

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | Assignment-01:  Text Summarization using Python & NLTK: TF-IDF Algorithm | | | | |  | |
|  |  | | | | | | |  |
|  | | | | SANNIDHI TARUN SRIKANTH |  | | | |
|  | | | | **STUDENT ID: 2023MT12014**November 17, 2024—AI & ML Techniques for Cyber Security—Prof. JAGDISH PRASAD |  | | | |
|  | | |  | | |  | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  |  | |  | | |
|  | INTRODUCTION | | | | | |  | | |
|  |  | | |  |  | |  | | |
|  |  |  | The objective of this assignment is to implement a text summarization technique utilizing Python and the NLTK (Natural Language Toolkit) library, specifically through the application of the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm. Text summarization is a crucial task in natural language processing that involves condensing lengthy documents into shorter summaries while preserving the essential information and overall meaning.  By employing the TF-IDF algorithm, this assignment aims to identify and extract the most relevant sentences from a given text based on their significance, as determined by their term frequency and inverse document frequency scores. This approach not only enhances the efficiency of information retrieval but also aids in presenting concise insights from extensive data sets, making it particularly valuable in fields such as AIML cybersecurity, where quick access to critical information is paramount. | | |  | |  |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Decorative | |  |  |  |  | |
|  | | Overall Solution Design and Process Architecture | | |  | |
|  | |  |  |  |  | |
|  | The solution design for text summarization using the **TF-IDF** algorithm involves a systematic approach that can be broken down into several key steps:   * **Text Input**: The process begins with the input of long pieces of text that need summarization. * **Sentence Tokenization**: The text is divided into sentences using sentence tokenization techniques. * **Frequency Matrix Creation**: A frequency matrix is created to count the occurrences of each word in every sentence. * **Term Frequency Calculation**: The term frequency (TF) is calculated to understand how often a word appears in a sentence relative to the total number of words in that sentence. * **Document Frequency Calculation**: A table is generated to track how many sentences contain each word, which is essential for calculating the inverse document frequency (IDF). * **IDF Calculation**: The IDF scores are computed to determine the importance of words across all sentences. * **TF-IDF Matrix Generation**: The TF and IDF scores are combined to create a TF-IDF matrix, which reflects the significance of words in each sentence. * **Sentence Scoring**: Each sentence is scored based on its TF-IDF values, allowing for the identification of key sentences. * **Threshold Determination**: An average score is calculated to set a threshold for selecting sentences for the summary. * **Summary Generation**: Sentences that meet or exceed the threshold are compiled into a coherent summary.   This structured approach ensures that the summary retains critical information while being concise. | | | | |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Text Summarization using Python & NLTK: TF-IDF Algorithm | PAGE 3 |  |

#### BLOCK DIAGRAM:

**Input Text Data**

**Sentence Tokenize**

**Text Processing (Cleaning, Stopwords)**

**Create TF Matrix (Term Frequency)**

**Create IDF Matrix (Inverse Document Frequency)**

**Calculate TF-IDF Matrix**

**Calculate Sentence Scores**

**Determine Threshold**

**Generate Summary**

**Output Summary**

#### Architectural Diagram:

|  |  |
| --- | --- |
| Text Summarization using Python & NLTK: TF-IDF Algorithm | PAGE 4 |

Text Summarization using TF-IDF Algorithm

Input Layer

**Raw Text Data**

Processing Layer

**Sentence Tokenization**

**Text Preprocessing**

**Term Frequency Matrix**

**Inverse Document Frequency Matrix**

Scoring Layer

**Calculate TF-IDF**

**Sentence Scoring**

Output Layer

**Determine Threshold**

**Generate Summary**

#### Explanation of Diagrams

* Block Diagram:
  + This diagram outlines the sequential steps involved in the text summarization process, from inputting raw text to generating a summary.
  + Each block represents a specific function or operation performed on the text.
* Architectural Diagram:
  + This diagram provides a high-level overview of the system architecture, showing how different layers interact with each other.
  + It highlights the input layer, processing layer, scoring layer, and output layer, making it clear how data flows through the entire system.

These diagrams can serve as visual aids for understanding the architecture and flow of a text summarization system using the TF-IDF algorithm.

|  |  |
| --- | --- |
| Text Summarization using Python & NLTK: TF-IDF Algorithm | PAGE 4 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Decorative | |  |  |  |  | |
|  | | Tool Used and Reasons to Use This Specific Tool | | |  | |
|  | |  |  |  |  | |
|  | **Used Tools**:  Python 3.1.3  NLTK Tool Kit  Py Charm IDE  The primary tool used for this assignment is **Python**, specifically leveraging the **NLTK (Natural Language Toolkit)** library. The reasons for choosing these tools include:   * **Ease of Use**: Python's syntax is straightforward, making it accessible for rapid development and prototyping. * **Rich Libraries**: NLTK provides extensive libraries for text processing, including tokenization, stemming, and stopword removal, which are essential for NLP tasks like summarization. * **Community Support**: Both Python and NLTK have large communities and abundant resources, facilitating troubleshooting and learning through available documentation and tutorials. * **Flexibility**: Python allows for easy integration with other libraries and tools, enabling enhancements or modifications to the summarization process as needed. | | | | |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Text Summarization using Python & NLTK: TF-IDF Algorithm | | | | PAGE 5 | |  | |
| Decorative | | |  |  |  | |  | | | |
|  | | | Final Output Results and Analysis of Results | | | |  | | | |
|  | | |  |  |  | |  | | | |
|  | Output Result for AIML\_GivenCode.py: Code Path: [GivenCode.py](https://github.com/TARUNSRIKANTH/AIML_Assignment1/blob/main/Python_Code/AIML_GivenCode.py)   Analysis of the Given Code and Output The code implements a text summarization technique using the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm in Python with the NLTK library. The objective is to condense a longer piece of text while retaining its key information and overall meaning. Below is a brief analysis of the code and its output.  **Code Analysis**   1. **Functionality**:    * The code begins by importing necessary libraries and defining functions that perform specific tasks in the summarization process, including:      + **Creating a frequency matrix**: This counts the occurrences of each word in each sentence while ignoring stopwords.      + **Calculating Term Frequency (TF)**: This normalizes the frequency of words within each sentence.      + **Calculating Inverse Document Frequency (IDF)**: This measures how unique or rare a word is across all sentences.      + **Computing TF-IDF scores**: This combines TF and IDF to determine the significance of each word in each sentence.      + **Scoring sentences**: Each sentence is scored based on its TF-IDF values to identify which sentences are most important.      + **Generating a summary**: Sentences that meet or exceed a calculated threshold score are selected to form the summary. 2. **Process Steps**:    * The code follows a systematic approach as outlined in the assignment, which includes:      + Tokenizing the input text into sentences.      + Creating frequency matrices and calculating TF and IDF values.      + Scoring sentences based on their importance derived from TF-IDF scores. 3. **Threshold Calculation**:    * The threshold for selecting sentences for the summary is calculated as the average score of all sentences. This ensures that only significant sentences are included in the final summary. 4. **Output Generation**:    * The final output consists of selected sentences that meet or exceed the threshold score, resulting in a concise summary that retains essential information from the original text.   **Output Analysis:**   * The output generated by the code is:   “The virion then releases RNA into the cell and forces the cell to produce and disseminate copies of the virion.”   * **Interpretation**:   + This sentence highlights a critical aspect of SARS-CoV-2's mechanism, indicating how it interacts with host cells. It reflects an important detail about viral replication, which is central to understanding COVID-19's pathology.   + The selection of this sentence as part of the summary demonstrates the effectiveness of the TF-IDF algorithm in identifying significant information within a larger context. * **Conclusion**:   + Overall, the code successfully implements a structured approach to text summarization using TF-IDF, effectively condensing complex information into a meaningful summary.   + The output illustrates that key elements from the original text are preserved, showcasing the algorithm's capability to extract relevant details efficiently.   This analysis confirms that the code meets its objective and provides valuable insights into how TF-IDF can be utilized for text summarization tasks.   |  |  |  | | --- | --- | --- | | Text Summarization using Python & NLTK: TF-IDF Algorithm | PAGE 7 |  |  Output Result for Assignment\_Advancement.py: Code Path: [Assignment\_Advancement.py](https://github.com/TARUNSRIKANTH/AIML_Assignment1/blob/main/Python_Code/Assignment_Advancement.py)   Key Enhancements Made to the Assignment\_Advancement Code:  1. **Dynamic Threshold Adjustment**: If no sentences are selected with the initial adjusted threshold (summary\_threshold\_multiplier \* threshold), we decrease this multiplier by 10% and attempt to generate a summary again. 2. **Debugging Output**: Added print statements to display sentence scores and thresholds during execution. This helps understand why certain thresholds may not yield any summaries. 3. **Robust Summary Generation**: The code now attempts to generate a summary even when higher multipliers are set by adjusting thresholds dynamically.  Additional Feature: **Dynamic Threshold Adjustment**:   * + The code includes logic to adjust the threshold dynamically if no sentences are selected initially. If no summary is generated with the initial threshold, it reduces the threshold multiplier by 10% and attempts to generate a summary again.  Output Analysis **The output generated by the code Assignment\_Advancement.py is:**  “  Sentence Scores: {  ‘Severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) is the strain of coronavirus that causes coronavirus disease 2019 (COVID-19), a respiratory illness.’: 0.6015439222003931,  ‘During the initial outbreak in Wuhan, China, the virus was commonly referred to as “coronavirus” or “Wuhan coronavirus”.’: 0.452267763668791,  ‘In March 2020, U.S. President Donald Trump referred to the virus as the “Chinese virus” in tweets, interviews, and White House press briefings.’: 0.5470341468411741,  ‘Based on whole genome sequence similarity, a pangolin coronavirus candidate strain was found to be less similar than RaTG13 but more similar than other bat coronaviruses to SARS-CoV-2.’: 0.6139861889484661,  ‘Arinjay Banerjee, a virologist at McMaster University, notes that “the SARS virus shared 99.8% of its genome with a civet coronavirus, which is why civets were considered the source’’’: 0.5684252605716873,  ‘The virion then releases RNA into the cell and forces the cell to produce and disseminate copies of the virion.’: **0.7074102276214943**  }  Threshold: 0.5817779183086677  No summary generated with adjusted threshold. Trying with lower multiplier.  Generated Summary:  The virion then releases RNA into the cell and forces the cell to produce and disseminate copies of the virion.  “  **Final Summary**:   * Ultimately, only one sentence was selected for inclusion in the summary:   “The virion then releases RNA into the cell and forces the cell to produce and disseminate copies of the virion.“   * This sentence highlights a crucial aspect of SARS-CoV-2's mechanism of infection, illustrating how it interacts with host cells—a vital detail relevant to understanding COVID-19.  3.Output Result for AIML\_Assignment1.py: Code Path: [AIML\_Assignment1.py](https://github.com/TARUNSRIKANTH/AIML_Assignment1/blob/main/Python_Code/AIML_Assignment1.py)   |  |  |  | | --- | --- | --- | | Text Summarization using Python & NLTK: TF-IDF Algorithm | PAGE 9 |  |  Analysis of the Assignment1 Code and Output: Same as before code Analysis Additional point as below: Key Enhancements Made to the Assignment1 Code:Dynamic Sentence Selection:The summary is generated by selecting the top N sentences based on their TF-IDF scores, ensuring that the most relevant information is included in the final output.Output Analysis:The output generated by the code is as follows: “  **Original Text:**  Severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) is the strain of coronavirus  that causes coronavirus disease 2019 (COVID-19), a respiratory illness.  During the initial outbreak in Wuhan, China, the virus was commonly referred to as "coronavirus" or  "Wuhan coronavirus". In March 2020, U.S. President Donald Trump referred to the virus as the  "Chinese virus" in tweets, interviews, and White House press briefings.  Based on whole genome sequence similarity, a pangolin coronavirus candidate strain was found to be less similar  than RaTG13, but more similar than other bat coronaviruses to SARS-CoV-2.  Arinjay Banerjee, a virologist at McMaster University, notes that "the SARS virus shared 99.8% of its genome  with a civet coronavirus, which is why civets were considered the source."  The virion then releases RNA into the cell and forces the cell to produce and disseminate copies of the virion.  **Summary:**  Severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) is the strain of coronavirus  that causes coronavirus disease 2019 (COVID-19), a respiratory illness. Based on whole genome sequence similarity, a pangolin coronavirus candidate strain was found to be less similar  than RaTG13, but more similar than other bat coronaviruses to SARS-CoV-2.  “ Summary of Analysis for all the 3 codes and same text: The expected output from this text summarization process is a concise summary that captures the essence of the original text while omitting extraneous details.For example, given an input text about SARS-CoV-2, the output summary might include key points such as:   * Identification of SARS-CoV-2 as the virus causing COVID-19. * Historical references to its naming during early outbreaks. * Information about genetic similarities with other coronaviruses.   **Analysis of Results**:   * The effectiveness of the summary can be evaluated based on its coherence, relevance, and retention of critical information from the original text. * Metrics such as ROUGE scores can be employed to quantitatively assess summary quality by comparing it against human-generated summaries.   Code Link:  https://github.com/TARUNSRIKANTH/AIML\_Assignment1/tree/main/Python\_Code | | | | | | | |  | |

|  |  |
| --- | --- |
| Text Summarization using Python & NLTK: TF-IDF Algorithm | PAGE 10 |