```
import pandas as pd
import numpy as np
import string

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

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

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
documents = [
    "The football match was exciting and full of energy",
    "The cricket team won the championship",
    "The government announced new election dates",
    "The prime minister addressed the nation",
    "Doctors recommend exercise for good health",
    "A healthy diet improves overall fitness",
    "Artificial intelligence is transforming technology",
    "Machine learning powers modern applications",
    "The smartphone uses advanced processor technology",
    "Cyber security is important for data protection",
    "Hospitals are improving patient care",
```

```
    "The parliament passed a new bill",
    "Basketball players train daily",
    "New software updates improve performance",
    "Mental health awareness is increasing",
    "Vaccines help prevent diseases",
    "Political debates influence public opinion",
    "Cloud computing supports scalable systems",
    "Athletes require physical fitness",
    "Technology innovation drives economic growth"
]

df = pd.DataFrame({'Text': documents})
df.head()
```

	Text
0	The football match was exciting and full of en...
1	The cricket team won the championship
2	The government announced new election dates
3	The prime minister addressed the nation
4	Doctors recommend exercise for good health

Next steps: Generate code with df | New interactive sheet

```
nltk.download('punkt_tab')

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    tokens = word_tokenize(text)
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]
```

```
    return " ".join(tokens)

df['Clean_Text'] = df['Text'].apply(preprocess)
df.head()
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

	Text	Clean_Text
0	The football match was exciting and full of en...	football match exciting full energy
1	The cricket team won the championship	cricket team championship
2	The government announced new election dates	government announced new election date
3	The prime minister addressed the nation	prime minister addressed nation
4	Doctors recommend exercise for good health	doctor recommend exercise good health

Next steps: Generate code with df | New interactive sheet

```
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(df['Clean_Text'])
```

```
cos_sim = cosine_similarity(tfidf_matrix)

cosine_df = pd.DataFrame(cos_sim, columns=df.index, index=df.index)
cosine_df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.153493	0.0	0.135906	0.000000	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.182091	0.0	0.0	0.0	0.0	0.0

Next steps: Generate code with `cosine_df` New interactive sheet

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

jaccard_scores = []

for i in range(len(df)):
    for j in range(i+1, len(df)):
        score = jaccard_similarity(df['Clean_Text'][i], df['Clean_Text'][j])
        jaccard_scores.append((i, j, score))

jaccard_scores[:5]
```

```
[(0, 1, 0.0), (0, 2, 0.0), (0, 3, 0.0), (0, 4, 0.0), (0, 5, 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])
    return None
```

```
word_pairs = [
    ('doctor', 'physician'),
    ('football', 'basketball'),
    ('government', 'parliament'),
    ('technology', 'software'),
    ('health', 'fitness'),
    ('disease', 'illness'),
    ('computer', 'machine'),
    ('exercise', 'training'),
    ('election', 'vote'),
    ('security', 'protection')
]

for w1, w2 in word_pairs:
    print(w1, "-", w2, ":", wordnet_similarity(w1, w2))
```

```
doctor - physician : 1.0
football - basketball : 0.782608695652174
government - parliament : 0.5333333333333333
technology - software : 0.23529411764705882
health - fitness : 0.375
disease - illness : 0.9473684210526315
computer - machine : 0.9411764705882353
exercise - training : 0.7058823529411765
election - vote : 0.625
security - protection : 0.2857142857142857
```

Close

```
Cosine similarity performs best for short
documents with overlapping vocabulary.
It considers word importance and frequency, making
it widely used.
Jaccard similarity relies purely on exact word
overlap,
so it fails when synonyms are used.
WordNet similarity captures semantic relationships
between words.
It works well when words differ but meaning is
```

Cosine similarity performs best for short documents with overlapping vocabulary. It considers word importance and frequency, making it widely used. Jaccard similarity relies purely on exact word overlap, so it fails when synonyms are used. WordNet similarity captures semantic relationships between words. It works well when words differ but meaning is related. However, WordNet struggles with full

```
It works well when words differ but meaning is
related.
However, WordNet struggles with full sentence
comparison.
Cosine and Jaccard are lexical methods,
while WordNet is semantic.
The scores disagree when texts use different
vocabulary for same meaning
```

sentence comparison. Cosine and Jaccard are lexical methods, while WordNet is semantic. The scores disagree when texts use different vocabulary for same meaning