```
pip install datasets
```

Show hidden output

```
!pip install huggingface_hub datasets
```

Show hidden output

```
from huggingface_hub import hf_hub_download
from huggingface_hub import login

login()

import json
```

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

```
Cloning into 'Indian_Financial_News'...
remote: Enumerating objects: 10, done.
remote: Total 10 (delta 0), reused 0 (delta 0), pack-reused 10 (from 1)
Unpacking objects: 100% (10/10), 3.06 KiB | 1.02 MiB/s, done.
```

```
import pandas as pd
from datasets import Dataset

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

```
Index(['URL', 'Content', 'Summary', 'Sentiment'], dtype='object')
```

```
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))
```

Show hidden output

```
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: 100%                                    26961/26961 [00:02<00:00, 11132.43 examples/s]
Filter: 100%                                    26254/26254 [00:01<00:00, 11957.61 examples/
```

```
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

```
import re
import nltk

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
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: 100% 26233/26233 [00:21<00:00, 1466.37 examples/

```
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: 100% 27/27 [00:03<00:00, 7.95ba/s]

```
113592985
```

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

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 /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

```
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')
```

```
Mounted at /content/drive
```

```
# 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 nex
Summary:
 as per the norms, the banks ought to have their tier-i capital at 9.5 per cer
```

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

```
pip install rouge-score
```

Show hidden output

```
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
```