2403A52031

AKULA MANOJ

Import Required Libraries

```
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
```

Download NLTK Resources

```
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
```

Create Our Own Dataset

```
documents = [
    # Sports
    "The cricket team won the championship match",
    "Football players trained hard for the tournament",
    "The match was exciting and full of energy",
    "The athlete broke the world record",

    # Politics
    "The government passed a new education policy",
    "Elections will be held next month",
    "The president addressed the nation",
    "Parliament discussed economic reforms",

    # Health
    "Doctors recommend regular exercise for good health",
    "The patient recovered after proper treatment",
    "A healthy diet improves immunity",
```

```
    "Hospitals use advanced medical equipment",

    # Technology
    "Artificial intelligence is transforming industries",
    "Software developers build scalable applications",
    "Cybersecurity is crucial for data protection",
    "Cloud computing enables remote access"
]

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


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 16,\n  \"fields\": [\n
{\n      \"column\": \"Text\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 16,\n
\"samples\": [\n          \"The cricket team won the championship
match\",\n          \"Football players trained hard for the
tournament\",\n          \"Elections will be held next month\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


Text Preprocessing Function

```
nltk.download('punkt_tab')

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

def preprocess(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    tokens = word_tokenize(text)
    tokens = [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\": 16,\n  \"fields\": [\n
{\n      \"column\": \"Text\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 16,\n
\"samples\": [\n          \"The cricket team won the championship
match\",\n          \"Football players trained hard for the
tournament\",\n          \"Elections will be held next month\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Clean_Text\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 16,\n        \"samples\": [\n
```

```
\"cricket team championship match\",\n          \"football players
trained hard tournament\",\n          \"elections held next month\"\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

```
cosine_sim = cosine_similarity(tfidf_matrix)

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


```
{"summary":"{\n  \"name\": \"cosine_df\",\n  \"rows\": 16,\n
\"fields\": [\n    {\n      \"column\": 0,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25172089390098024,\n
\"min\": 0.0,\n        \"max\": 1.0000000000000002,\n
\"num_unique_values\": 3,\n        \"samples\": [\n
1.0000000000000002,\n          0.0,\n          0.20179245719305272\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": 1,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25000000000000006,\n
\"min\": 0.0,\n        \"max\": 1.0000000000000002,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
1.0000000000000002,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 2,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.2517208939009802,\n
\"min\": 0.0,\n        \"max\": 1.0,\n        \"num_unique_values\":
3,\n        \"samples\": [\n          0.20179245719305272,\n
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 3,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1.0,\n
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 4,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25000000000000006,\n        \"min\": 0.0,\n        \"max\":
1.0000000000000002,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0000000000000002,\n          0.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 5,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
```

```
\"num_unique_values\": 2,\n        \"samples\": [\n          1.0,\n
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 6,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1.0,\n
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 7,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1.0,\n
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 8,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25000000000000006,\n        \"min\": 0.0,\n        \"max\":
1.0000000000000002,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0000000000000002,\n          0.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 9,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1.0,\n
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 10,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1.0,\n
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 11,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25000000000000006,\n        \"min\": 0.0,\n        \"max\":
1.0000000000000002,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0000000000000002,\n          0.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 12,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1.0,\n
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 13,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25000000000000006,\n        \"min\": 0.0,\n        \"max\":
1.0000000000000002,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0000000000000002,\n          0.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 14,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1.0,\n
```

```
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 15,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25000000000000006,\n        \"min\": 0.0,\n        \"max\":
1.0000000000000002,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0000000000000002,\n          0.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"cosine_df"}
```


Jaccard Similarity

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

# Create Jaccard matrix
jaccard_matrix = np.zeros((len(df), len(df)))

for i in range(len(df)):
    for j in range(len(df)):
        jaccard_matrix[i][j] = jaccard_similarity(
            df["Clean_Text"][i], df["Clean_Text"][j]
        )

jaccard_df = pd.DataFrame(jaccard_matrix, columns=df.index,
index=df.index)
jaccard_df.head()
```


```
{"summary":"{\n  \"name\": \"jaccard_df\",\n  \"rows\": 16,\n
\"fields\": [\n    {\n      \"column\": 0,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.2501700102202608,\n
\"min\": 0.0,\n        \"max\": 1.0,\n        \"num_unique_values\":
3,\n        \"samples\": [\n          1.0,\n          0.0,\n
0.14285714285714285\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 1,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.25,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1.0,\n
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 2,\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.2501700102202608,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 3,\n        \"samples\": [\n
0.14285714285714285,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 3,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
```

```
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 4,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 5,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 6,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 7,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 8,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 9,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 10,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 11,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 12,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 13,\n      \"properties\": {\n
```

```
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 14,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": 15,\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25,\n        \"min\": 0.0,\
n        \"max\": 1.0,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"jaccard_df"}
```


WordNet Semantic Similarity

```
def wordnet_similarity(word1, word2):
    syn1 = wordnet.synsets(word1)
    syn2 = wordnet.synsets(word2)

    if not syn1 or not syn2:
        return 0

    return syn1[0].wup_similarity(syn2[0])
```

Example Tests

```
pairs = [
    ("doctor", "physician"),
    ("car", "vehicle"),
    ("cricket", "football"),
    ("software", "program"),
    ("hospital", "clinic")
]

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

```
doctor - physician : 1.0
car - vehicle : 0.8
cricket - football : 0.09090909090909091
software - program : 0.26666666666666666
hospital - clinic : 0.11764705882352941
```

Comparison of All Methods Comparative Analysis (8–10 sentences)

Cosine similarity works best for medium-length text and captures frequency patterns.

Jaccard similarity depends heavily on exact word overlap and fails with synonyms.

WordNet similarity captures semantic meaning effectively.

For short sentences, cosine and WordNet outperform Jaccard.

Jaccard gives zero similarity even when meaning is close.

WordNet works well for concept similarity but not full sentences.

Cosine is widely used due to efficiency and scalability.

Lexical similarity differs from semantic similarity in meaning awareness.