NLP LAB 2

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
```

```
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}")
```

```
import os
import spacy
```

```
from collections import Counter
from gensim.models import Word2Vec

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

# Function to extract phrases using spaCy and Word2Vec
def extract_phrases(text):
    doc = nlp(text)
    phrases = []
    for sentence in doc.sents:
        phrases.extend([str(chunk) for chunk in sentence.noun_chunks])
    return phrases

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

# 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 phrases using parallel processing
phrases_list1 = []
for file_path in file_names1:
    phrases_list1.extend(process_document(file_path))

# Compute the top-k most frequent phrases
def top_k_phrases(phrases_list1, k):
    phrase_counts = Counter(phrases_list1)
    top_k = phrase_counts.most_common(k)
    return top_k

# Set the value of k for top-k phrases
```

```
k = 100

# Output the top-k most frequent phrases
top_k_phrases_list1 = top_k_phrases(phrases_list1, k)
print(f"Top-{k} Most Frequent Phrases:")
for phrase, count in top_k_phrases_list1:
    print(f"{phrase}: {count} occurrences")
```

```
import os
import spacy
from collections import Counter
from gensim.models import Word2Vec

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

# Function to extract phrases using spaCy and Word2Vec
def extract_phrases(text):
    doc = nlp(text)
    phrases = []
    for sentence in doc.sents:
        phrases.extend([str(chunk) for chunk in sentence.noun_chunks])
    return phrases

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

# 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 phrases using parallel processing
phrases_list2 = []
for file_path in file_names2:
    phrases_list1.extend(process_document(file_path))

# Compute the top-k most frequent phrases
def top_k_phrases(phrases_list2, k):
    phrase_counts = Counter(phrases_list2)
    top_k = phrase_counts.most_common(k)
    return top_k

# Set the value of k for top-k phrases
k = 100

# Output the top-k most frequent phrases
top_k_phrases_list2 = top_k_phrases(phrases_list2, k)
print(f"Top-{k} Most Frequent Phrases:")
for phrase, count in top_k_phrases_list1:
    print(f"{phrase}: {count} occurrences")
```

Outputs for topk most frequent phrases inin cnn stories :

```
Top-100 Most Frequent Phrases:
it: 3922 occurrences
he: 3876 occurrences
I: 3371 occurrences
that: 2844 occurrences
who: 2254 occurrences
they: 1873 occurrences
she: 1722 occurrences
you: 1545 occurrences
It: 1486 occurrences
we: 1466 occurrences
He: 1301 occurrences
which: 1278 occurrences
him: 923 occurrences
what: 914 occurrences
them: 893 occurrences
We: 813 occurrences
(CNN: 618 occurrences
people: 587 occurrences
```

She: 584 occurrences
CNN: 528 occurrences
They: 521 occurrences
me: 518 occurrences
this: 461 occurrences
@highlight: 452 occurrences
her: 390 occurrences
us: 375 occurrences
You: 340 occurrences
That: 331 occurrences
the United States: 324 occurrences
those: 311 occurrences
all: 298 occurrences
the world: 276 occurrences
-: 270 occurrences
Obama: 268 occurrences
something: 260 occurrences
This: 257 occurrences
police: 257 occurrences
the country: 250 occurrences
part: 243 occurrences
some: 241 occurrences
this report: 230 occurrences
a statement: 217 occurrences
What: 210 occurrences
women: 189 occurrences
Washington: 188 occurrences
authorities: 183 occurrences
Syria: 180 occurrences
a lot: 178 occurrences
time: 174 occurrences
the government: 165 occurrences
the time: 161 occurrences
China: 155 occurrences
the case: 154 occurrences
Iraq: 152 occurrences
anything: 141 occurrences
Congress: 139 occurrences
place: 138 occurrences
nothing: 135 occurrences
himself: 134 occurrences
Iran: 133 occurrences
others: 130 occurrences
the end: 129 occurrences
someone: 128 occurrences
reporters: 126 occurrences
officials: 126 occurrences

children: 125 occurrences
Afghanistan: 125 occurrences
New York: 119 occurrences
everything: 118 occurrences
Tuesday: 116 occurrences
anyone: 113 occurrences
the city: 113 occurrences
things: 112 occurrences
Thursday: 110 occurrences
life: 110 occurrences
London: 109 occurrences
California: 109 occurrences
the report: 108 occurrences
themselves: 107 occurrences
America: 106 occurrences
Monday: 105 occurrences
Republicans: 104 occurrences
information: 104 occurrences
North Korea: 103 occurrences
Friday: 102 occurrences
the state: 101 occurrences
June: 101 occurrences
Police: 95 occurrences
the way: 94 occurrences
March: 93 occurrences
president: 92 occurrences
Texas: 92 occurrences
Mexico: 90 occurrences
India: 90 occurrences
Sunday: 89 occurrences
the people: 89 occurrences
Democrats: 89 occurrences
Some: 88 occurrences
everyone: 88 occurrences
the day: 88 occurrences

Outputs for top k most frequent phraases in dailymail stories :

```
Top-100 Most Frequent Phrases:
it: 3922 occurrences
he: 3876 occurrences
I: 3371 occurrences
that: 2844 occurrences
who: 2254 occurrences
they: 1873 occurrences
she: 1722 occurrences
you: 1545 occurrences
It: 1486 occurrences
we: 1466 occurrences
He: 1301 occurrences
which: 1278 occurrences
him: 923 occurrences
what: 914 occurrences
them: 893 occurrences
We: 813 occurrences
(CNN: 618 occurrences
people: 587 occurrences
She: 584 occurrences
CNN: 528 occurrences
They: 521 occurrences
me: 518 occurrences
this: 461 occurrences
@highlight: 452 occurrences
her: 390 occurrences
us: 375 occurrences
You: 340 occurrences
That: 331 occurrences
the United States: 324 occurrences
those: 311 occurrences
all: 298 occurrences
the world: 276 occurrences
-: 270 occurrences
Obama: 268 occurrences
something: 260 occurrences
This: 257 occurrences
police: 257 occurrences
the country: 250 occurrences
part: 243 occurrences
some: 241 occurrences
this report: 230 occurrences
a statement: 217 occurrences
What: 210 occurrences
women: 189 occurrences
Washington: 188 occurrences
```

authorities: 183 occurrences
Syria: 180 occurrences
a lot: 178 occurrences
time: 174 occurrences
the government: 165 occurrences
the time: 161 occurrences
China: 155 occurrences
the case: 154 occurrences
Iraq: 152 occurrences
anything: 141 occurrences
Congress: 139 occurrences
place: 138 occurrences
nothing: 135 occurrences
himself: 134 occurrences
Iran: 133 occurrences
others: 130 occurrences
the end: 129 occurrences
someone: 128 occurrences
reporters: 126 occurrences
officials: 126 occurrences
children: 125 occurrences
Afghanistan: 125 occurrences
New York: 119 occurrences
everything: 118 occurrences
Tuesday: 116 occurrences
anyone: 113 occurrences
the city: 113 occurrences
things: 112 occurrences
Thursday: 110 occurrences
life: 110 occurrences
London: 109 occurrences
California: 109 occurrences
the report: 108 occurrences
themselves: 107 occurrences
America: 106 occurrences
Monday: 105 occurrences
Republicans: 104 occurrences
information: 104 occurrences
North Korea: 103 occurrences
Friday: 102 occurrences
the state: 101 occurrences
June: 101 occurrences
Police: 95 occurrences
the way: 94 occurrences
March: 93 occurrences
president: 92 occurrences
Texas: 92 occurrences

```
Mexico: 90 occurrences
India: 90 occurrences
Sunday: 89 occurrences
the people: 89 occurrences
Democrats: 89 occurrences
Some: 88 occurrences
everyone: 88 occurrences
the day: 88 occurrences
```