Plagiarism Detector

Team Mates

Abdul Rehman

Ashley Alex Jacob

Raja Ahmed

Problem Statement

Plagiarism in Digital Content

Industry Domain

1- Education

2- Digital Content

Plagiarism & Detection

Plagiarism detection or content similarity detection is the process of locating instances of plagiarism and/or copyright infringement within a work or document. The widespread use of computers and the advent of the Internet have made it easier to plagiarize the work of others.

Objectives

1- To Eradicate Plagiarism

2- To enhance digital transparency

3- Useful in Current Educational Industry Scenarios

Solution Approach

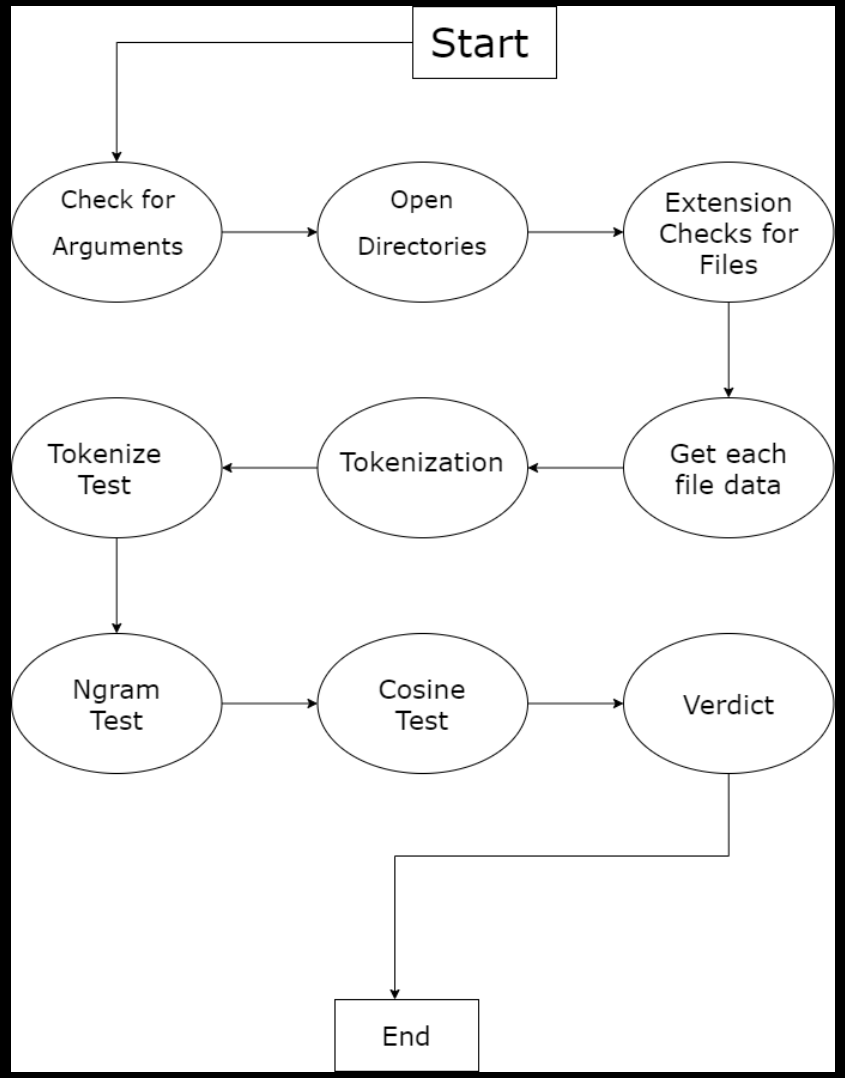
Tests

1. *Token frequency matching*
2. *N-Gram matching*
3. *Cosine matching*

Algorithm

1 Start
2 Check if terminal arguments less than 2 than throw error and close
3 Open the target directory
4 Check for each files extension
5 Then get each file and perform string cleaning
6 Open the database directory
7 Check for each files extension
8 Then get each file and perform string cleaning
9 Then convert each string to tokens for both target and database directories
10 Pass these tokens to test and get the scores and matching files results
11 Pass the results to verdict class to generate final output
12 end

Flow Chart


Start
Check for Arguments
Open Directories
Extension Checks for Files
Get each file data
Tokenization
Tokenize Test
Ngram Test
Cosine Test
Verdict
End

Working

1- The program is supposed to require Database File/Directory and Target File/Directory in parameter.

2- It would check the plagiarism based on combination of keywords from the Database with the target ones.

3- The project will include a library from which user defined function will be called by an object.

How to ?

1. Place all the reference text files in the **database** directory.
2. Place all the text files required to be checked in the **target** directory.
3. *(Optional)* Edit the stopwords.txt text file as per requirement, to add words which are to be ignored in the analysis.
4. Change the database and target_folder variables with the actual location of them.
5. Compile run.cpp in C++11 (or above), with the command g++ run.cpp -std=c++11.
6. Run the generated executable with the command ./a.out *(in Linux environment)*.

Technicalities

Language Implemented == C++

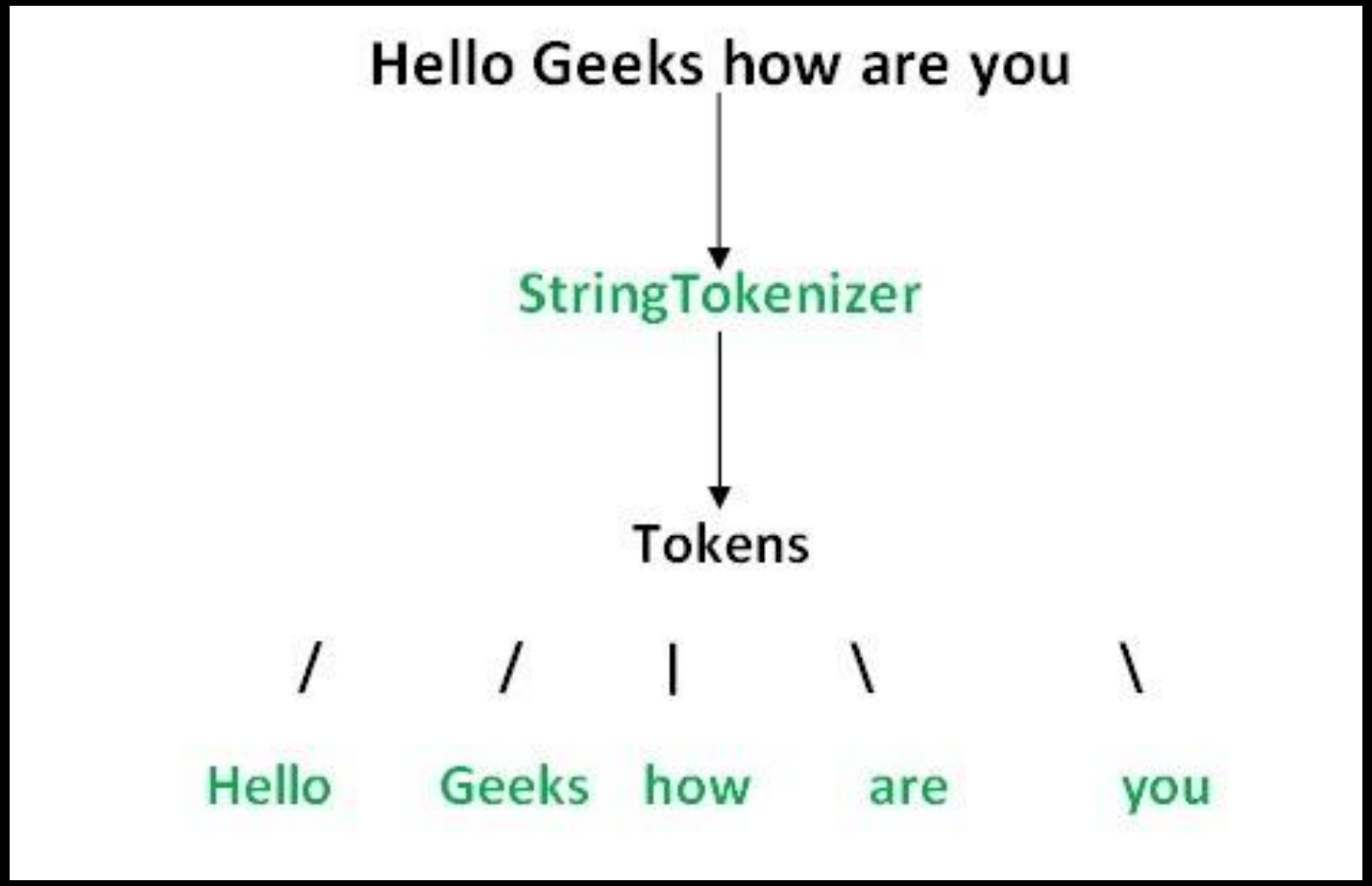
Libraries Used

1-dirent.h
2- algorithm.h

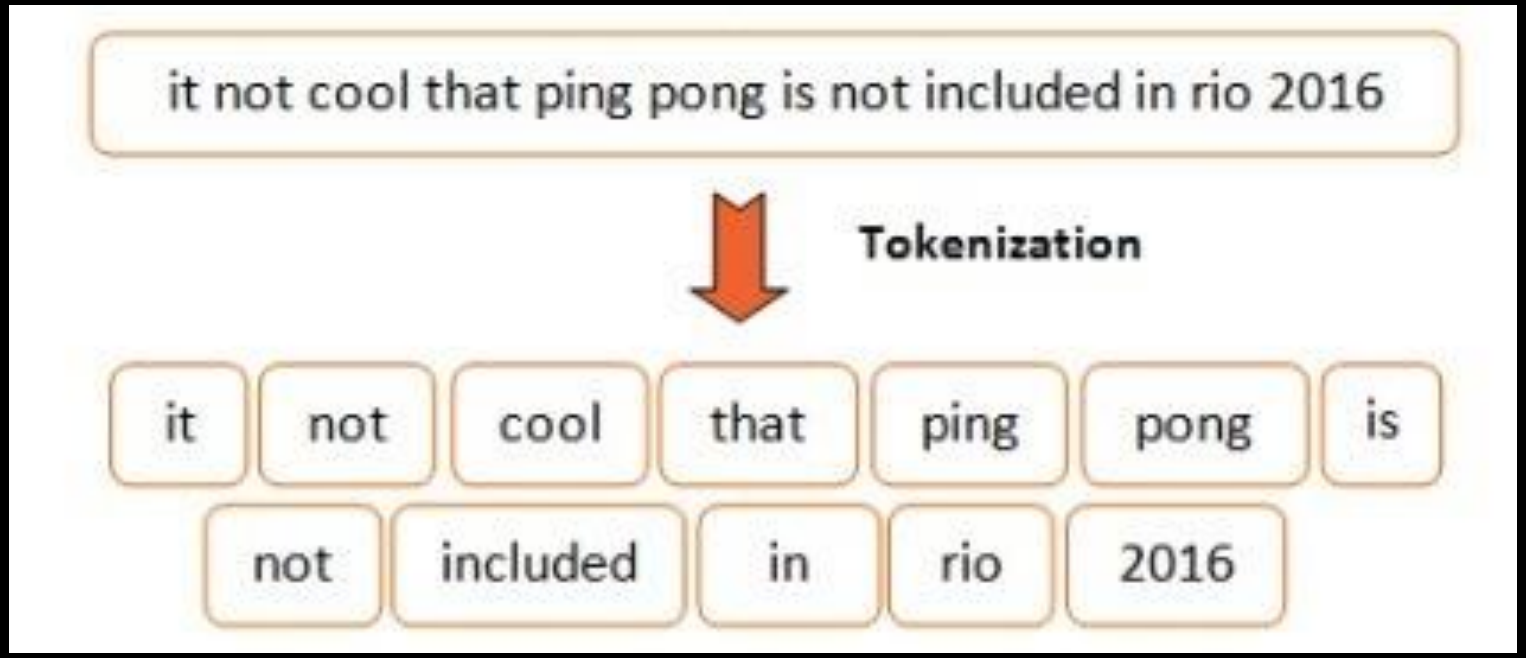
Tests Implemented

1. *Token frequency matching*
2. *N-Gram matching*
3. *Cosine matching*

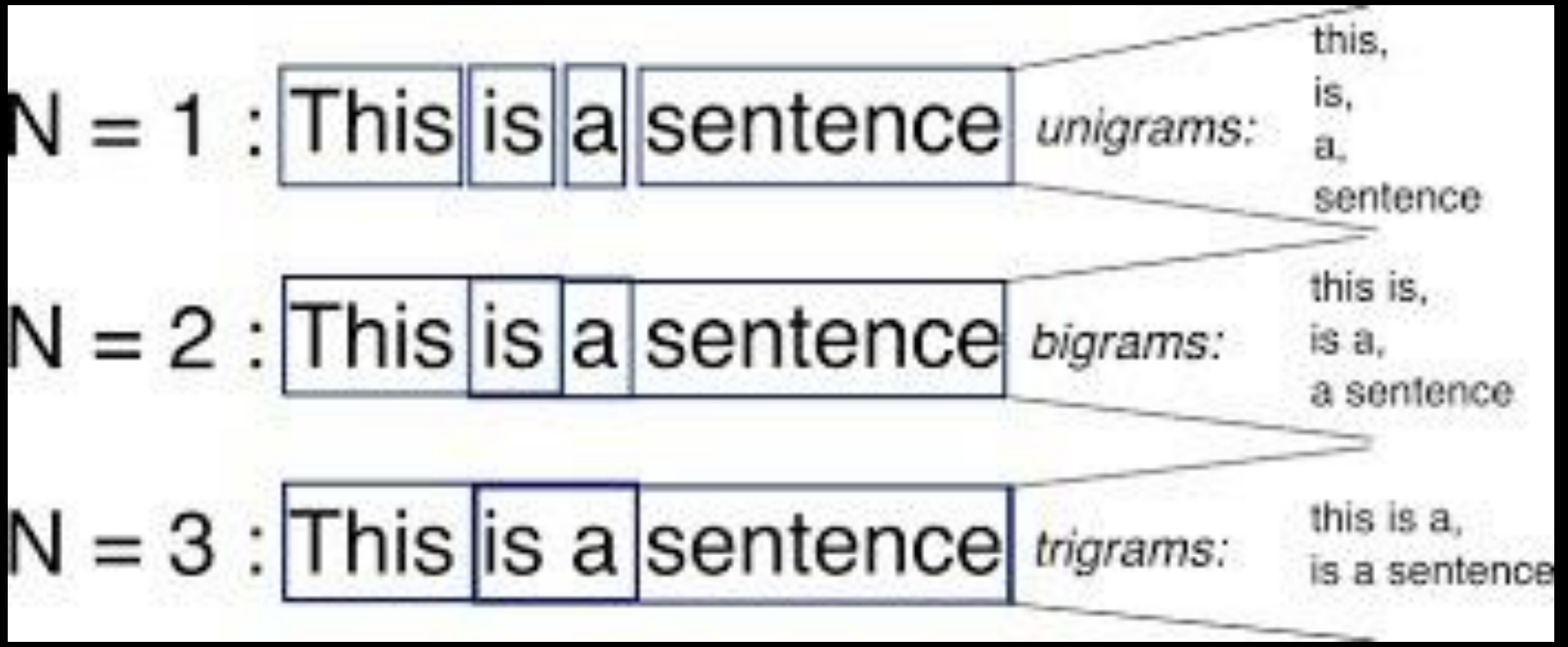
Tokenization



Tokenization



N-Gram


N = 1 : This is a sentence
unigrams:
this,
is,
a,
sentence
N = 2 : This is a sentence
bigrams:
this is,
is a,
a sentence
N = 3 : This is a sentence
trigrams:
this is a,
is a sentence

N-Gram


This is Big Data AI Book
Uni-Gram
This
Is
Big
Data
AI
Book
Bi-Gram
This is
Is Big
Big Data
Data AI
AI Book
Tri-Gram
This is Big
Is Big Data
Big Data AI
Data AI Book

N-Gram


This is Big Data AI Book
Uni-Gram
This
Is
Big
Data
AI
Book
Bi-Gram
This is
Is Big
Big Data
Data AI
AI Book
Tri-Gram
This is Big
Is Big Data
Big Data AI
Data AI Book

Cosine

Cosine similarity is a measure of similarity between two vectors

Levenshtein distance, edit distance, is a string metric
for measuring the difference between two sequences.

Cosine

Cosine similarity is a measure of similarity between two vectors

the bigger the return value is, the more similar the two texts are

Cosine

Cosine similarity is a measure of similarity between two vectors

Used for analyzing the string similarity.

The similarity scales between 0 and 1 (maximum).

The bigger the return value is, the more similar the two texts are.

Convert string to vector to get the degree between strings

$\cos 0° = 1$ 0° means that the two texts are equal
, since two sequences point to the same point.

$\cos 90° = 0$ 90° means that the two texts are totally different