**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Academic Year 2023-24**

**CAPSTONE PROJECT**

**Course Code & Name:** CSA1325 & THEORY OF COMPUTATION WITH STORAGE

**Course Faculty:** K.V. KANIMOZHI

**Title:** RECOGNIZING SIMILAR TEXT

**Team No: 2**

1. Reg No. – 192110001-Deendhayal. RR
2. Reg No. – 192110526-Kishore Kumar. S
3. Reg No. – 192211297-Akan. S
   1. **Objective:**

To create algorithms that can efficiently recognize and compare the similarity of various textual material. This has applications in plagiarism detection, information retrieval, text clustering, and content recommendation, allowing for more accurate and efficient processing of textual data.

* 1. **Introduction**

"Recognizing Similar Text" refers to methods for detecting text similarities, such as plagiarism detection, information retrieval, and content suggestion. This overview examines how these algorithms work in a variety of applications, including text clustering, translation enhancement, and data deduplication, highlighting their critical role in efficient information processing and analysis.

**A. The algorithm Overview:**

Learn the fundamentals of text similarity algorithms, including approaches for identifying similarities across texts.

**B. Plagiarism Detection:**

Understand how text similarity affects plagiarism detection systems and how to identify and address instances of copied or closely resembling text.

**C. Information Retrieval Techniques:**

Discuss the use of text similarity algorithms to improve search engine results through textual matching.

**D. Text Clustering Strategies:**

Explore how text similarity can be used to group related documents and improve organizational and analytical processes.

**E. Content Recommendation Systems:**

Understand how text similarity affects content recommendation, including how algorithms examine user preferences to recommend relevant articles, products, and content.

**F. Sentiment Analysis and Textual Similarity:**

Discuss the relationship between sentiment analysis and text similarity, including how algorithms find and measure similarities in sentiments expressed in various texts.

**G. Language Translation Enhancement:**

Explain how text similarity improves language translation systems by recognizing similar phrases or expressions in different languages, leading to more accurate translations.

**H. Data Deduplication Methods:**

Discuss the significance of text similarity in data deduplication, including how algorithms discover and remove duplicate or near-duplicate records to improve data quality.

Each section focuses on unique applications and elements of text similarity in a variety of disciplines.

* 1. **Literature Review:**

**1.**Sajeetha., & Mahesan, S. (2019). Sentiment Analysis in Tamil Texts: A Study on Machine Learning Techniques and Feature Representation. 2019 14th Conference on Industrial and Information Systems (ICIIS), 320-325. <http://doi.org/10.1109/ICIIS47346.2019.9063341>

**2.** Molina, M.., & Garip, F. (2019). Machine Learning for Sociology. Annual Review of Sociology. <http://doi.org/10.1146/ANNUREV-SOC-0731>

**3.**Hancock, E.., Wilson, Richard C.., Ulusoy & Escolano, Francisco. (2010). Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop, SSPR&SPR 2010, Izmir, Turkey, August 18-20, 2010. Proceedings, 6218. <http://doi.org/10.1007/978-3-642-14980-1>

**4.**Arts, Sam., Cassi man, B.., & Gomez, J. C. (2018). Text matching to measure patent similarity. Southern Medical Journal, 39, 62-84. <http://doi.org/10.1002/SMJ.2699>

**Code:**

#include <iostream>

#include <string>

#include <sstream>

#include <unordered\_set>

using namespace std;

int countSimilarWords(const string& sentence1, const string& sentence2) {

unordered\_set<string> words;

stringstream ss1(sentence1);

stringstream ss2(sentence2);

string word;

// Store words from sentence1 in the set

while (ss1 >> word) {

words.insert(word);

}

int similarWords = 0;

// Count similar words in sentence2

while (ss2 >> word) {

if (words.count(word) > 0) {

similarWords++;

}

}

return similarWords;

}

int main() {

string sentence1, sentence2;

// Input the first sentence

cout << "Enter the first sentence: ";

getline(cin, sentence1);

// Input the second sentence

cout << "Enter the second sentence: ";

getline(cin, sentence2);

// Count similar words

int similarWordCount = countSimilarWords(sentence1, sentence2);

cout << "Number of similar words between the two sentences: " << similarWordCount << endl;

return 0;

}