|  |
| --- |
| Experiment No. 8 |
| Implement Text Similarity Recognizer for the chosen text documents. |
| Date of Performance: |
| Date of Submission: |

**Aim:** Implement Text Similarity Recognizer for the chosen text documents.

**Objective:** Understand the importance of Implementing Text Similarity Recognizer for the chosen text documents.

**Theory:**

1. **Preprocess the Text Data:**
   * Tokenization
   * Stopwords removal
   * Lemmatization or stemming
2. **Feature Extraction:**
   * TF-IDF Vectorization
   * Word Embeddings (e.g., Word2Vec, GloVe)
   * Sentence Embeddings (e.g., Sentence-BERT)
3. **Compute Similarity:**
   * Cosine Similarity
   * Euclidean Distance
4. **Evaluate Similarity:**
   * Compare and interpret the similarity scores.

Code:

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

# Define two sentences to compare

sentenceOne = 'My house is empty today'

sentenceTwo = 'Nobody is at my home'

documents = [sentenceOne, sentenceTwo]

# Initialize the TF-IDF Vectorizer

tfidf = TfidfVectorizer()

# Fit and transform the documents to get the TF-IDF matrix

sparseMatrix = tfidf.fit\_transform(documents)

# Convert the sparse matrix to a dense matrix

docTermMatrix = sparseMatrix.todense()

# Create a DataFrame to visualize the TF-IDF matrix

df = pd.DataFrame(

    docTermMatrix,

    columns=tfidf.get\_feature\_names\_out(),

    index=['sentenceOne', 'sentenceTwo']

)

# Calculate the cosine similarity between the two sentences

simScore = cosine\_similarity(df, df)[0, 1]

# Identify the words that appear in both sentences (non-zero entries)

match\_keys = df.isin([0]).sum(axis=0)  # Sum of zeros in the columns

match\_words = match\_keys[match\_keys.values == 0].keys()  # Words with no zeros in both sentences

# Print results

print(f'Cosine Similarity: {round(simScore, 2)}')

print(f'Matching Words: {list(match\_words)}')

Output:

(venv) PS D:\Vartak college\sem 7\NLP\EXP\New folder> python .\exp8.py

Cosine Similarity: 0.25

Matching Words: ['is', 'my']

**Conclusion**:Implementing a Text Similarity Recognizer involves several crucial steps: preprocessing the text data, extracting meaningful features, computing similarity measures, and interpreting the results. In this example, we utilized TF-IDF Vectorization for feature extraction and Cosine Similarity for computing similarity between text documents.