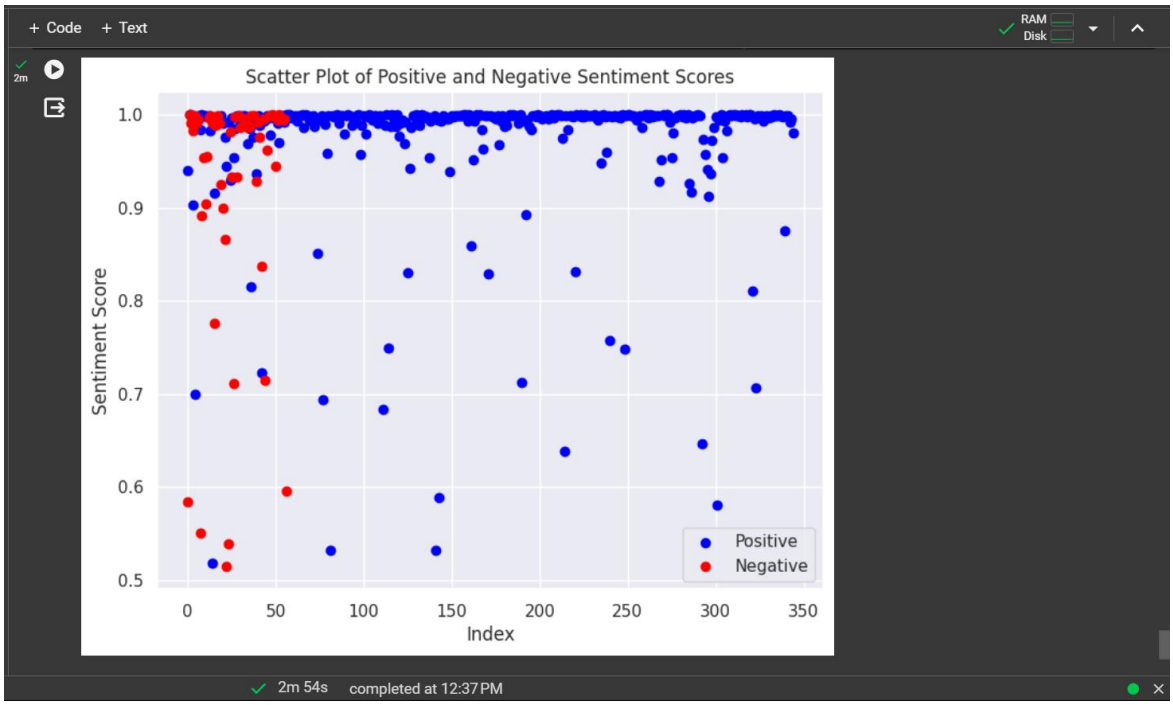
A PROJECT BASED ON SENTIMENT ANLYSIS FOR INDIAN BUDGET 2024 BY MEDICAL PROFESSIONALS

ABSTRACT :

After the budget 2024 has been announced we notices the reaction of different professional group , one of which is the medical professionals,
So just to capture what they have to say about it , I have scrapped there reaction from India Today and used various NLP features over it .
But as it can seen , they view the new budget in a positive light.
I am stating the conclusion first and showing the process next, note that this data is not very accurate as it has been done on a very small scale with a pre-trained model from hugging face, and can also contain some bias

CONCLUSION :



MEDICAL PROFESSIONALS VIEW INDIAN BUDGET 2024 AS A GOOD SIGN OF GROWTH IN VARIOUS SECTORS REGARDING MEDICAL FIELD AS WE CAN SEE MOST OF THE SENTIMENT RESPONSE WERE POSITIVE ON SCALE OF 0-1 , AND LESS NEGATIVE ON THE SAME SCALE OF 0-1.

PROCESS :

```
+ Code   + Text

[ ] pip install transformers

Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.35.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.13.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in /usr/local/lib/python3.10/dist-packages (from tr
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.23
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.19,>=0.14 in /usr/local/lib/python3.10/dist-packages (from transfor
Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hu
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from hugg
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from reques
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transfor
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->tr
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->tr

24s [1] from transformers import pipeline
        classifier = pipeline('sentiment-analysis',model="DistilBert-base-uncased-finetuned-sst-2-english")
        #classifier = pipeline('sentiment-analysis',model="xyz")
        # default model is DistilBert-base-uncased-finetuned-sst-2-english
        #DistilBert-base  - model

✓ 2m 54s  completed at 12:37 PM
```

```
24s # default model is DistilBert-base-uncased-finetuned-sst-2-english
    #DistilBert-base  - model
    #uncased - small case
    # sst-2 eng dataset type

/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: 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
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.
  warnings.warn(
config.json: 100%  629/629 [00:00<00:00, 5.96kB/s]
model.safetensors: 100%  268M/268M [00:03<00:00, 86.0MB/s]
tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 785B/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 1.84MB/s]

1s [2] classifier("i love coding")

[{'label': 'POSITIVE', 'score': 0.9996923208236694}]

0s [3] classifier("i love coding but I hate debugging")

✓ 2m 54s  completed at 12:37 PM
```

```
[{'label': 'POSITIVE', 'score': 0.9996923208236694}]
```

```
[3] classifier("i love coding but I hate debugging")
```

```
[{'label': 'POSITIVE', 'score': 0.5278215408325195}]
```

```
[12] a=classifier("i hate cricket")
     print(a[0]['label'])
     classifier("i hate cricket")
```

```
NEGATIVE
[{'label': 'NEGATIVE', 'score': 0.9991239905357361}]
```

```
from transformers import AutoTokenizer, TFAutoModelForSequenceClassification
#to tokenize , convert to token(on internet numbers)
```

```
[16] model_name = "DistilBert-base-uncased-finetuned-sst-2-english"
     # This model only exists in PyTorch, so we use the `from_pt` flag to import that model in TensorFlow.
     model = TFAutoModelForSequenceClassification.from_pretrained(model_name, from_pt=True)
     tokenizer = AutoTokenizer.from_pretrained(model_name)
     classifier = pipeline('sentiment-analysis', model=model, tokenizer=tokenizer)
```

```
pytorch_model.bin: 100% [=========================] 268M/268M [00:02<00:00, 109MB/s]
```

2m 54s completed at 12:37 PM

```
[16] All PyTorch model weights were used when initializing TFDistilBertForSequenceClassification.

     All the weights of TFDistilBertForSequenceClassification were initialized from the PyTorch model.
     If your task is similar to the task the model of the checkpoint was trained on, you can already use TFDistilBer
```

```
[ ] '''import requests
    from bs4 import BeautifulSoup

    websites = {
        "The Hindu": "https://www.thehindu.com/business/budget/budget-2024-live-updates-nirmala-sitharaman-present
        "Hindustan Times": "https://www.hindustantimes.com/india-news/parliament-budget-session-live-updates-rajya
        "Times of India": "https://timesofindia.indiatimes.com/business/budget",
    }'''
```

```
import urllib.request
from bs4 import BeautifulSoup

# here we have to pass url and path
# (where you want to save your text file)
urllib.request.urlretrieve("https://indiamedtoday.com/2024-budget-reactions/",
            "text_file.txt")

file = open("text_file.txt", "r")
contents = file.read()
```

2m 54s completed at 12:37 PM

```python
import urllib.request
from bs4 import BeautifulSoup

# here we have to pass url and path
# (where you want to save your text file)
urllib.request.urlretrieve("https://indiamedtoday.com/2024-budget-reactions/",
            "text_file.txt")

file = open("text_file.txt", "r")
contents = file.read()
soup = BeautifulSoup(contents, 'html.parser')

f = open("test1.txt", "w")

# traverse paragraphs from soup
for data in soup.find_all("p"):
  sum = data.get_text()
  f.writelines(sum)

f.close()
```

```python
# Import necessary libraries
import requests
```

2m 54s completed at 12:37 PM

```python
# Import necessary libraries
import requests
from bs4 import BeautifulSoup
import re
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

# Function to scrape a website and return its text content
def scrape_website(url):
    response = requests.get(url)
    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
        # Extract text from the soup
        return soup.get_text(separator=' ')#soup.find_all('p').
    else:
        print(f"Failed to retrieve content from {url}")
        return None
'''
# Function to check if text contains business-related content using regex
def has_business_content(text):
    # Define a simple regex pattern for demonstration
    business_pattern = r'\b(business|company|corporation|finance)\b'
    return bool(re.search(business_pattern, text, re.IGNORECASE))
'''
# Example usage
url = 'https://www.cnbc.com/finance/'  # Replace with the URL of the website you want to scrape
webpage_text = scrape_website(url)
```

2m 54s completed at 12:37 PM

Main Section

```python
import requests
from bs4 import BeautifulSoup
import re
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize, sent_tokenize

webpage_text = ""
paragraphs = soup.find_all('p')
for paragraph in paragraphs:
    webpage_text += paragraph.text

lines = webpage_text.split('.')

# Open the file in write mode
with open("output.txt", "w") as f:
    for line in lines:
        # Lowercase all words and remove special characters
        processed_line = re.sub(r"[^a-z0-9\s]", "", line.lower())  # Remove all non-alphanumeric characters
        f.write(processed_line + "\n")
        print(processed_line + "\n")
```

```
union finance minister nirmala sitaraman in her interim budget 202425 has emphasised the need for vaccinating
```

2m 54s completed at 12:37 PM

```python
        f.write(processed_line + "\n")
        print(processed_line + "\n")
```

```
union finance minister nirmala sitaraman in her interim budget 202425 has emphasised the need for vaccinating

 it is good to see the increasing concern of the government in addressing in growing menace

 this age range is perfectly suitable because it offers the best protection before girls become sexually acti

 i believe the awareness about safety of hpv vaccine is much needed to be spread and by this initiative of go

 looking forward for more such initiatives by government to treat other kinds of cancer as well

   having worked across diverse healthcare systems i can see that india is taking important steps towards bui

 while there are still gaps in certain areas the union budgets dedicated focus on consolidating maternal and

 additionally the announcement encouraging vaccination for girls to prevent cervical cancer is a proactive st

 medix global remains dedicated to collaborative efforts for a future defined by personalised care inclusivit

 i appreciate the governments push for a comprehensive health development for women in this budget

 the emphasis on promoting cervical cancer vaccination for girls aged 914 is a pivotal step towards raising a

 the budget announced other initiatives such as comprehensive programs for maternal and child healthcare sche

 these steps will help in improving overall national health indicators in future
```

2m 54s completed at 12:37 PM

```
to enable wider access of preventive care and health cover there is increased focus on vaccination for cervica

ne consolidation of maternal and child healthcare schemes into a comprehensive program alongside the expedited

these measures coupled with the intensified efforts of mission indradhanush signify a progressive leap towards

mission indradhanush is a critical health mission of the government of india aimed at achieving more than 90 f

ne missions goal complements the comprehensive healthcare approach outlined in budget 2024 reflecting the gove
```

```
import requests
from bs4 import BeautifulSoup
import re
from transformers import pipeline
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize, sent_tokenize
import csv

def score_and_write(line, writer):
    classifier = pipeline('sentiment-analysis')
    results = classifier(line)
    if results[0]['label']=='NEGATIVE' and results[0]['score'] > 0.7:
        writer.writerow(["bad", line])
    elif results[0]['label']=='POSITIVE' and results[0]['score'] > 0.7:
        writer.writerow(["good", line])

tion in the Edit menu ✕ r.writerow(["neutral", line])
```

2m 54s completed at 12:37 PM

```
    elif results[0]['label']=='POSITIVE' and results[0]['score'] > 0.7:
        writer.writerow(["good", line])
    else:
        writer.writerow(["neutral", line])

webpage_text = ""
paragraphs = soup.find_all('p')
for paragraph in paragraphs:
    webpage_text += paragraph.text

lines = webpage_text.split('.')

# Open the file in write mode and create a CSV writer
with open("output.csv", "w", newline="") as f:
    writer = csv.writer(f)
    # Write CSV header
    writer.writerow(["sentiment", "content"])

    # Process and write lines with scoring
    for line in lines:
        processed_line = re.sub(r"[^a-z0-9\s]", "", line.lower())
        score_and_write(processed_line, writer)

print("Processed content written to output.csv")
```

2m 54s completed at 12:37 PM

```
plied, defaulted to distilbert-base-uncased-finetuned-sst-2-english and revision af0f99b (https://huggingface.co
 without specifying a model name and revision in production is not recommended.
629/629 [00:00<00:00, 11.3kB/s]
00% 268M/268M [00:02<00:00, 77.9MB/s]
: 100% 48.0/48.0 [00:00<00:00, 1.01kB/s]
232k/232k [00:00<00:00, 2.27MB/s]
```

```
[34] #classifier = pipeline('sentiment-analysis')
     results = classifier("i  am sorry to say i am thoroughly disappointed with the interim budget")
     print(results)
```

```
[{'label': 'NEGATIVE', 'score': 0.9996238946914673}]
```

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

sns.set()  #  Seaborn style plt got discontinues since specific version

df = pd.read_csv('output.csv', encoding='ISO-8859-1')
print(df.head())
```

```
  sentiment                                            content
0      good  union finance minister nirmala sitaraman in he...
1      good   it is good to see the increasing concern of t...
2      good   this age range is perfectly suitable because ...
3      good   i believe the awareness about safety of hpv v...
4      good   looking forward for more such initiatives by ...
```

```
[37] df.isna().sum()
```

2m 54s completed at 12:37 PM

```
[37] df.isna().sum()
     df['sentiment'].value_counts()
     print(df.columns)
```

```
Index(['sentiment', 'content'], dtype='object')
```

```
[39] y=df['sentiment'].values
     y.shape
```

```
(400,)
```

```
[40] x=df['content'].values
     x.shape
```

```
(400,)
```

```
from sklearn.model_selection import train_test_split
```

```
(x_train,x_test,y_train,y_test)=train_test_split(x,y,test_size=0.4)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

2m 54s completed at 12:37 PM

```
df1=pd.DataFrame(x_train)
df1=df1.rename(columns={0:'news'})
df2=pd.DataFrame(y_train)
df2=df2.rename(columns={0:'sentiment'})
df_train=pd.concat([df1,df2],axis=1)
df_train.head()
```

	news	sentiment
0	Â to address the dearth of doctors in india th...	good
1	further the decision to expand medical educat...	good
2	however we feel if customs duty on medical de...	good
3	bringing maternal and child healthcare schemes...	good
4	these measures will not only enhance healthca...	good

Categorical distributions

```
[44] df3=pd.DataFrame(x_test)
     df3=df3.rename(columns={0:'news'})
     df4=pd.DataFrame(y_test)
```

2m 54s completed at 12:37 PM

```
df3=pd.DataFrame(x_test)
df3=df3.rename(columns={0:'news'})
df4=pd.DataFrame(y_test)
df4=df2.rename(columns={0:'sentiment'})
df_test=pd.concat([df3,df4],axis=1)
df_test.head()
```

	news	sentiment
0	Â Â Â the governments focus on nation first ac...	good
1	64 per cent	good
2	while the customs duty has not gone down whic...	good
3	around 65 per cent of our population still ac...	good
4	additionally the emphasis on innovation and t...	good

Categorical distributions

```
#removing punctuations
#library that contains punctuation
```

2m 54s completed at 12:37 PM

```python
#removing punctuations
#library that contains punctuation
import string
string.punctuation
#defining the function to remove punctuation
def remove_punctuation(text):
  if(type(text)==float):
    return text
  ans=""
  for i in text:
    if i not in string.punctuation:
      ans+=i
  return ans
#storing the puntuation free text in a new column called clean_msg
df_train['news']= df_train['news'].apply(lambda x:remove_punctuation(x))
df_test['news']= df_test['news'].apply(lambda x:remove_punctuation(x))
df_train.head()
#punctuations are removed from news column in train dataset
```

	news	sentiment
0	Â to address the dearth of doctors in india th...	good
1	further the decision to expand medical educat...	good

2m 54s completed at 12:37 PM

[45]

1	further the decision to expand medical educat...	good
2	however we feel if customs duty on medical de...	good
3	bringing maternal and child healthcare schemes...	good
4	these measures will not only enhance healthca...	good

```python
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

+ Code + Text

[47]

```python
#method to generate n-grams:
#params:
#text-the text for which we have to generate n-grams
#ngram-number of grams to be generated from the text(1,2,3,4 etc., default value=1)
def generate_N_grams(text,ngram=1):
  words=[word for word in text.split(" ") if word not in set(stopwords.words('english'))]
  print("Sentence after removing stopwords:",words)
  temp=zip(*[words[i:] for i in range(0,ngram)])
  ans=[' '.join(ngram) for ngram in temp]
  return ans
```

2m 54s completed at 12:37 PM

```
[47]  words=[word for word in text.split(" ") if word not in set(stopwords.words('english'))]
      print("Sentence after removing stopwords:",words)
      temp=zip(*[words[i:] for i in range(0,ngram)])
      ans=[' '.join(ngram) for ngram in temp]
      return ans
```

```
[48] #sample!
     generate_N_grams("The sun rises in the east",2)
```

```
Sentence after removing stopwords: ['The', 'sun', 'rises', 'east']
['The sun', 'sun rises', 'rises east']
```

```
from collections import defaultdict
positiveValues=defaultdict(int)
negativeValues=defaultdict(int)
neutralValues=defaultdict(int)
#get the count of every word in both the columns of df_train and df_test dataframes
#get the count of every word in both the columns of df_train and df_test dataframes where sentiment="positive"
for text in df_train[df_train.sentiment=="good"].news:
  for word in generate_N_grams(text):
    positiveValues[word]+=1
#get the count of every word in both the columns of df_train and df_test dataframes where sentiment="negative"
for text in df_train[df_train.sentiment=="bad"].news:
  for word in generate_N_grams(text):
    negativeValues[word]+=1
#get the count of every word in both the columns of df_train and df_test dataframes where sentiment="neutral"
```

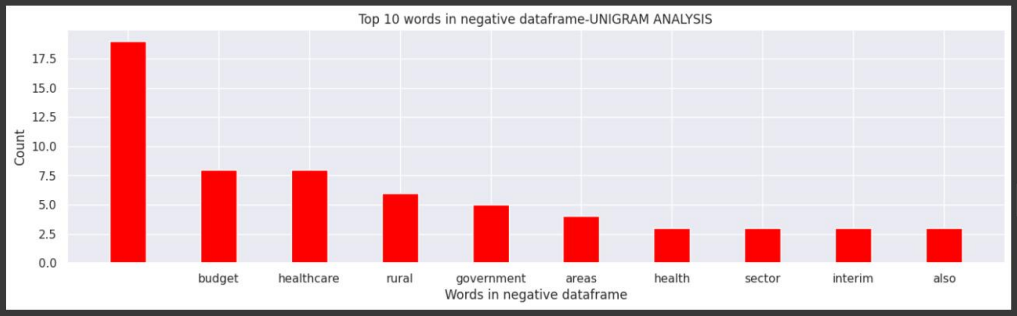
2m 54s completed at 12:37 PM

```
    neutralValues[word]+=1
#focus on more frequently occuring words for every sentiment=>
#sort in DO wrt 2nd column in each of positiveValues,negativeValues and neutralValues
df_positive=pd.DataFrame(sorted(positiveValues.items(),key=lambda x:x[1],reverse=True))
df_negative=pd.DataFrame(sorted(negativeValues.items(),key=lambda x:x[1],reverse=True))
df_neutral=pd.DataFrame(sorted(neutralValues.items(),key=lambda x:x[1],reverse=True))
pd1=df_positive[0][:10]
pd2=df_positive[1][:10]
ned1=df_negative[0][:10]
ned2=df_negative[1][:10]
nud1=df_neutral[0][:10]
nud2=df_neutral[1][:10]
plt.figure(1,figsize=(16,4))
plt.bar(pd1,pd2, color ='green',
        width = 0.4)
plt.xlabel("Words in positive dataframe")
plt.ylabel("Count")
plt.title("Top 10 words in positive dataframe-UNIGRAM ANALYSIS")
plt.savefig("positive-unigram.png")
plt.show()
```

```
Sentence after removing stopwords: ['Â\xa0to', 'address', 'dearth', 'doctors', 'india', 'government', 'though
Sentence after removing stopwords: ['', 'decision', 'expand', 'medical', 'educational', 'institutions', 'usin
Sentence after removing stopwords: ['', 'however', 'feel', 'customs', 'duty', 'medical', 'devices', 'reduced'
Sentence after removing stopwords: ['bringing', 'maternal', 'child', 'healthcare', 'schemes', 'one', 'umbrell
Sentence after removing stopwords: ['', 'measures', 'enhance', 'healthcare', 'accessibility', 'rural', 'areas
Sentence after removing stopwords: ['', 'aligns', 'global', 'sustainability', 'goals', 'also', 'signifies', '
```

2m 54s completed at 12:37 PM

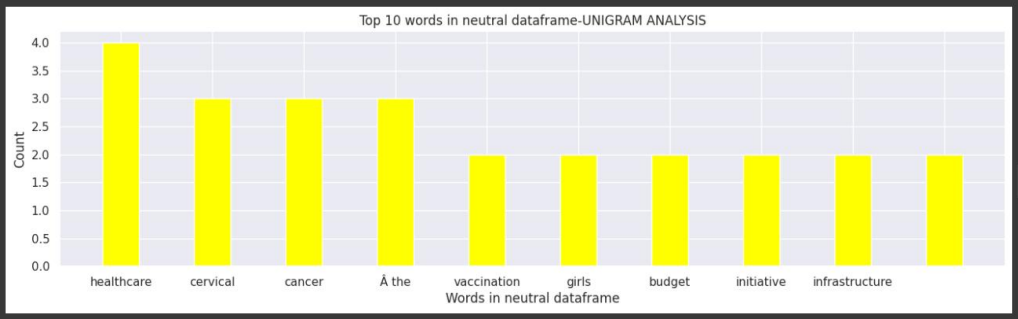
```
plt.figure(1,figsize=(16,4))
plt.bar(ned1,ned2, color ='red',
        width = 0.4)
plt.xlabel("Words in negative dataframe")
plt.ylabel("Count")
plt.title("Top 10 words in negative dataframe-UNIGRAM ANALYSIS")
plt.savefig("negative-unigram.png")
plt.show()
```




```
[64] plt.figure(1,figsize=(16,4))
     plt.bar(nud1,nud2, color ='yellow',
             width = 0.4)
```

2m 54s completed at 12:37 PM

```
plt.figure(1,figsize=(16,4))
plt.bar(nud1,nud2, color ='yellow',
        width = 0.4)
plt.xlabel("Words in neutral dataframe")
plt.ylabel("Count")
plt.title("Top 10 words in neutral dataframe-UNIGRAM ANALYSIS")
plt.savefig("neutral-unigram.png")
plt.show()
```




```
[73] import pandas as pd
```

2m 54s completed at 12:37 PM

+ Code + Text

RAM Disk

[73]

```python
import pandas as pd
import matplotlib.pyplot as plt

positiveValues2=defaultdict(int)
negativeValues2=defaultdict(int)
neutralValues2=defaultdict(int)

# Assuming df_train and df_test dataframes are defined somewhere
# Assuming sentiment column exists in df_train dataframe

# get the count of every word in both the columns of df_train and df_test dataframes where sentiment="positive
for text in df_train[df_train.sentiment=="positive"].news:
    for word in generate_N_grams(text,2):
        positiveValues2[word]+=1

# get the count of every word in both the columns of df_train and df_test dataframes where sentiment="negative
for text in df_train[df_train.sentiment=="negative"].news:
    for word in generate_N_grams(text,2):
        negativeValues2[word]+=1

# get the count of every word in both the columns of df_train and df_test dataframes where sentiment="neutral"
for text in df_train[df_train.sentiment=="neutral"].news:
    for word in generate_N_grams(text,2):
        neutralValues2[word]+=1
```

2m 54s completed at 12:37 PM

+ Code + Text

RAM Disk

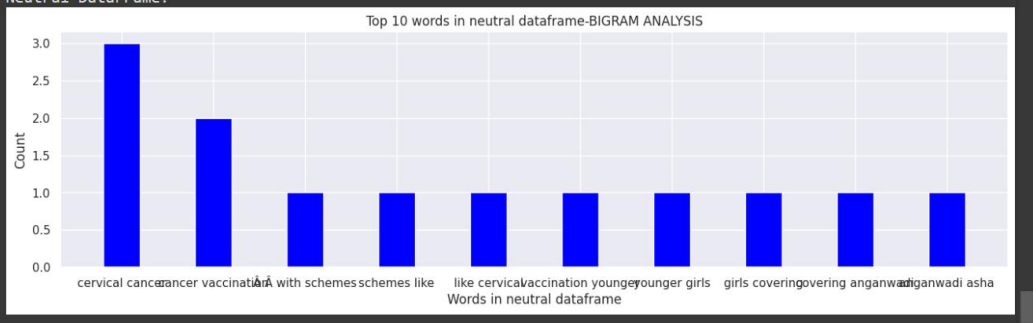
```python
# focus on more frequently occurring words for every sentiment => sort in descending order wrt values
df_positive2 = pd.DataFrame(sorted(positiveValues2.items(), key=lambda x: x[1], reverse=True))
df_negative2 = pd.DataFrame(sorted(negativeValues2.items(), key=lambda x: x[1], reverse=True))
#df_neutral2 = pd.DataFrame(sorted(neutralValues2.items(), key=lambda x: x[1], reverse=True))

if not df_positive2.empty:
    print("Positive DataFrame:")
    pd1bi = df_positive2[0][:10]  # Words
    pd2bi = df_positive2[1][:10]  # Counts
    plt.figure(figsize=(16, 4))
    plt.bar(range(len(pd1bi)), pd2bi, color='green', width=0.4)
    plt.xlabel("Words in positive dataframe")
    plt.ylabel("Count")
    plt.xticks(range(len(pd1bi)), pd1bi)  # Set x-axis labels to words
    plt.title("Top 10 words in positive dataframe-BIGRAM ANALYSIS")
    plt.savefig("positive-bigram.png")
    plt.show()

if not df_negative2.empty:
    print("Negative DataFrame:")
    nd1bi = df_negative2[0][:10]  # Words
    nd2bi = df_negative2[1][:10]  # Counts
    plt.figure(figsize=(16, 4))
    plt.bar(range(len(nd1bi)), nd2bi, color='red', width=0.4)
    plt.xlabel("Words in negative dataframe")
    plt.ylabel("Count")
```

2m 54s completed at 12:37 PM

```python
if not df_negative2.empty:
    print("Negative DataFrame:")
    nd1bi = df_negative2[0][:10]  # Words
    nd2bi = df_negative2[1][:10]  # Counts
    plt.figure(figsize=(16, 4))
    plt.bar(range(len(nd1bi)), nd2bi, color='red', width=0.4)
    plt.xlabel("Words in negative dataframe")
    plt.ylabel("Count")
    plt.xticks(range(len(nd1bi)), nd1bi)  # Set x-axis labels to words
    plt.title("Top 10 words in negative dataframe-BIGRAM ANALYSIS")
    plt.savefig("negative-bigram.png")
    plt.show()

if not df_neutral2.empty:
    print("Neutral DataFrame:")
    nud1bi = df_neutral2[0][:10]  # Words
    nud2bi = df_neutral2[1][:10]  # Counts
    plt.figure(figsize=(16, 4))
    plt.bar(range(len(nud1bi)), nud2bi, color='blue', width=0.4)
    plt.xlabel("Words in neutral dataframe")
    plt.ylabel("Count")
    plt.xticks(range(len(nud1bi)), nud1bi)  # Set x-axis labels to words
    plt.title("Top 10 words in neutral dataframe-BIGRAM ANALYSIS")
    plt.savefig("neutral-bigram.png")
    plt.show()
```

2m 54s completed at 12:37 PM

```
Sentence after removing stopwords: ['Â\xa0Â\xa0with', 'schemes', 'like', 'cervical', 'cancer', 'vaccination', '
Sentence after removing stopwords: ['Â\xa0the', 'initiative', 'establish', 'additional', 'medical', 'colleges',
Sentence after removing stopwords: ['Â\xa0the', 'finance', 'minister', 'also', 'outlined', 'prudent', 'backward
Sentence after removing stopwords: ['', 'however', 'crucial', 'aspect', 'cervical', 'cancer', 'vaccination', 'g
Sentence after removing stopwords: ['', 'initiative', 'crucial', 'india', 'currently', 'holds', 'unfortunate',
Sentence after removing stopwords: ['Â\xa0the', 'interim', 'budget', '202425', 'allocation', 'rs', '90171', 'cr
Neutral DataFrame:
```




```python
[75] positiveValues3 = defaultdict(int)
     negativeValues3 = defaultdict(int)
     neutralValues3 = defaultdict(int)
```

2m 54s completed at 12:37 PM

```python
positiveValues3 = defaultdict(int)
negativeValues3 = defaultdict(int)
neutralValues3 = defaultdict(int)

# Assuming df_train and df_test dataframes are defined somewhere
# Assuming sentiment column exists in df_train dataframe

# get the count of every word in both the columns of df_train and df_test dataframes where sentiment="positive
for text in df_train[df_train.sentiment=="positive"].news:
    for word in generate_N_grams(text,3):
        positiveValues3[word]+=1

# get the count of every word in both the columns of df_train and df_test dataframes where sentiment="negative
for text in df_train[df_train.sentiment=="negative"].news:
    for word in generate_N_grams(text,3):
        negativeValues3[word]+=1

# get the count of every word in both the columns of df_train and df_test dataframes where sentiment="neutral"
for text in df_train[df_train.sentiment=="neutral"].news:
    for word in generate_N_grams(text,3):
        neutralValues3[word]+=1

# focus on more frequently occurring words for every sentiment => sort in descending order wrt values
df_positive3 = pd.DataFrame(sorted(positiveValues3.items(), key=lambda x: x[1], reverse=True))
df_negative3 = pd.DataFrame(sorted(negativeValues3.items(), key=lambda x: x[1], reverse=True))
df_neutral3 = pd.DataFrame(sorted(neutralValues3.items(), key=lambda x: x[1], reverse=True))
```

2m 54s completed at 12:37 PM

```python
    print(plt.show())

if not df_negative3.empty:
    print("Negative DataFrame:")
    nd1tri = df_negative3[0][:10]  # Words
    nd2tri = df_negative3[1][:10]  # Counts
    plt.figure(figsize=(16, 4))
    plt.bar(range(len(nd1tri)), nd2tri, color='red', width=0.4)
    plt.xlabel("Words in negative dataframe")
    plt.ylabel("Count")
    plt.xticks(range(len(nd1tri)), nd1tri)  # Set x-axis labels to words
    plt.title("Top 10 words in negative dataframe-TRIGRAM ANALYSIS")
    plt.savefig("negative-trigram.png")
    print(plt.show())

if not df_neutral3.empty:
    print("Neutral DataFrame:")
    nud1tri = df_neutral3[0][:10]  # Words
    nud2tri = df_neutral3[1][:10]  # Counts
    plt.figure(figsize=(16, 4))
    plt.bar(range(len(nud1tri)), nud2tri, color='blue', width=0.4)
    plt.xlabel("Words in neutral dataframe")
    plt.ylabel("Count")
    plt.xticks(range(len(nud1tri)), nud1tri)  # Set x-axis labels to words
    plt.title("Top 10 words in neutral dataframe-TRIGRAM ANALYSIS")
    plt.savefig("neutral-trigram.png")
    print(plt.show())
```

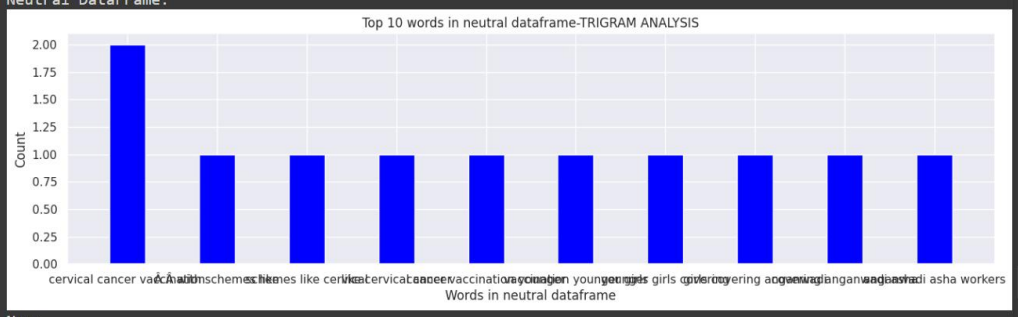
2m 54s completed at 12:37 PM

```
print(plt.show())

if not df_negative3.empty:
    print("Negative DataFrame:")
    nd1tri = df_negative3[0][:10]  # Words
    nd2tri = df_negative3[1][:10]  # Counts
    plt.figure(figsize=(16, 4))
    plt.bar(range(len(nd1tri)), nd2tri, color='red', width=0.4)
    plt.xlabel("Words in negative dataframe")
    plt.ylabel("Count")
    plt.xticks(range(len(nd1tri)), nd1tri)  # Set x-axis labels to words
    plt.title("Top 10 words in negative dataframe-TRIGRAM ANALYSIS")
    plt.savefig("negative-trigram.png")
    print(plt.show())

if not df_neutral3.empty:
    print("Neutral DataFrame:")
    nud1tri = df_neutral3[0][:10]  # Words
    nud2tri = df_neutral3[1][:10]  # Counts
    plt.figure(figsize=(16, 4))
    plt.bar(range(len(nud1tri)), nud2tri, color='blue', width=0.4)
    plt.xlabel("Words in neutral dataframe")
    plt.ylabel("Count")
    plt.xticks(range(len(nud1tri)), nud1tri)  # Set x-axis labels to words
    plt.title("Top 10 words in neutral dataframe-TRIGRAM ANALYSIS")
    plt.savefig("neutral-trigram.png")
    print(plt.show())
```

2m 54s completed at 12:37 PM

```
plt.title("Top 10 words in neutral dataframe-TRIGRAM ANALYSIS")
plt.savefig("neutral-trigram.png")
print(plt.show())
```

```
Sentence after removing stopwords: ['Â\xa0Â\xa0with', 'schemes', 'like', 'cervical', 'cancer', 'vaccination', '
Sentence after removing stopwords: ['Â\xa0the', 'initiative', 'establish', 'additional', 'medical', 'colleges',
Sentence after removing stopwords: ['Â\xa0the', 'finance', 'minister', 'also', 'outlined', 'prudent', 'backward
Sentence after removing stopwords: ['', 'however', 'crucial', 'aspect', 'cervical', 'cancer', 'vaccination', 'g
Sentence after removing stopwords: ['', 'initiative', 'crucial', 'india', 'currently', 'holds', 'unfortunate',
Sentence after removing stopwords: ['Â\xa0the', 'interim', 'budget', '202425', 'allocation', 'rs', '90171', 'cr
Neutral DataFrame:
```




```
None
```

+ Code + Text

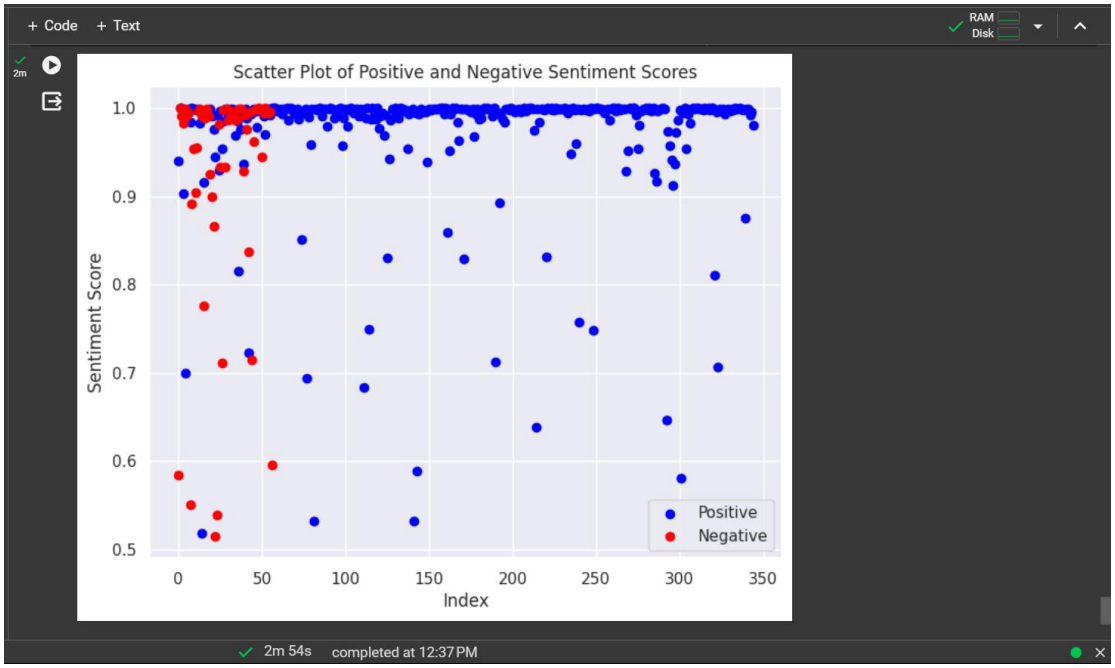
```python
import matplotlib.pyplot as plt

# Assuming classifier function is defined and accessible

# Function to generate sample data
def generate_data():
    positive_data = []
    negative_data = []

    # Open and read lines from the input file
    with open("output.txt", "r") as f:
        lines = f.readlines()
        for line in lines:
            # Classify the line and extract the sentiment score
            score = classifier(line)[0]['score']

            # Determine sentiment and append to appropriate list
            if classifier(line)[0]['label'] == 'POSITIVE':
                positive_data.append(score)
            else:
                negative_data.append(score)

    return positive_data, negative_data

# Generate sample data
positive_data, negative_data = generate_data()
```

2m 54s completed at 12:37 PM

+ Code + Text

```python
        for line in lines:
            # Classify the line and extract the sentiment score
            score = classifier(line)[0]['score']

            # Determine sentiment and append to appropriate list
            if classifier(line)[0]['label'] == 'POSITIVE':
                positive_data.append(score)
            else:
                negative_data.append(score)

    return positive_data, negative_data

# Generate sample data
positive_data, negative_data = generate_data()

# Plot scatter graph
plt.figure(figsize=(8, 6))
plt.scatter(range(len(positive_data)), positive_data, color='blue', label='Positive')
plt.scatter(range(len(negative_data)), negative_data, color='red', label='Negative')
plt.xlabel('Index')
plt.ylabel('Sentiment Score')
plt.title('Scatter Plot of Positive and Negative Sentiment Scores')
plt.legend()
plt.grid(True)
plt.show()
```

2m 54s completed at 12:37 PM



REFERENCES :

- https://www.analyticsvidhya.com/blog/2021/09/what-are-n-grams-and-how-to-implement-them-in-python/
- https://colab.research.google.com/drive/1Z_Qmx0D3umE0wa3U7TESCy1E5OlfPdti?usp=sharing
- https://indiamedtoday.com/2024-budget-reactions/