Import Kaggle - Original Notebook was in kaggle

```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'bbc-news-summary:https%3A%2F%2Fstorage.googleapis.com%2Fka

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=
except FileExistsError:
```

Resources

You are not subscribed. Learn more

You currently have zero compute units available.
Resources offered free of charge are not guaranteed.
Purchase more units here.

At your current usage level, this runtime may last up to 2 hours 10 minutes.

Manage sessions

Python 3 Google Compute Engine backend (GPU)

Showing resources from 6:50 PM to 2:35 PM

System RAM
6.2 / 12.7 GB

GPU RAM
14.7 / 15.0 GB

Disk
32.8 / 78.2 GB

```
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_direct
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes d
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destinati
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue
```

```
print('Data source import complete.')
```

```
Downloading bbc-news-summary, 9342346 bytes compressed
[==================================================] 9342346 bytes downloaded
Downloaded and uncompressed: bbc-news-summary
Data source import complete.
```

Some Pip Installs required *

```
!pip install bert-extractive-summarizer
!pip install contractions
!pip install evaluate rouge_score
!pip install gradio
!pip install accelerate -U
```

```
Requirement already satisfied: bert-extractive-summarizer in /usr/local/li
Requirement already satisfied: transformers in /usr/local/lib/python3.10/d
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/d
Requirement already satisfied: spacy in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/d
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/pyth
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/l
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/l
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/pytho
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/pyt
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in /usr/local/lib/pytho
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/pyth
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/pytho
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/p
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in /usr/local/lib/pyth
Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/pytho
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/pytho
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/p
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /us
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-pa
```

```
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dis
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.1
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/p
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
Requirement already satisfied: huggingface-hub<1.0,>=0.23.0 in /usr/local/
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/py
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/li
Requirement already satisfied: language-data>=1.2 in /usr/local/lib/python
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/py
Requirement already satisfied: pydantic-core==2.18.4 in /usr/local/lib/pyt
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/d
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.10/
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/li
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.1
Requirement already satisfied: marisa-trie>=0.7.7 in /usr/local/lib/python
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/pyt
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/p
Requirement already satisfied: wrapt in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dis
Requirement already satisfied: contractions in /usr/local/lib/python3.10/d
Requirement already satisfied: textsearch>=0.0.21 in /usr/local/lib/python
Requirement already satisfied: anyascii in /usr/local/lib/python3.10/dist-
```

Imports

```
import pandas as pd
import os
import numpy as np
from spacy.tokenizer import Tokenizer
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
import seaborn as sns
from sklearn.manifold import TSNE
# from rouge import Rouge
import transformers
from summarizer import Summarizer, TransformerSummarizer
import contractions
from sklearn.model_selection import KFold
from transformers import BigBirdPegasusForConditionalGeneration, PegasusTokenizer
import torch
from datasets import Dataset
import gradio as gr
from transformers import PegasusForConditionalGeneration, TrainingArguments, Auto
import evaluate
rouge = evaluate.load('rouge')
```

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docke
# For example, here's several helpful packages to load

# just to see the directory structure.
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list a

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets p
# You can also write temporary files to /kaggle/temp/, but they won't be saved out
```

Show hidden output

Create a Single Dataframe from all 5 categories of news.

```
# We need to create the dataframe and concatenate later.
def make_dataframes_from_file(text,summary):
    articles = {file for file in os.listdir(text) if file.endswith('.txt')}
    summaries = {file for file in os.listdir(summary) if file.endswith('.txt')}

    common_files = articles.intersection(summaries)
    # so that it matches the same files
    if len(common_files) != len(articles) or len(common_files) != len(summaries):
        raise ValueError("Source and target directories do not contain the same f

    records = []
    for filename in common_files:
        article_file_path = os.path.join(text, filename)
        summary_file_path = os.path.join(summary, filename)
        with open(article_file_path, 'r', encoding='latin-1') as article_file:
            article_text = article_file.read()

        with open(summary_file_path, 'r', encoding='latin-1') as summary_file:
            summary_text = summary_file.read()

        records.append({'article': article_text, 'summary': summary_text})
    df = pd.DataFrame.from_records(records)

    return df
```

Combine them

```
""" Loop thru the categories"""
categories = ['business', 'entertainment', "politics", "sport", "tech"]

base_article_path = "/kaggle/input/bbc-news-summary/BBC News Summary/News Articles
base_summary_path = "/kaggle/input/bbc-news-summary/BBC News Summary/Summaries/"

# Create dfs for all the categories (5)
dfs = [make_dataframes_from_file(f"{base_article_path}{category}", f"{base_summary

# Concatenate all dfs
df = pd.concat(dfs, ignore_index=True)
df.head(5)
```

	article	summary
0	Lufthansa flies back to profit\n\nGerman airli...	German airline Lufthansa has returned to profi...
1	Dollar hits new low versus euro\n\nThe US doll...	The US dollar has continued its record-breakin...
2	US economy shows solid GDP growth\n\nThe US ec...	Growth was at an annual rate of 4% in the thir...
3	Warning over US pensions deficit\n\nTaxpayers ...	With the Pension Benefit Guaranty Corporation ...
4	US in EU tariff chaos trade row\n\nThe US has ...	"Although the EU is a customs union, there is ...

Next steps: Generate code with df View recommended plots

```
#check specific words
keyword = 'wal-mart'
filtered_df = df[df['summary'].str.contains(keyword, case=False)]
filtered_df
```

	article	summary
66	Wal-Mart to pay $14m in gun suit\n\nThe world'...	"Although Wal-Mart has suspended gun sales in ...
121	Mixed Christmas for US retailers\n\nUS retaile...	Upscale department store Nordstrom said same s...
182	Winn-Dixie files for bankruptcy\n\nUS supermar...	Winn-Dixie, once among the most profitable of ...
197	Macy's owner buys rival for $11bn\n\nUS retail...	US retail giant Federated Department Stores is...
238	Wal-Mart fights back at accusers\n\nTwo big US...	Meanwhile, drugs group Eli Lilly is planning a...
318	US consumer confidence up\n\nConsumers' confid...	Wal-Mart, the largest US retailer, has said it...
811	Wal-Mart is sued over rude lyrics\n\nThe paren...	Wal-Mart said it was investigating the claims ...

Next steps: Generate code with `filtered_df` View recommended plots

```
df.summary
```

```
0       German airline Lufthansa has returned to profi...
1       The US dollar has continued its record-breakin...
2       Growth was at an annual rate of 4% in the thir...
3       With the Pension Benefit Guaranty Corporation ...
4       "Although the EU is a customs union, there is ...
                              ...
2220    The blurring of boundaries between TV and the ...
2221    Many feel that the most difficult and challeng...
2222    Critics say the law would favour large compani...
2223    It has to be said if the game did not have the...
2224    But, however careful you may be, if the organi...
Name: summary, Length: 2225, dtype: object
```

```
df.head()
```

	article	summary
0	Lufthansa flies back to profit\n\nGerman airli...	German airline Lufthansa has returned to profi...
1	Dollar hits new low versus euro\n\nThe US doll...	The US dollar has continued its record-breakin...
2	US economy shows solid GDP growth\n\nThe US ec...	Growth was at an annual rate of 4% in the thir...
3	Warning over US pensions deficit\n\nTaxpayers ...	With the Pension Benefit Guaranty Corporation ...
4	US in EU tariff chaos trade row\n\nThe US has ...	"Although the EU is a customs union, there is ...

Next steps: Generate code with df View recommended plots

Some Cleaning with contractions and newlines spotted.

```
df=df.dropna()
def remove_newlines(text):
    return text.replace('\n', ' ')

df['article'] = df['article'].apply(remove_newlines)
df['summary'] = df['summary'].apply(remove_newlines)

def fix_contractions(text):
    return contractions.fix(text)

df['article'] = df['article'].apply(fix_contractions)
df['summary'] = df['summary'].apply(fix_contractions)
```

Train test val split

```
X = df['article']
Y = df['summary']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_st
train_df = pd.DataFrame({'article_text': X_train, 'summary_text': Y_train})
test_df = pd.DataFrame({'article_text': X_test, 'summary_text': Y_test})

# val set
train_df, val_df = train_test_split(train_df, test_size=0.1, random_state=42)
```

```
train_df.shape
```

```
(1602, 2)
```

```
test_df.shape
```

```
(445, 2)
```

```
train_df.describe()
```

	article_text	summary_text
count	1602	1602
unique	1554	1528
top	More power to the people says HP The digital ...	CBI chief economist Ian McCafferty said the ec...
freq	2	2

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
```

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

Remove punctuations, stop words, tokenize, etc.

```
# clean the data, remove common things in nlp.
stop_words = set(stopwords.words('english'))

def preprocess_text(text):

    tokens = word_tokenize(text)

    tokens = [token.lower() for token in tokens]
    # punctuation
    tokens = [token for token in tokens if token not in string.punctuation]
    # stopwords
    tokens = [token for token in tokens if token not in stop_words]
    return ' '.join(tokens)

train_df['articles'] = train_df['article_text'].apply(preprocess_text)
train_df['summaries'] = train_df['summary_text'].apply(preprocess_text)

train_df['articles'].head()
```

```
0       lufthansa flies back profit german airline luf...
1911    playstation 3 processor unveiled cell processo...
1902    ibm puts cash behind linux push ibm spending 1...
1690    robben sidelined broken foot chelsea winger ar...
```

```
601     grammys honour soul star charles memory soul l...
Name: articles, dtype: object
```

```
train_df.head()
```

	article_text	summary_text	articles	summaries
0	Lufthansa flies back to profit German airline...	German airline Lufthansa has returned to profi...	lufthansa flies back profit german airline luf...	german airline lufthansa returned profit 2004 ...
1911	PlayStation 3 processor unveiled The Cell pro...	The Cell processor, which will drive Sony's Pl...	playstation 3 processor unveiled cell processo...	cell processor drive sony 's playstation 3 run...
1902	IBM puts cash behind Linux push IBM is spendi...	IBM said it had taken the step in response to ...	ibm puts cash behind linux push ibm spending 1...	ibm said taken step response greater customer ...
1690	Robben sidelined with broken foot Chelsea win...	Chelsea winger Arjen Robben has broken two met...	robben sidelined broken foot chelsea winger ar...	chelsea winger arjen robben broken two metatar...
601	Grammys honour soul star Charles The memory o...	His song Here We Go Again with Norah Jones won...	grammys honour soul star charles memory	song go norah jones record year best pop

Next steps: Generate code with `train_df` View recommended plots

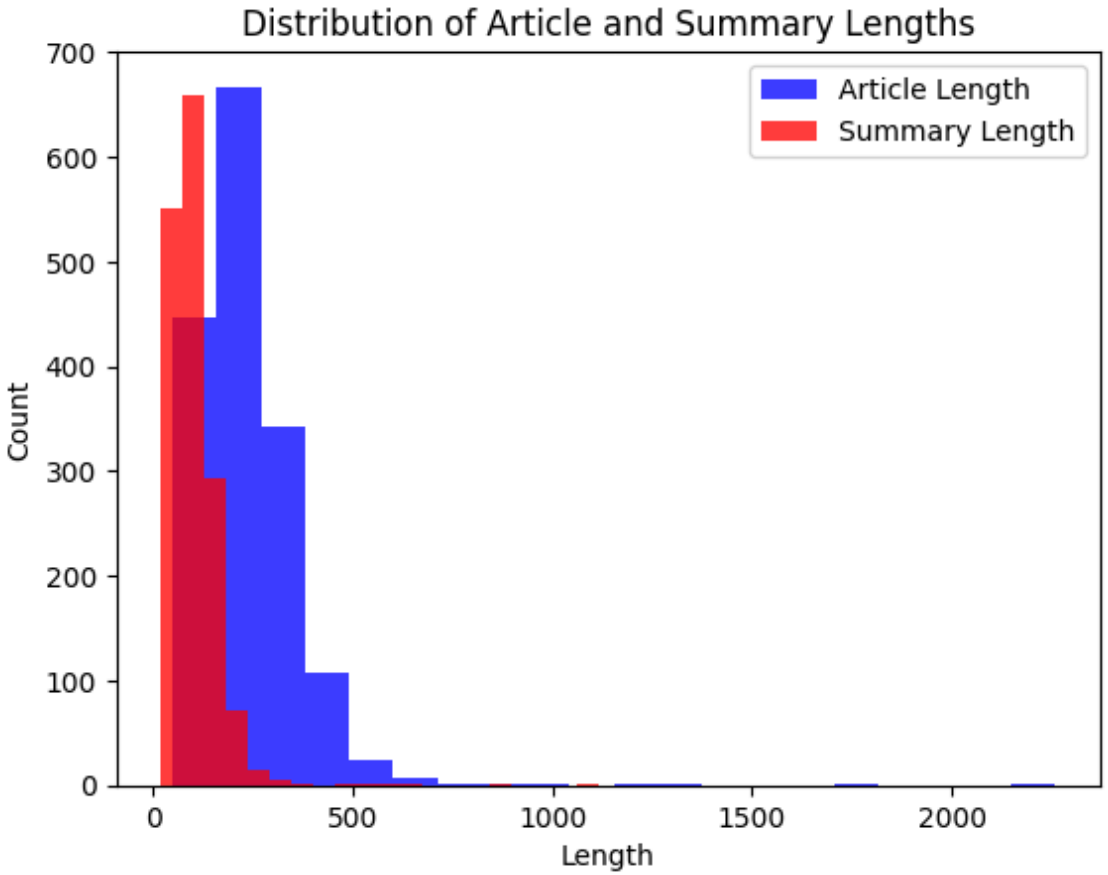
Some Stats

```
train_df['article_length'] = train_df['articles'].apply(lambda x: len(x.split()))
train_df['summary_length'] = train_df['summaries'].apply(lambda x: len(x.split()))

print("Biggest length of articles:")
print(train_df['article_length'].max())

print("Average length of summaries:")
print(train_df['summary_length'].mean())
```

```
Max length of articles:
2258
Max length of summaryies:
1114
Average length of summaries:
104.08302122347067
```

Histogram of article and summary length

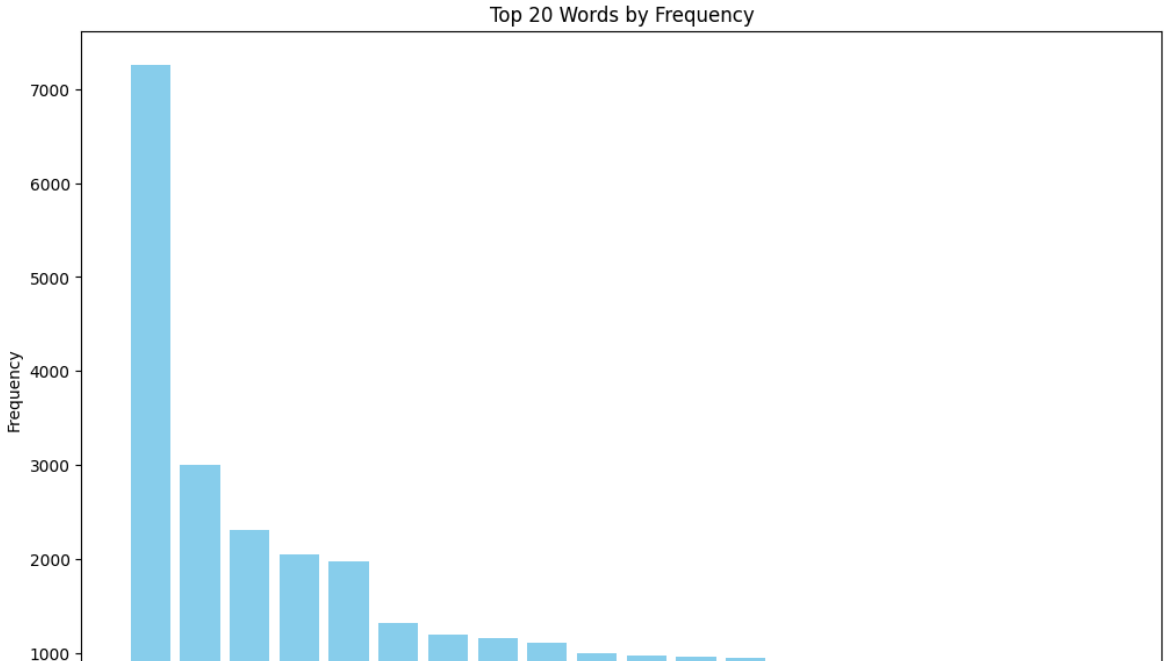
```
plt.hist(train_df['article_length'], bins=20, alpha=0.75, color='b', label='Artic
plt.hist(train_df['summary_length'], bins=20, alpha=0.75, color='r', label='Summa
plt.xlabel('Length')
plt.ylabel('Count')
plt.title('Distribution of Article and Summary Lengths')
plt.legend()
plt.show()
```



Find the most used words for a wordcloud later

```
vectorizer = CountVectorizer(stop_words='english')

# transform the articles for counting
X = vectorizer.fit_transform(df['article'])
# sum
word_frequencies = X.sum(axis=0)

word_feats = vectorizer.get_feature_names_out()

# Create a DataFrame to store frequencies
word_freq_df = pd.DataFrame(word_feats, columns=['word'])

word_freq_df['frequency'] = word_frequencies.tolist()[0]

# Sort words by frequency
word_freq_df = word_freq_df.sort_values(by='frequency', ascending=False)

# Display top 10 words by frequency
top_words = word_freq_df.head(20)
print("Top 20 words by frequency:")
print(top_words)

# Plotting top 20 words by frequency
plt.figure(figsize=(12, 8))
plt.bar(top_words['word'], top_words['frequency'], color='skyblue')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 20 Words by Frequency')
plt.show()
```

```
Top 20 words by frequency:
            word  frequency
22822        said       7255
17682          mr       3005
28933        year       2309
19527      people       2045
18095         new       1978
26390        time       1322
28764       world       1201
11988  government       1160
27092          uk       1115
28937       years       1003
3841         best        974
14809        just        957
16358        make        945
26477        told        911
10641        film        890
15792        like        879
11455        game        871
17786       music        839
15275      labour        804
1             000        804
```



Finding cosine similarity between each article and summary

```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer

# Example DataFrame df with 'article_text' and 'summary_text' columns
# Assuming df['article_text'] and df['summary_text'] contain preprocessed text dat

# Initialize TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words='english')

# Fit and transform article_text and summary_text
X = vectorizer.fit_transform(df['article'])
Y = vectorizer.transform(df['summary'])

# Calculate cosine similarity between article_text and summary_text
cosine_similarities = cosine_similarity(X, Y)

# Example: Print cosine similarity score between the first article and its corresp
print(f"Cosine similarity between first article and its summary: {cosine_similarit
```

```
Cosine similarity between first article and its summary: 0.8398752597963478
```

This is computational heavy so won't work, will figure another way

```
plt.figure(figsize=(10, 8))
sns.heatmap(cosine_similarities, cmap='coolwarm', vmin=0, vmax=1,
            xticklabels=df['summary'], yticklabels=df['article'],
            cbar_kws={'label': 'Cosine Similarity'})
plt.title('Cosine Similarity between Articles and Summaries')
plt.xlabel('Summary Text')
plt.ylabel('Article Text')
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```

Show hidden output

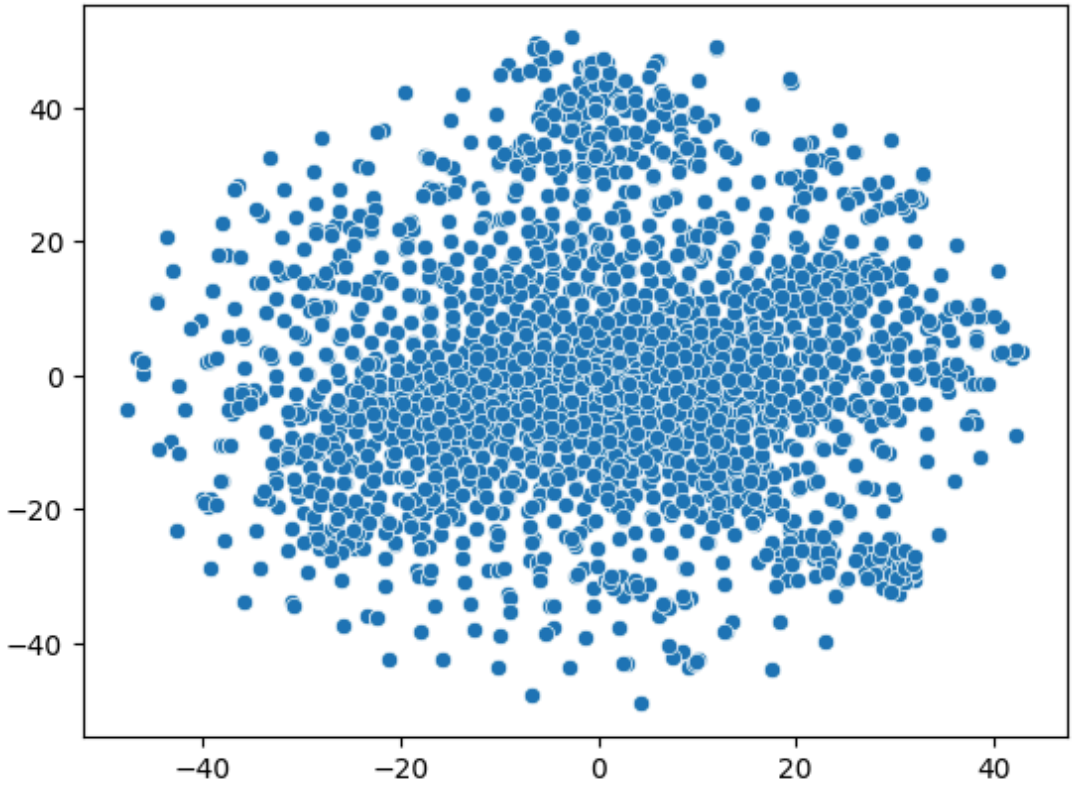
```
cv= CountVectorizer(max_df= .6, min_df= .05, ngram_range= (1, 3))
X_cv= cv.fit_transform(df.article)
X_cv.shape
```

```
(2225, 975)
```

Visualization using TSNE

```
tsne = TSNE(n_components= 2, learning_rate= 'auto', init= 'random')
X_cv_tsne= tsne.fit_transform(X_cv)
sns.scatterplot(x= X_cv_tsne[:,0], y= X_cv_tsne[:,1])
```

```
<Axes: >
```



Direct Apply BERT, GPT2, XLNet on a given text.

```
model_bert = Summarizer()

model_gpt2 = TransformerSummarizer(transformer_type = "GPT2", transformer_model_k

model_xlnet = TransformerSummarizer(transformer_type = "XLNet", transformer_model_
```

```
df['article'][12]
text = 'Putin backs state grab for Yukos  Russia\'s president has defended the pur
```

Summarizes the text using the models.

```
summary_1 = ''.join(model_bert(text, min_length=50))
summary_2 = ''.join(model_gpt2(text, min_length=50))
summary_3 = ''.join(model_xlnet(text, min_length=50))

print(f'Summary by {model_bert}:\n\n{summary_1} \n\n')
print(f'Summary by {model_gpt2}:\n\n{summary_2} \n\n')
print(f'Summary by {model_xlnet}:\n\n{summary_3} \n\n')
```

Load the MODEL - PEGASUS in this case.

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
# mostly it's not available
# Load the pre-trained tokenizer and model
# tokenizer = PegasusTokenizer.from_pretrained('google/bigbird-pegasus-large-arxi
tokenizer = AutoTokenizer.from_pretrained("google/bigbird-pegasus-large-arxiv")
model = BigBirdPegasusForConditionalGeneration.from_pretrained('google/bigbird-pe
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: U
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
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access pu
  warnings.warn(
```

```
device
```

```
device(type='cuda')
```

Constant prolem with google colab

```
pip install accelerate -U
```

```
import accelerate
print(accelerate.__version__)
```

```
0.31.0
```

```
!pip install transformers[torch] --upgrade
```

```
import torch
print(torch.__version__)
# Delete a model
del model
torch.cuda.empty_cache()
```

```
# Tokenize the data
def tokenize_data(df):
    model_inputs = tokenizer(df['article_text'], max_length=524, truncation=True,
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(df['summary_text'], max_length=128, truncation=True, p
    model_inputs['labels'] = labels['input_ids']
    return model_inputs

train_dataset = train_df.apply(tokenize_data, axis=1).tolist()
val_dataset = val_df.apply(tokenize_data, axis=1).tolist()
test_dataset = test_df.apply(tokenize_data, axis=1).tolist()


# Define the training arguments
training_args = TrainingArguments(
```

```
    output_dir='./output',
    eval_strategy='epoch',
    learning_rate=5e-5,
    per_device_train_batch_size=1,
    per_device_eval_batch_size=1,
    num_train_epochs=1,
    weight_decay=0.1,
    save_total_limit=3,
    warmup_steps=500,
    save_strategy='epoch',
    remove_unused_columns=True,
    disable_tqdm=True,
    load_best_model_at_end=True,
    metric_for_best_model="rouge2"
)

# all rouge metrics
def compute_metrics(pred):
    # Example metrics using rouge for summarization
    return {
        'rouge1_precision': pred.metrics['rouge1'].precision,
        'rouge1_recall': pred.metrics['rouge1'].recall,
        'rouge1_fmeasure': pred.metrics['rouge1'].fmeasure,
        'rouge2_precision': pred.metrics['rouge2'].precision,
        'rouge2_recall': pred.metrics['rouge2'].recall,
        'rouge2_fmeasure': pred.metrics['rouge2'].fmeasure,
    }

# Initialize the Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)
```

```
# K-fold cross-validation
kf = KFold(n_splits=6, shuffle=True, random_state=42)

# to do
X_train = train_df['article_text'].values
Y_train = train_df['summary_text'].values

def train_fold(train_dataset, val_dataset):
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=val_dataset,
        tokenizer=tokenizer,
        compute_metrics=compute_metrics,
    )
    trainer.train()
    return trainer
train_fold(train_dataset, val_dataset)


result = trainer.evaluate(eval_dataset=test_dataset)
print(f"Final evaluation: {result}")
```

```
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.p
  warnings.warn(
Attention type 'block_sparse' is not possible if sequence_length: 524 <= num g
{'loss': 5.8197, 'grad_norm': 1.8068251609802246, 'learning_rate': 0.005, 'epo
{'loss': 6.3102, 'grad_norm': 1.3469185829162598, 'learning_rate': 0.00273139
{'loss': 5.9718, 'grad_norm': 1.380008339881897, 'learning_rate': 0.000462794
---------------------------------------------------------------------------
OutOfMemoryError                          Traceback (most recent call last)
<ipython-input-29-760948a80dd3> in <cell line: 74>()
     72     trainer.train()
     73     return trainer
---> 74 train_fold(train_dataset, val_dataset)
     75
     76

                                 10 frames
/usr/local/lib/python3.10/dist-packages/transformers/trainer_pt_utils.py in
torch_pad_and_concatenate(tensor1, tensor2, padding_index)
     97
     98     if len(tensor1.shape) == 1 or tensor1.shape[1] ==
tensor2.shape[1]:
---> 99         return torch.cat((tensor1, tensor2), dim=0)
    100
    101     # Let's figure out the new shape

OutOfMemoryError: CUDA out of memory. Tried to allocate 172.00 MiB. GPU
```

Next steps: Explain error

Total params

```
total_params = sum(p.numel() for p in model.parameters())
total_params
```

RoUGE scores for evaluation

```
# Compute ROUGE scores
rouge_results = trainer.compute_metrics(result)
print(f"ROUGE scores on test set:")
print(f"ROUGE-1 Precision: {rouge_results['rouge1_precision']}")
print(f"ROUGE-1 Recall: {rouge_results['rouge1_recall']}")
print(f"ROUGE-1 F-measure: {rouge_results['rouge1_fmeasure']}")
print(f"ROUGE-2 Precision: {rouge_results['rouge2_precision']}")
print(f"ROUGE-2 Recall: {rouge_results['rouge2_recall']}")
print(f"ROUGE-2 F-measure: {rouge_results['rouge2_fmeasure']}")
```

Gradio Interface

```
def summarize_article(article):

    inputs = tokenizer(article, return_tensors="pt", max_length=2048, truncation=

    # model generates the summary
    summary_ids = model.generate(inputs['input_ids'].to(device))

    summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)

    return summary

article_input = gr.Textbox(lines=10, label="Input article")
summarize_button = gr.Button(text="Summarize")
```