Exploring the Usage of Existing Plagiarism Tools for Automated Student Assessment for Java Program

Sonal Jain, *Member*, *IACSIT*, Mayuri Singhal, and Axita Shah

***Abstract*—Plagiarism is the act of copying someone else work without acknowledging the person. Various plagiarism tools exist for natural and programming languages to check if the offence has been committed. This paper discusses the use of plagiarism tools for automated assessment of student java programs. Automated assessment involves evaluating or assessing student's work automatically with the aid of computer. Plagiarism tools are utilized for finding similarity between teacher and student solution. In order to test if the existing tools for source code plagiarism detection can be used to find similarity between student and teacher solutions and how the tools behave in case of different structural and lexical modifications, two of the existing tools; JPlag and Ferret were used to test teacher's solutions with students' solutions. JPlag was further explored with larger testbed, however variations in the results were observed making it difficult to conclude the usage of existing Plagiarism tool for automated assessment.**

***Index Terms*—Automated assessment, plagiarism tools, JPlag, ferret.**

I. Introduction

Contrast to the ordinary setup where students' work and assignments are evaluated by the teacher manually, automated assessment involves evaluating or assessing student's work automatically with the aid of computer. Evaluating and grading student's assignments manually is a very tedious task on part of the teacher, thus Automated assessment and grading systems not only eases this strenuous task of teachers but also are a great tool for assessing and grading in a higher educational setup such as engineering institutions. In case of engineering institutes the assignments submitted by the students are not just plain text but comprise of complicated academic work such as mathematical solutions, algorithms, coding in various programming languages, equations, diagrams etc. Thus more the complicated work more is the manual labor required on part of teacher for grading and assessing of such assignments. Massive Open Online Courses (MOOCs) are recent trend in distance learning, gaining momentum with support of many prestigious universities like Stanford University, Udemi, edX, Coursera and Udacity. However, with increase in popularity of MOOCs, issues and challenges are emerging. Assessing educational achievement and providing feedback to learners is crucial in online distance education. Assessment has been a difficult topic for MOOCs because it is impossible, or at least difficult, for an instructor to provide detailed feedback on performance of thousands of students [1]. Students who take online courses do not have many chances to communicate with their instructors and demonstrate mastery of course content in a direct way [2] fueling demand for online assessments. Automated student assessment systems in such higher educational institutes can provide great relief and support to the teachers who can utilize the time saved in grading and evaluation more effectively.

The main tasks of an automated assessment system are to evaluate assignments given by the instructors to the students to help them apply knowledge gained during the course. Plagiarism is the act of copying someone else work without acknowledging the person. Various plagiarism tools exist for natural and programming languages to check if the offence has been committed. This paper proposes that the plagiarism tools for source code plagiarism detection besides detecting plagiarism can also be employed in online courses for programming languages and educational institutes to check similarity between the teacher's and students' solutions. In this case the similarity is desired to prove that the student has submitted the correct program. Also this can prevent the students from submitting malicious codes that may crash the server as the plagiarism tools will be able to detect that the coding solution submitted by the student is similar to the teacher's solution or not. Section II describes source code plagiarism, Section III shows emergence of automated assessment systems, section IV describes authors' approach, section V includes working of JPlag and Ferret, Section VI demonstrates comparison results by JPlag and Ferret, Section VII includes testing JPlag with larger tested followed by conclusion and future work.

II. Source Code Plagiarism Detection Tools

Various plagiarism detection tools have been developed to detect plagiarism in source code files. These tools are based on various detection approaches which can be classified into three types: text based approach, attribute counting based approach and structure based approach [3]-[8]. Source code plagiarism tools such as JPlag [7], Moss [9], Ferret [9], SIM [10]-[12] etc. exist to find plagiarism in source code of various programming languages.

III. Automated Assessment of Programming Assignments

In case of computer science engineering major stress is on

Manuscript received August 25, 2014; revised October 30, 2014.

Sonal Jain and Mayuri Singhal are with the Department of Computer Science & Engineering, Institute of Engineering and Technology, JK Lakshmipat University, Jaipur, Rajasthan (e-mail: drsonalamitjain@gmail.com, mayuri.singhal81@gmail.com).

Axita Shah is with Elegant Microweb pvt. Ltd, Ahmedabad, India (e-mail: axitashah@gmail.com).

programming assignments. More than the theoretical knowledge of the students their programming skills are valued thus in engineering educational institutes more effort is made to enhance the programming skills of the students. For this purpose various programming labs are used to enhance the practical knowledge of the students. This also requires a great deal of effort on part of teachers to design and distribute programming assignments to the students that increase their programming capabilities. A great deal of effort is also required by the teachers to evaluate the coding solutions submitted by the students and provide them with effective and proper feedback. More complicated solutions require more strenuous task on part of teacher for grading and evaluation. In such a scenario automated assessment systems can prove to be of a great help.

IV. Our Approach

Source code tools are used to check whether similarity exists between original source code and suspicious source code. So if these tools can find similarity between two codes they can be utilized for automated student assessment in educational institutes to check students' programs and in case of MOOC (massive online open courses) where programming languages are taught to the users online. In case of automated student assessment in both the described scenarios, for various programming languages these tools can be used to intentionally find similarity between submitted student program and teacher's solutions at the server end. Teachers along with the question can also submit possible solutions. If similarity between student's solution and teacher's solution exists above a certain threshold that means that the solution provided by the student is almost correct and can be finally submitted to be checked or investigated by the teacher. Fig. 1 shows the flow of the system utilized in an engineering institution.

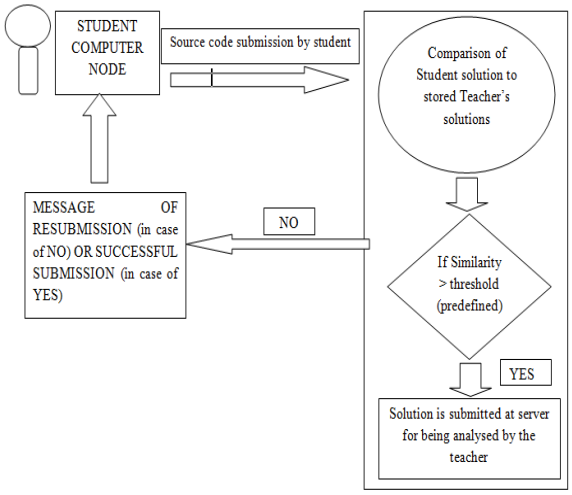


Fig. 1. System flow for usage of plagiarism tools for assignment submission.

As shown in Fig. 1, the student submits the source code solution for programming exercise question provided via online assessment system. The solution is passed to a plagiarism detection tool at the server end. The plagiarism tool checks for similarity between the given solution and stored teacher's solutions for the same question. If similarity is found above a predefined threshold level it is passed to the server for final submission otherwise a message for resubmitting the solution is sent to the student. The predefined threshold is a score that signifies that a match that has score equal to or greater than the defined score is almost similar.

When the plagiarism tools are employed in open online courses and educational institutes for automated student assessment of the programming assignments submitted by the user, it can prevent the students from submitting malicious code that may crash the server. If the student submits the source code solution that contains entirely different code (malicious code) as compared to the coding solution of the query, the code will be rejected by the plagiarism tool due to low similarity to the stored teacher's solutions.

V. Testing JPlag and Ferret for Students' Solutions with Various Structural Modifications

Five java programs were used to test the plagiarism tools, the programs given were: finding given number in an array, finding whether the number given by user is prime or not, first ten numbers in Fibonacci series, calculating factorial of the number given by user, finding the sum of first ten natural numbers. 30 students submitted programs. It was found that students submit programs with Lexical modifications like Space between lines, Additional Comments, Change of Output Statements, Change of variable names, Change of Declarations, Change of Order of Statements as well Structural modifications like Change of Control Structure. JPlag and Ferret are one of the two plagiarism detection tools that are available for free. These two plagiarism detection tools were used to detect similarity between a teacher's solution and students' solutions representing various lexical and structural modifications.

A. Ferret

Ferret [10] is a copy-detection tool, created at the University of Hertfordshire by members of the Plagiarism Detection Group. Ferret locates duplicate text or code in multiple text documents or source files. The program is designed to detect copying (collusion) within a given set of files. Ferret works equally well with documents in natural language (such as English, German, etc.) and with source-code files in a wide range of programming languages. Ferret computes a similarity measure based on the trigrams found within each of the two documents under comparison; this measure is a number from 0 (no copying) to 1 (everything has been copied). This measure should not be taken as an absolute measure of the amount of copying. Instead, the measure is intended to indicate the relative amount of copying that the current pair has compared with the rest of the group. Pairs which appear on top of the table of all similarity comparisons should be examined for possible copying, but the measure itself does not imply any reliable conclusion.

A document may contain natural language text, such as English, or computer programs, such as Java or C. A measure of similarity is computed for each pair of documents, based on matching sequences of words or tokens. Ferret is particularly suited to assist teachers in checking for plagiarism. Large collections of documents can be processed in a single run,

producing a table with every pair of compared documents arranged in order of similarity. A detailed analysis of each pair of documents shows every occurrence of copied text. Reports of the table of comparisons and the detailed analysis may be saved for later reviewing or printing. Fig. 2 shows similarity measure between two codes.

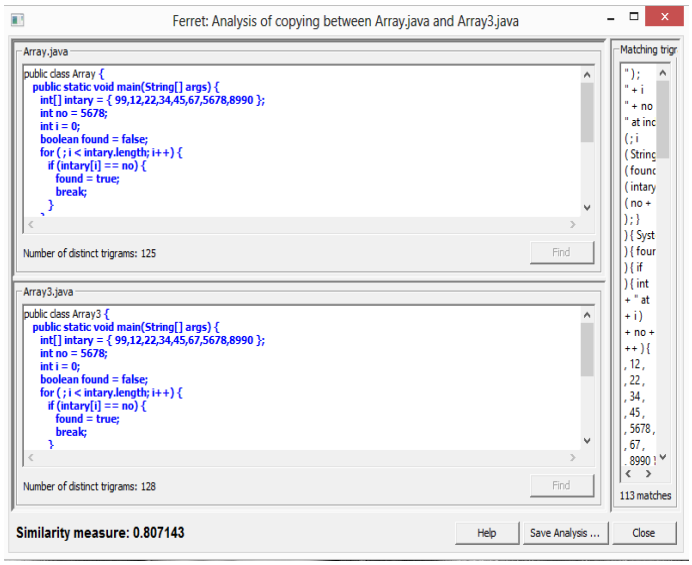


Fig. 2. Analysis showing common trigrams.

B. JPlag

JPlag is a web based plagiarism detection system that was developed by Guido Malpohl at the University of Karlsruhe. In 1996 it started out as a student research project and a few months later it evolved into a first online system. In 2005 JPlag was turned into a web service by Emeric Kwemou and Moritz Kroll. In this the source code is parsed and converted into token rings that represent the structure of the program thus it can be regarded as a tool based on structure approach. The similarity is detected by comparing token rings using Greedy String Tiling algorithm. JPlag includes some context of the program structure into the token strings, for example using the "BEGIN METHOD" token to indicate an open brace at the beginning of a method and "OPEN BRACE" to indicate other open braces. Whitespace, comments, and identifier names are ignored. The languages that are supported by JPlag are Ada, C#, C, C++, Scheme and natural language text. Java Web Start Client is available on the website of JPlag to upload files to the server that are to be compared. JPlag presents its results as a set of HTML pages. The pages are sent back to the client and stored locally. The main page is an overview that includes a table with the configuration used to run the query, a list of failed parses, a chart showing the distribution of the similarity values, and listings of the most similar pairs, sorted by average similarity as well as by maximum similarity.

JPlag compares source-code files submitted in folders or as single files. If each student's work is stored as a separate folder then JPlag will return a similarity score between the folders that contain the suspicious files. If, each student's work is stored as a single file, then JPlag will return a similarity score between the suspicious files detected. The user can begin the comparison process by selecting the folder that contains the student work to be compared as shown in Fig. 3. Fig. 4 shows results in histogram format.

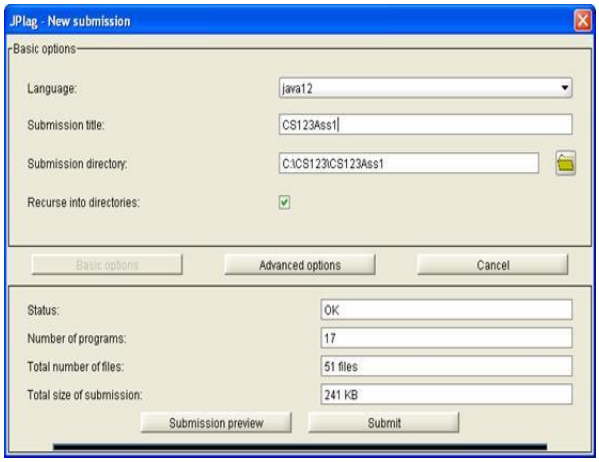
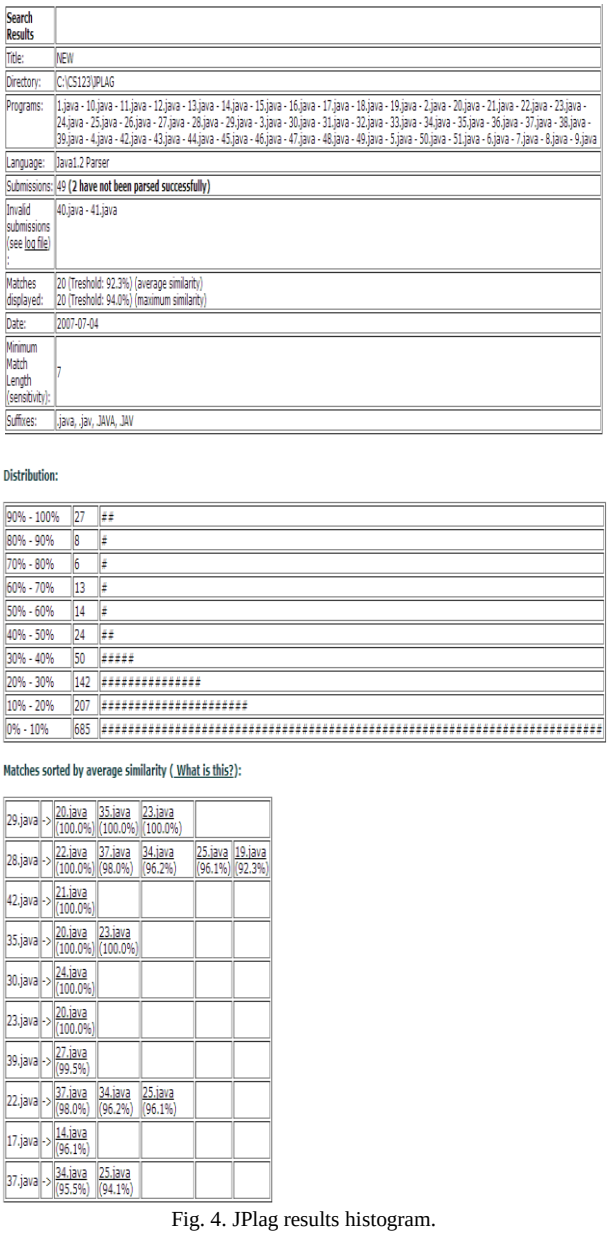


Fig. 3: JPlag Submission selection.



Search Results	
Title:	NEW
Directory:	C:\CS123\JPLAG
Programs:	1.java - 10.java - 11.java - 12.java - 13.java - 14.java - 15.java - 16.java - 17.java - 18.java - 19.java - 2.java - 20.java - 21.java - 22.java - 23.java - 24.java - 25.java - 26.java - 27.java - 28.java - 29.java - 3.java - 30.java - 31.java - 32.java - 33.java - 34.java - 35.java - 36.java - 37.java - 38.java - 39.java - 4.java - 42.java - 43.java - 44.java - 45.java - 46.java - 47.java - 48.java - 49.java - 5.java - 50.java - 51.java - 6.java - 7.java - 8.java - 9.java
Language:	Java1.2 Parser
Submissions:	49 (2 have not been parsed successfully)
Invalid submissions (see log file):	40.java - 41.java
Matches displayed:	20 (Treshold: 92.3%) (average similarity) 20 (Treshold: 94.0%) (maximum similarity)
Date:	2007-07-04
Minimum Match Length (sensitivity):	7
Suffixes:	.java, .jav, JAVA, JAV

Distribution:

90% - 100%	27	##
80% - 90%	8	#
70% - 80%	6	#
60% - 70%	13	#
50% - 60%	14	#
40% - 50%	24	##
30% - 40%	50	#####
20% - 30%	142	###############
10% - 20%	207	######################
0% - 10%	685	##

Matches sorted by average similarity (What is this?):

29.java	->	20.java (100.0%)	35.java (100.0%)	23.java (100.0%)		
28.java	->	22.java (100.0%)	37.java (98.0%)	34.java (96.2%)	25.java (96.1%)	19.java (92.3%)
42.java	->	21.java (100.0%)				
35.java	->	20.java (100.0%)	23.java (100.0%)			
30.java	->	24.java (100.0%)				
23.java	->	20.java (100.0%)				
39.java	->	27.java (99.5%)				
22.java	->	37.java (98.0%)	34.java (96.2%)	25.java (96.1%)		
17.java	->	14.java (96.1%)				
37.java	->	34.java (95.5%)	25.java (94.1%)			

Fig. 4. JPlag results histogram.

JPlag has worked very fast, and returned only the suspicious pairs of files. This tool allows the user to view the entire detected files clearly and provide a clear indication of the suspicious source-code fragments by colour indicating the suspicious fragments. It also provides easy navigation for

viewing the suspicious source-code fragments between files that contain similar code that has been rearranged in different positions. One of the drawbacks of JPlag is that it cannot handle files which do not parse. Because of this JPlag has missed the suspicious file pair 37 and 40. A nice feature of JPlag is that it displays the groupings of suspicious files found which is a very useful feature for catching suspected plagiarism between groups of students.

VI. Results by JPlag and Ferret for Students' Solutions with Various Structural Modifications

Table I shows the results for Ferret and JPlag for the student programs with lexical and structural modifications when compared with the teacher's solutions. Percentage indicates similarity between student's solution and teacher's solution.

TABLE I: Similarity Mesure between Programs Generated by Two Plagiarism Tools

Modification	Student Program	JPlag (%)	Ferret (%)
Space Between Lines (Lexical)	Array1.java	100	95
	Prime1.java	100	95
	Fibonacci1.java	100	93
	Factorial1.java	100	94
	Sum101.java	100	93
Additional Comments (Lexical)	Array2.java	100	76
	Prime2.java	100	78
	Fibonacci2.java	100	73
	Factorial2.java	100	74
	Sum102.java	100	68
Change of O/P Statements (Lexical)	Array3.java	100	80
	Prime3.java	100	79
	Fibonacci3.java	100	77
	Factorial3.java	100	75
	Sum103.java	100	79
Change of Variable Names (Lexical)	Array4.java	100	49
	Prime4.java	100	43
	Fibonacci4.java	100	24
	Factorial4.java	100	39
	Sum104.java	100	29
Joint Declaration (Lexical)	Array5.java	100	90
	Prime5.java	100	86
	Fibonacci5.java	100	72
	Factorial5.java	100	84
	Sum105.java	100	82
Change of Order of Statements (Structural)	Array8.java	59	89
	Prime6.java	58	86
	Fibonacci6.java	58	84
	Factorial6.java	56	85
	Sum106.java	59	80
Additional Print Statements (Structural)	Array9.java	87	77
	Prime7.java	89	75
	Fibonacci7.java	88	72
	Factorial7.java	88	84
	Sum107.java	87	70
Change of Control Structure (Structural)	Array10.java	48	79
	Prime8.java	47	75
	Fibonacci8.java	45	63
	Factorial8.java	48	76
	Sum108.java	48	57

JPlag performed well for all the modifications but there was a drop in similarity percentage in case of structural modifications. The drop was drastic in case of modifications where the order of the statements was changed and different control structure was used. Ferret performed well for most of the modifications with similarity above 70. But there was a drop in case of lexical modification where different variable names were used. It is a matter of concern as in case of online assessment system the student solution is more prone to this lexical modification as it is obvious that the variable names used by the student will be different from the variable names used by the teacher unless specified in the question that what variable names should be used. Specifying names is quite impractical as a program consists of a large number of identifiers such as variable names, class names, method names etc. The reason why JPlag not performed well for the structural modifications can't be stated as its working is not known. In case of Ferret, the reason it was not able to detect similarity in case of change of variable names might be the change of trigrams due to change in variables. It was concluded that JPlag was more suitable to test java programs particularly when intention is to assess similarity measure between student solution and teacher solution.

VII. Testing JPlag for Automated Assesment

To achieve goal of automated assessment, a testbed of java programming exercises was created and two folders named teacher and student were used to store the solutions for the programming exercises. The teacher folder comprised of solutions given by the teacher for the programming questions and the student folder is used to store the solutions provided by the students. The plagiarism detection tool JPlag was used to compare the source code files by the teacher with the solutions provided by the students as files stored in folders. Testbed included sample java progams shown in Table II.

TABLE II: Test Sample

1. Take ten numbers from the user as input and print even and odd numbers. Also display total number of even and odd numbers?
2. Write a program that consists of a separate class called Fibonacci with a method to find Fibonacci series of the number provided by the user.
3. Write a program that consists of class called Rectangle with two methods; one is for length and breadth (passed as parameters) and the other to calculate the area of rectangle.
4. Write a program that consists of a class called Room. This class consists of constructor for length and breadth and a method to calculate its area. Now extend this class, the subclass consists of constructor for variable height (length and breadth of super class using super keyword) and a method to calculate volume.

Table III shows the result of JPlag based on which solution is correct or incorrect was assumed.

The results received by the test are not constant and show variations. All the programs submitted by the student 2 are correct coding solutions but JPlag gave very low similarity percentage to all the solutions which will result in their rejection if JPlag is implemented to check similarity between student and teacher solution. One of the reasons of this low percentage is the difference in the structure of teacher's and student's coding solutions and less number of variations provided by the teacher. JPlag performs poorly in case

structural modifications which has been shown in section VI. For example, student 2 has used two loops for first java program number.java to print even and odd numbers whereas in the coding solution provided by the teacher only one loop is used for printing even and odd numbers. Also the first program submitted by student 3 ArrayEvenOddNumber.java is correct but JPlag was unable to parse the program and thus no comparison is done. The reason JPag was not able to parse the program is not known as JPlag is not an open source plagiarism tool. The coding solution Room.java submitted by the student 4 is incorrect but JPlag assigns it a score of 69 which is quite high, one of the reasons for this score can be the use of scanner statements in both teacher and student coding solution.

TABLE III: RESULT OF ASSESSMENT SYSTEM UTILIZING JPLAG

Student Number	Student Program	Correct / Incorrect	JPlag (max %)
1	Q1.java	C	100
	Q2.java	C	100
	Q3.java	C	66
	Q4.java	C	93
	Q5.java	C	83
2	number.java	C	18
	series.java	C	0-10
	area.java	C	37
	room.java	C	38
	series.java	C	0-10
3	ArrayEvenOddNumber.java	C	Parse error
	FibonacciSeries.java	I	0-10
	RectArea.java	I	0-10
	AreaofRoom.java	C	100
	Prime.java	I	32
4	FindEvenOrOddNumber.java	I	0-10
	Fibonacci.java	C	0-10
	Area.java	C	100
	Room.java	I	69
	PrimeNumbers.java	I	0-10
5	OddorEven.java	I	0-10
	Fibonacci.java	I	0-10
	CalculateRectArea.java	I	0-10
	AreaVolume.java	C	72
	Sum.java	I	0-10

VIII. CONCLUSION AND SUMMARY

Freeware plagiarism tools JPlag and Ferret were used to test teacher's programming solutions for five java programs with programming solutions of students with various lexical and structural modifications. JPlag performed well and thus was used with larger test bed and more variations in programs submitted by student. However, it cannot be concluded clearly that whether or not plagiarism tools can be used in online courses and educational institutes for checking similarity between student and teacher coding solutions and to prevent the students from submitting malicious code submission, as the results of the test were not constant and showed variations. The less number of teacher solution variations and JPlag parse error (reason unknown) are some of the reasons that a clear conclusion can't be stated. Also experiments with other existing plagiarism tools for source code plagiarism detection have not been conducted. In future, authors aim to continue to locate the reason behind the variations in results and solutions to reduce the same.

REFERENCES

[1] S. L. Richter and M. Krishnamurthi, "Preparing faculty for teaching a MOOC: Recommendations from research and experience," *International Journal of Information & Educational Technology*, vol. 4, no. 5, October 2014

[2] A. P. Rovai, "Online and traditional assessments: What is the difference?" *Internet and Higher Education*, vol. 3, no. 3, pp. 141-151, 2000.

[3] C. Georgina and M. Joy, "An approach to source-code plagiarism detection and investigation using latent semantic analysis," *IEEE Transactions on Computers*, vol. 61, no. 3, pp. 379-394, 2012.

[4] Đ. Zoran and D. Gašević, "A source code similarity system for plagiarism detection," *The Computer Journal*, vol. 56, no. 1, pp. 70-86, 2013.

[5] X. Chen, B. Francia, M. Li, B. Mckinnon, and A. Seker, "Shared information and program plagiarism detection," *IEEE Transactions on Information Theory*, vol. 50, no. 7, pp. 1545-1551, 2004.

[6] H. Jurriaan, P. Rademaker, and N. V. Vugt, "A comparison of plagiarism detection tools," Utrecht University, Utrecht, The Netherlands, vol. 28, 2010.

[7] JPlag. [Online]. Available: https://jplag.ipd.kit.edu/

[8] Aiken. [Online]. Available: http://theory.stanford.edu/~aiken/moss/

[9] Ferret. [Online]. Available: http://peterlane.info/software/ferret.html

[10] Dickgrune. [Online]. Available: http://dickgrune.com/Programs/similarity_tester/

[11] Heacademy. [Online]. Available: http://www.ics.heacademy.ac.uk/resources/assessment/plagiarism/demo_jplag.html

[12] A. Christian and S. M. Tahaghoghi, "Plagiarism detection across programming languages," in *Proc. the 29th Australasian Computer Science Conference*, vol. 48, Australian Computer Society, Inc., 2006.

Sonal Jain has over 14 years of academic experience. She is actively involved in teaching, training and research. Currently she is an associate professor at JK Lakshmipat University, Jaipur. Dr. Jain has several publications to her credit and has presented research papers at National and International conferences organized by institutes like IITB, IITK and, NTU Singapore. She has received the award for one of the research papers at National Conference. Dr. Jain has co-authored books including Personal Computer Software (Wiley India), Simplifying C (Dreamtech Publications) and six text books for Gujarat State Education Board. Dr. Jain is life member of Computer Society of India. She is the committee member with IACSIT Singapore. Her area of interests are in research, training and consultancy including technology for education, information retrieval and natural language processing.

Mayuri Singhal completed her schooling from Seedling Public School (Jaipur, Rajasthan, India) and did her graduation from Arya College of Engineering and Research Center (Kukas, Rajasthan, India) in Btech (information and technology). She completed her MTech (computer science) from JK Lakshmipat (Jaipur, Rajasthan, India). Her dissertation topic was "Possibilities of Usages of Existing Plagiarism tools for Student Assessment (Programming Languages)."

Axita Shah has 6 years of industry and academic experience, and she is pursuing Ph.D in computer science and working as a technology lead in Elegant Microweb pvt. Ltd. in business intelligence product. She has experience of teaching various core as well as advanced subjects such as data structure, data warehousing, data mining, and internet marketing to post graduate students. She has worked on predictive analytics techniques, data warehouse architecture in business intelligence product.