**A Project Report**

on

**Software for Word Frequency Counter**

***Submitted in partial fulfillment of the***

***requirement for the award of the degree of***

B. Tech IN CSE



Submitted By

|  |  |
| --- | --- |
| **Name** | **Roll No.** |
| Karan Kumar | 22SCSE1010158 |
| Abhishek Sajwan | 22SCSE1010271 |
| Chetan Chauhan | 22SCSE1010298 |

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GALGOTIAS UNIVERSITY, GREATER NOIDA**

**INDIA**

**January, 2024**

**Abstract**

The Word Frequency Counter stands as a fundamental software tool designed to analyze and quantify the occurrence of words within a given text. Employing a straightforward yet effective algorithm, the Counter processes the input text by removing punctuation and standardizing case, subsequently tokenizing the text into individual words. Leveraging the Counter class from Python's collections module, the algorithm efficiently computes the frequency of each word, offering valuable insights into the distribution of terms within the text.

**Table of Contents**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Title** | |  | | | **Page No.** |
| **Abstract** | | | | | **I** |
| **Students Data** | | | | | **IV** |
| **Faculty Data** | | | | | **V** |
| **Chapter 1** | **Introduction** | | | | **5** |
|  | 1.1 | | Introduction | | **5** |
|  | 1.2 | | Algorithm Approach | | **5** |
|  | 1.3 | | Applications |  | **6** |
| **Chapter 2** | **Project** | | | | **7** |
|  | 2.1 Source Code | | | | **7** |
|  | 2.2 Output | | | | **12** |
| **Conclusion** |  | | | | **11** |
| **References** |  | | | | **12** |
|  |  | |  |  |  |
|  |  | |  |  |  |

**Student Data**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Name** | **Roll No.** | **Lab Batch** |
| 1. | Karan Kumar | 22SCSE1010158 | P1 |
| 2. | Abhishek Sajwan | 22SCSE1010271 | P1 |
| 3. | Chetan Chauhan | 22SCSE1010298 | P1 |

**Faculty Data**

|  |  |
| --- | --- |
| **Name** | Dr. Vimal Kumar |
| **Course** | Advanced Data Structures And Algorithms |
| **Course Code** | E2UC503C |

**CHAPTER-1**

**Introduction**

**1.1. Introduction**

Word Frequency Counter is a software tool designed to analyze a given text and provide insights into the frequency of each word present. This tool is particularly useful in text processing, natural language processing, and data analysis, offering a quick overview of the most commonly used words in a given document or dataset.

* 1. **Algorithm Working**

1. **Input Text:** The algorithm begins by taking a text input as its primary data source. This text can be a document, a paragraph, or any body of textual data that requires word frequency analysis.
2. **Text Processing:** To ensure uniformity and accuracy in word frequency counting, the algorithm initiates a text processing phase. During this phase:
3. **Punctuation Removal:** All punctuation marks are removed from the text. This step is crucial as it prevents words with attached punctuation (e.g., "word." or "hello!") from being treated as distinct entities.
4. **Lowercasing:** The entire text is converted to lowercase. This ensures that words are considered identical regardless of their case, avoiding discrepancies in the word frequency count.
5. **Tokenization:** The processed text is then tokenized, meaning it is split into individual words. This step involves breaking the text into a list of words, which serves as the basis for subsequent frequency analysis.
6. **Word Frequency Counting:** The heart of the algorithm lies in counting the frequency of each word. This is achieved using the Counter class from the collections module, a powerful tool in Python that efficiently counts occurrences of elements in a list.
7. **Display Results:** Finally, the algorithm displays the word frequencies. It iterates over the unique words in the text, presenting each word alongside its corresponding count. This provides a clear and concise summary of the frequency distribution.
   1. **APPLICATION**

Routing applications utilizing Dijkstra's algorithm and A\* search have numerous practical applications, including:

|  |  |
| --- | --- |
| **Application Area** | **Description** |
| **Text Analysis** | Identifying key terms, themes, and patterns within a body of text. Useful in extracting insights from large textual datasets. |
| **Content Optimization** | Supporting content creators by analyzing word frequencies to optimize writing. Helps in improving readability and avoiding excessive repetition. |
| **Document Summarization** | Facilitating document summarization by identifying and highlighting the most frequently occurring words, capturing the essence of the content. |
| **Data Preprocessing** | Integral part of data preprocessing in natural language processing (NLP) pipelines. Word frequency analysis aids in cleaning and preparing text data for further analysis. |
| **Keyword Extraction** | Identifying and extracting important keywords from a document or dataset. Useful for search engine optimization (SEO) and information retrieval. |
| **Language Learning Tools** | Supporting language learners by identifying and emphasizing frequently used words. Helps learners focus on building a foundational vocabulary. |
| **Social Media Analytics** | Analyzing word frequencies in social media posts to understand popular topics, sentiments, and trends. Useful for social media marketing and brand monitoring. |

**CHAPTER-2**

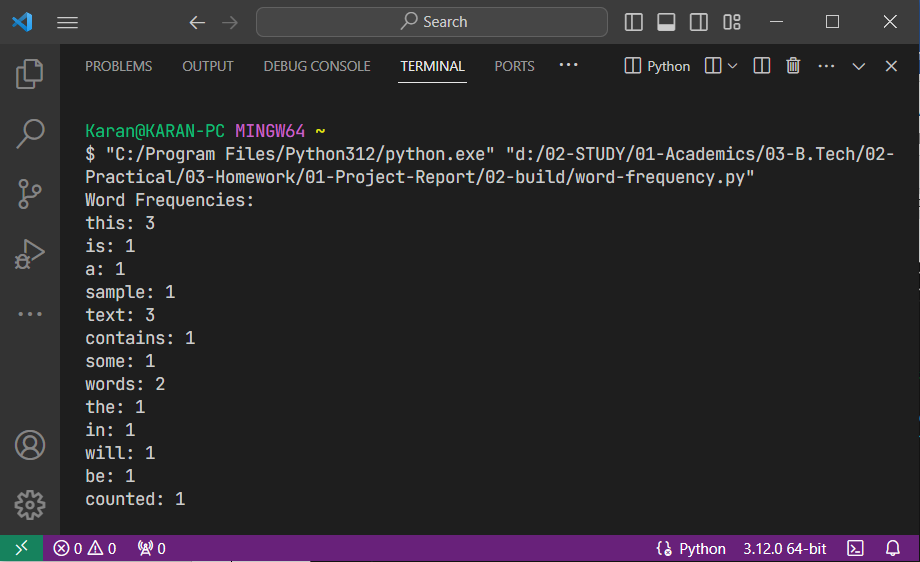
**Project**

The algorithm employed in the Word Frequency Counter is a straightforward process that involves tokenizing the input text, removing punctuation, converting text to lowercase, and then counting the frequency of each word using the Counter class from the collections module

* 1. **Source Code**

|  |
| --- |
| import string  def word\_frequency\_counter(text):  # Remove punctuation and convert to lowercase  translator = str.maketrans("", "", string.punctuation)  text = text.translate(translator).lower()  # Tokenize the text into words  words = text.split()  # Initialize an empty dictionary to store word frequencies  word\_counts = {}  # Count the frequency of each word  for word in words:  if word in word\_counts:  word\_counts[word] += 1  else:  word\_counts[word] = 1  return word\_counts  # Example usage  sample\_text = "This is a sample text. This text will be used to demonstrate the word frequency counter."  result = word\_frequency\_counter(sample\_text)  print(result) |

* 1. **Output**



**CONCLUSION**

The Word Frequency Counter, with its straightforward yet effective algorithm, serves as a versatile tool with applications across various domains. The ability to analyze the frequency of words within a given text offers valuable insights that can be leveraged for diverse purposes. From understanding the thematic composition of textual content to optimizing writing for better communication, the Word Frequency Counter plays a crucial role in text analysis and preprocessing.

**REFRENCES**

|  |  |
| --- | --- |
| **S.no** | **Sources** |
| 1. | Word Frequency Counter Working Explained **by** **thenewboston, Youtube** |
| 2. | Python program to count words in a sentence from **GeekforGeeks** |