```
"""
This cell imports all the required libraries for
text preprocessing, vectorization, and similarity computation.
"""

import numpy as np
import pandas as pd
import re

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

# Feature extraction and similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Download required NLTK datasets
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
```

```
"""
This dataset contains 30 short documents across
four domains: sports, politics, health, and technology.
"""

documents = [

    # -------- SPORTS --------
    "The football team won the final match",
    "Cricket players practiced for the tournament",
    "The athlete broke the world record",
    "Basketball is a popular indoor sport",
    "The coach planned strategies for the game",
    "Olympic players trained daily for excellence",
    "The referee controlled the football match",
    "Fans celebrated the team victory",

    # -------- POLITICS --------
    "The government introduced a new policy",
    "Election campaigns influence voters",
    "The prime minister addressed the public",
    "Political parties debated economic reforms",
    "Parliament passed a new bill",
    "Democracy allows citizens to vote",
    "The minister discussed foreign relations",
    "The opposition criticized the budget",

    # -------- HEALTH --------
    "Doctors recommend exercise for fitness",
    "A balanced diet improves health",
    "Hospitals provide treatment to patients",
    "Mental health awareness is important",
    "Vaccination prevents serious diseases",
    "Healthy sleep improves immunity",
    "Nutrition plays a vital role in wellness",

    # -------- TECHNOLOGY --------
    "Artificial intelligence improves automation",
    "Machine learning analyzes large data",
    "Cybersecurity protects digital systems",
    "Smartphones use advanced processors",
    "Technology evolves rapidly every year",
    "Software development requires logical thinking"
]

# Create DataFrame
df = pd.DataFrame(documents, columns=["Text"])
df.head()
```

1 to 5 of 5 entries Filter

index	Text
0	The football team won the final match
1	Cricket players practiced for the tournament
2	The athlete broke the world record
3	Basketball is a popular indoor sport
4	The coach planned strategies for the game

Show 25 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.

Next steps: Generate code with df New interactive sheet

```
import nltk
nltk.download('punkt_tab')
"""
This step cleans and preprocesses text data by:
- converting to lowercase
- removing punctuation and numbers
- removing stopwords
- tokenizing words
- lemmatizing tokens
"""

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

def preprocess_text(text):
    """
    Preprocesses a given text string.

    Parameters:
    text (str): Input sentence

    Returns:
    str: Cleaned and preprocessed sentence
    """

    # Convert text to lowercase
    text = text.lower()

    # Remove punctuation and numbers
    text = re.sub(r'[^a-z\s]', '', text)

    # Tokenize the text
    tokens = word_tokenize(text)

    # Remove stopwords and lemmatize tokens
    tokens = [
        lemmatizer.lemmatize(word)
        for word in tokens
        if word not in stop_words
    ]

    # Join tokens back to string
    return " ".join(tokens)

# Apply preprocessing
df["Processed_Text"] = df["Text"].apply(preprocess_text)
df.head()
```

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

1 to 5 of 5 entries Filter

index	Text	Processed_Text
0	The football team won the final match	football team final match
1	Cricket players practiced for the tournament	cricket player practiced tournament
2	The athlete broke the world record	athlete broke world record
3	Basketball is a popular indoor sport	basketball popular indoor sport
4	The coach planned strategies for the game	coach planned strategy game

Show 25 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.

Next steps: Generate code with df New interactive sheet

```
"""
```

```
TF-IDF converts text into numerical vectors by
assigning importance to words based on frequency.
"""

# Initialize TF-IDF vectorizer
vectorizer = TfidfVectorizer()

# Create TF-IDF matrix
tfidf_matrix = vectorizer.fit_transform(df["Processed_Text"])

# Display matrix shape
tfidf_matrix.shape
```

```
(29, 112)
```

```
"""
Cosine similarity measures the angle between
TF-IDF vectors to determine similarity.
"""

# Compute cosine similarity matrix
cosine_sim_matrix = cosine_similarity(tfidf_matrix)

# Convert to DataFrame for readability
cosine_df = pd.DataFrame(cosine_sim_matrix)

# Display first few rows
cosine_df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	19	20	21	22	23	24	25	26	27	28
0	1.0	0.0	0.0	0.0	0.0	0.000000	0.455686	0.221547	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.186035	0.000000	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.000000	0.000000	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	1.0	0.000000	0.000000	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 29 columns

```
"""
Jaccard similarity compares word overlap
between two documents.
"""

def jaccard_similarity(text1, text2):
    """
    Computes Jaccard similarity between two texts.

    Parameters:
    text1 (str), text2 (str)

    Returns:
    float: Jaccard similarity score
    """
    set1 = set(text1.split())
    set2 = set(text2.split())

    return len(set1.intersection(set2)) / len(set1.union(set2))

# Compute Jaccard similarity for sample pairs
for i in range(5):
    score = jaccard_similarity(
        df["Processed_Text"][i],
        df["Processed_Text"][i+1]
    )
    print(f"Jaccard similarity Doc {i} & Doc {i+1}: {score}")
```

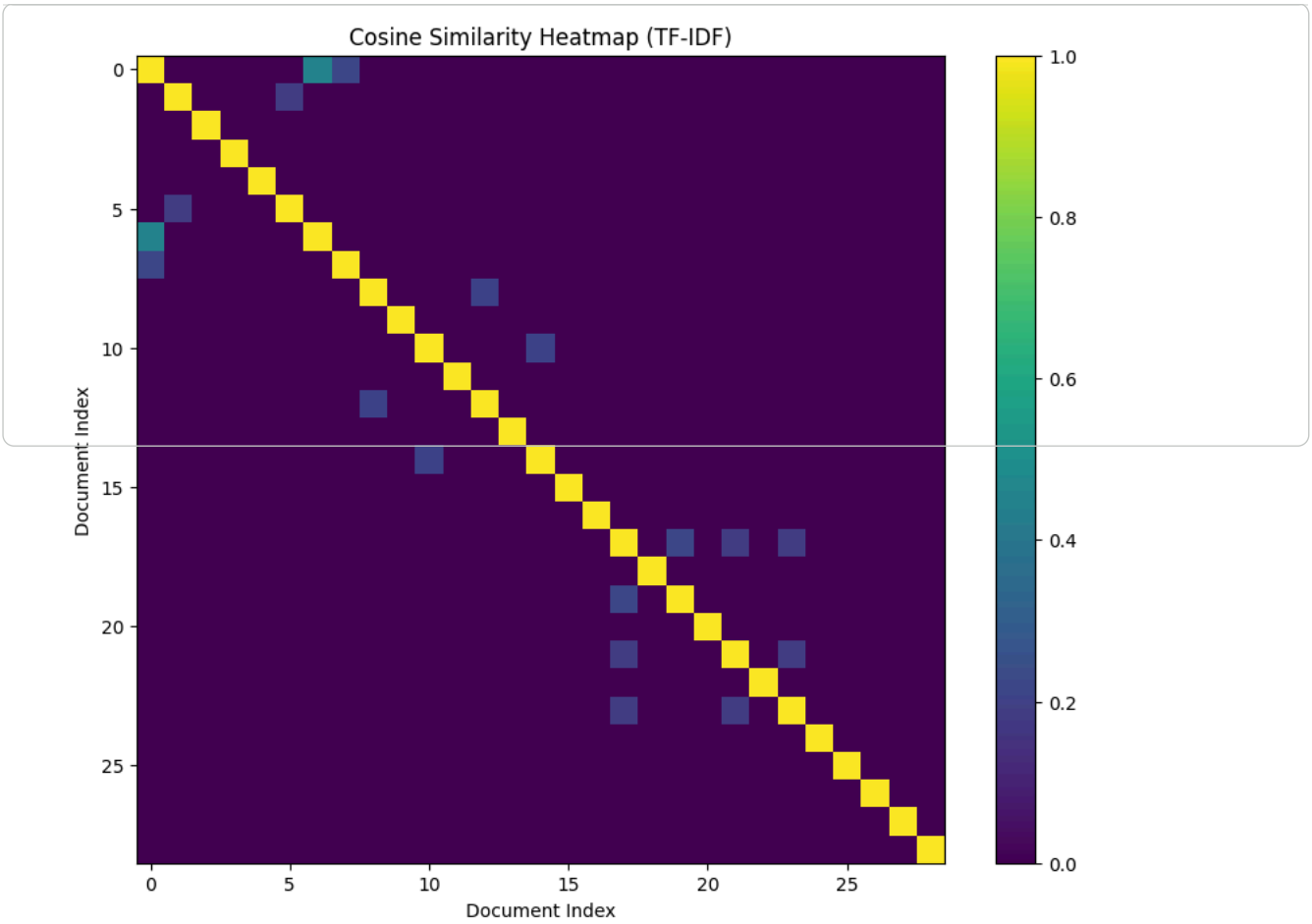
```
Jaccard similarity Doc 0 & Doc 1: 0.0
Jaccard similarity Doc 1 & Doc 2: 0.0
Jaccard similarity Doc 2 & Doc 3: 0.0
Jaccard similarity Doc 3 & Doc 4: 0.0
Jaccard similarity Doc 4 & Doc 5: 0.0
```

```
"""
WordNet similarity measures semantic closeness
between words using lexical databases.
```

```
"""

def wordnet_similarity(word1, word2):
    """
    Computes Wu-Palmer similarity between two words.

    Parameters:
    word1 (str), word2 (str)

    Returns:
    float or None
    """
    synsets1 = wordnet.synsets(word1)
    synsets2 = wordnet.synsets(word2)

    if synsets1 and synsets2:
        return synsets1[0].wup_similarity(synsets2[0])
    return None

# Test semantic similarity on word pairs
word_pairs = [
    ("doctor", "physician"),
    ("football", "soccer"),
    ("diet", "nutrition"),
    ("government", "administration"),
    ("technology", "innovation")
]

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

```
doctor - physician : 1.0
football - soccer : 0.96
diet - nutrition : 0.3333333333333333
government - administration : 0.26666666666666666
technology - innovation : 0.125
```

```
"""
Visualization of cosine similarity matrix using a heatmap.
Brighter colors indicate higher similarity between documents.
"""

import matplotlib.pyplot as plt

# Create a figure
plt.figure(figsize=(10, 8))

# Display cosine similarity as an image
plt.imshow(cosine_sim_matrix)

# Add color bar
plt.colorbar()

# Add labels
plt.title("Cosine Similarity Heatmap (TF-IDF)")
plt.xlabel("Document Index")
plt.ylabel("Document Index")

# Show plot
plt.show()
```


Cosine Similarity Heatmap (TF-IDF)
Document Index
Document Index
0
5
10
15
20
25
1.0
0.8
0.6
0.4
0.2
0.0