```
# 1import libraries requried
import nltk
import re
import numpy as np
import pandas as pd

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

from nltk.corpus import stopwords, wordnet
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# Download required NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
True
```

```
# 2Prepare Dataset
documents = [
    "Machine learning is a branch of artificial intelligence",
    "Artificial intelligence enables machines to think",
    "Deep learning is a subset of machine learning",
    "Natural language processing deals with text data",
    "NLP helps computers understand human language",
    "Python is widely used in data science",
    "Data science involves statistics and programming",
    "A doctor treats patients in a hospital",
    "A physician works in a medical clinic",
    "Students learn programming in computer science"
]

df = pd.DataFrame(documents, columns=["Text"])
print("Dataset Sample:")
print(df.head())
```

```
Dataset Sample:
                                             Text
0  Machine learning is a branch of artificial int...
1  Artificial intelligence enables machines to think
2      Deep learning is a subset of machine learning
```

```
3   Natural language processing deals with text data
4      NLP helps computers understand human language
```

```
# 3Text Preprocessing
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    text = text.lower()                          # Lowercasing
    text = re.sub(r'[^a-z\s]', '', text)         # Remove punctuation & numbers
    tokens = word_tokenize(text)                 # Tokenization
    tokens = [word for word in tokens if word not in stop_words]  # Stopword removal
    tokens = [lemmatizer.lemmatize(word) for word in tokens]      # Lemmatization
    return " ".join(tokens)

df["Processed_Text"] = df["Text"].apply(preprocess_text)
print("\nPreprocessed Text Sample:")
print(df.head())
```

```

Preprocessed Text Sample:
                                                Text  \
0  Machine learning is a branch of artificial int...
1  Artificial intelligence enables machines to think
2      Deep learning is a subset of machine learning
3   Natural language processing deals with text data
4      NLP helps computers understand human language

                                    Processed_Text
0  machine learning branch artificial intelligence
1    artificial intelligence enables machine think
2            deep learning subset machine learning
3       natural language processing deal text data
4      nlp help computer understand human language
```

```
# Text Representation
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(df["Processed_Text"])
```

```
# 5Cosine similarity
cosine_sim = cosine_similarity(tfidf_matrix)

print("\nCosine Similarity Matrix:")
print(np.round(cosine_sim, 2))

print("\nExample Interpretation:")
print("Sentence 1 & 3 similarity:",
      cosine_sim[0][2])
print("Higher value indicates more similar meaning.")
```

```

Cosine Similarity Matrix:
[[1.   0.52 0.44 0.   0.   0.   0.   0.   0.   0.  ]
 [0.52 1.   0.12 0.   0.   0.   0.   0.   0.   0.  ]
 [0.44 0.12 1.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   1.   0.13 0.12 0.12 0.   0.   0.  ]
 [0.   0.   0.   0.13 1.   0.   0.   0.   0.   0.15]
 [0.   0.   0.   0.12 0.   1.   0.28 0.   0.   0.14]
 [0.   0.   0.   0.12 0.   0.28 1.   0.   0.   0.33]
 [0.   0.   0.   0.   0.   0.   0.   1.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   1.   0.  ]
 [0.   0.   0.   0.   0.15 0.14 0.33 0.   0.   1.  ]]

Example Interpretation:
Sentence 1 & 3 similarity: 0.44402467338850365
Higher value indicates more similar meaning.
```

```
# 6Jaccard similarity
def jaccard_similarity(text1, text2):
    set1 = set(text1.split())
    set2 = set(text2.split())
    intersection = set1.intersection(set2)
    union = set1.union(set2)
    return len(intersection) / len(union)

print("\nJaccard Similarity Examples:")
for i in range(5):
    score = jaccard_similarity(df["Processed_Text"][0],
                               df["Processed_Text"][i])
    print(f"Sentence 0 & {i}:", round(score, 2))
```

```

Jaccard Similarity Examples:
Sentence 0 & 0: 1.0
Sentence 0 & 1: 0.43
Sentence 0 & 2: 0.29
Sentence 0 & 3: 0.0
Sentence 0 & 4: 0.0
```

```
# 7WordNet-based Semantic Similarity
def wordnet_similarity(word1, word2):
    syns1 = wordnet.synsets(word1)
    syns2 = wordnet.synsets(word2)
    if not syns1 or not syns2:
        return 0
    # Using Wu-Palmer similarity, which measures the depth of the least common subsumer
    # and the depth of the two synsets in the WordNet taxonomy.
    return syns1[0].wup_similarity(syns2[0])

def sentence_wordnet_similarity(sent1, sent2):
    words1 = sent1.split()
    words2 = sent2.split()
    scores = []
    for w1 in words1:
        max_score = 0
        for w2 in words2:
            sim = wordnet_similarity(w1, w2)
            if sim and sim > max_score:
                max_score = sim
        scores.append(max_score)
    return np.mean(scores) if scores else 0

print("\nWordNet Semantic Similarity Examples:")
pairs = [(7, 8), (0, 2), (3, 4)]
```

```
for i, j in pairs:
    score = sentence_wordnet_similarity(
        df["Processed_Text"][i],
        df["Processed_Text"][j]
    )
    print(f"Sentence {i} & {j}:", round(score, 2))
```

```
WordNet Semantic Similarity Examples:
Sentence 7 & 8: 0.55
Sentence 0 & 2: 0.64
Sentence 3 & 4: 0.58
```

```
# 8Comparison Summary
print("\nComparison Summary:")
print("""
Cosine Similarity works well for short text with shared context.
Jaccard Similarity depends heavily on exact word overlap.
WordNet captures semantic meaning even when words differ.
WordNet performs better for synonyms like doctor and physician.
Cosine is widely used due to efficiency and vector-based nature.
""")
```

```
Comparison Summary:

Cosine Similarity works well for short text with shared context.
Jaccard Similarity depends heavily on exact word overlap.
WordNet captures semantic meaning even when words differ.
WordNet performs better for synonyms like doctor and physician.
Cosine is widely used due to efficiency and vector-based nature.
```