

Lab 7.3: Text Similarity Measures

2403A52006

MD.Mustafa

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

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

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')

[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...
[nltk_data]  Unzipping corpora/wordnet.zip.
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

True

Dataset Preparation

```
documents = [
    # Sports
    "The football team won the match",
    "Cricket players are training hard",
    "The athlete won a gold medal",
    "The coach discussed game strategy",
    "Fans celebrated the victory",

    # Politics
    "The government passed a new law",
    "Elections will be held next year",
    "The president addressed the nation",
    "Parliament discussed economic policy",
    "The minister announced reforms",
```

```

# Health
"Doctors recommend regular exercise",
"The patient received medical treatment",
"Healthy diet improves immunity",
"Hospitals provide emergency care",
"Vaccination prevents diseases",

# Technology
"Artificial intelligence is transforming technology",
"Machine learning improves data analysis",
"The software update fixed bugs",
"Cybersecurity protects digital systems",
"Cloud computing enables scalability"
]

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

{"summary": "{\n    \"name\": \"df\",\n    \"rows\": 20,\n    \"fields\": [\n        {\n            \"column\": \"Text\",\n            \"properties\": {\n                \"dtype\": \"string\",\n                \"num_unique_values\": 20,\n                \"samples\": [\n                    \"The football team won the match\",\n                    \"The software update fixed bugs\",\n                    \"Artificial\nintelligence is transforming technology\"\n                ],\n                \"semantic_type\": \"\",\n                \"description\": \"\"\n            }\n        }\n    ]\n}", "type": "dataframe", "variable_name": "df"}

```

Text Preprocessing

```

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.

{"summary": "{\n    \"name\": \"df\",\n    \"rows\": 20,\n    \"fields\": [\n        {\n            \"column\": \"Text\",\n            \"properties\": {\n
```

```

\"dtype\": \"string\",\\n      \"num_unique_values\": 20,\\n
\"samples\": [\n      \"The football team won the match\",\\n
      \"The software update fixed bugs\",\\n      \"Artificial\nintelligence is transforming technology\"\\n      ],\\n
      },\\n      {\n        \"column\": \"Clean_Text\",\\n
      \"properties\": {\n        \"dtype\": \"string\",\\n
        \"num_unique_values\": 20,\\n        \"samples\": [\n          \"football team match\",\\n          \"software update fixed bug\",\\n
          \"artificial intelligence transforming technology\"\\n        ],\\n
        \"semantic_type\": \"\",\\n        \"description\": \"\"\n      }\\n
    ]\\n  },\\n  \"type\": \"dataframe\", \"variable_name\": \"df\"}

```

Text Representation using TF-IDF

```

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

```

Cosine Similarity

```

cos_sim = cosine_similarity(tfidf_matrix)

print("Cosine Similarity Matrix (Rows = Documents):\n")

for i in range(len(df)):
    row = []
    for j in range(len(df)):
        row.append(f"{cos_sim[i][j]:.2f}")
    print(f"Doc {i}: " + " ".join(row))

```

Cosine Similarity Matrix (Rows = Documents):

Doc 0:	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Doc 1:	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Doc 2:	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Doc 3:	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.20	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Doc 4:	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Doc 5:	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Doc 6:	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Doc 7:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Doc 8:	0.00	0.00	0.00	0.20	0.00	0.00	0.00	0.00	1.00	0.00	0.00

```

0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 9: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 10: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 11: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 12: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 1.00 0.00 0.00 0.00 0.18 0.00 0.00 0.00 0.00
Doc 13: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 14: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 15: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00
Doc 16: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.18 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00
Doc 17: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00
Doc 18: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00
Doc 19: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00

```

Jaccard Similarity

```

print("Jaccard Similarity Matrix (Rows = Documents):\n")

for i in range(len(df)):
    row = []
    for j in range(len(df)):
        score = jaccard_similarity(df["Clean_Text"][i],
df["Clean_Text"][j])
        row.append(f"{score:.2f}")
    print(f"Doc {i}: " + " ".join(row))

```

Jaccard Similarity Matrix (Rows = Documents):

```

Doc 0: 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 1: 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 2: 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 3: 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.14 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 4: 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 5: 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00

```

```

0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 6: 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 7: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 8: 0.00 0.00 0.00 0.14 0.00 0.00 0.00 0.00 1.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 9: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 10: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 11: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 12: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 1.00 0.00 0.00 0.00 0.12 0.00 0.00 0.00 0.00
Doc 13: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 14: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 15: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
Doc 16: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.12 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00
Doc 17: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00
Doc 18: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00
Doc 19: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00

```

WordNet-based Semantic Similarity

```

word_pairs = [
    ("doctor", "physician"),
    ("car", "vehicle"),
    ("computer", "technology"),
    ("football", "sport"),
    ("medicine", "health"),
    ("hospital", "clinic"),
    ("software", "program"),
    ("diet", "nutrition"),
    ("government", "administration"),
    ("law", "policy")
]

print("WordNet Semantic Similarity Scores:\n")

for w1, w2 in word_pairs:

```

```
score = wordnet_similarity(w1, w2)
print(f"{w1} ↔ {w2} : {score}")
```

WordNet Semantic Similarity Scores:

```
doctor ↔ physician : 0.07692307692307693
car ↔ vehicle : 0.3333333333333333
computer ↔ technology : 0.07692307692307693
football ↔ sport : 0.07692307692307693
medicine ↔ health : 0.1111111111111111
hospital ↔ clinic : 0.3333333333333333
software ↔ program : 0.0833333333333333
diet ↔ nutrition : 0.1111111111111111
government ↔ administration : 1.0
law ↔ policy : 0.2
```