NLP LAB 1

Ankit Rathwa - 202001190

code :

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

```
import tarfile
import os

# Path to the .tgz file in Google Drive
path_to_tgz = '/content/drive/MyDrive/nlp/cnn_stories.tgz'

# Path to save the extracted files
extracted_folder_path = '/content/Extracted/'

# Create the extracted folder if it doesn't exist
os.makedirs(extracted_folder_path, exist_ok=True)

# Extract the .tgz file
with tarfile.open(path_to_tgz, 'r:gz') as tar:
    tar.extractall(extracted_folder_path)
```

```
mport tarfile
import os

# Path to the .tgz file in Google Drive
path_to_tgz = '/content/drive/MyDrive/nlp/dailymail_stories.tgz'

# Path to save the extracted files
extracted_folder_path = '/content/Extracted/'

# Create the extracted folder if it doesn't exist
```

```
os.makedirs(extracted_folder_path, exist_ok=True)

# Extract the .tgz file
with tarfile.open(path_to_tgz, 'r:gz') as tar:
    tar.extractall(extracted_folder_path)
```

```
mport os

# Path to the extracted folder
extracted_folder_path = '/content/Extracted/cnn/stories/'

# List files in the extracted folder
extracted_files = os.listdir(extracted_folder_path)

# Print the list of extracted files
print("Extracted Files:")
for file_name in extracted_files:
    print(file_name)
```

```
import os

# Path to the extracted folder
extracted_folder_path = '/content/Extracted/dailymail/stories/'

# List files in the extracted folder
extracted_files = os.listdir(extracted_folder_path)

# Print the list of extracted files
print("Extracted Files:")
for file_name in extracted_files:
    print(file_name)
```

```
import os
import spacy
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from multiprocessing import Pool

# Load spaCy's English model
```

```python
nlp = spacy.load("en_core_web_sm")

# Function to extract noun phrases using spaCy
def extract_noun_phrases(text):
    doc = nlp(text)
    noun_phrases = [chunk.text for chunk in doc.noun_chunks]
    return ' '.join(noun_phrases)

# Function to process a single document and extract noun phrases
def process_document(file_path):
    with open(file_path, 'r') as file:
        text = file.read()
        noun_phrases_text = extract_noun_phrases(text)
        return noun_phrases_text

# Function to compute cosine similarity between noun phrases
def compute_cosine_similarity(noun_phrases_list):
    # Initialize TF-IDF vectorizer
    tfidf_vectorizer = TfidfVectorizer()

    # Fit and transform the text to obtain the TF-IDF matrix
    tfidf_matrix = tfidf_vectorizer.fit_transform(noun_phrases_list)

    # Compute cosine similarity between all pairs of noun phrases
    cosine_similarities = cosine_similarity(tfidf_matrix)
    return cosine_similarities

# Path to the directory containing the documents
documents_dir1 = '/content/Extracted/cnn/stories'

# List files in the directory and select the first 1000 files
file_names1 = [os.path.join(documents_dir1, file_name) for file_name in
os.listdir(documents_dir1) if file_name.endswith('.story')][:1000]

# Process documents and extract noun phrases using parallel processing
with Pool() as pool:
    noun_phrases_list1 = pool.map(process_document, file_names1)

# Output all extracted noun phrases
print("Extracted Noun Phrases:")
```

```
for noun_phrases_text in noun_phrases_list1:
    print(noun_phrases_text)
```

```
# Compute cosine similarities between all pairs of noun phrases
cosine_similarities1 = compute_cosine_similarity(noun_phrases_list1)

# Output cosine similarities
print("\nCosine Similarities:")
for i in range(len(cosine_similarities1)):
    for j in range(i+1, len(cosine_similarities1[i])):
        similarity = cosine_similarities1[i, j]
        print(f"Cosine similarity between noun phrases {i} and {j}:
{similarity}")
```

```
import os
import spacy
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from multiprocessing import Pool

# Load spaCy's English model
nlp = spacy.load("en_core_web_sm")

# Function to extract noun phrases using spaCy
def extract_noun_phrases(text):
    doc = nlp(text)
    noun_phrases = [chunk.text for chunk in doc.noun_chunks]
    return ' '.join(noun_phrases)

# Function to process a single document and extract noun phrases
def process_document(file_path):
    with open(file_path, 'r') as file:
        text = file.read()
        noun_phrases_text = extract_noun_phrases(text)
        return noun_phrases_text

# Function to compute cosine similarity between noun phrases
def compute_cosine_similarity(noun_phrases_list):
    # Initialize TF-IDF vectorizer
```

```
    tfidf_vectorizer = TfidfVectorizer()

    # Fit and transform the text to obtain the TF-IDF matrix
    tfidf_matrix = tfidf_vectorizer.fit_transform(noun_phrases_list)

    # Compute cosine similarity between all pairs of noun phrases
    cosine_similarities = cosine_similarity(tfidf_matrix)
    return cosine_similarities

# Path to the directory containing the documents
documents_dir2 = '/content/Extracted/dailymail/stories'

# List files in the directory and select the first 1000 files
file_names2 = [os.path.join(documents_dir2, file_name) for file_name in
os.listdir(documents_dir2) if file_name.endswith('.story')][:1000]

# Process documents and extract noun phrases using parallel processing
with Pool() as pool:
    noun_phrases_list2 = pool.map(process_document, file_names2)

# Output all extracted noun phrases
print("Extracted Noun Phrases:")
for noun_phrases_text in noun_phrases_list2:
    print(noun_phrases_text)
```

```
# Compute cosine similarities between all pairs of noun phrases
cosine_similarities2 = compute_cosine_similarity(noun_phrases_list2)

# Output cosine similarities
print("\nCosine Similarities:")
for i in range(len(cosine_similarities2)):
    for j in range(i+1, len(cosine_similarities2[i])):
        similarity = cosine_similarities2[i, j]
        print(f"Cosine similarity between noun phrases {i} and {j}:
{similarity}")
```

Outputs for noun-phraases in cnn stories :

```
for noun_phrases_text in noun_phrases_list:
    print(noun_phrases_text)

Extracted Noun Phrases:
(CNN The great Davos talking shop delegates all levels importance shapes nationalities this quaint , alpine town the world rights many ordinary peop

The World Economic Forum invitation 2,500 participants over 100 countries The roll call CEOs representatives the world's biggest corporations politi

@highlight

World Economic Forum invitation @highlight a business each delegate Davos accomodation the pocket @highlight a helicopter Zurich Davos
(CNN Rick Perlstein his book Paranoia Richard Nixon the guy who these tensions a new kind politics Rick Perlstein Perlstein's history the 1960s earl

"Nixonland the ' 60s Richard Nixon who era's tensions his ends

@highlight

Book politics and pop culture gaps @highlight We Nixonland author
Washington (CNN)The massacre some France's top cartoonists a American nightmare Western jihadists war terror battlefields carnage home reality the t

Paris attack underscores spectacular, large-scale attacks foreign terror groups homegrown radicals jihadist organizations more limited strikes
(CNN ethical shopping a star it Danny Seo the celebrity eco stylist who green consumerism the pages Organic Style magazine he editor his books TV ap

Danny Seo the 'new Martha Stewart the green generation

@highlight

Upbeat practical advice Hollywood stars

@highlight 'Think his advice a cheaper and better option
2-year-old Riley Ann Sawyers helts her hair the room water an affidavit the Galveston County Sheriff's Office Police 2-year-old Riley Ann Sawyers Ba
```

Connected to Python 3 Google Compute Engine backend

Outputs for cosine similarities among noun-phraases in cnn stories :

```
print(f'Cosine similarity between noun phrases {i} and {j}: {similarity}')

Streaming output truncated to the last 5000 lines.
Cosine similarity between noun phrases 98 and 201: 0.04978188136740312
Cosine similarity between noun phrases 98 and 202: 0.05209788625826228
Cosine similarity between noun phrases 98 and 203: 0.057025458883218326
Cosine similarity between noun phrases 98 and 204: 0.06668678878939725
Cosine similarity between noun phrases 98 and 205: 0.057775718073279685
Cosine similarity between noun phrases 98 and 206: 0.06330325941840034
Cosine similarity between noun phrases 98 and 207: 0.0557219931032107
Cosine similarity between noun phrases 98 and 208: 0.04905457190510986
Cosine similarity between noun phrases 98 and 209: 0.06658270201139865
Cosine similarity between noun phrases 98 and 210: 0.07000306001650991
Cosine similarity between noun phrases 98 and 211: 0.04778970693882325
Cosine similarity between noun phrases 98 and 212: 0.06532723592487706
Cosine similarity between noun phrases 98 and 213: 0.044260957678368296
Cosine similarity between noun phrases 98 and 214: 0.029848325574256655
Cosine similarity between noun phrases 98 and 215: 0.05946750130430663
Cosine similarity between noun phrases 98 and 216: 0.08144777067278695
Cosine similarity between noun phrases 98 and 217: 0.037107042703269415
Cosine similarity between noun phrases 98 and 218: 0.044440405079945886
Cosine similarity between noun phrases 98 and 219: 0.061880712314178965
Cosine similarity between noun phrases 98 and 220: 0.07090015352769197
Cosine similarity between noun phrases 98 and 221: 0.04088164482774044
Cosine similarity between noun phrases 98 and 222: 0.06664829371789814
Cosine similarity between noun phrases 98 and 223: 0.05582209720565299
Cosine similarity between noun phrases 98 and 224: 0.07851146310680757
Cosine similarity between noun phrases 98 and 225: 0.07495417929614323
Cosine similarity between noun phrases 98 and 226: 0.11106527628274902
Cosine similarity between noun phrases 98 and 227: 0.09617752716046962
Cosine similarity between noun phrases 98 and 228: 0.06781449823844958
```

Connected to Python 3 Google Compute Engine backend

Outputs for noun-phraases in dailymail stories :

```
Extracted Noun Phrases:
Mark Duell Matt Chorley Mailonline Political Editor :

08:22 EST 20 March 2013


| 13:36 EST 20 March 2013

Millions Britain's hard-pressed taxpayers themselves the coalition its flagship tax cut Chancellor George Osborne the amount money that workers they

Rising trend This personal allowance graph figures the Institute Fiscal Studies website a person it It part a bid the burden squeezed family budgets
decision VAT 17.5 per cent 20 per cent the coalition the
Tories Lib Dems the personal allowance this a
longer term policy objective the income tax threshold April the average worker The Government the policy other tax cuts the Tory policy Inheritance
personal allowance inflation the Exchequer £500million free market the Adam
Smith Institute Mr Osborne the allowance which full-time minimum wage workers any
tax It a report a
£12,875 personal allowance the take-home pay all those up to 1,297,000 people the tax system The institute's policy director Sam
Bowman It a national scandal we the people the
bottom society they a basic
standard living Payments April the average taxpayer £705 less tax the 2010 general election Meanwhile hopes the Chancellor the strain savers recent
cash four years ultra-low interest rates living costs wages Almost a dozen tax loopholes the government those who payments the exchequer the Chancel
website Moneyfacts the impact inflation savings average interest tax 20 per cent the spending power Former Government policy adviser Ros
Altmann who director-general Saga the rules tax-free Isas people all their allowance cash Isas they the current system only half the £11,280 allowan
remainder stocks shares Dr Altmann Older people who their savings the stock
market younger generations who a house deposit It the
dramatic drop savings interest rates the Government savers them cash other
assets Mr Osborne that corporation tax 20 per cent He the headline rate 28 to 24 per cent it cent 21 per cent April he corporation tax 20 per cent i
cutbacks public spending a fall labour productivity a
Connected to Python 3 Google Compute Engine backend
```

Outputs for cosine similarities among noun-phraases in dailmail stories :

```
Streaming output truncated to the last 5000 lines.
Cosine similarity between noun phrases 61 and 304: 0.04954440345067586
Cosine similarity between noun phrases 61 and 305: 0.038187357971115006
Cosine similarity between noun phrases 61 and 306: 0.029037189088138203
Cosine similarity between noun phrases 61 and 307: 0.06952623307989433
Cosine similarity between noun phrases 61 and 308: 0.06701341773770296
Cosine similarity between noun phrases 61 and 309: 0.053808198534256925
Cosine similarity between noun phrases 61 and 310: 0.05514518867863258
Cosine similarity between noun phrases 61 and 311: 0.04594606243630534
Cosine similarity between noun phrases 61 and 312: 0.0516978942711775
Cosine similarity between noun phrases 61 and 313: 0.061807915657465656
Cosine similarity between noun phrases 61 and 314: 0.08441397243468785
Cosine similarity between noun phrases 61 and 315: 0.04160734755105666
Cosine similarity between noun phrases 61 and 316: 0.0386090445971357
Cosine similarity between noun phrases 61 and 317: 0.050319428887923415
Cosine similarity between noun phrases 61 and 318: 0.06467952952793624
Cosine similarity between noun phrases 61 and 319: 0.039346566206513965
Cosine similarity between noun phrases 61 and 320: 0.06847769230986281
Cosine similarity between noun phrases 61 and 321: 0.0513990214619156
Cosine similarity between noun phrases 61 and 322: 0.046179748958685964
Cosine similarity between noun phrases 61 and 323: 0.05253135265341709
Cosine similarity between noun phrases 61 and 324: 0.024556593643118504
Cosine similarity between noun phrases 61 and 325: 0.030114311128986094
Cosine similarity between noun phrases 61 and 326: 0.05477987118556557
Cosine similarity between noun phrases 61 and 327: 0.03528177784858059
Cosine similarity between noun phrases 61 and 328: 0.03486954206997862
Cosine similarity between noun phrases 61 and 329: 0.024598978415634257
Cosine similarity between noun phrases 61 and 330: 0.07055585063977915
Cosine similarity between noun phrases 61 and 331: 0.05878651362824487
Cosine similarity between noun phrases 61 and 332: 0.05278982458710728
Connected to Python 3 Google Compute Engine backend
```