GA-ANFIS Approach to Language Independent Source Code Plagiarism Detection
Ankur Nagaich
M.Tech Scholar
Digital Communication
SRCEM, Banmore

Anuj Bhargava
Professor
Digital Communication
SRCEM, Banmore

Prashant Badal
Professor
Digital Communication
SRCEM, Banmore

Abstract- Plagiarism detection in source code of different programming language is an area of concerns for identification similar or identical fragments of source code in source-code files. Mainly, the system computes and displays the similarity value between each pair of programs in a dataset. In this paper, a brief discussion of source code Plagiarism is presented as well as comparative study of the application of various techniques in textual similarity processing on source code. This paper is designed an approach based on machine learning tools to cluster source-code fragments for detecting similar or identical tokens in input dataset and hence finding plagiarized files. Fuzzy clustering technique is an appropriate technique for detection of plagiarism of source-code due to their potential to capture the qualitative and semantic elements of similarity. In this paper a genetic algorithm optimized neuro-fuzzy based approach is proposed for clustering source-code for detecting clusters which could contain similar and hence plagiarized files. In particular, this paper compares the performance of proposed model models with some existing technique's implementation of ANFIS, SVM, RF and LDA. As a result it is seen that proposed approach appears to overcome the several problems currently encountered by plagiarism detection algorithms and many of the existing algorithms and tools. The result shows that PCA based GA-ANFIS having higher accuracy than several existing techniques.

Keywords:- Source-code; Tokenization; Plagiarism detection; PCA; ANFIS; Genetic Algorithm; Accuracy;

I. INTRODUCTION

Plagiarism of source-code is a growing problem due to the growth of source-code repositories, and digital documents found on the Internet [1]. Plagiarism is considered as one of the most severe problems in academia, due to the availability of source-code found on-line and also due to sharing of source-code solutions among students. Essentially, plagiarism in computer programs occurs when a person reuses source code authored by someone else and fails to acknowledge to author.

Consequently, the issues in plagiarism detection systems is to recover files which have noteworthy code, and not to overwhelmingly detect files which contain several small and insignificant fragments occurring in several files (this is considered as noise in the data), as this will add the extra burden of time on the academic, searching through a large number of files to detect the ones which contain proof of plagiarism. Similar source-code fragments which are common across many files, essentially add noise to the problem of plagiarism detection [3]. Plagiarism is commonly classified into two categories:

A. Plagiarism in documents : Most of the work in document plagiarism has been done for academic purpose. Detecting plagiarism is important to judge and mark students’ work especially for postgraduates who are strictly prohibited from cheating, rewording, rephrasing, or restating without referencing. In this respect, a number of plagiarism detection tools have been developed. Such systems can be classified into two main categories, web-enabled systems and stand-alone systems.

B. Plagiarism in code: Source code plagiarism or it called programming plagiarisms generally done by students in universities and colleges is outlined act or trial to use,
reuse, convert and modify or copy the complete or the region of the source code written by some other person and used in your programming whereas not citation to the owners. source code detection primarily wants human intervention if they use Manual or automatic source code plagiarism detection to form a choice or to figure out whether or not or not the similarity because of the plagiarism or not. Manual detection of source code throughout a vast variety of student homework’s or project it's so difficult and wishes highly effort and stronger memory, it seems that not possible for an enormous variety of sources.

To understand the method used to create the tools and the offered functionality, it is necessary to understand the basic concepts involved. Plagiarism detection system or algorithms utilized in source-code similarity detection are often classified as based on either [4]:

**Strings:** Search for precise textual matches of segments, as an example five-word runs. Fast, however are often confused by renaming identifiers.

**Tokens:** Like strings, however using a lexer to convert the program into tokens 1st. This discards whitespace, comments, and identifier names, making the system more robust to easy text replacements. Most academic plagiarism detection systems work on this level, using completely different algorithms to measure the similarity between token sequences.

**Parse Trees:** Build and compare parse trees. This permits higher-level similarities to be detected. As an example, tree comparison will normalize conditional statements, and find equivalent constructs as just like each other.

**Program Dependency Graphs (PDGs):** A PDG captures the actual flow of control in a program, and permits a lot of higher-level equivalences to be placed, at a larger expense in quality and calculation time

**Metrics:** Metrics capture ‘scores’ of code segments according to certain criteria; as an example, “the variety of loops and conditionals”, or “the variety of different variables used”. Metrics are easy to calculate and may be compared quickly.

**Structural Information:** This is information from the structure of a programming language. An example would be the concept of comments, conditional controls, among others.

II. RELATED WORK

Parker and Hamblen [1], defined plagiarism in software as: “a program which has been produced from another program with a small number of routine transformations”. The most challenging aspect in code-plagiarism detection is the techniques that the implicated students tend to use to disguise the copied code in order to mislead the grader. The Running Karp-Rabin Greedy-String-Tiling algorithm (RKR-GST) is a well-known token matching algorithm developed initially within the YAP3 plagiarism detection tool.

Steve Engels et al [5] defined the metrics of source code and extracted feature based on neural network for detecting plagiarism. This algorithm works in two stages. In the first stage of the algorithm metrics from source code are extracted. Extracted metrics acts as a feature vector for further classification of source using neural network. Result of the feature-based neural network was illustrated or represented in terms of precision and recall. Precision for plagiarized source code was 0.6464 and recall for plagiarized source code was 0.4158.

Lange and Mancoridis [6] have proposed a source code plagiarism detection method, which uses source code metric histogram and genetic algorithm. In the very first step of algorithm, source code metrics are extracted from source code dataset. Further the normalized histograms were generated for each source code metric. These histograms were by k-nearest neighbor classifier. A genetic algorithm is used in order to identify the optimized set of source code metrics. The results shows that the accuracy of the proposed algorithm is about 55 percent.

Dian and Samadzadeh [7] have proposed a research work for determination of source code authors using metrics of source code. This research work is basically dependent on extraction a large number of source code metrics. In this work, author has extracted about 56 source code metrics that subjects to various statistical techniques in order to recognize the authors of each source code.

Upul Bandara and Gamini Wijayarathna [8] presented a latest plagiarism finding method and this method is based on machine learning techniques. In this research author have trained his system with the help of three algorithms that are Naive Bayes Classifier, K-Nearest Neighbor (KNN) and AdaBoost Meta-learning Algorithm. The system was designed and it is based on learning approach so it used the steps that are generating a set of tokens from the Source Code Files, Training Dataset, Implementing the System and Training the System. Java programming language has been used for developing the entire system. The benefit of this paper is that this model is capable to get 86.64 percent correctness by using the same dataset as used by Lange and Mancoridis.
Enrique Flores et al. [9] developed a tool named as DeSoCoRe and we can discover source code re-use with the help of DeSoCoRe. The tool is working at the various level of functions. The outputs generated by this tool are easy to recognize to the reviewer. The Framework of the DeSoCoRe consists of the modules that are Source code Splitter, Similarity Estimator and Pair Selector. Source code can be divided into functions with the help of Source code Splitter. After that divided functions will be compared with the help of similarity estimator. The similarity estimator firstly does the preprocessing of the function for removing of tabs, extra spaces and line breaks then after it will extract the feature by using character n gram extraction. Reviewers can be decide the sources are reused or not with the help of this tool. It cannot be do the comparison at fragment level, where a fragment is considered a part of a method or a set of method.

Zoran Djuric and Dragan Gasevic. [10] worked on Running-Karp-Rabin Greedy-String-Tiling (RKR-GST) based technique. In this research work preprocessing, tokenization, exclusion, similarity measurement and finishing similarity measurement are five stages which are recommended by the author. First two stages are used to eliminate the noise and are programming dependant. Stages 3 to 5 are programming independent. The approach is based on the two algorithms RKR-GST (Running-Karp-Rabin Greedy-String-Tiling) and Winningow algorithm. The benefit of this research work is that Source Code Similarity Detection System (SCSDS) has better performance than Jplag and drawback is that SCSDS is slower than existing system because in similarity detection process SCSDS will use different similarity detection algorithms.

Enrique Flores et al. [11] worked on Character N-Grams based technique include [10]. Character N-Grams Comparison Model is used for C++, JAVA & Python Language. The projected model is separated in three stages that are Pre-processing: In it they will removed tabs, spaces and line breaks, Features extraction: in it extracted character n-grams and weighted based on normalized and Comparison: in it we will used cosine similarity evaluation for comparison of the source code. The benefit of this research is to identify plagiarism between Source Code which is written in cross-language. The drawback is that best result is obtained only when comments are ignored.

Khurram Zeeshan et al. [12] introduced a technique named Greedy-String-Tiling technique. For source code plagiarism detection author has used Greedy String tiling. There are two phase in this technique i.e. Parsing and pre-Tokenization is done in first stage and Greedy String Tiling (GST) algorithm is applied in second stage. The benefit of this research is that in the given technique all features can be possessed like other tools but it also possesses some other features as similarity, inclusion, coverage and positive clone. The drawback of this research is that if anyone function names/variable names different then the proposed technique is suitable otherwise not suitable.

Huang, Liuliu et al. [13] introduced tokenization and Parse Tree based technique for code similarity a method have been proposed by the authors by joining structure metric with semantic computing methods. It is capable of recognizing not only primary cheating means in code copy, but also the senior ones, such as changing control structures with corresponding structures. In this research outcome shows that proposed method is better than MOSS and Structure only method. The benefit of this research or technique is that in the proposed method is better than MOSS and Structure only method. The drawback is that if threshold should be correct then the proposed method is suitable.

Arwin and Tahaghoghi [14], lists the most common disguises; which are changing comments, changing formatting, changing identifiers, changing the order of operands in expressions, changing data types, replacing expressions by equivalents, adding redundant statements, changing the order of time-independent statements, changing the structure of iteration statements, changing the structure of selection statements, replacing procedure calls by the procedure body, introducing no structured statements, combining original and copied program fragments and the translation of source code from one language to another.

Kustanto and Liem [15] proposed a tool for detecting source-code plagiarism among programs written in the LISP and Pascal programming languages. Their approach is a token-based approach which essentially comprises of two steps: firstly, it parses the source code and transforms it into tokens, and thereafter compares each pair of token strings obtained in the first step using the RRKRGST algorithm.

More recently, Muddu et al. [16] propose a token representation approach for programs written in the Java programming language, and then use the RRKRGST algorithm to detect code similarity. They compared their approach to other plagiarism detection tools, namely the Copy Paste Detector (CPD). One of the problems encountered is that files must parse to be included in the comparison for plagiarism, and this can cause similar files that were not parsed to be missed.
Ajmal et al. [17] proposed a source-code plagiarism detection system which also utilizes a greedy string tiling algorithm. However, Fuzzy logic approaches have been successfully applied to cluster source-code for the recovery of source-code design patterns, source code mining, to assess similarity within program. Samples, to derive rules in order to detect security defects in programs, to find traceability links between reports and source-code.

Giovanni Acampora and Georgina Cosma [18], proposes a novel Fuzzy-based approach to source-code plagiarism detection, based on Fuzzy C-Means and the Adaptive-Neuro Fuzzy Inference System (ANFIS).

Source code plagiarism can be experienced in institutions or even at commercial business points. It is considered as against the law if the others work is copyright or bad mannered practice otherwise in the society. Manually identification of source code plagiarism in educational institutions is not an easy task; the reason behind this is possibility of same copied source code but with different patterns. Some diplomatic changing are commonly in practice for making the code look different from the copied one but result in the same output. It actually decreased the efficiency of plagiarism detection tool.

III. PROPOSED METHODOLOGY

This section introduces innovative computational intelligence architecture, as shown in figure 1, for the purpose of analyzing source-code in the context of source-code plagiarism detection. A novel Fuzzy-based approach to source-code plagiarism detection, based on Fuzzy C-Means and the Adaptive-Neuro Fuzzy Inference System (ANFIS). The proposed framework consists of three phases i.e. Preprocessing, Post Processing Phase and Plagiarism Detection Phase.

A. Preprocessing Phase: The purpose of the Source-code Pre-processing phase is to pre-process the source-code files in such way that it removes unnecessary and meaningless terms and characters in order to reduce the size of the data to more efficiently capture the semantic representation of each source-code file. The first step towards Preprocessing phase is called tokenisation, the lexical analysis process of breaking a stream of text up into words, called tokens. Thereafter the following pre-processing parameters are also applied removing terms that occur in one file because such terms hold no information about relationships among terms across files, removing terms solely composed of numeric characters, removing syntactical tokens (e.g. semi-colons, colons), removal of comments, etc.

B. Post-Processing Phase: Once pre-processing is applied, the pre-processing Module creates the Feature Vector that represents the source code dataset. Feature Vector matrix represents in which each row i represents the source code file and j represents the occurrence of each token in the source code. Hence, each cell aij of Feature Vector matrix contains the occurrence of token at which a dictionary term j appears in a source-code file i.

Dimensionality Reduction Phase: The aim Dimensionality Reduction phase is to further reduce the database size and hence space complexity by removing noise and irrelevant data. This task is accomplished by using the Principal Component Analysis (PCA).
GA-ANFIS based classification phase: Let V(n*k) be the reduced source-code file-by-dimensions matrix containing only the selected features which resulted after applying PCA and Dimensionality reduction. This module takes as input matrix V(n*k) and applies a neuro-fuzzy learning algorithm. The learning algorithm works in two sequential steps.

**Step 1: Fuzzy C-Means (FCM) clustering**

It is applied to generate a collection of clusters where each cluster contains the source-code file characterized by a similar collection of identifiers (i.e. the terms found in source-code files after pre-processing). One of the processes to divide data elements into classes is data clustering. In data clustering, items in the same class are as similar as possible, and items in different classes are as dissimilar as possible. Depending on the nature of the data and the purpose for which clustering is being used, different measures of similarity may be used to place items into classes, where the similarity measure controls how the clusters are formed. Formally, let V(n*k) me be the n* k source-code file by dimension matrix containing source-code file vectors vi such that V = [v1; v2; v3; : : : ; vn], where each vi file vector contains k number of features selected after applying PCA and dimensionality reduction. Let c be the number of clusters and n be the total number of source-code files such that 2 <= c < n. Matrix V is input into the FCM algorithm which returns a list of cluster centres X = x1; : : : : ; xc and a membership matrix U, where each element holds the total membership of a source-code file vk belonging to cluster ci. FCM updates the cluster centres and the membership grades of each source-code file using the objective function.

**Step 2: GA-ANFIS Classification**

Fuzzy clustering is using in inference models based on adaptive neuro fuzzy systems. An ANFIS is a kind of artificial neural network that is based on Takagi Sugeno fuzzy inference system. Since it integrates both neural networks and fuzzy logic principles, it has potential to capture the benefits of the both in a single framework. And Optimized classification is obtained using Genetic algorithm based ANFIS (GA-ANFIS).

C. **Plagiarism Detection Phase:** In this phase a new source code is tested that it is plagiarized or not. During analysis, a new file is first of all uploaded and further preprocessing phase is carried out. During preprocessing features are extracted and dimension are reduced using PCA. Further GA-ANFIS rules are used to predict the degree of plagiarism of the source code.

A set of predicted clusters and known classes. Precision, Recall Fscore (or F-measure) and Accuracy can be calculated as below:

**Accuracy:** It measures how often classifier makes the correct prediction. It is the ratio of number of correct predictions to the total number of predictions (number of test data points).

\[
\text{Accuracy} = \frac{\text{Correct}}{\text{Total}}
\]

**Precision:** It measures and answers the question: out of the items which are predicted true, how many are actually true?

\[
\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}
\]

**Recall:** If answers the question: out of all the items which are true, how many are found to be true by classifier?

\[
\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}
\]

**F-measure:** F-measure combines both precision and recall. This score comes in between 0 and 1. 0 is worst and 1 is ideal.

\[
F1 = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}
\]

**IV. RESULT ANALYSIS**

In this paper a dataset is prepared from different sources codes and result is analyzed on different types of machine learning algorithms such as Support Vector Machine (SVM), Linear Discriminant Analysis (LDA), Adaptive Neuro Fuzzy Inference System (ANFIS), etc.

The result is analyzed in terms of recall, precision, accuracy and F_measure. After comparative analysis of proposed PCA based GA-ANFIS with some existing techniques it is seen that the proposed work is efficient.

Below table I shows the comparative result of different existing machine learning source code plagiarism detection algorithm. This result is analyzed on four parameters i.e. recall, precision, accuracy and F_measure and it is analyzed that proposed PCA based GA-ANFIS gives best result as compared to all other existing algorithm in terms of recall, accuracy and F_measure.

<table>
<thead>
<tr>
<th>Technique/Measures</th>
<th>Recall</th>
<th>Precision</th>
<th>Accuracy</th>
<th>F_measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA-GA-ANFIS</td>
<td>1</td>
<td>0.9091</td>
<td>0.9167</td>
<td>0.9524</td>
</tr>
<tr>
<td>SVD-GA-ANFIS</td>
<td>1</td>
<td>0.8333</td>
<td>0.8333</td>
<td>0.9091</td>
</tr>
<tr>
<td>SVM</td>
<td>0.7</td>
<td>1</td>
<td>0.75</td>
<td>0.8235</td>
</tr>
<tr>
<td>RF</td>
<td>0.6</td>
<td>1</td>
<td>0.6667</td>
<td>0.75</td>
</tr>
<tr>
<td>LDA</td>
<td>0.4</td>
<td>1</td>
<td>0.5</td>
<td>0.5714</td>
</tr>
</tbody>
</table>
The figure 2 shows the comparative analysis of recall of proposed algorithm i.e. PCA based GA-ANFIS with SVD based ANFIS, SVM, Random Forest and LDA.

![Figure 2: Comparative Analysis of Recall of Source Code Plagiarism Detection Techniques](image)

The figure 3 shows the comparative analysis of precision of proposed algorithm i.e. PCA based GA-ANFIS with SVD based ANFIS, SVM, Random Forest and LDA.

![Figure 3: Comparative Analysis of Precision of Source Code Plagiarism Detection Techniques](image)

The figure 4 shows the comparative analysis of accuracy of proposed algorithm i.e. PCA based GA-ANFIS with SVD based ANFIS, SVM, Random Forest and LDA.

![Figure 4 Comparative Analysis of Accuracy of Source Code Plagiarism Detection Techniques](image)

The figure 5 shows the comparative analysis of F_measure of proposed algorithm i.e. PCA based GA-ANFIS with SVD based ANFIS, SVM, Random Forest and LDA.

![Figure 5: Comparative Analysis of F_measure of Source Code Plagiarism Detection Techniques](image)

V. CONCLUSION

Plagiarism in source-code submissions is a serious problem that has motivated researchers to find effective automated detectors. This paper proposed a plagiarism detection engine for source-code files. Plagiarism detection in source code of different programming language is area of concerns for identification similar or identical fragments of source code in source-code files. Mainly, the system computes and displays the similarity value between each pair of programs in a dataset.

This paper designed an approach based on machine learning tools to cluster source-code fragments for detecting similar or identical tokens in input dataset and hence finding plagiarized files. Fuzzy clustering approaches are a suitable solution to detecting source-code plagiarism due to their capability to capture the qualitative and semantic elements of similarity.

Further a genetic algorithm optimized neuro-fuzzy based approach to clustering source-code for detecting clusters which could contain similar and hence plagiarized files. In particular, this paper compares the performance of proposed model models with some existing technique’s implementation of ANFIS, SVM, RF and LDA.
The proposed approach uses data mining techniques to represent the pre-processed source-code files into a Vector Space Model and removes noise from the data prior to applying the FCM clustering algorithm to cluster the data and then data reduction technique by using Principle Component Analysis (PCA) and finally GA-ANFIS to optimize the performance of the detection system. As a result it is seen that proposed approach appears to overcome the several problems currently encountered by plagiarism detection algorithms and many of the existing algorithms and tools. To detect source code plagiarism, the proposed improves the efficiency and accuracy of finding degree of plagiarism. In future work the proposed algorithm may be presented on more datasets. The further research work in the proposed area will provide a more generalized overview of the code clone detection in a program as well as the proposed algorithm would be extended to detect plagiarism in text written in different language such as hindi, kannad, malyalam, etc.

REFERENCES