PAREENITA A.SHIRSATH ROLL.NO : 57 B.E.A.I.&D.S.

NLP EXPERIMENT NO : 09

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


documents = [
    "India has some of the most spectacular and unforgettable rail journeys in the world.",
    "Here you experience a simple way to find out everything you need to know in one easy place",
    "There's no better way to enjoy India's outback, cities, coastal towns and regional areas in comfort."
]


vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(documents)


cosine_similarities = cosine_similarity(tfidf_matrix)


for i, doc1 in enumerate(documents):
    for j, doc2 in enumerate(documents):
        if i < j:
            print(f"Similarity:- {cosine_similarities[i, j]:.2f}")
```

```
Similarity:- 0.02
Similarity:- 0.11
Similarity:- 0.13
```

```
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Download necessary NLTK resources
nltk.download('stopwords')
nltk.download('punkt')

# Initialize stemmer and stopwords list
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()

# Preprocess text
def preprocess_text(text):
    text = text.lower()  # Convert to lowercase
    text = text.translate(str.maketrans('', '', string.punctuation))  # Remove punctuation
    tokens = text.split()  # Tokenization
    tokens = [stemmer.stem(word) for word in tokens if word not in stop_words]  # Remove stopwords and stem
    return " ".join(tokens)

# Take multiple documents as input
num_documents = int(input("Enter the number of documents: "))
documents = []

for i in range(num_documents):
    doc_text = input(f"Enter text for Document {i+1}: ")
    documents.append(doc_text)

# Preprocess documents
preprocessed_documents = [preprocess_text(doc) for doc in documents]

# Print preprocessed documents for debugging
print("\nPreprocessed Documents:")
for idx, doc in enumerate(preprocessed_documents, 1):
    print(f"Document {idx}: {doc}")

# TF-IDF vectorization
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(preprocessed_documents)

# Calculate cosine similarity
cosine_similarities = cosine_similarity(tfidf_matrix, tfidf_matrix)

# Print all cosine similarities for debugging
print("\nCosine Similarity Matrix:")
for i in range(len(documents)):
```

```
    for j in range(i + 1, len(documents)):
        print(f"Cosine Similarity between Document {i+1} and Document {j+1}: {cosine_similarities[i][j]:.2f}")

# Print pairs of documents that are similar above the threshold
print("\nSimilar Document Pairs:")
similarity_threshold = 0.4  # Lower the threshold to capture more similarities
similar_pairs = []

for i in range(len(documents)):
    for j in range(i + 1, len(documents)):
        if cosine_similarities[i][j] >= similarity_threshold:
            similar_pairs.append((i, j, cosine_similarities[i][j]))

if len(similar_pairs) == 0:
    print("No similar documents found above the threshold.")
else:
    for pair in similar_pairs:
        doc1, doc2, similarity = pair
        print(f"Document {doc1 + 1} and Document {doc2 + 1} are similar (Cosine Similarity: {similarity:.2f})")
```

```
[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!
Enter the number of documents: 3
Enter text for Document 1: THIS IS MY FIRST REPORT
Enter text for Document 2: THIS IS MY SECOND REPORT
Enter text for Document 3: THIS IS MY THIRD REPORT

Preprocessed Documents:
Document 1: first report
Document 2: second report
Document 3: third report

Cosine Similarity Matrix:
Cosine Similarity between Document 1 and Document 2: 0.26
Cosine Similarity between Document 1 and Document 3: 0.26
Cosine Similarity between Document 2 and Document 3: 0.26

Similar Document Pairs:
No similar documents found above the threshold.
```