IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

Plagiar: Plagiarism Detection using Tool for Text Document and Source Code

Teja I[1], Smitha Shekar B[2]

M.Tech, IV Sem, Dr. A.I.T., Department of Computer Science& Engg, Bengaluru[1]

Associate Professor, Dr. A.I.T., Department of Computer Science& Engg Bengaluru[2]

Abstract: Source code written falsification has been a sympathy toward numerous instructors in software engineering field, given to the simplicity of accessibility of substance in this period of web. The tool built up is an instrument for identifying written falsification in source codes of understudies learning programming dialects, to take into account the requirements of educators and help them screen understudies source codes. Right now the instrument underpins Java Programming Language. The instrument works in three stages. Tokenization took after by N-Gram representation of source codes and afterward examination utilizing Greedy String Tiling calculation. Reaction time of the device is one moment for 50 source code records of length 75 lines of code (LOC). According to the exploration, results given by the instrument are ninety-nine percent right. The goal of this anticipate is to build up a device for distinguishing written falsification in both the source code and non-specific printed information documents for conquering the downsides of the current methodologies.

Keywords: Plagiarism, Source Code, Text Document, Tokens, Detection Tool.

I. INTRODUCTION

Unoriginality is the "wrongful assignment" and "taking and distribution" of another creator's "dialect, musings, thoughts, or expressions" and the representation of them as one's own unique work. The thought stays hazardous with hazy definitions and vague standards. Written falsification is viewed as scholastic unscrupulousness and a break of journalistic morals. It is liable to assents like punishments, suspension, and even ejection. As of late, instances of 'great counterfeiting' have been distinguished in the scholarly world.

Written falsification is viewed as scholastic deceitfulness and a break of journalistic morals. It is liable to authorizations like punishments, suspension, and even ejection. As of late, instances of 'great written falsification' have been recognized in the educated community.

Literary theft is not in itself a wrongdoing, but rather can constitute copyright encroachment. In the scholarly world and industry, it is a genuine moral offense. Literary theft and copyright encroachment cover to an impressive degree, yet they are not equal ideas, and numerous sorts of counterfeiting don't constitute copyright encroachment, which is characterized by copyright law and might be arbitrated by courts. Counterfeiting is not characterized or rebuffed by law, but instead by foundations (counting proficient affiliations, instructive establishments, and business substances, for example, distributed organizations).

Types of Source Code Plagiarism

Literary Similarity: Two codes are said to be literarily comparable if the words, variables in the source codes are comparable.

1. Sort 1: This type has one code that is a duplicate of another code aside from changes in space, line dispersing.
2. Sort 2: Same as sort 1 aside from changes to variable names, capacity names.
3. Sort 3: Some lines are added to or expelled from the code which has been replicated, which does not hold any significance.

Utilitarian Similarity: Two codes are said to be practically comparative in the event that they are utilizing the same semantics or performing the same activity. Because of predetermined number of educators, as the quantity of understudies builds it turns out to be more troublesome and tedious for the instructors to physically distinguish literary theft. The manual assessment many–a–times has a tendency to be shallow as the instructor generally just performs an output of the code, and does not check all angles.

Given to the expanding number of understudies, contrasting every source code record and all others to discover literary theft is an exceptionally bulky assignment. Say, in the event that we have around 200 understudies experiencing a programming test, for all intents and purposes it's impractical for an instructor to look at which understudy replicated which part of the project from the other, physically. Indeed, even after strict manual checking probability of hints of written falsification still perseveres. To let take educators a moan of alleviation we have thought of a device for consequently recognizing source code copyright infringement, "Plagiar".

There are for the most part two methodologies utilized as a part of copyright infringement recognition instruments:

1. Trait Based
2. Structure Based

In trait construct approach the center is with respect to specific properties of the system like number of lines, number of words, and number of characters. Two documents with same succession of these properties are viewed as a possibility for written falsification. In structure construct approach the center is in light of structure of the project. The system is initially tokenized and after that tokens are looked at utilizing Greedy String Tiling calculation. The entire center of this paper from now onwards would be on identifying source code literary theft as it is the necessity of our venture "Plagiar". Underneath to sum things up the outline and usage of our apparatus is talked about.

II. RELATED WORK

A writing overview is done to highlight a few thoughts and contentions in the field of learning. The work done in this is to examine the theme of interest and to comprehend the different methods, methodologies and difficulties confronted in it. A portion of the literary theft techniques [1] and designs utilized as a part of the reports, source codes or some other archives are examined and talked about here. The copyright infringement examples or events influences the way the counterfeiting discovery procedures can be composed and actualized. Taking after are a percentage of the papers utilized for writing review.
Salha M. Alzahrani.et al (2012) [4], Plagiarism can be of various natures, running from replicating writings to receiving thoughts, without offering credit to its originator. This paper displays another scientific categorization of literary theft that highlights contrasts between exacting copyright infringement and savvy written falsification, from the liar's behavioral perspective. LifangHan.et al (2010) [3], Plagiarism essentially happens as duplicate and-glue of the code, supplanting the name of capacities or variables, reordering the grouping of the announcement, sort redefinition, et cetera. At present, there are three homologous programming discovery innovation strategies available: content based closeness location, token-based comparability recognition and punctuation structure-based similitude identification. Ahmed Hamza Osman.et al (2013) [5], this paper talks about another unoriginality location technique for content reports called Tree-based Conceptual Matching. The proposed technique not just speaks to the substance of a content report as a tree; however it likewise caught the fundamental semantic significance as far as the connections among its ideas. The strategy was embraced to distinguish copyright infringement in content archives. The tree-based assumed a critical part in this strategy. It took a gander at the measure of identifying appropriated sentences from the first records. [2] Nousheen Samuel.et al (2010), some of the literary theft recognition instruments work just as a standalone apparatuses, some of them are hardcoded into college wide learning administration frameworks. XML based dialect for depiction of literary theft recognition results will serve asa light-weight mix stage that permits written falsification location instruments to be effectively incorporated with existing learning administration frameworks or college gateways. Norman Meuschke.et al(2014) [6], This paper proposes a half breed way to deal with literary theft identification in scholarly reports that coordinates discovery techniques utilizing references, semantic contention structure, and semantic word likeness with character-based strategies to accomplish a higher recognition execution for masked counterfeiting frames. At present accessible programming for unoriginality identification only performs content string correlations. These frameworks discover duplicates, yet neglect to recognize camouflaged copyright infringement, for example, rewords, interpretations, or thought literary theft.
Some of the plagiarism methods [1] and patterns used in the documents, source codes or any other documents are studied and discussed here. The plagiarism patterns or occurrences affect the way the plagiarism detection techniques can be designed and implemented. Following are some of the papers used for literature survey.

Salha M. Alzahrani.et al (2012) [4], Plagiarism can be of many different natures, ranging from copying texts to adopting ideas, without giving credit to its originator. This paper presents a new taxonomy of plagiarism that highlights differences between literal plagiarism and intelligent plagiarism, from the plagiarist's behavioural point of view. LifangHan.et al (2010) [3], Plagiarism mainly happens as copy-and-paste of the code, replacing the name of functions or variables, reordering the sequence of the statement, type redefinition, and so on. At present, there are three homologous software detection technology methods on the market: text-based similarity detection, token-based similarity detection and syntax structure-based similarity detection. Ahmed Hamza Osman.et al (2013) [5], This paper discusses a new plagiarism detection method for text documents called Tree-based Conceptual Matching. The proposed method not only represents the content of a text document as a tree, but it also captured the underlying semantic meaning in terms of the relationships among its concepts. The method was adopted to detect plagiarism in text documents. The tree-based played a very important role in this method. It looked at the amount of detecting plagiarized sentences from the original documents. Nousheen Samuel.et al (2010)[2], Some of the plagiarism detection tools work only as a standalone tools, some of them are hardcoded into university-wide learning management systems. XML based language for description of plagiarism detection results will serve a sa light-weight integration platform that allows plagiarism detection tools to be easily integrated with existing learning management systems or university portals. Norman Meuschke.et al(2014) [6]**,** This paper proposes a hybrid approach to plagiarism detection in academic documents that integrates detection methods using citations, semantic argument structure, and semantic word similarity with character-based methods to achieve a higher detection performance for disguised plagiarism forms. Currently available software for plagiarism detection exclusively performs text string comparisons.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

These systems find copies, but fail to identify disguised plagiarism, such as paraphrases, or idea plagiarism.

III. ARCHITECTURE

Plagiar is an online grading tool for programming languages. It is available on the web for easy access. An organization can also use Plagiar for checking scholars programming capabilities.

A. Plagiar Approach

The tool has utilized Structure based methodology, in the same way as other Plagiarism discovery devices talked about above, wherein a group of documents or catalogs needs the source code to be as a matter of first importance tokenized. The procedure of Tokenization is clarified quickly in outline and execution some portion of this paper. It's extremely key for any written falsification indicator instrument to be extendible, as in any course, learning one dialect is not adequate considering the opposition and evolving innovations. So Plagiarism locator ought to have the capacity to include another dialect for copyright infringement recognition as and when required. We have fused this element in our instrument. New dialects can be included effectively without critical changes in code, so making our device extendible. This device offers extendibility; you simply need to present the arrangement of watchwords of the dialect you need to include.

Achieve capacity is another critical angle for any apparatus, as though it's not reachable then its ease of use will likewise be less. The instrument is sent on web, as a "Plagiar" so achieve capacity is not an issue. Educators can sign in, whenever, from any terminal which has web on it. The terminals with least setup can likewise get the outcomes, as yield is only a html page. No additional establishment of any product is required. The interface is likewise exceptionally easy to understand, instructor simply needs to choose the records, utilizing check boxes, which he/she supposes are copied and can see the outcomes inside a moment or something like that. The yield is a rundown of documents with a rate of coordinating code in them. Additionally there is a catch to see coordinating code. As educator snaps on the catch, the two documents with coordinating codes are indicated one next to the other. The coordinating codes are spoken to with various hues. A tick on one side of coordinating code brings the comparative code on other side in core interest. A preview of the yield is appeared in figure 3 underneath. This gives ease in looking at the outcomes. The educator doesn't need to hold up much to get the outcomes; our instrument offers an incredible reaction time, inside one moment the outcomes can be seen for a cluster of source code documents. That spares time and disappointment, of the instructor, and finds the broken understudies.

B. Design and Implementation

The engineering configuration of our apparatus comprises of a web interface, where records to be checked for written falsification are chosen and results are shown. At that point comes Tokenization and Comparison. Tokenization is dialect subordinate stage and examination autonomous. Tokenizer handles all alone parsing of various dialect source codes. Tokenized source code documents are thought about for literary theft in comparator. Fig. 1 demonstrates the design of Plagiar Plagiarism finder. It has three modules. Web-Interface is the thing that the client sees when he gives a solicitation for recognizing copyright infringement. The solicitation gets sent to Tokenizer which tokenizes the source codes. At that point the yield of Tokenizer is bolstered to comparator and it's here the source codes are examined for similitudes.The Figure 1 shows the architecture diagram of the tool.

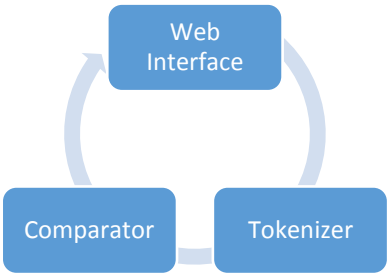

Figure 1: Architecture Diagram

1) Tokenization

Tokenization is the process of converting source code into a sequence of tokens which can then be compared against another token sequence. Tokenization is done in 3 steps
(a) First convert the codes into a lexeme format
(b) Translate into tokens
(c) Change tokens into an appropriate number.
As far as plagiarism detector tool used by Plagiar is concerned we have divided the whole source code into tokens of type
• Header files
• Keywords
• Identifiers
• Operators
• Numerals
e.g.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  printf("Plagiar Software");
}
```

2) Algorithm Rabin Greedy-String Tiling

Definition: A maximal-match is where a substring Pp of the pattern string starting at p, matches, element by element, a substring Tt of the text string starting at t. The match is assumed to be as long as possible, i.e until a non-match or end-of-string is encountered, or until one of the elements is found to be marked. (Marking will be discussed presently.) Maximal-matches are temporary and possibly not unique associations, i.e. a substring involved in one maximal-match may form part of several other maximal-matches.
Definition: A tile is a permanent and unique (one-to-one) association of a substring from P with a matching

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

substring from T. In the process of forming a tile from a maximal-match, tokens of the two substrings are marked, and thereby become unavailable for further matches. With the definitions of tiles and maximal-matches in place it is worth noting that in many situations isolated, short maximal-matches can be ignored.
Definition: A minimum-match-length is defined such that maximal-matches (and hence tiles) below this length are ignored. The minimum-match-length can be 1, but in general will be an integer greater than 1.

The Rabin greedy string tiling algorithm pseudo code is given below
Step 1: starting with the top queue, while there is a non-empty queue do if the current queue is empty then drop to next queue
/* corresponding to smaller maximal-matches */
Step 2: else remove match(p, t, L) from queue
/* Assume the length of maximal-matches in the current queue is L */
Step 3: if match not occluded then
Step 4: if for all j: 0 . . . s − 1, Pp + j =Tt + j then
/* IE match is not hash artefact */
Step 5: for j:= 0 to L - 1 do
Step 6:mark_token(Pp + j)
Step 7:mark_token(Tt + j)
Step 8: length_of_tokens_tiled := length_of_tokens_tiled + L
Step 9: else if L –Loccluded ≥ s then
/* IE the unmarked part remaining of the maximal-match */
Step 10: replace unmarked portion on list of queues

IV. TESTING

Plagiar has undergone load testing for a large number of programs in all languages that it supports. Results have been quite satisfactory and Response time of our tool is one minute for 50 source code files of length 75 lines of code (LOC). A snap shot of the output of the "Plagiar" is shown in the below figures.
The Figure 2 is the snap shot of the Proposed Tool which is showing the file storing repository where all the files uploaded by the user is stored.

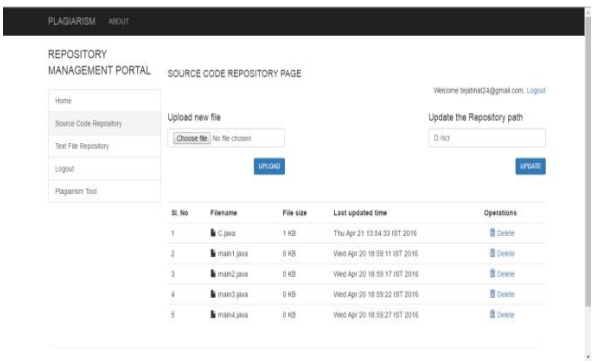

Figure 2: Repository of the Portal

The Figure 3 shows the snapshot of the report generated by the tool which clearly shows the percentage of

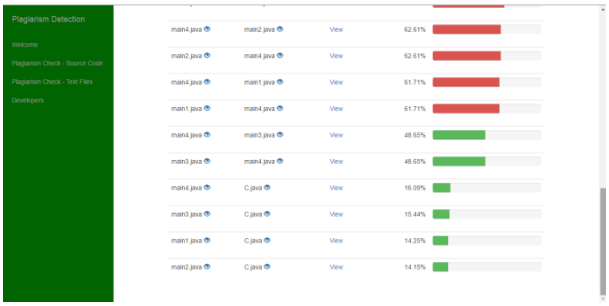

Figure 3: Plagiarism Report

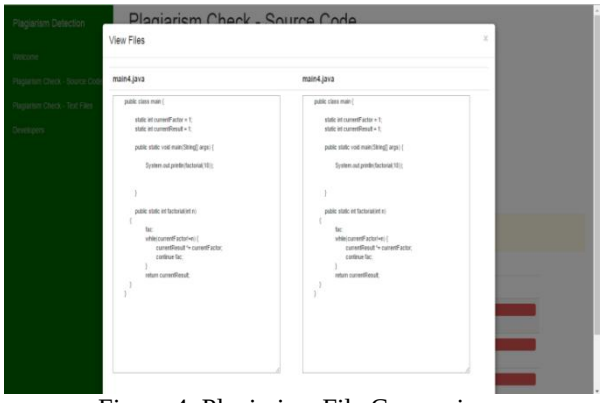

Figure 4: Plagiarism File Comparison

The figure 4 is going to show the snapshot of file comparison after the plagiarism is checked. The files show the content matching in the snapshot.

V. CONCLUSION

In today's world there are many plagiarism Tools available in market. We have tried to make our tool better by overcoming certain flaws in these tools. In "Plagiar" their case sensitivity while matching the text documents. In this tool tokenization technique is used for both source code matching and text document matching, at present plagiar tool is implemented only java platform, if required it can be extended more languages like C#, C, C++ etc.

REFERENCES

[1]. Jurriaan Hage, "Plagiarism detection for Java: a tool comparison" Utrecht University, TB Utrecht, The Netherlands
[2]. Nousheen Samuel, Naima Samuel, Sergey Butakov "XML Based Format for exchange of Plagiarism Detection Results" SolBridge International School of Business Woosong University Daejeon, South Korea. IEEE 978-1-4244-5943-8/10 ©2010.
[3]. Lifang Han, Baojing Cui, Jianxin Wang, YongleHao "Type Redifinition Plagirism Detection of tocken-Based Comparison", International Conference on Multimedia Information Networking and Security 2010.
[4]. Salha M. Alzahrani, Naomie Salim, and Ajith Abraham" Understanding Plagiarism Linguistic Patterns,Textual Features, and Detection Methods" IEEE Transactions On Systems, Man, And Cybernetics—Part C: Applications And Reviews, Vol. 42, No. 2, March 2012.
[5]. Ahmed Hamza Osman1, Naomie Salim2 and Ammar Ahmed E.Elhadi "A Tree-based Conceptual Matching For Plagiarism Detection", International Conference On Computing, Electrical And Electronic Engineering (ICCEEE) 2013.
[6]. Norman Meuschke, Bela Gipp "Reducing Computational Effort for Plagiarism Detection by using Citation Characteristics to Limit Retrieval Space" IEEE 2014.