CISB5123 Text Analytics
Lab 5
Bag of Words and Term Frequency-Inverse Document Frequency (TF-IDF)

The Bag of Words (BoW) model represents text by counting the occurrence of words within a document. In bag of words, we take all unique words from the corpus, note the frequency of occurrence and sort them in descending order. All the word - vector mapping we have will be used to represent each sentence.

tf–idf or TFIDF, short for term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

Bag of Words Implementation

1. Import necessary libraries:

```
from collections import Counter #count word occurrences
import pandas as pd
```

2. Load pre-processed file:

```
file_path = "Processed_Reviews.csv"
df = pd.read_csv(file_path)
```

3. Extract the tokenized text data:

```
tokenized_reviews = df['tokenized'].dropna().apply(eval)
```

4. List all unique words:

```
all_words = [word for review in tokenized_reviews for word in review]
unique_words = list(set(all_words))
```

5. Create word frequency table:

```
word_freq = Counter(all_words)
sorted_word_freq = dict(sorted(word_freq.items(), key=lambda item: item[1], reverse=True))
```

6. Create document vectors:

```
document_vectors = []
for review in tokenized_reviews:
    document_vector = [1 if word in review else 0 for word in sorted_word_freq.keys()]
    document_vectors.append(document_vector)
```

7. Convert document vectors to DataFrame:

```
doc_vectors_df = pd.DataFrame(document_vectors, columns=sorted_word_freq.keys())
```

8. Save document vectors to CSV file:

```
doc_vectors_df.to_csv("document_vectors.csv", index=False)
```

9. Display word frequency table:

```
word_freq_df = pd.DataFrame(list(sorted_word_freq.items()), columns=["Word", "Frequency"])
print("Word Frequency Table:")
print(word_freq_df)
```

TF-IDF Implementation

1. Import necessary libraries:

```
import pandas as pd
import math # use for log function
from collections import Counter
```

2. Load pre-processed file:

```
file_path = "Processed_Reviews.csv"
df = pd.read_csv(file_path)
```

3. Extract the tokenized text data:

```
tokenized_reviews = df['tokenized'].dropna().apply(eval)
```

4. Function to compute Term Frequency (TF)

```
def compute_tf(document):
    word_count = Counter(document)
    tf = {word: count / len(document) for word, count in word_count.items()}
    return tf
```

5. Function to compute Inverse Document Frequency (IDF)

```
def compute_idf(documents):
    N = len(documents)  # Total number of documents
    idf = {}
    all_words = set(word for doc in documents for word in doc)  # Unique words
    for word in all_words:
        count = sum(1 for doc in documents if word in doc)
        idf[word] = math.log(N / count)
    return idf
```

6. Function to compute TF-IDF:

```
def compute_tfidf(document, idf):
    tfidf = {}
    tf = compute_tf(document)  # Get TF values for the document
    for word, tf_value in tf.items():
        tfidf[word] = tf_value * idf[word]  # Multiply TF and IDF
    return tfidf
```

7. Convert tokenized reviews into a List:

```
documents = tokenized_reviews.tolist()
```

8. Compute TF scores, convert to DataFrame and save to a CSV file:

```
tf_data = [compute_tf(doc) for doc in documents]
tf_df = pd.DataFrame(tf_data).fillna(0)
tf_df.to_csv("tf_scores.csv", index=False)
```

9. Compute IDF scores, convert to DataFrame and save to a CSV file:

```
idf = compute_idf(documents)
idf_df = pd.DataFrame([idf]).fillna(0)
idf_df.to_csv("idf_scores.csv", index=False)
```

10. Compute TF-IDF scores, convert to DataFrame and save to a CSV file:

```
tfidf_data = [compute_tfidf(doc, idf) for doc in documents]
tfidf_df = pd.DataFrame(tfidf_data).fillna(0)
tfidf_df.to_csv("tfidf_scores.csv", index=False)
```

Inspect the saved files, and understand what those scores mean.