```
# Text preprocessing
import nltk
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# Similarity calculations
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# WordNet semantic similarity
from nltk.corpus import wordnet

# Download resources (only first time)
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
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 punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

```
# Example dataset (expand to 20–25 docs)
documents = [
    "Cricket is a popular sport in India.",
    "Football leagues attract millions of fans worldwide.",
    "The parliament passed a new education reform bill.",
    "Elections are vital for democratic governance.",
    "Regular exercise improves cardiovascular health.",
    "Mental health awareness has increased in recent years.",
    "Artificial Intelligence is transforming industries.",
    "5G networks promise faster connectivity and innovation."
]

# Display sample
for i, doc in enumerate(documents[:5]):
    print(f"Doc {i+1}: {doc}")
```

```
Doc 1: Cricket is a popular sport in India.
Doc 2: Football leagues attract millions of fans worldwide.
Doc 3: The parliament passed a new education reform bill.
Doc 4: Elections are vital for democratic governance.
Doc 5: Regular exercise improves cardiovascular health.
```

```
# Preprocessing function
def preprocess(text):
    # Lowercase
    text = text.lower()
    # Remove punctuation & numbers
    text = re.sub(r'[^a-z\s]', '', text)
    # Tokenize
    tokens = word_tokenize(text)
    # Remove stopwords
    tokens = [t for t in tokens if t not in stopwords.words('english')]
    # Lemmatize
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(t) for t in tokens]
    return " ".join(tokens)

# Apply preprocessing
clean_docs = [preprocess(doc) for doc in documents]
print(clean_docs[:5])
```

```
['cricket popular sport india', 'football league attract million fan worldwide',
```

```
# TF-IDF representation
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(clean_docs)

print("TF-IDF Matrix Shape:", tfidf_matrix.shape)
```

```
TF-IDF Matrix Shape: (8, 39)
```

```
cosine_sim = cosine_similarity(tfidf_matrix)

# Display similarity matrix
import pandas as pd
cosine_df = pd.DataFrame(cosine_sim)
print(cosine_df.head())
```

```
     0    1    2    3    4         5    6    7
0  1.0  0.0  0.0  0.0  0.0  0.000000  0.0  0.0
1  0.0  1.0  0.0  0.0  0.0  0.000000  0.0  0.0
2  0.0  0.0  1.0  0.0  0.0  0.000000  0.0  0.0
3  0.0  0.0  0.0  1.0  0.0  0.000000  0.0  0.0
4  0.0  0.0  0.0  0.0  1.0  0.135638  0.0  0.0
```

```
def jaccard_similarity(doc1, doc2):
    set1, set2 = set(doc1.split()), set(doc2.split())
    return len(set1 & set2) / len(set1 | set2)
```

```
# Example comparisons
for i in range(5):
    print(f"Jaccard between Doc1 and Doc{i+2}: {jaccard_similarity(clean_docs[0
```

```
Jaccard between Doc1 and Doc2: 0.0
Jaccard between Doc1 and Doc3: 0.0
Jaccard between Doc1 and Doc4: 0.0
Jaccard between Doc1 and Doc5: 0.0
Jaccard between Doc1 and Doc6: 0.0
```

```
def wordnet_similarity(word1, word2):
    syn1 = wordnet.synsets(word1)
    syn2 = wordnet.synsets(word2)
    if syn1 and syn2:
        return syn1[0].wup_similarity(syn2[0])  # Wu-Palmer similarity
    return None

print("Similarity doctor vs physician:", wordnet_similarity("doctor", "physicia
print("Similarity car vs automobile:", wordnet_similarity("car", "automobile"))
```

```
Similarity doctor vs physician: 1.0
Similarity car vs automobile: 1.0
```