Assignment-01: Text Summarization using Python & NLTK: TF-IDF Algorithm

Course: AIML-CZG-567 (AI & ML Techniques for Cyber Security)
Name & BITS ID: Cherukupally Sarika, 2023AC05780

Problem Statement

Text summarization involves generating concise summaries of lengthy text documents while retaining key information and meaning. The goal of this assignment is to implement a text summarization system using the TF-IDF algorithm in Python, focusing on:

- Extracting key sentences from the text.
- Ranking sentences based on their importance using term frequency and inverse document frequency.

1. Overall Process Description & Solution Approach

The solution is based on the following process steps:

Step 1: Tokenize Sentences

- To split the input text into individual sentences, I have used NLTK's **‘sent_tokenize’** method.

Step 2: Create Frequency Matrix of Words in Each Sentence

- Tokenized each sentence into words using **‘word_tokenize’** method.
- Word occurrences have been counted by excluding ‘**stop_words’** method and non-alphanumeric characters.

Step 3: Calculate Term Frequency (TF)

- For computing the Term Frequency (TF), I have calculated the normalized frequency for each word within the sentence:

TF = Total Words in Sentence / Word Count in Sentence

Step 4: Create Document Per Word Table

- Counted the number of sentences containing each word.

Step 5: Calculate Inverse Document Frequency (IDF)

- For each word, calculate the IDF value:

IDF = log(Total Sentences / Number of Sentences containing Word)

Step 6: Calculate TF-IDF

- Multiplied the TF and IDF values for each word to generate the TF-IDF score.

TF-TDF = TF x IDF

Step 7: Score the Sentences

- Calculated the score for each sentence by summing the TF-IDF scores of its words.

Step 8: Find the Threshold

- Determine a threshold for significant sentences based on the average sentence score and a threshold factor:

  Average Score Calculation: Computing the mean score of all sentences:

  Average Score = Sum of All Sentence Scores / Number of Sentences

  Threshold Factor: The threshold is adjusted by multiplying the average score with the threshold factor:

  Threshold = Average Score × Threshold Factor

Step 9: Generate the Summary

- Selected the sentences with scores above the threshold to form the summary.

---

2. Tool used and reasons to use this specific tool:

- **Python 3:** For implementing the solution due to its flexibility and rich library support.
- **NLTK Toolkit:** Provides robust methods for text tokenization, stop_word handling, and processing.
- **IDE/Text Editor:** Used Jupyter Notebook for efficient coding and debugging.

---

3. Source code snippets

1. Tokenize sentences

```
from nltk.tokenize import sent_tokenize

text = """Artificial Intelligence (AI) is a branch of computer science that aims to create machines as intelligent as humans.
AI has applications in various fields, including healthcare, finance, and transportation.
Machine Learning (ML), a subset of AI, enables computers to learn from data without being explicitly programmed.
Natural Language Processing (NLP) and Computer Vision are key domains within AI.
"""
# Tokenize Sentences
def tokenize_sentences(text):
    return sent_tokenize(text)

sentences = tokenize_sentences(text)
print("<< Step 1: Tokenized Sentences >>")
for i, sentence in enumerate(sentences, 1):
    print(f"\n Sentence {i}: {sentence}")
```

2. Create Frequency Matrix of Words in Each Sentence.

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from collections import defaultdict

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

# Create Frequency Matrix
def create_frequency_matrix(sentences):
    frequency_matrix = {}
    for sentence in sentences:
        words = word_tokenize(sentence.lower())
        frequency_matrix[sentence] = {}
        for word in words:
            if word.isalnum() and word not in stop_words:
                frequency_matrix[sentence][word] = frequency_matrix[sentence].get(word, 0) + 1
    return frequency_matrix

frequency_matrix = create_frequency_matrix(sentences)
print("\n<< Step 2: Frequency Matrix >>")
for sentence, freqs in frequency_matrix.items():
    print(f"\n Sentence: \"{sentence}\"")
    print(f">> Frequencies: {freqs}")
```

3. Calculate Term Frequency and Generate Matrix.

```
# Calculate Term Frequency
def calculate_term_frequency(frequency_matrix):
    tf_matrix = {}
    for sentence, freqs in frequency_matrix.items():
        tf_matrix[sentence] = {}
        total_words = sum(freqs.values())
        for word, count in freqs.items():
            tf_matrix[sentence][word] = count / total_words
    return tf_matrix

tf_matrix = calculate_term_frequency(frequency_matrix)
print("\n<< Step 3: Term Frequency (TF) Matrix >>")
for sentence, tf_scores in tf_matrix.items():
    print(f"\n Sentence: \"{sentence}\"")
    print(f">> TF Scores: {tf_scores}\n")
```

4. Create a table for documents per words.

```
# Document Per Word Table
def create_documents_per_word_table(frequency_matrix):
    word_document_counts = defaultdict(int)
    for sentence, freqs in frequency_matrix.items():
        for word in freqs.keys():
            word_document_counts[word] += 1
    return word_document_counts

word_document_counts = create_documents_per_word_table(frequency_matrix)
print("\n<< Step 4: Document Per Word Table >>\n")
for word, doc_count in word_document_counts.items():
    print(f"{word}: {doc_count}")
```

5. Calculate IDF and generate matrix.

```
import math

# Calculate Inverse Document Frequency
def calculate_idf(sentences, word_document_counts):
    total_documents = len(sentences)
    idf_scores = {}
    for word, count in word_document_counts.items():
        idf_scores[word] = math.log10(total_documents / (1 + count))
    return idf_scores

idf_scores = calculate_idf(sentences, word_document_counts)
print("\n### Step 5: Inverse Document Frequency (IDF) Scores ###\n")
for word, idf_score in idf_scores.items():
    print(f"{word}: {idf_score}")
```

6. Calculate TF-IDF and Generate Matrix. </b

```
# Calculate TF-IDF Matrix
def calculate_tfidf(tf_matrix, idf_scores):
    tfidf_matrix = {}
    for sentence, tf_scores in tf_matrix.items():
        tfidf_matrix[sentence] = {}
        for word, tf_score in tf_scores.items():
            tfidf_matrix[sentence][word] = tf_score * idf_scores.get(word, 0)
    return tfidf_matrix

tfidf_matrix = calculate_tfidf(tf_matrix, idf_scores)
print("\n### Step 6: TF-IDF Matrix ###\n")
for sentence, tfidf_scores in tfidf_matrix.items():
    print(f"Sentence: \"{sentence}\"")
    print(f">> TF-IDF Scores: {tfidf_scores}\n")
```

7. Score the sentences.

```python
# Score Sentences
def score_sentences(tfidf_matrix):
    sentence_scores = {}
    for sentence, tfidf_scores in tfidf_matrix.items():
        sentence_scores[sentence] = sum(tfidf_scores.values())
    return sentence_scores

sentence_scores = score_sentences(tfidf_matrix)
print("\n### Step 7: Sentence Scores ###")
for sentence, score in sentence_scores.items():
    print(f"\n Sentence: \"{sentence}\" - Score: {score}")
```

8. Find the threshold.

```python
# Calculate Threshold
def calculate_threshold(sentence_scores):
    threshold_factor = 1
    return sum(sentence_scores.values()) / len(sentence_scores) * threshold_factor

threshold = calculate_threshold(sentence_scores)
print("\n### Step 8: Threshold Value ###")
print(f"Threshold (Average Score): {threshold}")
```

9. Generate the Summary.

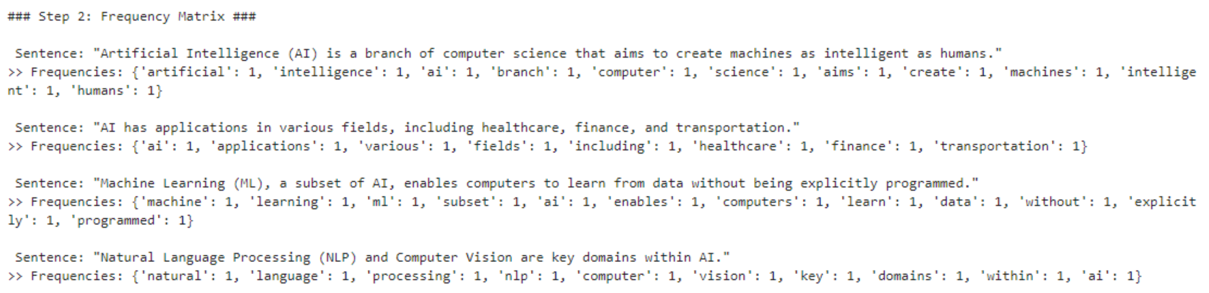
```python
# Generate Summary
def generate_summary(sentences, sentence_scores, threshold):
    return [sentence for sentence in sentences if sentence_scores[sentence] > threshold]

summary = generate_summary(sentences, sentence_scores, threshold)
print("\n### Step 9: Generated Summary ###\n")
print(" ".join(summary))
```

4. Final output results and analysis of results

Step 1: Tokenize sentences

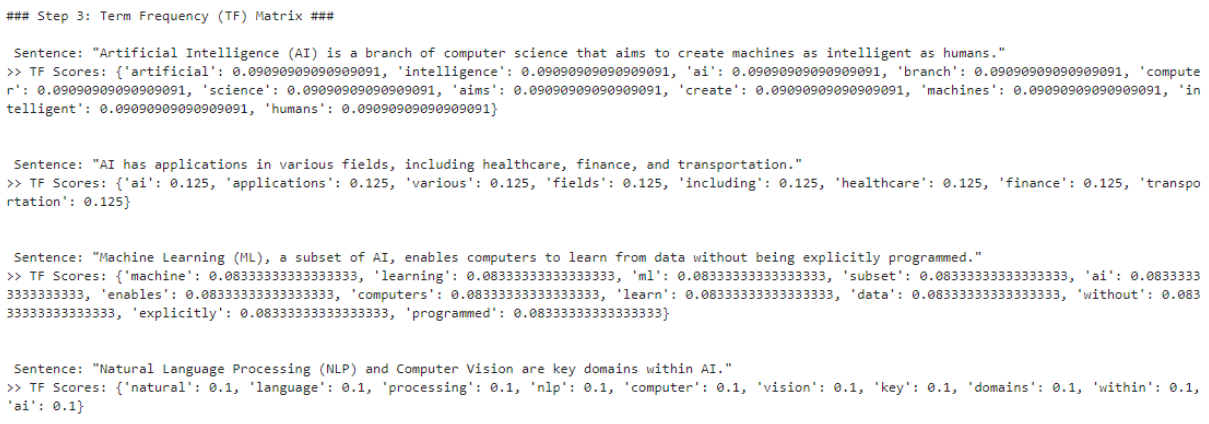
```
### Step 1: Tokenized Sentences ###

 Sentence 1: Artificial Intelligence (AI) is a branch of computer science that aims to create machines as intelligent as humans.

 Sentence 2: AI has applications in various fields, including healthcare, finance, and transportation.

 Sentence 3: Machine Learning (ML), a subset of AI, enables computers to learn from data without being explicitly programmed.

 Sentence 4: Natural Language Processing (NLP) and Computer Vision are key domains within AI.
```

Step 2: Create Frequency Matrix of Words in Each Sentence.



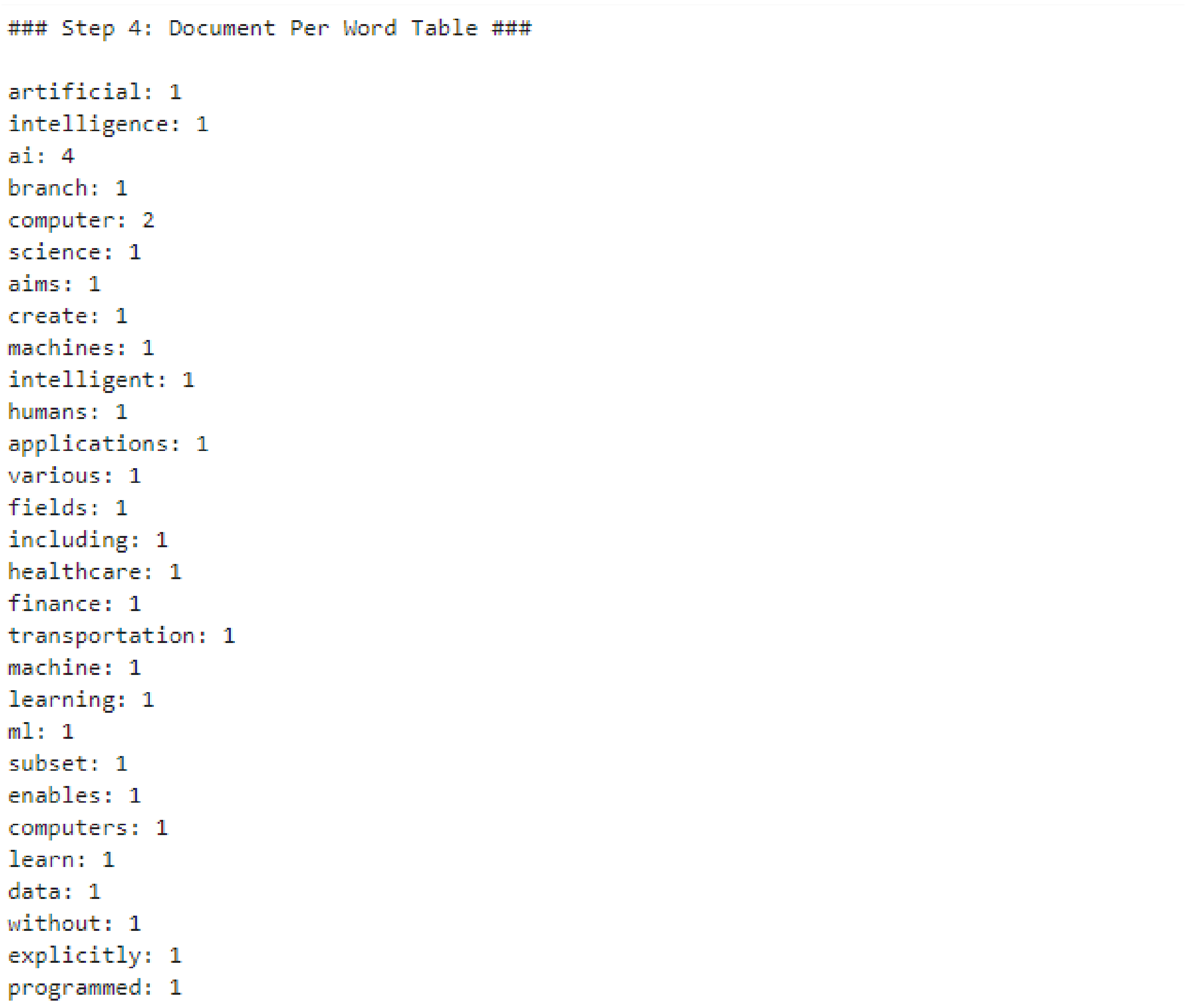
```
### Step 2: Frequency Matrix ###

 Sentence: "Artificial Intelligence (AI) is a branch of computer science that aims to create machines as intelligent as humans."
>> Frequencies: {'artificial': 1, 'intelligence': 1, 'ai': 1, 'branch': 1, 'computer': 1, 'science': 1, 'aims': 1, 'create': 1, 'machines': 1, 'intellige
nt': 1, 'humans': 1}

 Sentence: "AI has applications in various fields, including healthcare, finance, and transportation."
>> Frequencies: {'ai': 1, 'applications': 1, 'various': 1, 'fields': 1, 'including': 1, 'healthcare': 1, 'finance': 1, 'transportation': 1}

 Sentence: "Machine Learning (ML), a subset of AI, enables computers to learn from data without being explicitly programmed."
>> Frequencies: {'machine': 1, 'learning': 1, 'ml': 1, 'subset': 1, 'ai': 1, 'enables': 1, 'computers': 1, 'learn': 1, 'data': 1, 'without': 1, 'explicit
ly': 1, 'programmed': 1}

 Sentence: "Natural Language Processing (NLP) and Computer Vision are key domains within AI."
>> Frequencies: {'natural': 1, 'language': 1, 'processing': 1, 'nlp': 1, 'computer': 1, 'vision': 1, 'key': 1, 'domains': 1, 'within': 1, 'ai': 1}
```

Step 3: Calculate Term Frequency and Generate Matrix.

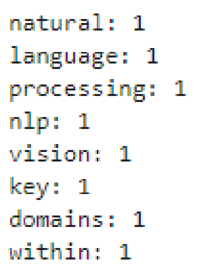


```
### Step 3: Term Frequency (TF) Matrix ###

 Sentence: "Artificial Intelligence (AI) is a branch of computer science that aims to create machines as intelligent as humans."
>> TF Scores: {'artificial': 0.09090909090909091, 'intelligence': 0.09090909090909091, 'ai': 0.09090909090909091, 'branch': 0.09090909090909091, 'compute
r': 0.09090909090909091, 'science': 0.09090909090909091, 'aims': 0.09090909090909091, 'create': 0.09090909090909091, 'machines': 0.09090909090909091, 'in
telligent': 0.09090909090909091, 'humans': 0.09090909090909091}


 Sentence: "AI has applications in various fields, including healthcare, finance, and transportation."
>> TF Scores: {'ai': 0.125, 'applications': 0.125, 'various': 0.125, 'fields': 0.125, 'including': 0.125, 'healthcare': 0.125, 'finance': 0.125, 'transpo
rtation': 0.125}


 Sentence: "Machine Learning (ML), a subset of AI, enables computers to learn from data without being explicitly programmed."
>> TF Scores: {'machine': 0.08333333333333333, 'learning': 0.08333333333333333, 'ml': 0.08333333333333333, 'subset': 0.08333333333333333, 'ai': 0.0833333
3333333333, 'enables': 0.08333333333333333, 'computers': 0.08333333333333333, 'learn': 0.08333333333333333, 'data': 0.08333333333333333, 'without': 0.083
33333333333333, 'explicitly': 0.08333333333333333, 'programmed': 0.08333333333333333}


 Sentence: "Natural Language Processing (NLP) and Computer Vision are key domains within AI."
>> TF Scores: {'natural': 0.1, 'language': 0.1, 'processing': 0.1, 'nlp': 0.1, 'computer': 0.1, 'vision': 0.1, 'key': 0.1, 'domains': 0.1, 'within': 0.1,
'ai': 0.1}
```

Step 4: Create a table for documents per words.

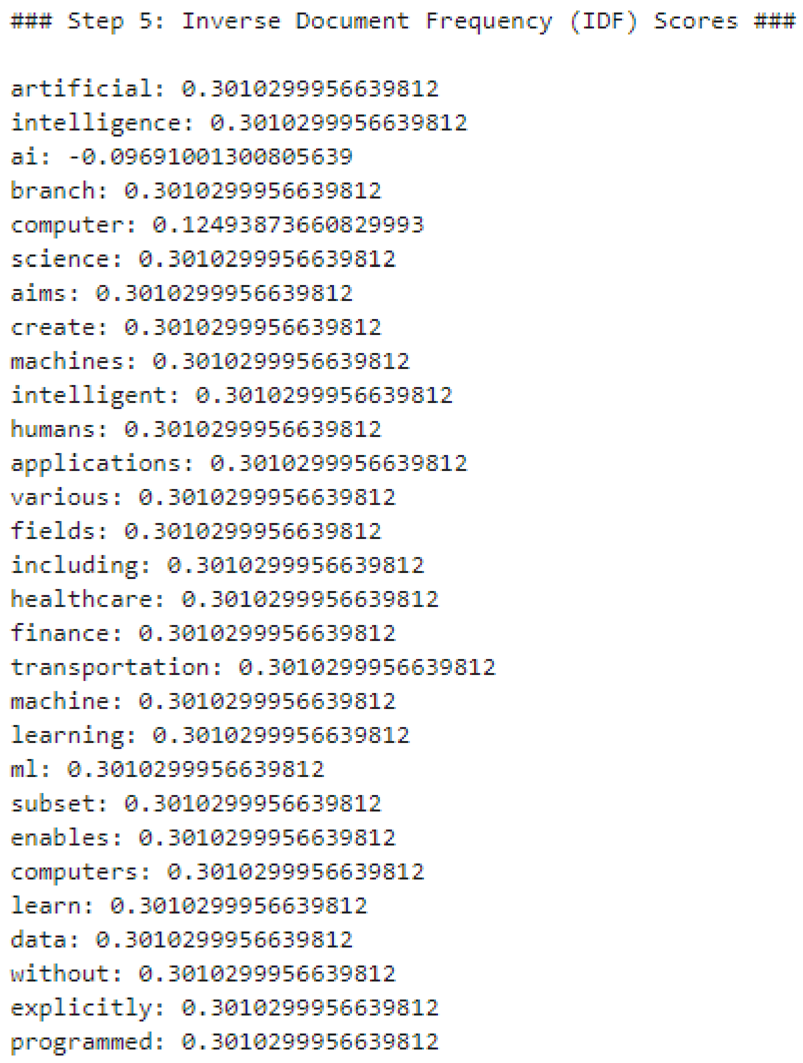


```
### Step 4: Document Per Word Table ###

artificial: 1
intelligence: 1
ai: 4
branch: 1
computer: 2
science: 1
aims: 1
create: 1
machines: 1
intelligent: 1
humans: 1
applications: 1
various: 1
fields: 1
including: 1
healthcare: 1
finance: 1
transportation: 1
machine: 1
learning: 1
ml: 1
subset: 1
enables: 1
computers: 1
learn: 1
data: 1
without: 1
explicitly: 1
programmed: 1
```

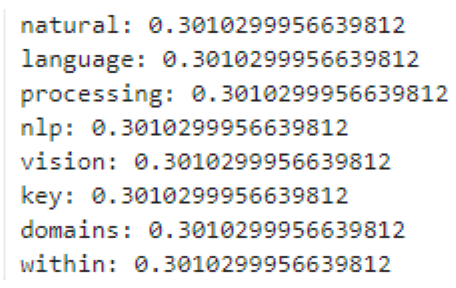


```
natural: 1
language: 1
processing: 1
nlp: 1
vision: 1
key: 1
domains: 1
within: 1
```

Step 5: Calculate IDF and generate matrix.

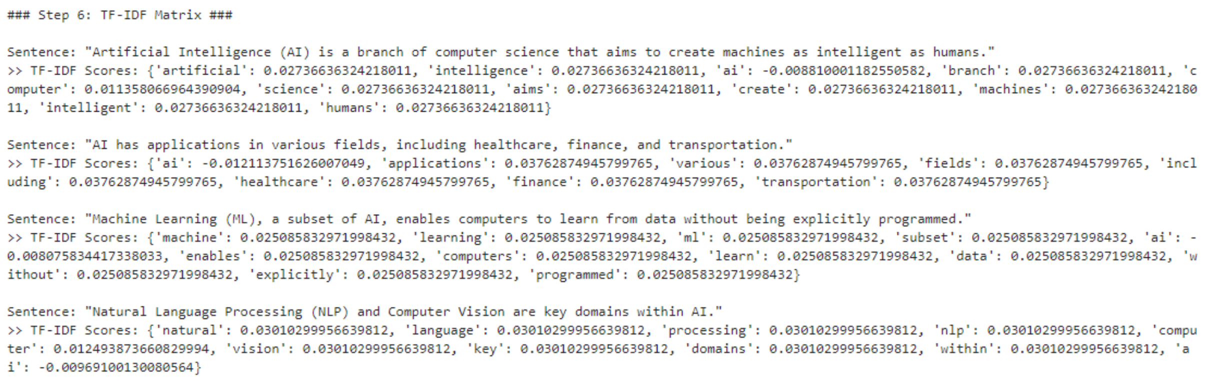


```
### Step 5: Inverse Document Frequency (IDF) Scores ###

artificial: 0.3010299956639812
intelligence: 0.3010299956639812
ai: -0.09691001300805639
branch: 0.3010299956639812
computer: 0.12493873660829993
science: 0.3010299956639812
aims: 0.3010299956639812
create: 0.3010299956639812
machines: 0.3010299956639812
intelligent: 0.3010299956639812
humans: 0.3010299956639812
applications: 0.3010299956639812
various: 0.3010299956639812
fields: 0.3010299956639812
including: 0.3010299956639812
healthcare: 0.3010299956639812
finance: 0.3010299956639812
transportation: 0.3010299956639812
machine: 0.3010299956639812
learning: 0.3010299956639812
ml: 0.3010299956639812
subset: 0.3010299956639812
enables: 0.3010299956639812
computers: 0.3010299956639812
learn: 0.3010299956639812
data: 0.3010299956639812
without: 0.3010299956639812
explicitly: 0.3010299956639812
programmed: 0.3010299956639812
```



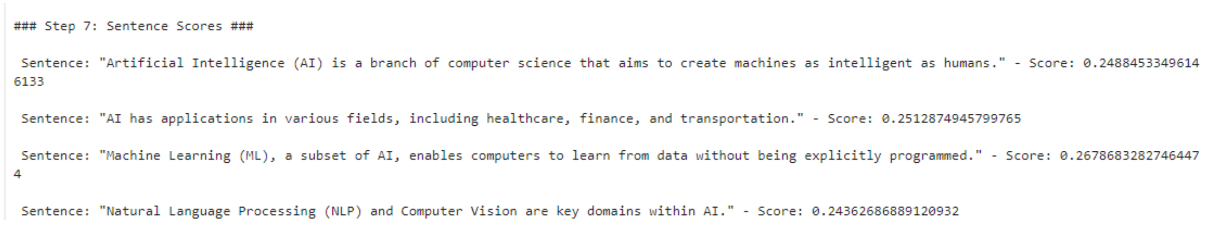
```
natural: 0.3010299956639812
language: 0.3010299956639812
processing: 0.3010299956639812
nlp: 0.3010299956639812
vision: 0.3010299956639812
key: 0.3010299956639812
domains: 0.3010299956639812
within: 0.3010299956639812
```

Step 6: Calculate TF-IDF and Generate Matrix.



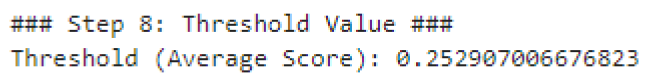
```
### Step 6: TF-IDF Matrix ###

Sentence: "Artificial Intelligence (AI) is a branch of computer science that aims to create machines as intelligent as humans."
>> TF-IDF Scores: {'artificial': 0.02736636324218011, 'intelligence': 0.02736636324218011, 'ai': -0.008810001182550582, 'branch': 0.02736636324218011, 'c
omputer': 0.011358066964390904, 'science': 0.02736636324218011, 'aims': 0.02736636324218011, 'create': 0.02736636324218011, 'machines': 0.027366363242180
11, 'intelligent': 0.02736636324218011, 'humans': 0.02736636324218011}

Sentence: "AI has applications in various fields, including healthcare, finance, and transportation."
>> TF-IDF Scores: {'ai': -0.012113751626007049, 'applications': 0.03762874945799765, 'various': 0.03762874945799765, 'fields': 0.03762874945799765, 'incl
uding': 0.03762874945799765, 'healthcare': 0.03762874945799765, 'finance': 0.03762874945799765, 'transportation': 0.03762874945799765}

Sentence: "Machine Learning (ML), a subset of AI, enables computers to learn from data without being explicitly programmed."
>> TF-IDF Scores: {'machine': 0.025085832971998432, 'learning': 0.025085832971998432, 'ml': 0.025085832971998432, 'subset': 0.025085832971998432, 'ai': -
0.008075834417338033, 'enables': 0.025085832971998432, 'computers': 0.025085832971998432, 'learn': 0.025085832971998432, 'data': 0.025085832971998432, 'w
ithout': 0.025085832971998432, 'explicitly': 0.025085832971998432, 'programmed': 0.025085832971998432}

Sentence: "Natural Language Processing (NLP) and Computer Vision are key domains within AI."
>> TF-IDF Scores: {'natural': 0.03010299956639812, 'language': 0.03010299956639812, 'processing': 0.03010299956639812, 'nlp': 0.03010299956639812, 'compu
ter': 0.012493873660829994, 'vision': 0.03010299956639812, 'key': 0.03010299956639812, 'domains': 0.03010299956639812, 'within': 0.03010299956639812, 'a
i': -0.00969100130080564}
```

Step 7: Score the sentences.



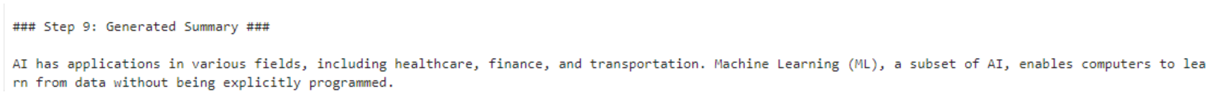
```
### Step 7: Sentence Scores ###

 Sentence: "Artificial Intelligence (AI) is a branch of computer science that aims to create machines as intelligent as humans." - Score: 0.2488453349614
6133

 Sentence: "AI has applications in various fields, including healthcare, finance, and transportation." - Score: 0.2512874945799765

 Sentence: "Machine Learning (ML), a subset of AI, enables computers to learn from data without being explicitly programmed." - Score: 0.2678683282746447
4

 Sentence: "Natural Language Processing (NLP) and Computer Vision are key domains within AI." - Score: 0.24362686889120932
```

Step 8: Find the threshold.



```
### Step 8: Threshold Value ###
Threshold (Average Score): 0.252907006676823
```

Step 9: Generate the Summary.



```
### Step 9: Generated Summary ###

AI has applications in various fields, including healthcare, finance, and transportation. Machine Learning (ML), a subset of AI, enables computers to lea
rn from data without being explicitly programmed.
```

Input Text:

Artificial Intelligence (AI) is a branch of computer science that aims to create machines as intelligent as humans.
AI has applications in various fields, including healthcare, finance, and transportation.
Machine Learning (ML), a subset of AI, enables computers to learn from data without being explicitly programmed.
Natural Language Processing (NLP) and Computer Vision are key domains within AI.

Generated Summary:

AI has applications in various fields, including healthcare, finance, and transportation.
Machine Learning (ML), a subset of AI, enables computers to learn from data without being explicitly programmed.

Analysis:

- The summary effectively highlights key points, such as AI applications and the significance of ML.
- Irrelevant details or examples are omitted, ensuring conciseness.
- The threshold factor dynamically adjusts the inclusion of significant sentences.

---

5. Conclusion

This assignment demonstrates the implementation of a text summarization system using the TF-IDF algorithm. The step-by-step process provides a clear understanding of how to pre-process text, calculate importance scores, and extract key sentences. The results validate the system's ability to retain essential information while discarding extraneous details. Adjusting the threshold factor allows flexibility in the summarization granularity.