Worksheet 4

Student Name: GOITA Aboubacar

UID: 24MCI10225

Branch: MCA AIML

Section/Group: MAM 4-A

Semester: 3rd

Date of Performance: 12.09.2025

Subject Name: Natural Language Processing

Subject Code: 24CAH724

1. Aim/Overview of the practical:

To understand and implement the TF-IDF (Term Frequency–Inverse Document Frequency) technique to extract important keywords from a document.

Task to be done:

1. Input a document or a set of documents.
2. Preprocess the text: lowercasing, removing punctuation, and stop words.
3. Use TfidfVectorizer to compute TF-IDF scores.
4. Sort and display words by their TF-IDF score.
5. Identify top keywords with the highest scores.

2. Code

Part a-)

```
# Experiment: Calculate TF-IDF scores and identify keywords

from sklearn.feature_extraction.text import TfidfVectorizer

import pandas as pd

# Sample input document

document = """Natural Language Processing (NLP) enables computers to understand human language.

TF-IDF is a statistical measure used to evaluate how important a word is to a document in a collection or corpus."""

# Step 1: Preprocess the document (convert to lowercase, remove punctuations if necessary)

processed_doc = document.lower()

# Step 2: Create a TF-IDF Vectorizer

vectorizer = TfidfVectorizer(stop_words='english')
```

```
# Step 3: Fit and transform the document

tfidf_matrix = vectorizer.fit_transform([processed_doc])

# Step 4: Get feature names (i.e., words)

feature_names = vectorizer.get_feature_names_out()

# Step 5: Create a DataFrame of words and their TF-IDF scores

df_tfidf = pd.DataFrame(tfidf_matrix.T.toarray(), index=feature_names, columns=["TF-IDF Score"])

df_tfidf = df_tfidf.sort_values(by=["TF-IDF Score"], ascending=False)

# Step 6: Display top keywords

print("Top keywords based on TF-IDF scores:")

print(df_tfidf.head(10))
```

Part b-)

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

# Load dataset
df = pd.read_csv("/Users/aboubacargoita/Documents/NLP/AI_Human.csv")
print("Columns in dataset:", df.columns)

# Use only text column for documents
documents = df["text"].astype(str).tolist()

# Apply TF-IDF with limited features
vectorizer = TfidfVectorizer(max_features=1000, stop_words="english")
tfidf_matrix = vectorizer.fit_transform(documents)

# Convert to DataFrame
df_tfidf = pd.DataFrame(tfidf_matrix.toarray(), columns=vectorizer.get_feature_names_out())

# Find top features
mean_scores = df_tfidf.mean().sort_values(ascending=False)
print("Top TF-IDF features across AI_Humans dataset:")
print(mean_scores.head(10))
```

3. Output

```
aboubacargoita@Aboubacars-MacBook-Air NLP % /usr/bin/python3 /Users/aboubacargoita/Documents/NLP/exp4dataset.py
Columns in dataset: Index(['text', 'generated'], dtype='object')
Top TF-IDF features across AI_Humans dataset:
students     0.070468
people       0.051574
car          0.048839
cars         0.047087
electoral    0.045801
school       0.043106
college      0.037437
like         0.035245
venus        0.033293
vote         0.031922
dtype: float64
aboubacargoita@Aboubacars-MacBook-Air NLP %
```

```
Problems  Output  Debug Console  Terminal  Ports                                   Python

/usr/bin/python3 /Users/aboubacargoita/Documents/NLP/exp4.py
aboubacargoita@Aboubacars-MacBook-Air NLP % /usr/bin/python3 /Users/aboubacargoita/Documents/NLP/exp4.py
Top keywords based on TF-IDF scores:
             TF-IDF Score
language         0.426401
measure          0.213201
used             0.213201
understand       0.213201
tf               0.213201
statistical      0.213201
processing       0.213201
nlp              0.213201
natural          0.213201
collection       0.213201
```

4. Learning outcomes:

- **Understanding of TF-IDF principles**
    - You learned how TF-IDF assigns higher weights to terms that are frequent in a document but rare across other documents, making them effective for keyword extraction.
- **Practical implementation using scikit-learn**
    - You gained hands-on experience with `TfidfVectorizer` in Python to preprocess text, transform it into numerical features, and extract important words.

- **Feature representation of text data**
    - You discovered how textual data can be converted into a **sparse vector representation**, enabling its use in machine learning models and similarity analysis.
- **Keyword identification and dataset-level insights**
    - From a single document, you identified top keywords (Part a), and from a full dataset, you extracted **global important terms** (Part b), demonstrating the flexibility of TF-IDF for different scales.
- **Foundation for NLP tasks**
    - You realized that TF-IDF forms the basis of many **downstream NLP applications** like document classification, clustering, search engines, and summarization.