NAME : SAI CHARAN . P

ROLL:NO: 2403a52343

LAB : 7

Title: Text Similarity Measures — Cosine, Jaccard, and WordNet-based Similarity

Dataset

```
documents = [
    "Machine learning is a subset of artificial intelligence",
    "Machine learning is an important subset of artificial intelligence",
    "Artificial intelligence includes machine learning techniques",
    "Artificial intelligence uses machine learning algorithms",
    "Machine learning algorithms are used in artificial intelligence",
    "Artificial intelligence and machine learning are related fields",
    "Machine learning is widely used in artificial intelligence systems",
    "Artificial intelligence applications use machine learning models",
    "Machine learning plays a major role in artificial intelligence",
    "Artificial intelligence depends on machine learning methods",
    "Machine learning techniques improve artificial intelligence",
    "Artificial intelligence systems rely on machine learning",
    "Machine learning helps build intelligent systems",
    "Artificial intelligence builds intelligent systems",
    "Intelligent systems use artificial intelligence and machine learning"
]
```

STEP 1: Text Preprocessing

```
import nltk
import string
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

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

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

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

```
    return tokens
```

```
[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 to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

STEP 2: Feature Representation

```
from sklearn.feature_extraction.text import TfidfVectorizer

documents = [
    "Machine learning is a subset of artificial intelligence",
    "Machine learning is an important subset of artificial intelligence",
    "Artificial intelligence includes machine learning techniques",
    "Artificial intelligence uses machine learning algorithms",
    "Machine learning algorithms are used in artificial intelligence",
    "Artificial intelligence and machine learning are related fields",
    "Machine learning is widely used in artificial intelligence systems",
    "Artificial intelligence applications use machine learning models",
    "Machine learning plays a major role in artificial intelligence",
    "Artificial intelligence depends on machine learning methods",
    "Machine learning techniques improve artificial intelligence",
    "Artificial intelligence systems rely on machine learning",
    "Machine learning helps build intelligent systems",
    "Artificial intelligence builds intelligent systems",
    "Intelligent systems use artificial intelligence and machine learning"
]

vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = vectorizer.fit_transform(documents)

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

```
TF-IDF matrix shape: (15, 29)
```

STEP 3: Cosine Similarity

```
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd

cos_sim = cosine_similarity(tfidf_matrix)

cosine_df = pd.DataFrame(
    cos_sim,
    index=[f"Doc {i+1}" for i in range(len(documents))],
    columns=[f"Doc {i+1}" for i in range(len(documents))]
)

print(cosine_df.round(2))
```

```
pairs = []

for i in range(len(documents)):
    for j in range(i + 1, len(documents)):
        pairs.append((f"Doc {i+1}", f"Doc {j+1}", cos_sim[i][j]))

pairs = sorted(pairs, key=lambda x: x[2], reverse=True)[:15]

print("Top 15 Most Similar Document Pairs:\n")
for d1, d2, score in pairs:
    print(f"{d1} - {d2} : {score:.3f}")
```

```
         Doc 1  Doc 2  Doc 3  Doc 4  Doc 5  Doc 6  Doc 7  Doc 8  Doc 9  Doc 10
Doc 1     1.00   0.74   0.29   0.29   0.31   0.27   0.26   0.24   0.23    0.27
Doc 2     0.74   1.00   0.21   0.21   0.23   0.20   0.20   0.18   0.17    0.20
Doc 3     0.29   0.21   1.00   0.21   0.23   0.20   0.20   0.18   0.17    0.20
Doc 4     0.29   0.21   0.21   1.00   0.59   0.20   0.20   0.18   0.17    0.20
Doc 5     0.31   0.23   0.23   0.59   1.00   0.22   0.54   0.19   0.18    0.22
Doc 6     0.27   0.20   0.20   0.20   0.22   1.00   0.19   0.17   0.16    0.19
Doc 7     0.26   0.20   0.20   0.20   0.54   0.19   1.00   0.16   0.16    0.19
Doc 8     0.24   0.18   0.18   0.18   0.19   0.17   0.16   1.00   0.14    0.17
Doc 9     0.23   0.17   0.17   0.17   0.18   0.16   0.16   0.14   1.00    0.16
Doc 10    0.27   0.20   0.20   0.20   0.22   0.19   0.19   0.17   0.16    1.00
Doc 11    0.29   0.21   0.55   0.21   0.23   0.20   0.20   0.18   0.17    0.20
Doc 12    0.31   0.23   0.23   0.23   0.25   0.22   0.40   0.19   0.19    0.22
Doc 13    0.12   0.09   0.09   0.09   0.09   0.08   0.22   0.07   0.07    0.08
Doc 14    0.14   0.11   0.11   0.11   0.11   0.10   0.27   0.09   0.09    0.10
Doc 15    0.29   0.21   0.21   0.21   0.23   0.20   0.37   0.46   0.17    0.20

         Doc 11  Doc 12  Doc 13  Doc 14  Doc 15
Doc 1      0.29    0.31    0.12    0.14    0.29
Doc 2      0.21    0.23    0.09    0.11    0.21
Doc 3      0.55    0.23    0.09    0.11    0.21
Doc 4      0.21    0.23    0.09    0.11    0.21
Doc 5      0.23    0.25    0.09    0.11    0.23
Doc 6      0.20    0.22    0.08    0.10    0.20
Doc 7      0.20    0.40    0.22    0.27    0.37
Doc 8      0.18    0.19    0.07    0.09    0.46
Doc 9      0.17    0.19    0.07    0.09    0.17
Doc 10     0.20    0.22    0.08    0.10    0.20
Doc 11     1.00    0.23    0.09    0.11    0.21
Doc 12     0.23    1.00    0.26    0.32    0.43
Doc 13     0.09    0.26    1.00    0.37    0.46
Doc 14     0.11    0.32    0.37    1.00    0.56
Doc 15     0.21    0.43    0.46    0.56    1.00
Top 15 Most Similar Document Pairs:

Doc 1 - Doc 2 : 0.743
Doc 4 - Doc 5 : 0.585
Doc 14 - Doc 15 : 0.557
Doc 3 - Doc 11 : 0.552
Doc 5 - Doc 7 : 0.537
Doc 13 - Doc 15 : 0.464
Doc 8 - Doc 15 : 0.457
Doc 12 - Doc 15 : 0.433
Doc 7 - Doc 12 : 0.399
Doc 13 - Doc 14 : 0.375
```

```
Doc 7 - Doc 15 : 0.366
Doc 12 - Doc 14 : 0.316
Doc 1 - Doc 12 : 0.313
Doc 1 - Doc 5 : 0.306
Doc 1 - Doc 3 : 0.288
```

STEP 4: Jaccard Similarity

Jaccard similarity compares overlapping word sets.

$$J(A, B)$$

$|A \cap B|$ $|A \cup B|$ J(A,B)= |A∪B| |A∩B|

```
import itertools

def jaccard_similarity(doc1, doc2):
    set1 = set(preprocess(doc1))
    set2 = set(preprocess(doc2))
    union_size = len(set1 | set2)
    return len(set1 & set2) / union_size if union_size != 0 else 0

jaccard_scores = []
for i, j in itertools.combinations(range(len(documents)), 2):
    score = jaccard_similarity(documents[i], documents[j])
    jaccard_scores.append((f"Doc {i+1}", f"Doc {j+1}", score))

top_5_jaccard = sorted(jaccard_scores, key=lambda x: x[2], reverse=True)[:5]

print("Top 5 Jaccard Similarity Pairs:")
for d1, d2, score in top_5_jaccard:
    print(f"{d1} & {d2} -> {score:.3f}")
```

```
Top 5 Jaccard Similarity Pairs:
Doc 1 & Doc 2 -> 0.833
Doc 3 & Doc 11 -> 0.714
Doc 4 & Doc 5 -> 0.714
Doc 5 & Doc 7 -> 0.625
Doc 7 & Doc 12 -> 0.625
```

STEP 5: WordNet Semantic Similarity

```
import itertools

wordnet_pairs = []

# Iterate through all unique pairs of documents
for i, j in itertools.combinations(range(len(documents)), 2):
    doc1_text = documents[i]
    doc2_text = documents[j]
    sim_score = sentence_similarity(doc1_text, doc2_text)
```

```
        if sim_score is not None:
            wordnet_pairs.append((f"Doc {i+1}", f"Doc {j+1}", sim_score))

# Sort pairs by similarity score in descending order and get the top 15
wordnet_pairs = sorted(wordnet_pairs, key=lambda x: x[2], reverse=True)[:15]

print("Top 15 Most Similar Document Pairs (WordNet):")
for d1, d2, score in wordnet_pairs:
    print(f"{d1} - {d2} : {score:.3f}")
```

```
Top 15 Most Similar Document Pairs (WordNet):
Doc 1 - Doc 2 : 0.394
Doc 3 - Doc 10 : 0.388
Doc 3 - Doc 11 : 0.386
Doc 1 - Doc 3 : 0.383
Doc 1 - Doc 10 : 0.381
Doc 10 - Doc 11 : 0.381
Doc 1 - Doc 11 : 0.376
Doc 2 - Doc 3 : 0.369
Doc 1 - Doc 8 : 0.368
Doc 1 - Doc 5 : 0.365
Doc 2 - Doc 10 : 0.365
Doc 1 - Doc 15 : 0.362
Doc 3 - Doc 5 : 0.362
Doc 5 - Doc 8 : 0.361
Doc 2 - Doc 11 : 0.361
```

STEP 6: Comparison & Analysis

Comparison Results:

Cosine similarity was most effective in detecting near-copy texts, as TF-IDF emphasizes shared weighted terms. Jaccard similarity performed well only when documents shared many exact words, but failed for paraphrased content. WordNet-based similarity helped detect semantic equivalence, especially when synonyms were used instead of exact words. However, WordNet sometimes produced false positives for conceptually related but contextually different sentences. Jaccard produced the highest number of false negatives. Cosine similarity balanced accuracy and efficiency. Semantic similarity was computationally expensive but valuable for paraphrase detection.

STEP 7: Lab Report Section

This lab demonstrated how different text similarity measures behave under lexical and semantic variations. Lexical approaches are fast and effective for detecting copied content, while semantic approaches are essential for understanding meaning. Combining both yields the most reliable results for real-world NLP applications.