IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

Plagiarism Tool

Mangesh Chandane[1], Medha Sarang[2], Sudarshan Borade[3], Amol Dhamale[4]

Student, Computer Engineering, ICEM, Pune, India [1,2,3,4]

Abstract: Plagiarism is a growing problem in academia. Academics always use plagiarism detection tools to find similar source-code files. Once similar files are detected, the academic investigates the process which involves identifying the similar source-code fragments within them. These source code fragments could be used as evidence for proving plagiarism. The tool implements a new approach for investigating the similarity between source-code files with a view to gathering evidence for proving plagiarism. Graphical evidence is presented that allows for the investigation of source-code fragments with regards to their contribution toward evidence for proving plagiarism. The graphical evidence indicates the relative importance of the given source-code fragments across files in a corpus. This is done by using the Latent Semantic Analysis information retrieval technique to detect how important they are within the specific files under investigation in relation to other files in the corpus.

Keywords: Java Corpus, LSA, Cosine similarity.

I. INTRODUCTION

In the computer science field, there are number of probabilities that code theft problems are occurred. In the education system, students submit their projects work and the programming assignments, there may be possibility of duplication in source code. The manually plagiarism detection in the source code is a very difficult task. Mostly the people in computer science are using programming assignments of another one. In the plagiarism detection process, there are two parts. In the first part, it generates a representation from a given program. The intermediate representation is used for evaluating the similarity between two programs or projects. A token sequence is often used by intermediate representation. Plagiarism detection system uses the token sequence.

In the second part, system evaluates several techniques and methods are developed for identifying similar code software projects with similar codes. In plagiarism is easy to do, but it is not easy to detect. Usually, when students solve the same problem by using the same programming language source code, there is a high possibility that their assignment solutions will be more similar. Strategies of source code modification that can be used to mask source code plagiarism. Examples of such strategies are renaming identifiers and combining several segments copied from different source code files.

LATENT SEMANTIC ANALYSIS

Latent Semantic Analysis is an information retrieval technique comprising mathematical algorithms that are applied to text collections. Initially a text collection is preprocessed and represented as a term-by-file matrix containing terms and their frequency counts in files. Matrix transformations are applied such that the values of terms in files are adjusted depending on how frequently they appear within and across files in the collection.

SYSTEM ARCHITECTURE

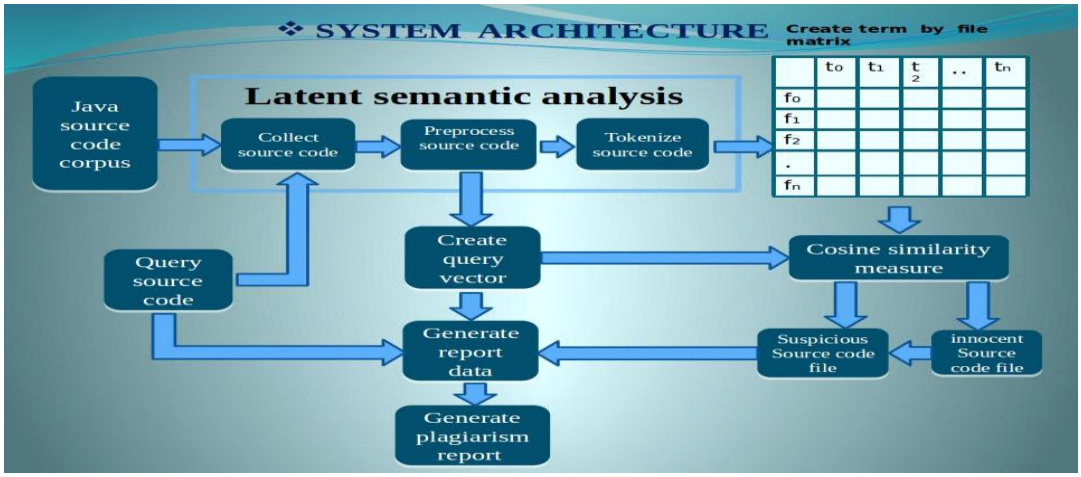

Fig: System Architecture

IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

Java Source Code Corpus:
It contain multiple or number of java application(desktop and web) .Any Java application contain Java source code file, configuration file, libraries, JSP/HTML pages and Java doc.

Collect source code:
We collect all the .java files from each application. We use recursive function to collect all the source code files or java files.

Preprocess source code:
For each .java file we preprocess the data in the following ways:
1.Removing line comments.
e.g. //……..//, ///……..///
2.Removing block comments.
For example, /*…….
……...
*/
3.Removing multiple white spaces excluding single space.
4.Removing multiple line breaks excluding single line break.

Tokenize source code:
After preprocessing each Java file we collect distinct tokens and further we calculate union set of all the collected tokens.

Create term by file matrix:
The term by file matrix will consist of the terms i.e. tokens of source code files as columns and files i.e. Java files as rows. This matrix will contain that how many tokens have occurred in each file.

Query source code:
Query source code is any .java file which is taken as an input. This .java file will undergo in the same technique i.e. Latent Semantic Analysis(LSA).

Create query vector:
In this module, we will get one row as a vector which will contain the tokens.

Cosine similarity measures:
In this we calculate cosine similarity between the query vector and term by file matrix, which we have generated from LSA.

Suspicious source code file and innocent source code file:
When we calculate cosine similarity, we get two files: suspicious source code files and innocent source code files. Suspicious source code files are those files which are copied files and innocent files are those which are uncopied. Suspicious source code file and innocent source code file are recognized between 0 and 1. If we get the value of files greater than 0.5 then it is called as suspicious file. If we get the value of file less than 0.5 then it is called as innocent file.

Generate report data:
We generate the ratio of suspicious source code file and innocent source code file. We need to calculate percentage of plagiarism from each suspicious file. We find maximum percentage of plagiarism in the file which will be the most plagiarized file.

Generate plagiarism report:
We create a pdf of plagiarism report which contain suspicious to innocent ratio, plagiarism percentage is detected in each suspicious file, maximum percentage of plagiarism, overall percentage of plagiarism and the copied tokens will be highlighted from the plagiarized files.

ALGORITHM

- Step 1: Start
- Step 2: Read java source files
- Step 3: Collect source codes, preprocess source code
- Step 4: Generate tokens for each file and union it and create matrix of term vs. file
- Step 5: Read query java source file
- Step 6: Goto step 3
- Step 7: Goto step 4
- Step 8: Find cosine similarity for each file with query source file
- Step 9: If cosinesim > 0.5 then suspicious file else innocent file
- Step 10: If plagiarism found then generate report in pdf
- Step 11: Else Goto step 1

LITERATURE SURVEY

In March 2012 Mike Joy and Georgina Cosma proposed that a novel approach based on the Latent Semantic Analysis information retrieval technique for enhancing the plagiarism detection and investigation process. The main aim is to detect source code files missed by current plagiarism detection tools, to provide visualization of the relative similarity between files and to provide a facility for investigating similar source code fragments and indicate the ones that could be used as strong evidence for proving plagiarism.

In the year 2013, Frederik Hattingh, Albertus A. K. Buitendag and Jacobus S. van der Walt suggests that plagiarism remains an active and ongoing problem and threat to academic institutions. Reactive and proactive methods of plagiarism prevention and detection do not align and complement each other. In traditional source code plagiarism detection engines that consider plagiarism detection a pattern matching problem, no indication is given on the reason for the student choosing to plagiarize. The proposed implementation of detecting plagiarism by utilizing metrics gained over time which is reported in real-time aims to improve the link between proactive and reactive methods of plagiarism detection. The formative

assessment and feedback model presented as part of this paper aims to limit the practice of plagiarism by providing real-time feedback to the student as well as the educator. This process could act as a deterrent for the practice as well as enhance the learning processes of the individual student. By incorporating real time feedback the student is proactively warned of possible plagiarism infringement and can correct the situation. The educator will still have the final say on a case by case basis maintaining the ability to reactively respond to plagiarism.

In Nov 2014 Tapan P. Gondaliya,Hiren D. Joshi (PhD) and Hardik Joshi described the real meaning of source code
plagiarism after that described the different source code plagiarism detection tools and compared its function, characteristics and technique. In the last phase authors discussed the different research papers and compared in tabular form with its technique, method, characteristics, functionality and its result. More and more contributions work towards achieving superlative efficiency and accuracy in the existing solutions.

In 2014, Divya Luke, Divya P.S, Sony L Johnson, Sreeprabha S and Elizabeth.B.Varghese compared six plagiarism detection tools with respect to ten tool features. Performance was compared by a sensitivity analysis on a collection of intentionally plagiarized programs and on a set of real life submissions. The performance was also compared by examining the top 10 results for each tool to the results of the others. The results of the comparison give good insight into the strong and weak points of the different tools.

In the year 2013, Daniela Marinescu, Alexandra Băicoianu, Sebastian Dimitriu described an application used for detection of this kind of plagiarism. This tool analyses more projects in the same time and, by using an efficient algorithm, it can decide if the code is unique or not. This application was designed for helping teachers from Transilvania University of Braşov to effectively detect and thereby prevent plagiarism between students. The application Source Code Plagiarism Detector was proved to be effective and of a great help for teachers. Of course an important role has the lab tutor in formulate the exercises in a special matter so that to be difficult for a student to find the solution on the Internet. It remains the possibility to copy from another student and this application is made to prevent this kind of plagiarism. By reducing the impact of plagiarism, this program has an important role in education of the new generation not only for student period but also for their future life.

In July 2016, Bharamadeo Vishnu Deokate & Dinesh Bhagwan Hanchate, conclude that LSA used in plagiarism detection system, source code and project has been introduced. The problem of the source-code plagiarism becomes more complicated task with the availability of the internet and the growing more web sites. To detect programing plagiarism, an efficient work to optimize the speed and the accuracy of the detection process is important as well as required. This is a difficult and challenging task which is an extension of this work.

FUTURE SCOPE

A scope of the approach is to generate term by file matrix of source code corpus and query vector. After finding this matrix we generate cosine similarity between these query vector and source code files. Finally we generate a plagiarism report which comprises percentage of suspicious files. The input files will be of java only and the system is only concerned with the code part. The system is useful in colleges, academics, IT industries. It can be also used in research centers. The system will be useful in all the areas where innovation is needed and must be aware not to use copied data.

CONCLUSION

LSA used in plagiarism detection system, source code and project has been introduced. The problem of the source-code plagiarism becomes more complicated task with the availability of the internet and the growing more web sites. To detect programming plagiarism, an efficient work to optimize the speed and the accuracy of the detection process is important as well as required. This is a difficult and challenging task which is an extension of this work.

REFERENCES

[1] Mike Joy and Georgina Cosma, "An approach to Source code plagiarism detection and investgation usin Latent Semantic Analysis", 2012
[2] Frederik Hattingh, Albertus A. K. Buitendag and Jacobus S. van der Walt, “Presenting an Alternative Source Code Plagiarism Detection Framework for Improving the Teaching and Learning of Programming ”2013
[3] Tapan P. Gondaliya,Hiren D. Joshi (PhD) and Hardik Joshi,“Source Code Plagiarism Detection SCPDet: A Review ” 2014
[4] Divya Luke, Divya P.S, Sony L Johnson, Sreeprabha S and Elizabeth. B.Varghese, “Software Plagiarism Detection Techniques: A Comparative Study ”2014
[5] Daniela Marinescu, Alexandra Bicoianu, Sebastian Dimitriu,“A Plagiarism Detection System in Computer Source Code ”2013
[6] Bharamadeo Vishnu Deokate,Dinesh Bhagwan Hanchate,“Software SourceCode Plagiarism Detection Using Latent Semantic Analysis ” 2013