

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


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

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

```
# Download NLTK resources
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
```

```
# Make sure to upload 'text_similarity_dataset.csv' to your Colab environment.
# A common path for uploaded files is '/content/'.
df = pd.read_csv("/content/text_similarity_dataset.csv")
df.head()
```

	Category	Text	
0	sports	The football team won the championship match	
1	sports	Cricket players practiced hard for the tournament	
2	sports	The athlete broke the world record in running	
3	sports	Basketball is a popular sport worldwide	
4	sports	The coach trained the players for fitness	

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
print("Number of documents:", len(df))
print("\nCategories:\n", df["Category"].value_counts())
```

Number of documents: 20

Categories:

Category	
sports	5
politics	5
health	5
technology	5

Name: count, dtype: int64

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

def preprocess(text):
    text = text.lower() # lowercase
    text = text.translate(str.maketrans('', '', string.punctuation)) # remove
    tokens = word_tokenize(text) # tokenization
    tokens = [w for w in tokens if w not in stop_words and not w.isdigit()] #
    tokens = [lemmatizer.lemmatize(w) for w in tokens] # lemmatization
    return " ".join(tokens)
```

```
df["Cleaned_Text"] = df["Text"].apply(preprocess)
df.head()
```

Category		Text	Cleaned_Text	
0	sports	The football team won the championship match	football team championship match	
1	sports	Cricket players practiced hard for the tournament	cricket player practiced hard tournament	
2	sports	The athlete broke the world record in running	athlete broke world record running	
3	sports	Basketball is a popular sport worldwide	basketball popular sport worldwide	
4	sports	The coach trained the players for	coach trained player fitness	

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(df["Cleaned_Text"])
```

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

TF-IDF Matrix Shape: (20, 87)

```
cosine_sim = cosine_similarity(tfidf_matrix)
```

```
cosine_df = pd.DataFrame(cosine_sim)
cosine_df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1.0	0.000000	0.0	0.0	0.000000	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	1.000000	0.0	0.0	0.182091	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	0.000000	1.0	0.0	0.000000	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	0.000000	0.0	1.0	0.000000	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	0.182091	0.0	0.0	1.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Next steps:

[Generate code with cosine_df](#)
[New interactive sheet](#)

```
print("Cosine similarity between Doc 0 and Doc 1:", cosine_sim[0][1])
print("Cosine similarity between Doc 0 and Doc 10:", cosine_sim[0][10])
```

Cosine similarity between Doc 0 and Doc 1: 0.0
Cosine similarity between Doc 0 and Doc 10: 0.0

```
def jaccard_similarity(text1, text2):
    set1 = set(text1.split())
    set2 = set(text2.split())
    return len(set1 & set2) / len(set1 | set2)
```

```
# Jaccard similarity for first 5 documents with Doc 0
for i in range(1, 6):
    score = jaccard_similarity(df["Cleaned_Text"][0], df["Cleaned_Text"][i])
    print(f"Jaccard Similarity (Doc 0 & Doc {i}): {score}")
```

Jaccard Similarity (Doc 0 & Doc 1): 0.0
Jaccard Similarity (Doc 0 & Doc 2): 0.0
Jaccard Similarity (Doc 0 & Doc 3): 0.0
Jaccard Similarity (Doc 0 & Doc 4): 0.0
Jaccard Similarity (Doc 0 & Doc 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 0
```

```
word_pairs = [
    ("doctor", "physician"),
    ("football", "sport"),
    ("government", "minister"),
    ("disease", "illness"),
    ("technology", "software"),
    ("health", "fitness"),
    ("athlete", "player"),
    ("hospital", "clinic"),
    ("learning", "education"),
    ("computer", "machine")
]

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

```
doctor - physician : 1.0
football - sport : 0.8888888888888888
government - minister : 0.13333333333333333
disease - illness : 0.9473684210526315
technology - software : 0.23529411764705882
health - fitness : 0.375
athlete - player : 0.6666666666666666
hospital - clinic : 0.11764705882352941
learning - education : 0.42857142857142855
computer - machine : 0.9411764705882353
```

```
print("Cosine uses vector similarity (TF-IDF based)")
print("Jaccard uses exact word overlap")
print("WordNet captures semantic meaning")
```

```
Cosine uses vector similarity (TF-IDF based)
Jaccard uses exact word overlap
WordNet captures semantic meaning
```

```
cosine_df.to_csv("cosine_similarity_results.csv", index=False)
print("Cosine similarity file saved")
```

```
Cosine similarity file saved
```

