Download the followings in your prompt if you are using local environment

```
pip install datasets
```

```
!pip install huggingface_hub datasets
```

Go to https://huggingface.co/, register an acc, click on ur profile and find setting, go to "access tokens" tab, create your own token, then paste the token link in the login session

```
from huggingface_hub import hf_hub_download
from huggingface_hub import login

login()

import json
```

```
VBox(children=(HTML(value='<center>
<img\nsrc=https://huggingface.co/front/assets/huggingface_logo-noborder.sv
```

```
!git clone https://huggingface.co/datasets/kdave/Indian_Financial_News
```

If the git clone block doesnt work, use this one

```
from datasets import load_dataset

dataset = load_dataset("kdave/Indian_Financial_News")
```

```
Downloading readme: 0.00B [00:00, ?B/s]
```

```
import pandas as pd
from datasets import Dataset

df = pd.read_csv("Indian_Financial_News/training_data_26000.csv")
print(df.columns)
```

Run this one if imported from dataset instead of git clone

```
import pandas as pd

df = dataset["train"].to_pandas()
print(df.columns)
print(df.head())
```

```
Index(['URL', 'Content', 'Summary', 'Sentiment'], dtype='object')
                                                 URL  \
0  https://www.moneycontrol.com/news/business/eco...
1  https://www.businesstoday.in/top-story/state-r...
2  https://www.financialexpress.com/economy/covid...
3  https://www.moneycontrol.com/news/business/mar...
4  https://www.financialexpress.com/industry/six-...

                                             Content  \
0  US consumer spending dropped by a record in Ap...
1  State-run lenders require an urgent Rs 1.2 tri...
2  Apparel exporters on Wednesday urged the gover...
3  Asian shares battled to extend a global reboun...
4  After India’s sovereign credit rating fell to ...

                                             Summary Sentiment
0  consumer spending plunges 13.6 percent in Apri...  Negative
1  government will have to take a bulk of the tab...  Negative
2  exporters are facing issues in terms of raw ma...  Negative
3  the dollar loses some ground on the safe haven...  Negative
4  six Indian public-sector undertakings have tak...  Negative
```

```
from datasets import Dataset

ds = Dataset.from_pandas(df)
print(ds[0])
```

```
{'URL': 'https://www.moneycontrol.com/news/business/economy/covid-19-pandemic-
```

```
print("Number of records:", len(ds))
print("Columns:", ds.column_names)
```

```
Number of records: 26961
Columns: ['URL', 'Content', 'Summary', 'Sentiment']
```

```
summary_lengths = [len(x.split()) for x in ds['Summary'] if isinstance(x, str)]
content_lengths = [len(x.split()) for x in ds['Content'] if isinstance(x, str)]

import numpy as np

print("\nSummary Stats:")
print("  Avg length:", np.mean(summary_lengths))
print("  Min length:", np.min(summary_lengths))
print("  Max length:", np.max(summary_lengths))

print("\nContent Stats:")
print("  Avg length:", np.mean(content_lengths))
print("  Min length:", np.min(content_lengths))
print("  Max length:", np.max(content_lengths))
```

```

Summary Stats:
  Avg length: 61.362115648529354
  Min length: 1
  Max length: 108

Content Stats:
  Avg length: 609.7049070880161
  Min length: 9
  Max length: 8328
```

```
def is_content_valid(example):
    return isinstance(example["Content"], str) and len(example["Content"].split()

def is_summary_valid(example):
    return isinstance(example["Summary"], str) and len(example["Summary"].split()

a_filtered_ds = ds.filter(is_content_valid)

b_filtered_ds = a_filtered_ds.filter(is_summary_valid)
print("Filtered dataset size:", len(b_filtered_ds))
```

```
Filter:   0%|          | 0/26961 [00:00<?, ? examples/s]
Filter:   0%|          | 0/26254 [00:00<?, ? examples/s]
Filtered dataset size: 26233
```

```
summary_lengths = [len(x.split()) for x in b_filtered_ds['Summary'] if isinstance
content_lengths = [len(x.split()) for x in b_filtered_ds['Content'] if isinstance

import numpy as np

print("\nSummary Stats (Filtered for extreme short entry):")
print("  Avg length:", np.mean(summary_lengths))
print("  Min length:", np.min(summary_lengths))
print("  Max length:", np.max(summary_lengths))

print("\nContent Stats: (Filtered for extreme short entry)")
print("  Avg length:", np.mean(content_lengths))
print("  Min length:", np.min(content_lengths))
print("  Max length:", np.max(content_lengths))
```

```
Summary Stats (Filtered for extreme short entry):
  Avg length: 61.58052834216445
  Min length: 13
  Max length: 98

Content Stats: (Filtered for extreme short entry)
  Avg length: 624.5348225517478
  Min length: 100
  Max length: 8328
```

```
print(b_filtered_ds[1])
```

```
{'URL': 'https://www.businesstoday.in/top-story/state-run-banks-need-urgent-ca
```

now summary and content has been filtered by length b_filtered_ds

```
# use this if not aleady downloaded
!pip install nltk


import re
import nltk

nltk.download('punkt')  # only once
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\qq258\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
def clean_text(text):
    # Lowercase
    text = text.lower()

    # Remove URLs
    text = re.sub(r'https?://\S+|www\.\S+', '', text)

    # Remove extra whitespace and newlines
    text = re.sub(r'\s+', ' ', text).strip()

    # Replace unicode smart quotes and dashes with ASCII
    replacements = {
        '’': "'",
        '‘': "'",
        '“': '"',
        '”': '"',
        '–': '-',
        '—': '-',
        '\u2013': '-',  # en dash
        '\u2014': '-',  # em dash
    }
    for k, v in replacements.items():
        text = text.replace(k, v)

    # Remove any remaining non-printable characters
    text = ''.join(c for c in text if c.isprintable())

    return text

def filter_entry(entry):
    # For example, skip if content or summary too short
    if len(entry['Content'].split()) < 30:
        return False
    if len(entry['Summary'].split()) < 5:
        return False
    return True
```

```
def clean_example(example):
    example['Content'] = clean_text(example['Content'])
    example['Summary'] = clean_text(example['Summary'])
    return example

# Assuming b_filtered_ds is your Dataset object:
b_filtered_ds = b_filtered_ds.map(clean_example)
```

```
Map:   0%|          | 0/26233 [00:00<?, ? examples/s]
```

```
print(b_filtered_ds[1])
```

```
{'URL': 'https://www.businesstoday.in/top-story/state-run-banks-need-urgent-ca
```

```
b_filtered_ds.to_csv("cleaned_dataset.csv")
```

```
Creating CSV from Arrow format:   0%|          | 0/27 [00:00<?, ?ba/s]
113619219
```

```
# saving files
df = b_filtered_ds.to_pandas()
#df.to_json("cleaned_dataset.jsonl", orient="records", lines=True, index=False)
df.to_json("cleaned_dataset.jsonl", orient="records", lines=True)
```

DATASET PRE-PROCESSED, NOW BASELINE WIHT TEXTRANK

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

def split_sentences(text):
    return nltk.sent_tokenize(text)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\qq258\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\qq258\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

```
# download networkx if module error
!pip install networkx
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: networkx in c:\all files\anaconda3\lib\site-pac
```

```
import networkx as nx
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

def textrank(sentences, top_n=5):
    # vectorize sentences with TF-IDF
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(sentences)

    # compute similarity matrix
    sim_matrix = cosine_similarity(tfidf_matrix)

    # build graph and apply PageRank
    nx_graph = nx.from_numpy_array(sim_matrix)
    scores = nx.pagerank(nx_graph)

    # rank sentences and pick top N
    ranked_sentences = sorted(((scores[i], s) for i, s in enumerate(sentences)),
    selected = [s for _, s in ranked_sentences[:top_n]]

    return ' '.join(selected)
```

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

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[23], line 1
----> 1 from google.colab import drive
      2 drive.mount('/content/drive')

ModuleNotFoundError: No module named 'google'
```

```
# suppose dataset is a list of dicts or a pandas DataFrame
entry = b_filtered_ds[1]  # or dataset.iloc[1] if pandas

text = entry['Content']  # get the content field

# assuming you have your baseline pipeline functions from before:


sentences = split_sentences(text)
summary = textrank(sentences, top_n=5)

print("Original Content:\n", text)
print("Summary:\n", summary)
```

```
Original Content:
 state-run lenders require an urgent rs 1.2 trillion in the capital in the ne
Summary:
 as per the norms, the banks ought to have their tier-i capital at 9.5 per ce
```

```
def baseline_summary_pipeline(text):
  sentences = split_sentences(text)
  summary = textrank(sentences, top_n=5)
  return summary


pip install rouge-score
```

```
from rouge_score import rouge_scorer

scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=Tru

rouge1_scores = []
rouge2_scores = []
rougeL_scores = []

for entry in b_filtered_ds.select(range(20)):
    generated = baseline_summary_pipeline(entry['Content'])  # your generated sum
    reference = entry['Summary']  # human reference summary

    score = scorer.score(reference, generated)

    rouge1_scores.append(score['rouge1'].fmeasure)
    rouge2_scores.append(score['rouge2'].fmeasure)
    rougeL_scores.append(score['rougeL'].fmeasure)

print("Average ROUGE-1 F1:", sum(rouge1_scores)/len(rouge1_scores))
print("Average ROUGE-2 F1:", sum(rouge2_scores)/len(rouge2_scores))
print("Average ROUGE-L F1:", sum(rougeL_scores)/len(rougeL_scores))
```

```
Average ROUGE-1 F1: 0.3117687753680703
Average ROUGE-2 F1: 0.18014261244658297
Average ROUGE-L F1: 0.22399678411130425
```

```
len(b_filtered_ds)
```

```
26233
```

Keyword Extraction

Tokenize and filter keywords

we will use nltk to tokenize the text, then remove stopwords and punctuation.

```
import nltk
nltk.data.path.append(r"C:\Users\qq258\AppData\Roaming\nltk_data")
from nltk.corpus import stopwords
from nltk import pos_tag, word_tokenize



nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')

def extract_pos_filtered_tokens(text):
    stop_words = set(stopwords.words('english'))
    words = word_tokenize(text.lower())
    words = [w for w in words if w.isalpha() and w not in stop_words]

    tagged = pos_tag(words)
    # Keep only nouns and adjectives
    keep_tags = {'NN', 'NNS', 'NNP', 'NNPS', 'JJ', 'JJR', 'JJS'}
    filtered = [word for word, tag in tagged if tag in keep_tags]

    return filtered
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\qq258\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\qq258\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\qq258\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Build co-occurrence graph

Create edges between words that co-occur within a fixed window size (e.g., 2–4), then use networkx to build and rank nodes.

```
import networkx as nx
from itertools import combinations

def build_cooccurrence_graph(words, window_size=4):
    graph = nx.Graph()
    for i in range(len(words)):
        for j in range(i+1, min(i+window_size, len(words))):
            if words[i] != words[j]:
                graph.add_edge(words[i], words[j])
    return graph
```

Apply pagerank to rank keywords

```
def extract_keywords(text, top_n=10):
    tokens = extract_pos_filtered_tokens(text)
    graph = build_cooccurrence_graph(tokens)
    scores = nx.pagerank(graph)
    ranked = sorted(scores.items(), key=lambda x: x[1], reverse=True)
    return [word for word, _ in ranked[:top_n]]


text = b_filtered_ds[1]['Content']
keywords = extract_keywords(text, top_n=10)
print("Extracted Keywords:", keywords)
```

```
Extracted Keywords: ['capital', 'banks', 'rs', 'government', 'report', 'fisca
```

Accumulate all keywords across the dataset and find the top 100 most frequent keywords

```
from collections import Counter

all_keywords = []

# Iterate over each summary in your DataFrame
for summary in df["Summary"]:
    keywords = extract_keywords(summary, top_n=10)  # your existing function
    all_keywords.extend(keywords)  # collect all keywords

# Count frequency of each keyword
keyword_freq = Counter(all_keywords)

# Get top 100 most common keywords across all summaries
top_100_keywords = [word for word, _ in keyword_freq.most_common(100)]

print(top_100_keywords)
```

```
['government', 'rs', 'india', 'market', 'cent', 'company', 'crore', 'economy',
```

Image Lookup Table

```
import requests
import os

def download_image_from_pexels(keyword, save_dir="./images"):
    os.makedirs(save_dir, exist_ok=True)

    headers = {
        "Authorization": "API_KEY"
    }

    response = requests.get(
        "https://api.pexels.com/v1/search",
        headers=headers,
        params={"query": f"insurance {keyword}", "per_page": 1}
    )


    if response.status_code == 200:
        data = response.json()
        photo_url = data["photos"][0]["src"]["original"]  # use 'medium' or 'smal
        image_data = requests.get(photo_url).content
        file_path = os.path.join(save_dir, f"{keyword}.jpg")

        with open(file_path, "wb") as f:
            f.write(image_data)

        print(f"Downloaded image for '{keyword}' to {file_path}")
    else:
        print(f"Failed to fetch image for {keyword} — HTTP {response.status_code}

keywords = ['bank', 'government', 'capital', 'market']  # example
keywords1 = ['government', 'rs', 'india', 'market', 'cent', 'company', 'crore', '
             'growth', 'new', 'markets', 'percent', 'vaccine', 'pandemic', 'globa
             'points', 'last', 'investors', 'companies', 'economic',
             'coronavirus', 'world', 'oil', 'sector', 'herd', 'index', 'trade',
             'years', 'people', 'prices', 'lakh', 'sensex', 'indian', 'rate', 'nu
             'stocks', 'business', 'financial', 'high', 'natural', 'lockdown', 't
             'infection', 'nifty', 'banks', 'investment', 'months', 'quarter', 'w
             'industry', 'march', 'higher', 'due', 'dollar', 'china', 'day', 'spo
             'tax', 'stock', 'capital', 'fund', 'demand', 'minister', 'state', 'r
             'top', 'month', 'gold', 'price', 'equity', 'operations', 'likely', '
             'fiscal', 'funds', 'key', 'total', 'domestic', 'major', 'countries',
             'report', 'first', 'interest', 'strong', 'shares', 'sales', 'rupee',
```

```
            'states', 'women', 'end', 'gdp', 'many', 'trading', 'recovery', 'pos
for word in keywords1:
    download_image_from_pexels(word)
```

```
Downloaded image for 'government' to ./images\government.jpg
Downloaded image for 'rs' to ./images\rs.jpg
Downloaded image for 'india' to ./images\india.jpg
Downloaded image for 'market' to ./images\market.jpg
Downloaded image for 'cent' to ./images\cent.jpg
Downloaded image for 'company' to ./images\company.jpg
Downloaded image for 'crore' to ./images\crore.jpg
Downloaded image for 'economy' to ./images\economy.jpg
Downloaded image for 'bank' to ./images\bank.jpg
Downloaded image for 'year' to ./images\year.jpg
Downloaded image for 'growth' to ./images\growth.jpg
Downloaded image for 'new' to ./images\new.jpg
Downloaded image for 'markets' to ./images\markets.jpg
Downloaded image for 'percent' to ./images\percent.jpg
Downloaded image for 'vaccine' to ./images\vaccine.jpg
Downloaded image for 'pandemic' to ./images\pandemic.jpg
Downloaded image for 'global' to ./images\global.jpg
Downloaded image for 'country' to ./images\country.jpg
Downloaded image for 'points' to ./images\points.jpg
Downloaded image for 'last' to ./images\last.jpg
Downloaded image for 'investors' to ./images\investors.jpg
Downloaded image for 'companies' to ./images\companies.jpg
Downloaded image for 'economic' to ./images\economic.jpg
Downloaded image for 'coronavirus' to ./images\coronavirus.jpg
Downloaded image for 'world' to ./images\world.jpg
Downloaded image for 'oil' to ./images\oil.jpg
Downloaded image for 'sector' to ./images\sector.jpg
Downloaded image for 'herd' to ./images\herd.jpg
Downloaded image for 'index' to ./images\index.jpg
Downloaded image for 'trade' to ./images\trade.jpg
Downloaded image for 'years' to ./images\years.jpg
Downloaded image for 'people' to ./images\people.jpg
Downloaded image for 'prices' to ./images\prices.jpg
Downloaded image for 'lakh' to ./images\lakh.jpg
Downloaded image for 'sensex' to ./images\sensex.jpg
Downloaded image for 'indian' to ./images\indian.jpg
Downloaded image for 'rate' to ./images\rate.jpg
Downloaded image for 'number' to ./images\number.jpg
Downloaded image for 'stocks' to ./images\stocks.jpg
Downloaded image for 'business' to ./images\business.jpg
Downloaded image for 'financial' to ./images\financial.jpg
Downloaded image for 'high' to ./images\high.jpg
Downloaded image for 'natural' to ./images\natural.jpg
Downloaded image for 'lockdown' to ./images\lockdown.jpg
Downloaded image for 'time' to ./images\time.jpg
Downloaded image for 'infection' to ./images\infection.jpg
```

```
Downloaded image for 'nifty' to ./images\nifty.jpg
Downloaded image for 'banks' to ./images\banks.jpg
Downloaded image for 'investment' to ./images\investment.jpg
Downloaded image for 'months' to ./images\months.jpg
Downloaded image for 'quarter' to ./images\quarter.jpg
Downloaded image for 'week' to ./images\week.jpg
Downloaded image for 'industry' to ./images\industry.jpg
Downloaded image for 'march' to ./images\march.jpg
Downloaded image for 'higher' to ./images\higher.jpg
Downloaded image for 'due' to ./images\due.jpg
Downloaded image for 'dollar' to ./images\dollar.jpg
Downloaded image for 'china' to ./images\china.jpg
Downloaded image for 'day' to ./images\day.jpg
Downloaded image for 'spokesman' to ./images\spokesman.jpg
```

The images folder is too big to be uploaded, so we saved them in a google drive folder. Here is a link:

https://drive.google.com/drive/folders/1pk4AJnD0eAIziylF7lWwn2snwX2D0dEj?usp=sharing

Now we are able to download images based on the keywords. below is a plot function that will plot out the downloaded images that correspond to the keywords.

```
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Display images corresponding to the given list of keywords.
def show_keyword_images(keywords, image_dir="./images"):

    image_files = [f"{word}.jpg" for word in keywords if os.path.exists(os.path.joi

    if not image_files:
        print("No matching images found.")
        return

    cols = 5
    rows = (len(image_files) + cols - 1) // cols  # Round up

    plt.figure(figsize=(15, 3 * rows))

    for i, img_file in enumerate(image_files):
        img_path = os.path.join(image_dir, img_file)
        img = mpimg.imread(img_path)

        plt.subplot(rows, cols, i + 1)
        plt.imshow(img)
        plt.axis("off")
        plt.title(img_file.split(".")[0])

    plt.tight_layout()
    plt.show()
```

```
show_keyword_images(keywords)
```