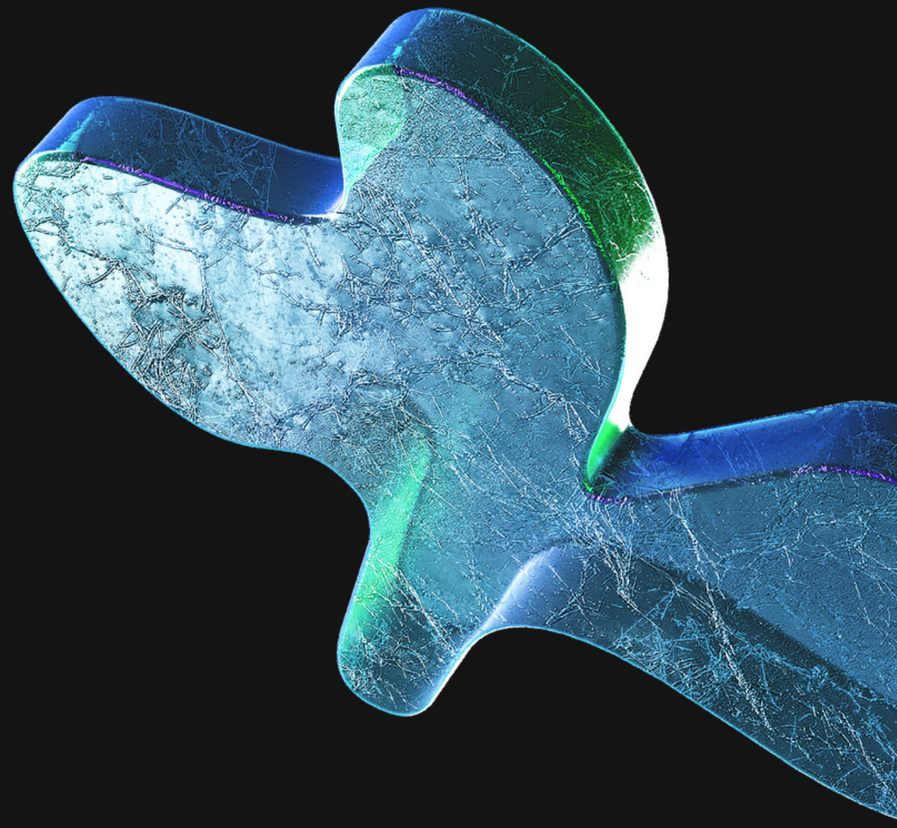
Mini-Project on

Text Plagiarism Checker



Team Members:

Mohit Singh

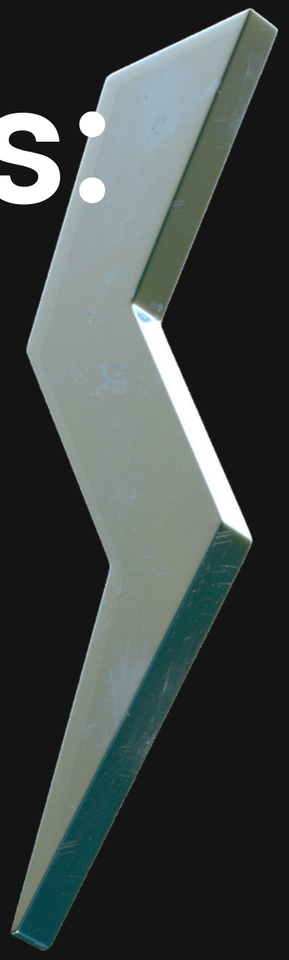
Mihir Gupta

Assessed by:

Payal Garg Mam

Contents:

What you need to know

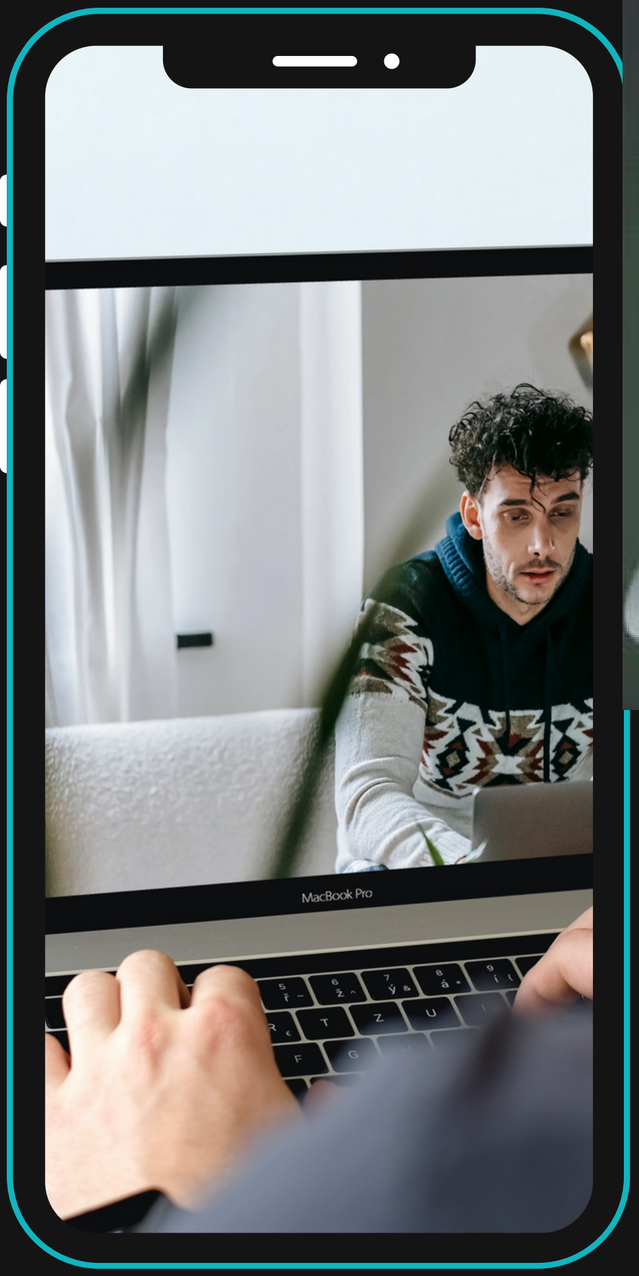


Introduction

Mini-Project Semester III

We have tried to deal with a very significant real -life problem through our mini-project . Plagiarsm is major issue in this world of web-technologies . Every college/university student must have came across it at some point of their lives.

The problem of Plagiarsm goes beyond the campus and ca cause major issues in the sectors of journalism, government activities.



What is Plagiarism?

Plagiarism is presenting someone else's work or ideas as your own, with or without their consent, by incorporating it into your work without full acknowledgement.

Plagiarsm in programming world

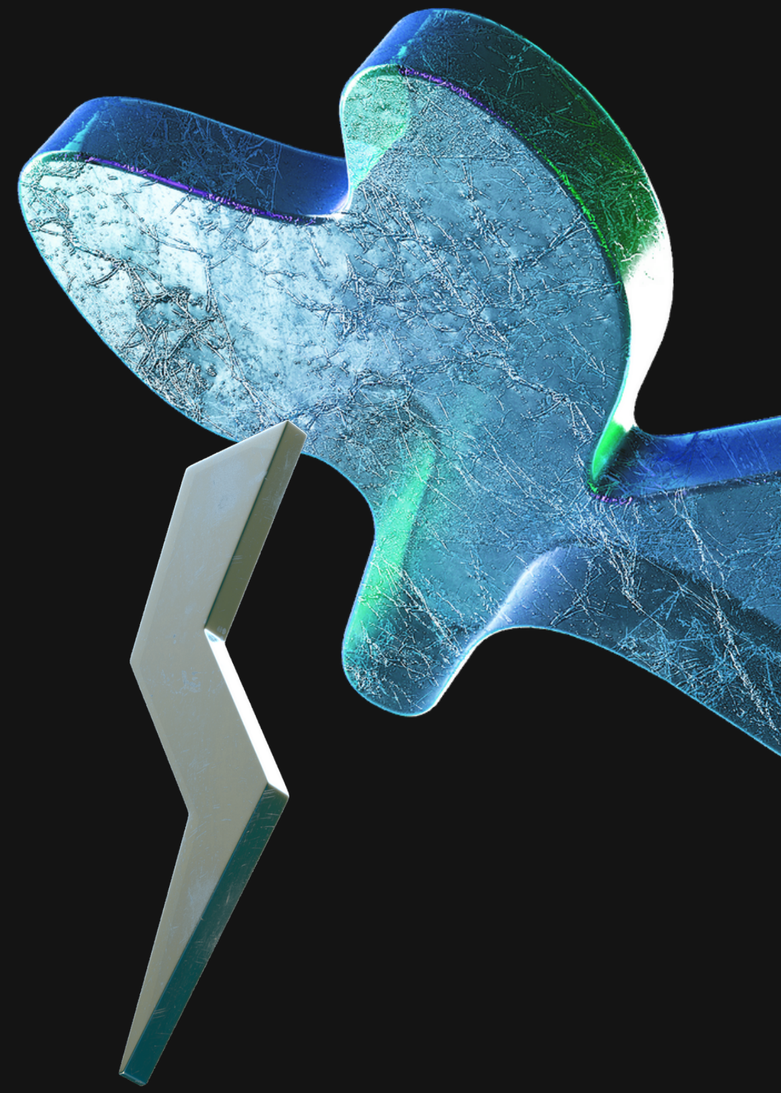
WHAT QUALIFIES AS CODE PLAGIARISM?

Source code plagiarism is defined as copying or reproducing source code without written permission from the original creator. That includes adapting the code minimally, moderately or including fragments of the original author's code in your own code. Converting the original code to a different programming language is still plagiarism, although there are arguments that there is a fine line.

THE DAMAGE DONE BY CHEATING

Careers have been lost over plagiarism that wasn't left in check. The big problem here is that it's the adults in charge that take the brunt of the damage. Students are young and can recover from a fallout based on plagiarism.

A Solution Exists



Instead of suspecting plagiarism, you can be 100% sure of its existence with code plagiarism detection tools. It is a rising problem, and needs to be dealt with in a quick, precise and ultimately discreet way.

Using The Tool To Gain An Advantage
What makes our checker so effective at code plagiarism detection? The platform uses proprietary code similarity algorithms and combines it with an ever-growing database of past submissions. It uses past plagiarist's submissions as their own tools of destruction.

Rabin-Karp Algo

This algorithm uses array tables and Hashing methods in the operation. This hashing method is used primarily to increase search speed by increasing equality testing in the text.

The worst case Time-complexity is O(mn)

where -n is number of text

-m is length of text

ALGORITHM USED

Formula used

Mathematical Equation
(for hash table)

t = (d * t + text[i]% q;

where
*d = 256(char range)
*q = modulo value

./file3.py

File size: 2568 bytes
Comment size: 2.14%
String size: 21.77%
Imports size: 2.18%

```
#coding: utf-8
#Python

import os
import shutil
import platform
import psutil

def duplicate_file(filename):
    if os.path.isfile(filename):
        newfile = filename + ".dupl"
        shutil.copy(filename, newfile)
        if os.path.exists(newfile):
            print("File", newfile, "successful create!")
            return True
        else:
            print("Some problem with copy.")
            return False


def sys_info():
            print("System Info")
            print("Dir:", os.getcwd())
            print("User:", os.getlogin())
            print("OS:", platform.release())
            print("CPU:", os.cpu_count())


def delete_dublies(dir_name):
        i = 0
        file_list = os.listdir(dir_name)
        for f in file_list:
            fullname = os.path.join(dir_name,f)
            if fullname.endswith('.dupl'):
                print ("File",fullname, "deleted." )
                os.remove(fullname)
                i +=1
            else:
                    pass
        return i

print("Hello")
name = input("Enter yor name&:")
```

./file2.py

File size: 2630 bytes
Comment size: 0.42%
String size: 26.35%
Imports size: 1.18%

```
import os, sys, shutil
import psutil

def remove_dubl(dir_name):
    if os.path.isdir(dir_name):
        file_list = os.listdir(dir_name)
        count = 0
        for f in file_list:
            full_path_file_name = os.path.join(dir_name, f)
            if full_path_file_name.endswith('.dupl'):
                os.remove(full_path_file_name)
                count += 1
    else:
        print(dir_name, " is not a directory!")
    return count

def duplicate_file(file_name):
    if os.path.isfile(file_name):
        new_file = file_name + '.dupl'
        shutil.copy(file_name, new_file)
        if os.path.exists(new_file):
            print("File ", file_name, " was duplicated")
            return True
        else:
            print("File ", file_name, " was NOT duplicated")
            return False
    else:
        print(file_name, " is not a file!")
def sys_info():
    print("Currernt directory: ", os.getcwd())
    print("Windows version: ", sys.getwindowsversion())
    print("Default encoding: ", sys.getdefaultencoding())
    print("Login:", os.getlogin())
    print("CPU count: ", os.cpu_count())
    # return 0

print("Welcome to python play ground!")
name = input("Enter your name: ")
print("Nice to see u", name)

answer = ''
while answer != 'q':
    answer = input("Want to work? (y/n/q)")
```

Source Code -->

```cpp
#include "bits/stdc++.h"
using namespace std;
typedef long long ll;
#define quicky ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define rex(n) for(int i = 0; i < n; i++)
#define endl "\n"
#define d 256
#define q 1000

int main(){
    vector < int > hash(1000,0);
    string text;
    float count = 0;
    float tot=0;
    float tat=0;


    ifstream file("in.txt");              // Source Text
    ifstream bfile("out.txt");            // Plagiarized Text


    while (file >> text)
    {
        tot++;
        int t=0;
        int m = text.length();


        if(text[m-1]=='s' || text[m-1] == 'S'){           // omitting s from text (example toys and toy)
            m=m-1;
        }
```

```
33
34      while (file >> text)
35      {
36          tot++;
37          int t=0;
38          int m = text.length();
39
40
41          if(text[m-1]=='s' || text[m-1] == 'S'){            // omitting s from text (example toys and toy)
42              m=m-1;
43          }
44
45
46          for (int i = 0; i < m; i++)
47          {
48              if(text[i] < 91){
49                  text[i] = (char)(text[i]+32);             // converting all uppercase to lower case
50              }
51
52              t = (d * t + text[i]) % q ;                   // calculating hash value of each text
53          }
54
55          hash[t]++;                                        // incrementing value in hash table
56      }
57      while(bfile >> text){
58          int t=0;
59          tat++;
60          int m = text.length();
61
62
63          if(text[m-1]=='s' || text[m-1] == 'S'){
64              m=m-1;
65          }
66
```

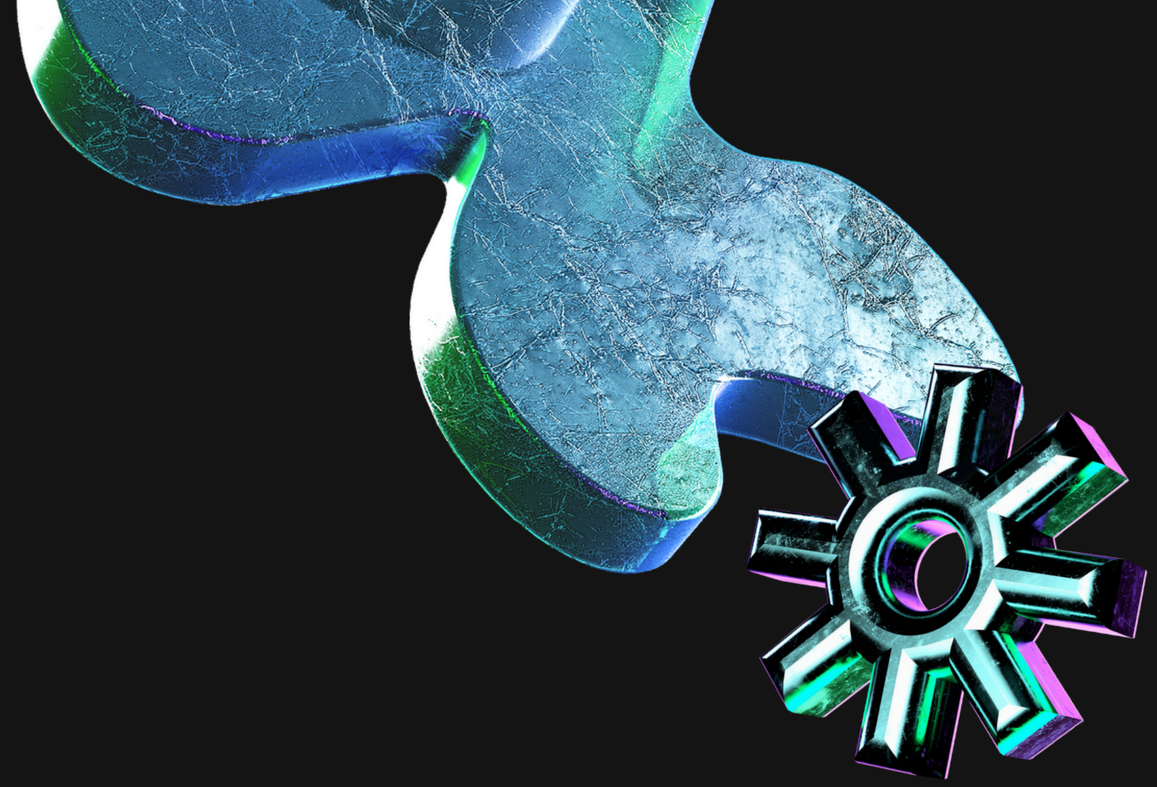
```
    while(brile >> text){
        int t=0;
        tat++;
        int m = text.length();


        if(text[m-1]=='s' || text[m-1] == 'S'){
            m=m-1;
        }


        for (int i = 0; i < m; i++)
        {
            if(text[i] < 91){
                text[i] = (char)(text[i]+32);
            }

            t = (d * t + text[i]) % q;
        }


        if(hash[t]>0){                                  // checking the hash table to count the similar words
            count++;
        }
    }

    cout<<endl<<"percent match = "<<fixed<<setprecision(4)<<(count/max(tot,tat))*100;
}
```

Source Code

https://pastebin.com

References

wikipedia
medium



Conclusion

- There is lot of scope of improvement in this Plagiarism checker.
- Since it shows the percentage of similarity in code it's much easier to decide level of punishment for cheating.
- Coding Competitions will be much more fairer.
- Stealing the source code will be much difficult now.