Journal of Theoretical and Applied Information Technology
15th February 2018. Vol.96. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN ITERATIVE GENETIC ALGORITHM BASED SOURCE CODE PLAGIARISM DETECTION APPROACH USING NCRR SIMILARITY MEASURE

[1]M. BHAVANI, [2]DR.K.THAMMI REDDY, [3]DR.P.SURESH VARMA

[1]Dept. of Information Technology, GITAM University, Visakhapatnam, Andhra Pradesh.
[2]Dept. of Computer Science Engineering, GITAM University, Visakhapatnam, Andhra Pradesh, India.
[3]Dept. of Computer Science, Adhikavi Nannayya University, Rajahmundry, Andhra Pradesh, India.

E-Mail: [1]bhavani.mm@gmail.com

ABSTRACT: With the advent of WWW and improvements in globally accessible software warehouses attained that source code is enthusiastically reachable to software designers. Even though, reusing of source code has its individual benefits, precaution is to be taken to guarantee that patented software [19] does not invade any authorizations. In this Context, Plagiarism Detection plays a very significant role. Although several existing detection approaches has been introduced, most of the approaches work on one to many similarity measures. However, this might not be very much helpful in case large number of datasets where many-to-many relationship exist. In this paper, an intelligent detection model is purposed by employing the iterative genetic algorithm with two different fitness evaluation functions. Prior to the detection model, the source code is preprocessed to remove noise and dimensionality reduction techniques are employed. The experimental results for the proposed approach are carried out using two different data sets. From the experimental results, it is found that the proposed model has good performance compared to the other existing approaches such as fuzzy clustering based Detection system and Incremental Genetic Algorithm.

Keywords: *Source Code, Plagiarism detection, Genetic Algorithm, Singular Vector Decomposition, Similarity Measure, Normalized Cumulative Reciprocal Rank, Euclidean Distance*

1. INTRODUCTION

The internet and globally available source code software's are increasing day by day in the present world. Such kind of code replication deprived of seeking the authorization or referring from the original authors for his contribution is known as plagiarism. Source Code plagiarism identification in programming language is a task that several higher education instructors accomplishes [3]. Whenever learners process the source-code validated through someone else, either intentionally or unintentionally, and fail to adequately admit the fact that the specified source-code is not its individual, the source code plagiarism ensues at this condition.

Learners, especially, at higher education are identified to be involved in certain kinds of academic deception [15, 16] where programming courses are not exclusion. Fair Evaluation is a crucial aspect for the achievement of the learning system, specifically at advanced Educational learning system. If Fair Evaluation is to be done, the novelty of the approach needs to be defined. They replicate the code partitions from entire resources accessible (internet, records, their class associates and their elders), in that way that the data samples where the plagiarism need to be identified rises.

Usually, Plagiarism detection issues are of two kinds: one-to-many where the individual desires to examine a unique document in a group and many-to-many the user wishes identify plagiarized pairs within a collection. In the prior situation, it is essential to limit whether a specific document is a plagiarized collection of a document in a group. In the subsequent situation it is essential to define the records that are plagiarized collections of one another. Approaches that perform good for one-to-many conditions does not essentially measure healthy to many-to-many condition. This paper mainly deals with the many-to-many condition. The proposed detection system generates a group of programs paired in two, categorized by similarity. Programs that are extremely identical to the replicas are in the topmost.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Numerous Plagiarism detection approaches are developed in the literature especially for one to many cases [3][10][11].An intelligent approach is proposed for many to many cases. In addition to this, there are also similarity detection techniques that is effortlessly affected by altering the identifier or program testimonial order, it fails to provide adequate provision for plagiarism detection. Thus, the proposed approach employed an Evolutionary Iterative Genetic Algorithm to obtain the similar pairs of detected codes by iteratively selecting the pair of documents from the pre-processed and diminished matrix .The normalized Euclidean distance is evaluated between the pair of source code and the cumulative reciprocal rank between the retrieved document pairs are also evaluated.

A brief introduction to source code detection and its importance along with the motivation for the suggested methodology is given in this section. The section 2 briefly discusses the existing methodologies in source code detection techniques. The proposed approach is briefly illuminated in the section 3. The experimental outcomes and its analysis for the proposed approach is given in section 4 followed by conclusion and references given in section 5 and section 6 correspondingly.

2. LITERATURE SURVEY

This section summarizes the diverse source-code plagiarism recognition devices which is present within the survey along with emphasis on the most of the current methodologies. In [13], a token illustration technique is suggested for programs inscribed in the Java language, and subsequently employs the Running Karp-Rabin Greedy String Tiling (RKRGST) approach to identify code resemblance. These approaches are matched with the other plagiarism recognition devices, such as Copy Paste Detector (CPD), Sherlock, CCFinder and Plaggie and observed that these techniques outstripped the other techniques.

A source-code plagiarism recognition approach was presented in [14] which exploits the greedy string tiling techniques. Summarizing, this technique primarily chooses a seed source code document from the data sample and recognizes the top K identical records by means of source code metric procedures (like McCabe's Cyclomatic Complication, counts of logical, physical, comment, and blank lines and counts of attributes and procedure).In the subsequent stage, the identified archives are matched with the source record by means of Greedy String Tiling approach. The archives, whose text resembles with the source file with identity more than the edge value, are recognized as identical.

An algorithm depending on kernelized fuzzy C-means (KFCM) is presented in [8] that employ the research of source code mining, to resolve the issue having huge number of quantities, numerous features and maximum of them are discrete software engineering. Using this approach, the efficacy of extraction is enhanced and pursue quicker and further efficient clustering technique. This likewise resolved the issue that the KFCM methodology could not group textual information openly. Subsequently, it could over the deficiency of merely being capable to acquire the minimal values through incorporating KFCM and genetic algorithm. Lastly, the experimentation demonstrations that the enhanced KFCM approach has a better clustering performance and higher competence on data mining.

A methodology that leveraged the lexical data and fuzzy clustering is presented in [9] to minimize the numerous design configuration examples that prevailing methodologies depends on structural data (i.e., directing the dependences amongst software features) inaccurately recover in source code. To evaluate the efficiency of the procedures, the outcome of a case study performed on four open source software systems executed in java are proposed. The data analysis specifies that the usage of lexical data and fuzzy clustering enhances the exactness of the outcomes attained through previous design pattern retrieval methods depending on structural knowledge, however conserving the design pattern examples appropriately acknowledged.

An incorporated methodology depending on Latent Semantic Indexing (LSI) and Stylometry approach is given in [10] for inherent plagiarism recognition. LSI is employed for the term document matrix of data samples, however, stylometry is employed for inherent estimation of individual inscription style. This accompanied a sequence of experimentations to examine the competency of dimensionality minimization constraint as the fundamental for LSI approach so as to achieve intuitions into its effects employing some tiny repositories. The comparative evaluation for the proposed approach was carried out through the LSI and Stylometry disjointedly, and subsequently performing them together. The result exhibited that the efficiency of the suggested

Journal of Theoretical and Applied Information Technology
15th February 2018. Vol.96. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

technique was enhanced with an incorporated methodology comprising of LSI and stylometry was exploited.

An evolutionary neural network approach was presented in [11] to introduce an intrinsic plagiarism recognition classifier that is proficient of developing the weights and configuration of a neural network. The neural network is empirically experimented on archives or records and is exhibited to function better. In [12], genetic algorithm is used to measure the identity amongst two codes by resolving an error rectifying sub graph isomorphism issues on dependency graphs. He proposed a novel price function for this issue that replicates the features of the source codes. An

The proposed approach focuses on many to many code detection approaches where the user attempts to detect the plagiarism within a group of documents where no restrictions is given on the type and number of documents. The Architecture for the proposed intelligent code detection methodology is given in Figure 1. This approach is

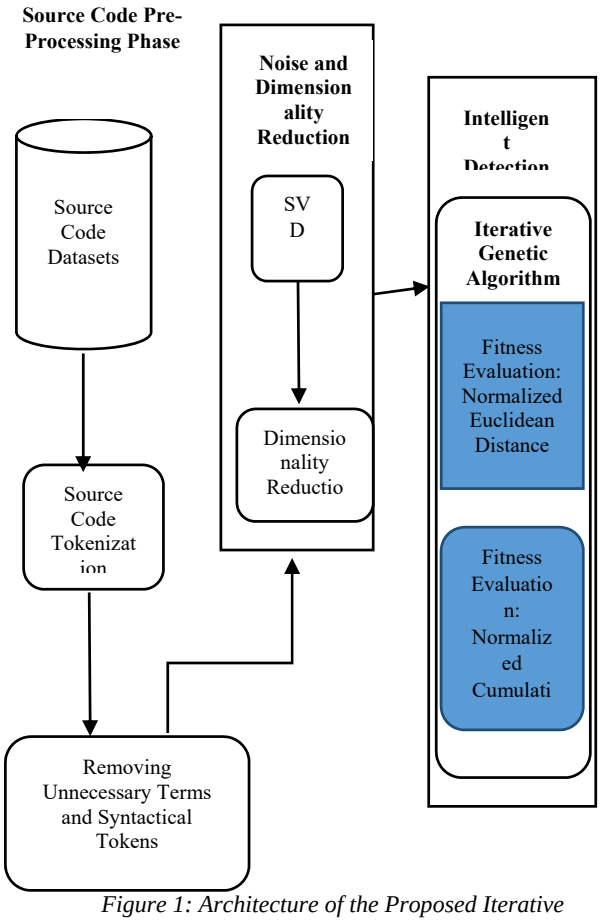

Figure 1: Architecture of the Proposed Iterative Genetic Algorithm Aided Detection System

incremental GA is employed to resolve the issue. The dimension of the graph need to be explored which is slowly rising in course of evolutionary approach. Novel operations are introduced for the methodology, and the entire system is verified on certain practical data. Experimental outcomes presented that the technique productively functions on code plagiarism identification and malicious recognition. The resemblance evaluated through the system has retrospected the similarity amongst the codes appropriately.

3. PROPOSED SYSTEM ON INTELLIGENT GENETIC ALGORITHM BASED SOURCE CODE DETECTION APPROACH

mainly divided into three phases for easy working of the methodology distinctively. They are:

- Source Code Pre-processing Phase
- Noise and Dimensionality Reduction Phase
- Intelligent Detection Phas

3.1 Source Code Pre-Processing

This phase is particularly employed to pre-process the source code document as to improve the retrieval of semantic information for code recognition where the irrelevant and unwanted information such as meaningless terms and characters, symbols or words etc. are removed. This phase is necessary to minimize the dimension of the information to further effectively seize the semantic depiction of every source-code file. The goal of this module is to accumulate the large number of source code into a processed format to detect the plagiarized source code relevantly.

Pre-Processing the source code can be of two forms such as pre-processing constraints that explicit to source code and parameters that are not specific to source code . Pre-processing constraints that specific to source code involves:

- Eliminating commentaries
- Merging or separating terms comprising of compound words
- Eliminating source code identifiers containing complex parameters that joined two words together found within terms and treated it as single term such as ‘student name’ to ‘studentname’.
- Plotting alternative words to a single form like function being plotted to procedure
- Reorganization the procedure according to the order of its function calling

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- Eliminating entire tokens that does not have the lexicon of the target language such as eliminating entire words which are not language reserved words

Certain conceivable pre-processing constraints not specific to source-code achieves involves:

- Eliminating words present in single document or entire document as these words preserves no additional knowledge regarding the association among the documents.
- Eliminating words merely comprising of numerical symbols,
- Eliminating syntactical tokens like semi-colons, colons, comma etc.
- Eliminating words comprising of a one alphabets, and
- Translating upper case alphabets to lower case.

After the pre-processing is performed, the Source Code pre-processing phase forms the Vector Space Model (VSM) that represents the source code data samples. In the VSM, a term document matrix is represented as $B_{m\times n} = [b_{ij}]$ where every row i have the rate of processed terms such as terms obtained in source-code document next to pre-processing, and every column j signifies the source-code document. Therefore, every element b_{ij} of B comprises of the rate at which the vocabulary term i occurs in a source-code file j. From the, term document matrix the normalized term frequency is obtained by applying probability inverse global weighting method (IDFP) [4] to modify the rate of terms relating to the whole group of source-code archives.[18] Similarly, document length normalization is performed to adjust the frequencies depending on the dimension of every document file. The two estimations are given below:

$$g_i = \log\left(\frac{N-n_i}{n_i}\right) \quad (1)$$

$$l_i = \frac{1}{\sqrt{(\sum_i (g_i.a_{ij}))^2}} \quad (2)$$

Here g_i is the probability inverse global weighting method, N is number of source code files in the group, n_i is the number of source code files where the word$i$occurs. l_i represents document length normalization. After the evaluation, every entry of the matrix A is updated as:

$$b_{ij} = b_{ij} \times g_i \times l_i \quad (3)$$

3.2 Noise And Dimensionality Reduction

The aim of noise and dimensionality reduction module is to minimize the size of the database and remove the noisy and irrelevant elements from the data sample. Once, the source code pre-processing module is accomplished, the obtained matrix A is a sparse matrix which need to be further evaluated to minimize its dimension and also remove the noisy term from the matrix. This task is efficiently accomplished by using the Singular Vector Decomposition (SVD) approach which is a useful tool for linear algebra. Using the SVD approach, the inherent higher order arrangement can be obtained in associations of terms with the source code files through specifying the SVD of huge sparse term in source code file matrices. SVD [5] is employed to evaluate the arrangement by means of using source code files. Retrieval is further accomplished by means of data sample of singular values and arrays accessible from the trimmed SVD. Performance information exhibited that these numerically obtained values are added strong pointers of importance compared to single individual words.

Given a matrix B of size $m \times n$, where deprived of loss of generalization $m \times n$ and $rank(B) = r$, the singular value decomposition of B, symbolized by SVD(B), is specified as

$$B = U\Sigma V^T \quad (4)$$

Here$U^T U = V^T V = I_n$and $\Sigma = diag(\sigma_1, \sigma_2, \ldots . \sigma_n); \sigma_i > 0$ for $1 \le i \le r$; $\sigma_j = 0$ for $j \ge r+1$. The initial r columns of orthogonal matrices U and Voutline the orthonormal eigenvectors linked with r non-zero eigen values of $BB^T$and $B^T B$, correspondingly. The columns of U and V are stated as the left and right singular vectors, respectively, and the singular values of B are given as the diagonal features of Σ that are the non-negative square roots of n eigen values of BB^T [5].These matrices represent an interruption to the original associations to a linearly independent vectors or feature values. SVD of matrix B specified in equation (4), with $r = rank(B) \le p = \min(m, n)$ defines

$$B_k = \sum_{i=0}^{k} u_i.\sigma_i.v_i^T \quad (5)$$

It is important to note that, the obtained B_k matrix do not rebuild the original term document matrix A precisely. The trimmed SVD, in one way, seizures maximum significant primary

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

structure in the association of words and files, however simultaneously eliminates the noise or inconsistency in term usage that plagues term dependent retrieval approach. Instinctively, as magnitude, k, is much lesser compared to the number of distinctive terms, m and negligible variances in vocabulary would be unnoticed. Terms that are present in alike files, would be nearer to one another in the k-dimensional feature domain though they never occur in the similar file. This reflects that certain files that do not share any terms with the other source code files might not be nearer to k-space. This resulted illustration that seizes term-term relations are employed for identification.

3.3 Intelligent Detection

The proposed algorithm is to detect the pair of source code files that are accurately similar to one another. Once the Noise and Dimensionality Reduction phase is accomplished, the obtained $B_{n\times k}$ which is minimized in terms of number of dimensions or features is employed to identify the resemblance within the pair of documents. For this, purpose, the proposed methodology used the intelligent evolutionary genetic algorithm to accomplish the task. The work flow process for the proposed approach is given in Figure 2. Evolutionary algorithms are motivated with Darwinian Theory, and are recognized as human competitive for various optimization issues [1]. The main benefit of evolutionary approach is that merely minimum degree of human skill sets is needed to achieve beneficial outcomes [2].The Genetic Algorithm (GA) is one of the exploratory approach that imitate the procedure of natural evolution and is therefore consistently employed to produce advantageous results to optimization and explore the similarity issues pertaining to the fitness evaluation employed in the algorithm. This technique is especially exploited in evaluating the precise or estimated outcomes to the search optimization issues.

In GA, the candidates are known as individuals or chromosomes. The initialization of populace is produced arbitrarily; selection and dissimilarity function are implemented in a loop till certain termination strategy is achieved.Every iteration of the loop is known as generation. The selection operation is envisioned to enhance the average quality of the population through providing chromosomes of highest quality with the highest possibility to be replicated into further iteration. The eminence of a chromosome is measured using the fitness evaluation. A fitness evaluation proposes the optimality of an outcome in the genetic algorithm such that a specific individual might be rated in contradiction to entire other chromosomes. For every iteration, genetic operations are employed and hence the population progresses.

Steps in Genetic Algorithm based Intelligent Detection Phase:

i. Population Initialization: The initialization of population for the performance of the proposed Detection is selected from the reduced matrix $B_{n\times k}$. It is observed that the matrix has n number of source code files with k terms within each source code file i.e. from $1, 2, \ldots . n$. The source code files with k terms are considered as individuals or chromosomes .In this approach the source code files are selected randomly from 1 to n of length k , the row elements of the matrix is a chromosome which is represented as the complete solution.

$$X = [b_{1k}, b_{2k}, \ldots \ldots b_{nk}]$$

ii. Recombination: This is similar to the organic systems, candidates are merged to generate their offspring in every programmatic loop known as iteration or generation. Generally, a pair of individuals or more are recombined to obtain the candidate solution. In this approach, a pair of source code documents are recombined as to find the similarities between the two. The picking of the pairs of source code documents are done randomly and a form of permutation and combination technique is applied to select the individual source code files.

iii. Fitness Evaluation: A value for suitability is allotted to every individual pertaining on how closely and truly is to resolve the issue hence reaching the outcome of the preferred problem. The fitness evaluation is a metric of the objective to be attained having it to be a maximal or a minimal value [6]. Fitness evaluation is augmented by means of genetic operations and computes every outcome to choose whether it would subsidize to the subsequent generation of results [7]. In this proposed approach two forms of fitness evaluation are made. They are:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Normalized Euclidean Distance between the pair of selected individual's source code documents as to obtain the value of similarity between them. The Normalized Euclidean Distance is given as:

$$ED(x,y) = \frac{1}{2}\left(\sum_{i=1}^{k}\sqrt{\frac{((x_i-\mu(x))-(y_i-\mu(y)))^2}{\sigma(x)-\sigma(y)}}\right) \quad (6)$$

The Similarity Measure that is employed for the proposed approach is Normalized Cumulative Reciprocal Rank which is evaluated as:

$$NCRR = \sum_{i=1}^{D}(plag(D_i) \times \frac{1}{i}) \div \sum_{i=1}^{|R|}\frac{1}{i} \quad (7)$$

Where D is the group of obtained document pairs, R is the set of acknowledged plagiarized document pairs and $plag(d)$ returns 1 for a plagiarized document pair and 0 for a non-plagiarized pair. The fitness values range from 0.0 to 1.0. NCRR similarity measures shows the proportion of the retrieved documents with respect to the relevant documents.

iv. Genetic Operations: Three different genetic operations are performed in the proposed approach like selection, crossover and mutation operations. These are the most influencing operator in the optimized evolutionary algorithms. Each operation has its own importance in the optimization of the algorithm to obtain the relevant outcomes.

- Selection: This approach executes the selection operation where most-suitable individuals of the populace persist, and the least-suitable individuals are eradicated. The procedure is the phase that monitors the methodology in the direction of even-better results. The most-fit condition for this approach is defined through the Euclidean Distance. If the Euclidean Distance is less than the threshold value 0.6, then those pairs of source files are considered for the computation of next generation.

- Crossover: This operation is achieved by attempting to merge individuals i.e. decision values of previous results so as to generate a novel result, having certain characteristics of every "parent". The crossover operation that is used is the point crossover with the probability of 0.5.

- Mutation: This operation occasionally attains arbitrary modifications in one or more individuals of the present populace, generating a novel outcome that might be superior to the preceding ones. This is achieved by altering the information of one parents with the information of other parent with the probability of 0.1.

v. Termination Criteria: This is also one of the important parameter in the evolution of solutions. The criteria for the proposed approach is the number of generation. The fitness function that is normalized Euclidean distance used in initial genetic approach terminates the process till number of iteration reaches 20. For the subsequent genetic algorithm the termination criteria are the condition where NCRR attains a near value to 1.0 with same number of iterations.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

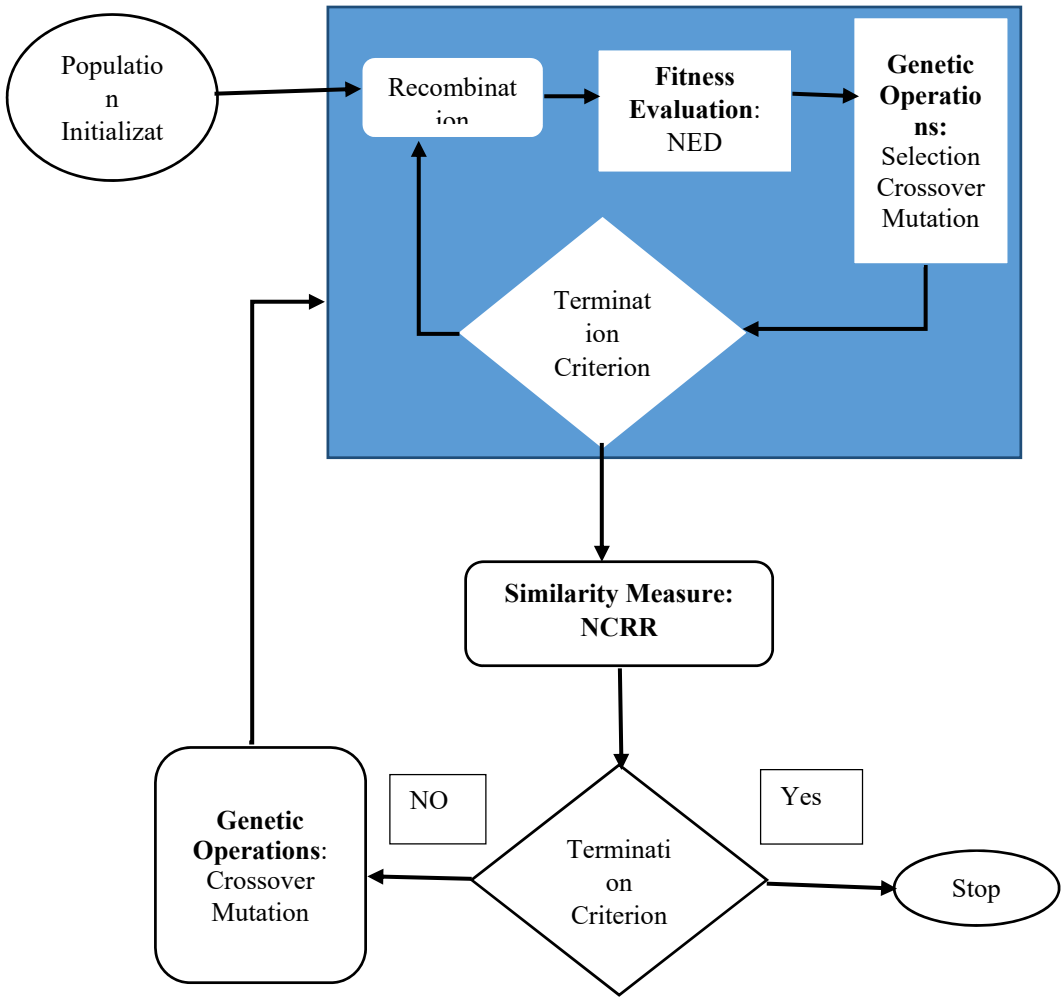

Figure 2: Work Flow Process of the Intelligent Detection Phase

5. EXPERIMENTAL RESULTS AND ITS ANALYSIS

The Experimental Results for the proposed Iterative Genetic Algorithm based Source Code Detection System is carried out using two different kinds of datasets. One is a set of simple source codes in Java programming Language with 30 in number and the other is the set of simple source codes in C Programming Language with 40 in number. The source code files with a maximum of 50 lines are considered for experimentation. The proposed approach is compared with the existing detection systems such as detection system using incremental genetic algorithm by means of sub graphs [12] and detection system using Fuzzy based Approach [17].

Performance Evaluation Measures:

Recall and Precision are two benchmarked and utmost recurrently employed metric in information retrieval system to estimate. These measures are exploited to estimate the efficiency of plagiarism detection. For the reason of estimation, the following terms are given:

i. Suspicious file pairs: Every suspicious pair, sp, comprises of documents that are being judged through human graders as suspicious. A class of suspicious pairs is given as $SP = \{sp_1, sp_2, \ldots . sp_x\}$ where the total amount of known suspicious pairs in a class (i.e., data sample) is $x = |SP|$.

ii. Innocent file pairs: These are the pairs that does not share any type suspicious identity however been identified as suspicious by proposed and existing detection systems.

iii. Detected file pairs: These pairs are obtained by means of the proposed Detection approaches. A class of identified pairs is referred as $DF = SD \cup NS = \{sd_1, sd_2, \ldots\ sd_x\} \cup \{ns_1, ns_2, \ldots\ ns_y\}$ where $SD \subseteq S$. The total number of detected file pairs is $|DF|$. The total amount of suspicious pairs identified is given by$|SD|$, and the total amount of innocent file pairs identified is denoted by $|NS|$.

Recall is given as R, where $R \in [0, 1]$, is the percentage of suspicious pairs that are recognized depending on the limit value, $\emptyset$. Recall is 1.00 whenever entire suspicious pairs are recognized.

$$Recall = \frac{|SD|}{|SP|} = \frac{number\_of\_suspicious\_file\_pair\_detected}{total\_suspicious\_file\_pairs}$$

Precision is given as P, where $P \in [0, 1]$, is the percentage of suspicious pairs that are recognized in the group of document pairs identified. Precision is 1.00 whenever each document pair identified is suspicious.

$$Precision = \frac{|SD|}{|DF|} = \frac{number\_of\_suspicious\_file\_pair\_detected}{total\_file\_pairs\_detected}$$

The complete performance of every device will be estimated through merging the precision and recall metrics. As a unique measure for estimating the efficiency of device for plagiarism detection, the weighted total of precision and recall would be evaluated as

$$Fscore_\beta = \frac{(\beta^2 + 0.1)(Precision \times Recall)}{\beta^2 \times (Precision + Recall)}, \in [0, 1]$$

Theβ coefficient obtains a way to bias $Fscore$ in the direction of Precision or Recall. Specifically, the value β=0.5 biases in the direction of precision, value β = 1.0evaluates precision and recall similarly and valueβ = 2.0 biases in the direction of recall. In the experimentations, entire three conditions are verified to define the comparative efficiency of several algorithms while highlighting recall or precision, and both. Henceforth, to penalize false negatives further powerfully compared to false positives through picking a value β> 1, therefore provides higher weightage to Recall.

The Analysis of the proposed approach is also performed by means of the Fitness Functions or the similarity measures that employed in the proposed approach. The average Normalized Euclidean Distance (NED) and Normalized Cumulative Reciprocal Rank (NCRR) measures is used to analyze the proposed approach specifically the Iterative Genetic Algorithm with Pre-processing and Dimensionality Reduction (GA with Pre-processing) against Simple Iterative Genetic Algorithm without any kind of Pre-processing and Dimensionality Reduction (GA without Pre-processing). The comparison is accomplished against the number of generation or iterations employed in the proposed Iterative Genetic Algorithm.

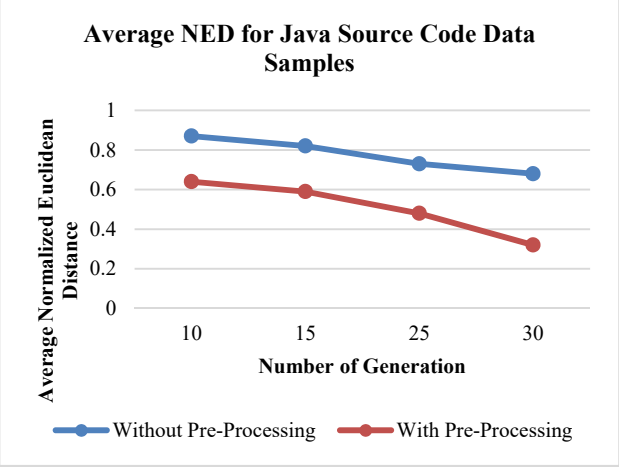

Figure 3: Average NED for Java Source Code Data Samples

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

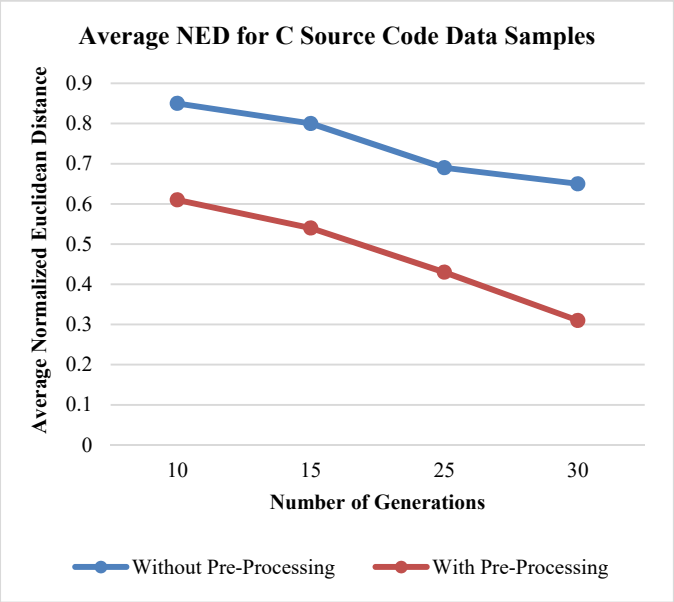

Figure 4: Average NED for C Source Code Data Samples

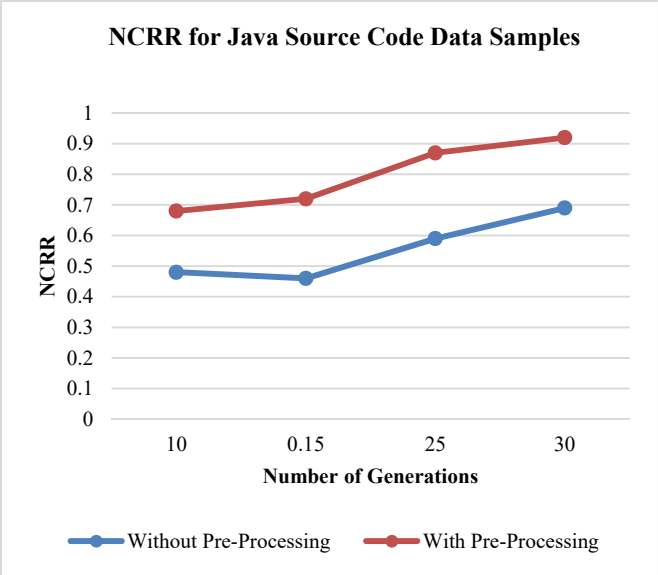

Figure 5: NCRR for Java Source Code Data Samples

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

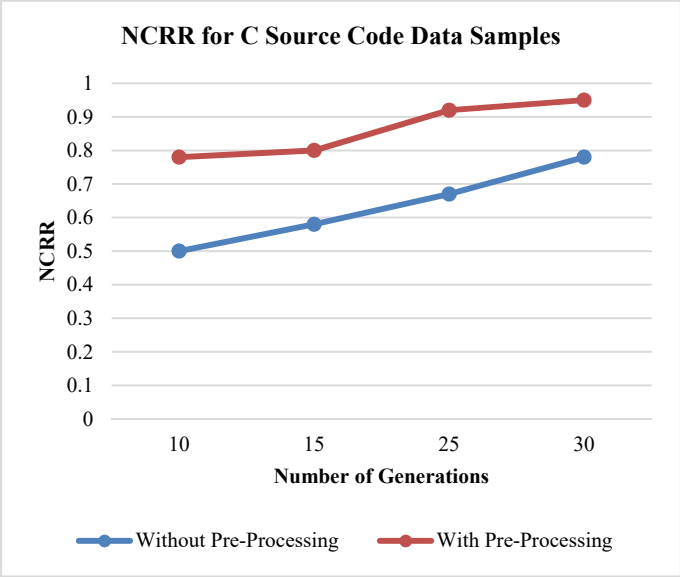

Figure 6: NCRR for C Source Code Data Samples

The precision, recall and $Fscore$ with different β are evaluated for the proposed Iterative Genetic Algorithm based Source Code Detection System against the existing approaches given in [12] and [17] for two different sets of data samples separately. Table 1 represents the Performance Measures of proposed IGA on Java Source Code Data Samples and C Programming Source Code Data Samples. Table 2 and Fig 7 represent the precision of the proposed approach that are matched with the existing detection system using the java programming source code data samples.

From table 1, table 2 and Fig 7, it is clearly observed that the performance measures of the suggested method are higher when compared to the other two techniques. It can also be inferred that the computational complexity of the proposed approach is also less compared to the other two approaches.

Table 1: Performance Measures of proposed IGA on Java Source Code Data Samples and C Programming Source Code Data Samples

Performance Measures	Java Source Code Data Samples	C Programming Source Code Data Samples
Precision	0.99	0.98
Recall	0.79	0.8
$Fscore_{0.5}$	0.615	0.616
$Fscore_{1.0}$	0.483	0.484
$Fscore_{2.0}$	0.4503	0.4514

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: Precision of Proposed approach with Existing methods.

Precision	Fuzzy Clustering based Detection System	Incremental Genetic Algorithm based Detection System	Proposed Iterative Genetic Algorithm based Approach
Java Source Code Data Samples	0.95	0.97	0.99
C Programming Source Code Data Samples	0.95	0.96	0.98

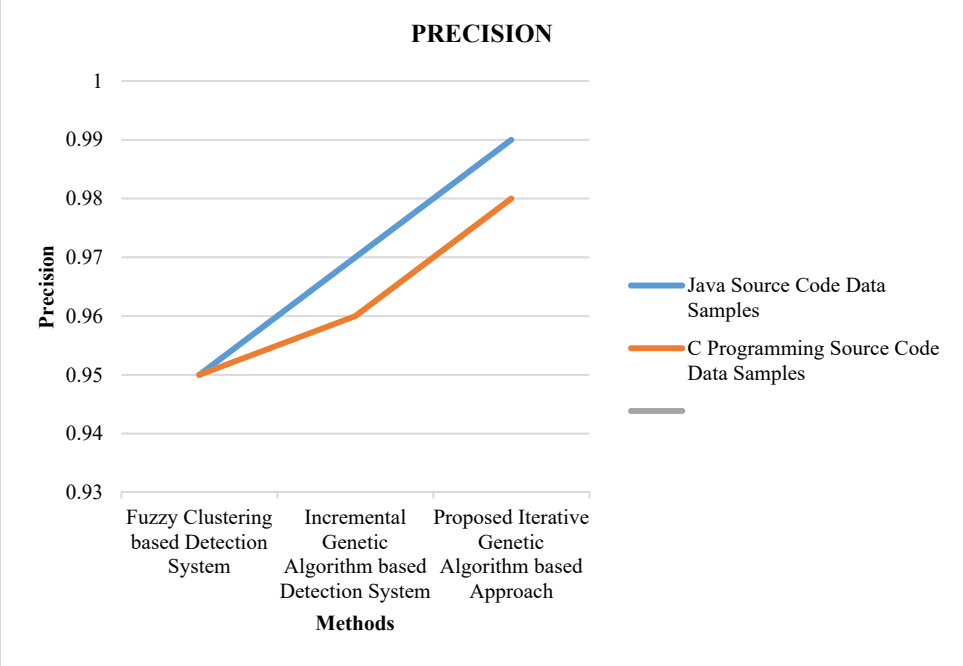

Figure 7 Precision of proposed approach with existing methods

6. CONCLUSIONS

The proposed intelligent iterative genetic algorithm employed two different fitness evaluation functions that compared the similarities between the pair of source code documents. The Source Code Pre-Processing and Noise and Dimensionality Reduction Module have further minimized the computational complexity of the proposed approach. The experimental results is been performed using two java and C programming data samples distinctively. The experimental outcomes showed that the suggested approach has good performance in term of precision and recall when compared with the other two existing approaches.

REFERENCES

[1] Koza, J. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, Norwell, MA.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[2] Koza, J. (1992). Genetic Programming: On the Programming of Computers by Means of NaturalSelection. MIT Press, Cambridge, MA.

[3] Arwin, C. and Tahaghoghi, S. M. M. (2006). Plagiarism Detection across Programming Languages. In Proceedings of the 29th Australasian Computer Science Conference, volume 48, pages 277–286,Hobart, Australia. Australian Computer Society, Inc.

[4] W. B. Croft, and D. J. Harper. Using probabilistic models of documentretrieval without relevance information. J. Documentation, 35(4): pp.285-295, 1979

[5] G. Golub and C. V. Loan, Matrix Computations, Johns-Hopkins, Baltimore, second ed., 1989.

[6] Praveen Pathak, Michael Gordon, Weiguo Fan, "Effective Information Retrieval Using Genetic Algorithms Based Matching Functions Adaptation," hicss, vol. 02, no. 2,pp. 2011, February 2000.

[7] Luger G F, "Artificial Intelligence- structure and strategies for complex problem solving", 4th edition, Pearson Education, 2002.

[8]X. Wang, "Kfcm algorithm based on the source code mining method study," in Intelligent Systems Design and Engineering Applications(ISDEA), 2014 Fifth International Conference on, June 2014, pp. 586–588.

[9] S. Romano, G. Scanniello, M. Risi, and C. Gravino, "Clustering and lexical information support for the recovery of design pattern in sourcecode," in Software Maintenance (ICSM), 2011 27th IEEE International Conference on, Sept 2011, pp. 500–503.

[10] Rahat Iqbal, Saad Amin, Anne James, Muna Alsallal,"Intrinsic Plagiarism Detection Using Latent Semantic Indexing and Stylometry", Sixth International Conference on Developments in eSystems Engineering (DeSE), 2013, IEEE.

[11] Dara Curran, "An Evolutionary Neural Network Approach to Intrinsic Plagiarism Detection", Irish Conference on Artificial Intelligence and Cognitive Science, Artificial Intelligence and Cognitive Science, pp 33-40, 2009, Springer.

[12] Jinhyun Kim, HyukGeun Choi, Hansang Yun, Byung-Ro Moon, "Measuring Source Code Similarity by Finding Similar Subgraph with an Incremental Genetic Algorithm", In Proceeding of the Genetic and Evolutionary Computation Conference, pp. 925-932, ACM, 2016.

[13] B. Muddu, A. Asadullah, and V. Bhat, "Cpdp: A robust technique for plagiarism detection in source code," 7th International Workshop on in Software Clones (IWSC), pp. 39–45, 2013.

[14] O. Ajmal, M. Missen, T. Hashmat, M. Moosa, and T. Ali, "Eplag: A two layer source code plagiarism detection system," in Digital Information Management (ICDIM), 2013 Eighth International Conference on, Sept2013, pp. 256–261.

[15] S. Burrows, S. M. M. Tahaghoghi and J. Zobel, "Efficient plagiarism detection for large code repositories", Software Practice and Experience, vol.37, pp. 151-175, 2006.

[16] Noh, Seo-Young, Sangwoo Kim, and S. K. Gaida. "An XML plagiarism detection model for procedural programming languages." In Proceedings ofthe 2nd International Conference on Computer Science and its Applications.2004.

[17] Giovanni Acampora, Georgina Cosma, "A Fuzzy-based Approach to Programming LanguageIndependent Source-Code Plagiarism Detection", IEEE International Conference on Fuzzy Systems, 2015.

[18] K.Thammi Reddy, M.Shashi and L.Pratap Reddy," Hybrid Clustering Approach for Concept Generation ",IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.4, April 2007.

[19]www.bitlaw.com/software-patent/patentable.html