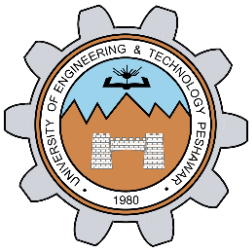
Plagiarism Detector

Final Report



Fall 2020

CSE302L System Programming Lab

Class Section: **A**

Batch**: 20**

Submitted by	**Registration No**
Abdul Rehman	18PWCSE1623
Ashley Alex Jacob	18PWCSE1649
Raja Ahmed	18PWCSE1632

Submitted to:

Engr. Madiha Sher

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Working:

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

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

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

The program will be based on Data structures.

Background:

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.

Problem Statement:

Ease of digital content over World Wide Web caused an abrupt increase in Plagiarism.

Industry Domain:

1. Educational Industry
2. Digital Content Industry

Objectives:

1- To Eradicate Plagiarism
2- To enhance digital transparency
3- Useful in Current Educational Industry Scenarios

Technologies:

We will be going to use to following technologies:

Programming language:

C++:

C++ is a general-purpose programming language created by Bjarne Stroutstrup as an extension of the C programming language, or "C with Classes"

Libraries:

We will be using following libraries for implementation:

1-dirent.h

2- algorithm.h

1-dirent.h:

The <dirent. h> header shall define the ino_t type as described in <sys/types. h>. The character array d_name is of unspecified size, but the number of bytes preceding the terminating null byte shall not exceed {NAME_MAX}.

2- algorithm.h:

The header <algorithm> defines a collection of functions especially designed to be used on ranges of elements.

A range is any sequence of objects that can be accessed through iterators or pointers, such as an array or an instance of some of the STL containers. Notice though, that algorithms operate through iterators directly on the values, not affecting in any way the structure of any possible container (it never affects the size or storage allocation of the container).

Getting started

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)*.

Uses naive methods to detect observable plagiarism in text files.

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

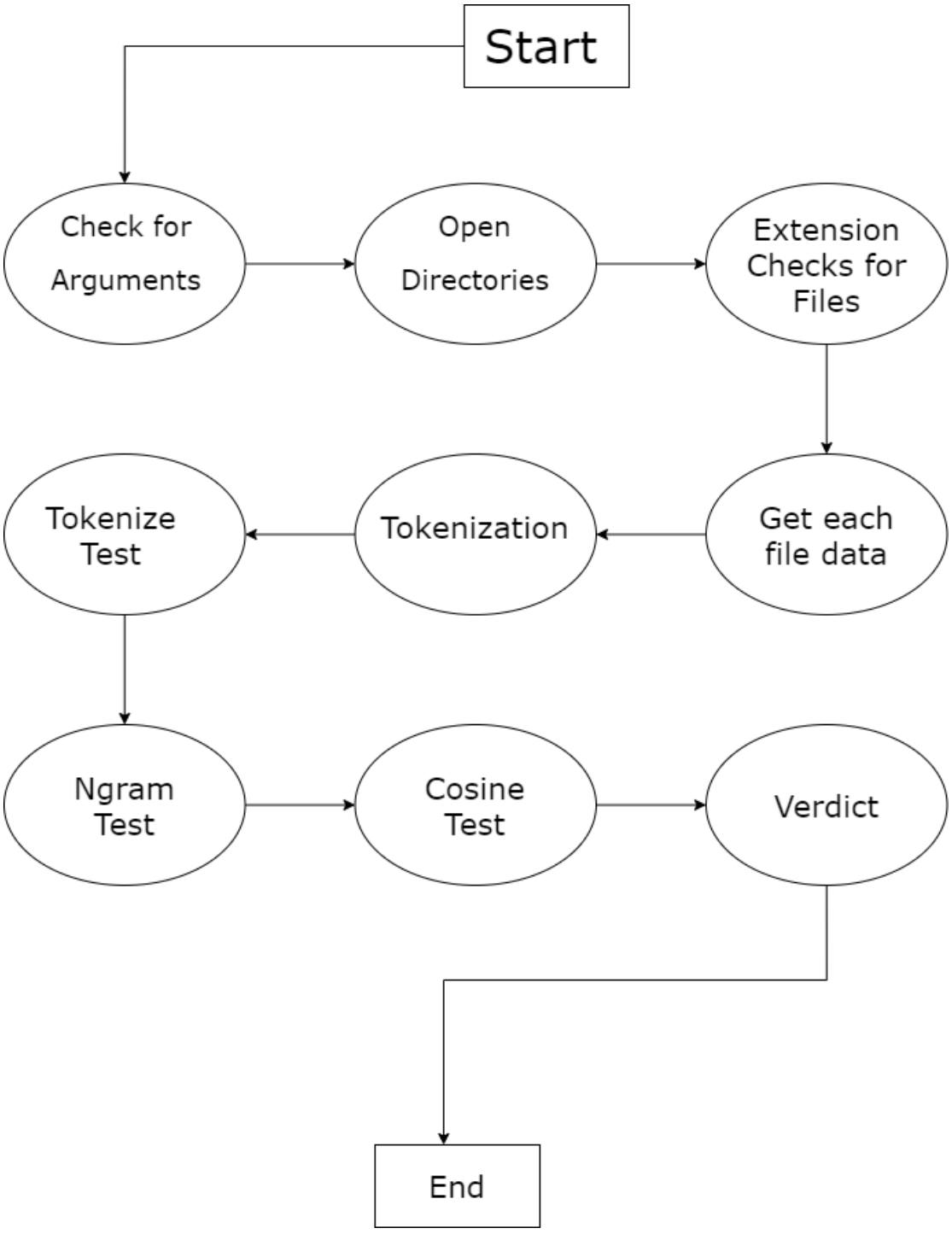
9 Then convert each string to tokens for both target and database directories

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

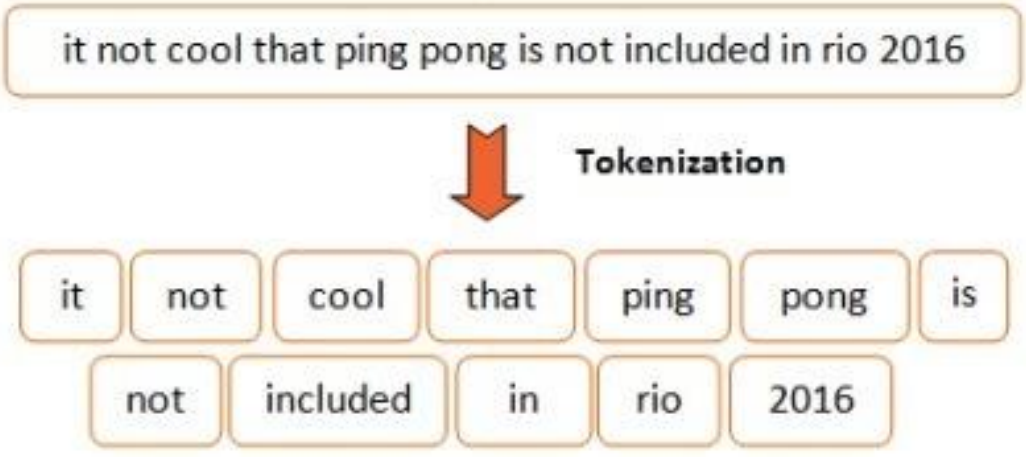
Flow Chart:


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

Tests implemented

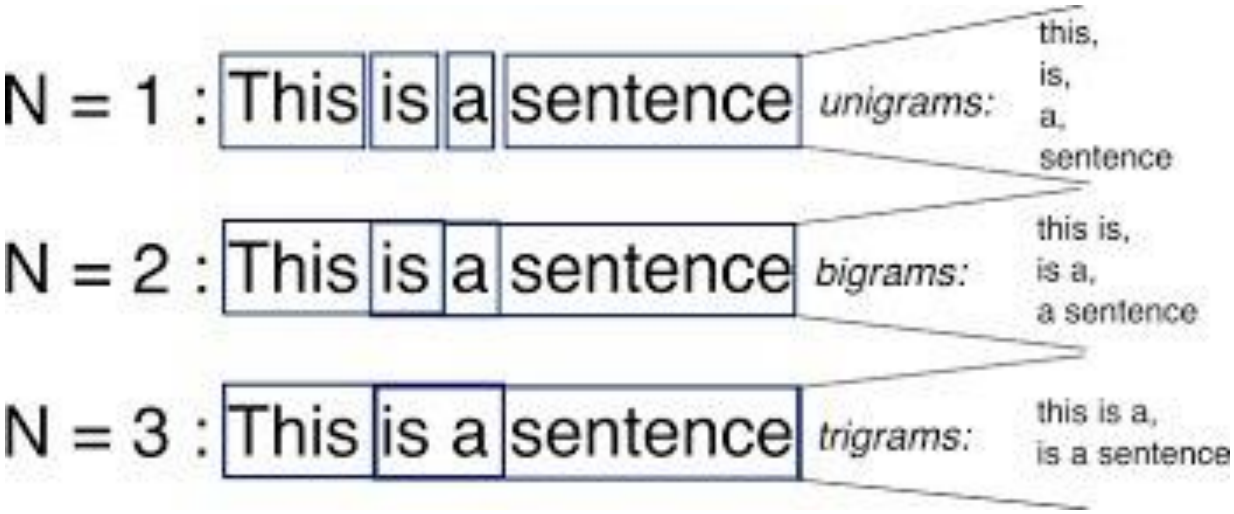
1. Token frequency matching

Target file's word count is matched with each individual text file's token counts in the database folder to find a high degree of similarity in the tokens and their frequency of use.



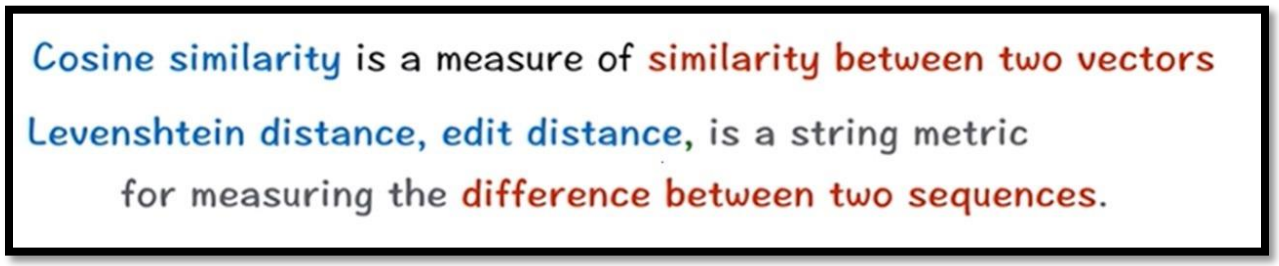
2. N-Gram matching

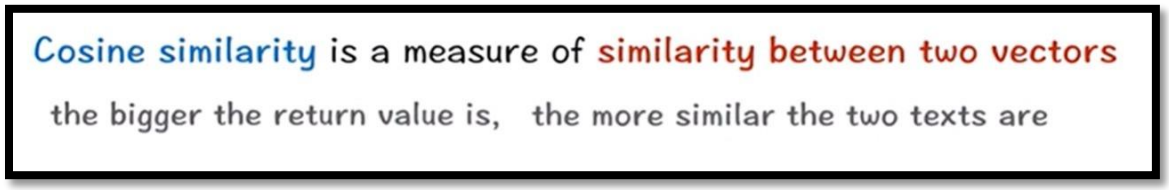
A direct match of consecutive tokens *(or ngrams)* was performed to detect similarity in patterns and neighbourhood of tokens. The value of **N** for the N-Gram generation was varied and a cumulative result was obtained by a weighted average over all of the results.

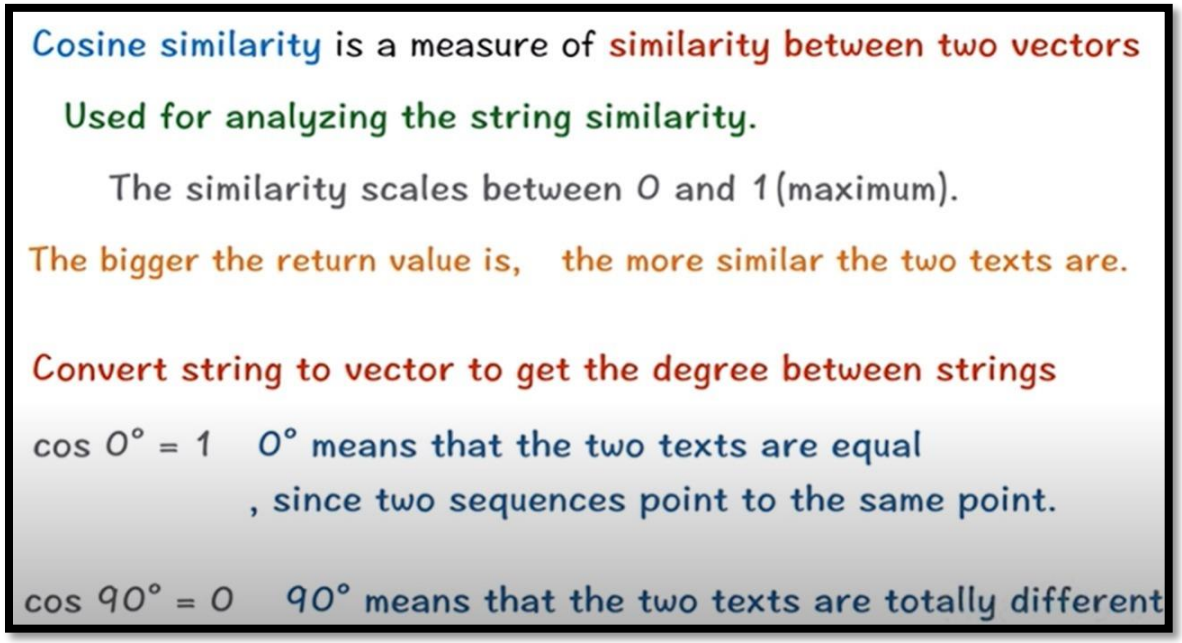


3. Cosine matching

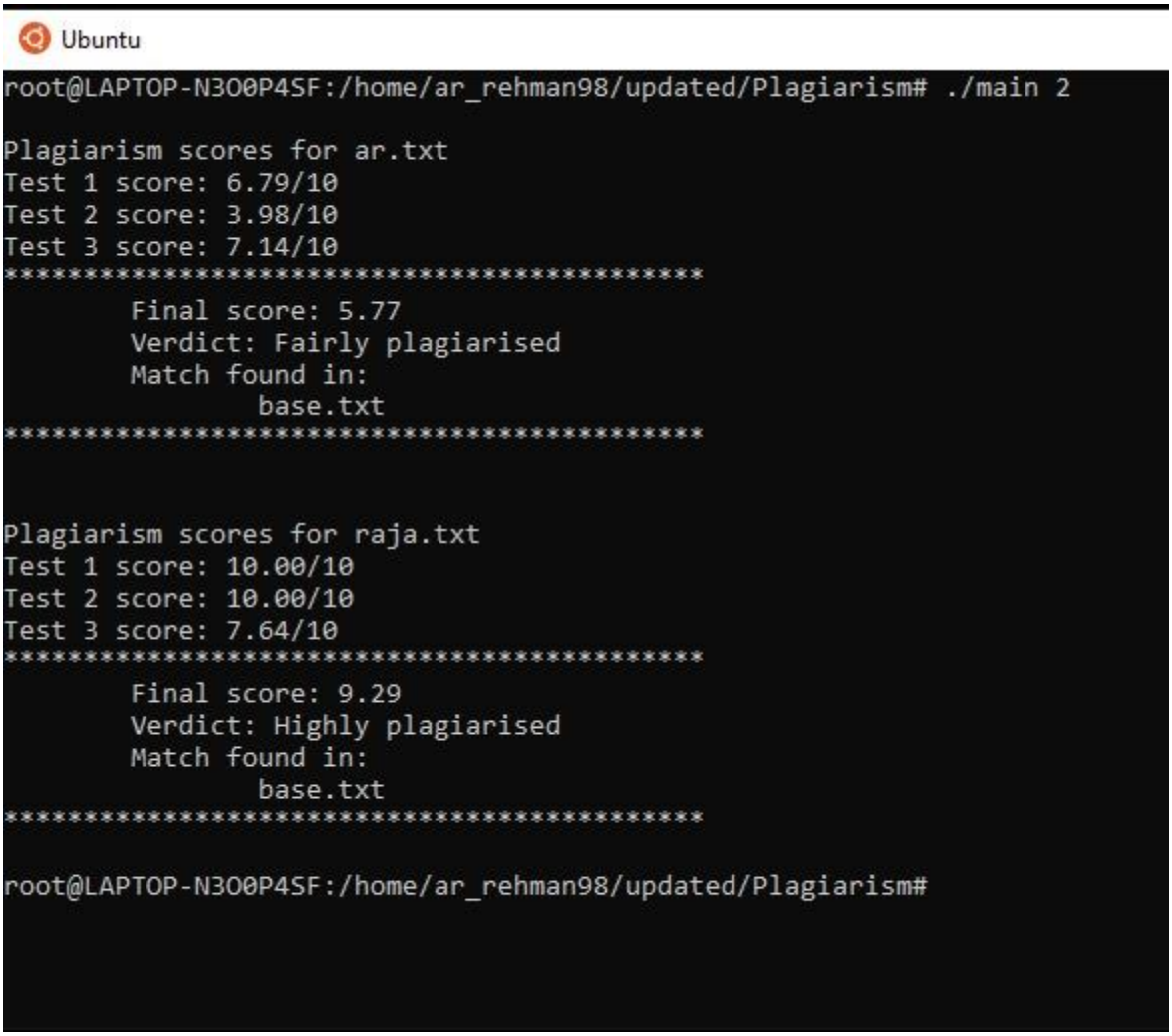
Cosine of the angle between the vectors obtained from the target and the base text files is computed to estimate the similarity in the token vectors of both the files.







Output of CLI Based Project:



Future Work:

We will be going to implement **GUI Version** of this project for better user experience.

GitHub Repository for all the Resources:

https://github.com/AshleyAlexJacob/Plagiarism-Detector-Systems-Programming-Lab-Project

Conclusion:

Via the help of this project we can up to much extent resolve the problem of plagiarism and hence the project is assisting us to solve a real world problem.