a competitive advantage by automat
```

```
long_text = """
Artificial intelligence (AI) has become one of the most transformative technologies of the 21st century. It is now impacting vi

In healthcare, AI-powered tools assist doctors and medical professionals with diagnosis, personalized treatment recommendations

Financial institutions are leveraging AI for fraud detection, credit scoring, personalized banking, and algorithmic trading, al

The transportation industry is undergoing a significant transformation thanks to AI. The rise of autonomous vehicles promises t

Education is also being reshaped through AI-driven personalized learning platforms that adapt content and pace based on individ

The entertainment industry leverages AI to create personalized content recommendations on streaming platforms, improving user e

Despite these advances, the rapid development of AI raises important ethical and societal questions that require careful consid

Ensuring responsible AI development requires transparency, fairness, and accountability from developers, companies, and policym

As AI continues to evolve rapidly, collaboration between technology experts, industry leaders, governments, academia, and civil

In summary, AI stands as a powerful tool with immense possibilities, poised to reshape our world in profound ways. However, it
"""
```