Abstract- Information technology is taking a lead in making the processes of education to be automated. One of key activity in education system is teaching learning process. In teaching learning process, one of the important activities is evaluation of learner’s capability through either assignments or examination. In the present day the teachers are more towards the submission through online than offline. It is tedious process for an individual to evaluate each and every assignment submitted by a set of students to find the similarity if exists. We have proposed a method to find similarity among the submitted assignments. The submitted assignments are transformed into intermediate form by transforming it by removing white spaces and comments, converting keywords and identifiers into standard notation to produce language free structure called pseudo code. The run length is calculated on derived pseudo code and it is represented in the form of vector. In similar way the run length is calculated on all the programs submitted by set of students. Using cosine similarity, the proximity is calculated between the assignments submitted by students and source codes in repository and output is shown pictorially. The proposed method is successful in finding similarity among the submitted assignments.

Keywords- source code, plagiarism, tokens, run length.

I. INTRODUCTION

The education system revolves around three important activities i.e. Teaching: art of lecturing by teachers, Learning: acquiring knowledge, Evaluation: test the ability of learning. In general the teaching and learning process participants are teachers, students where the mode of instruction being in the class room. In this process the student acquires knowledge and evaluation to be done to judge the progress of student. In the current system the evaluation is done internally or externally or both. In most of the cases, the external evaluation is paper based and needs human effort to judge the student. Internal evaluation can be done by conducting class tests, quizzes and by giving assignments. In earlier days, the assignments were submitted by the student to the faculty physically i.e. offline submission and faculty evaluates the assignments. Lot of effort is needed to grade the students and to find similarity among the submitted ones. Now these tasks can be automated with the help of technology which will substantially increase benefits to the system being built. The assignments submitted by the students could be replicated from the content available on Internet or previous submissions of their classmates. The assignments submitted by the student community may contain theoretical, images and code components. Especially in the engineering field the submission may contain program specific assignments i.e. in C/C++/Java. The students may submit the assignments by putting their own efforts or copying from their class mates or from multiple sources. A challenge that exists now is to find similarity among the submitted assignments for evaluation without proper acknowledgement. Gathering information from multiple sources without proper acknowledgement is referred as plagiarism.

Joy and Luck [1] define plagiarism as “unacknowledged copying of documents or programs”. Plagiarism can be either text plagiarism or code plagiarism. Text Plagiarism has shown much advance and various detection techniques are already in use. The available software’s like Turnitin, display the source from where the student copied and percentage of similarity.

The various techniques used by available software’s are attribute counting, token based and structure based comparison and all these techniques have their own merits and demerits. The focus of this paper is to compute similarity among programs written in some programming language known as source code. Source code can be copied either from same programming language or from other programming language and identifying its source of copy is plagiarism. To differentiate a duplicate program from original one, the only means is to compare with all submitted programs. Manual comparison is efficient if the numbers of submission are less otherwise the task becomes tedious. So, there is need to automate detection mechanism for comparison. The rest of the paper is organized as follows. Section 2 explains the related work carried out, section 3 briefs the proposed methodology, section 4 explains the experimental values. Finally, section 5 gives the conclusion and future improvements.

II. RELATED WORK

Various techniques have been proposed to compute similarity among source codes such as attribute counting, token-based, syntax based techniques[ 6][7][8][9][10]. In the attribute counting technique a Halstead’s parameter for used for
similarity comparison. This method computes similarity by comparing the number of operands and operators in both the source codes. This method is efficient and simple to understand. The drawback in this technique is structural changes cannot be identified.

To identify the structural changes, techniques like token based comparison, tree based comparison etc are used. In token based technique, the similarity is detected in two phases i.e. tokenization and comparison phase. In tokenization phase, the given code is divided into tokens where the keywords are represented by standard notation and identifiers are represented by <id> and numerical value by <val>.

The statement X=X+3
Can be represented as
{id}=id1+<val>.

After the tokenization phase, the codes are represented in form of strings. In comparison phase, the strings are compared using various strings comparison algorithm like greedy tiling etc to compute similarity. The efficiency in this method is generalization is done to higher degree. In this technique the lower abstraction details are lost. The token-based method cannot compute similarity accurately if functions are included and doesn’t emphasis on flow of control in source code.

Structural changes are identified in structure – based comparison technique where similarity is computed in two phases i.e. preprocessing and comparison phase. In preprocessing phase the codes are preprocessed and are represented in intermediate forms like Abstract syntax tree (AST) , program dependency graph(PDG) etc. In comparison phase, the intermediate forms like AST or PDG are compared for computing similarity using parse tree algorithm, parameterized algorithm etc. One of the advantage in this technique is control flow in the program can be found.

III. PROPOSED SYSTEM

The proposed work aims at finding the similarity between two source codes. The similarity is computed by computing the run length. The cosine similarity is used to calculate the proximities between submitted codes and codes in data repository and result is shown pictorially.

In the proposed method, the similarity is computed between test source code and source codes present in the repository. The repository is a semi-static data base updated from time to time. First the Manto Carlo test [17] is done to select the target source codes from the repository. The preprocessing is applied on test program as well as target source code. After preprocessing the codes are converted into strings which are collection of tokens. The run lengths are computed for the strings and are represented by vectors. Similarity is computed between vectors using cosine similarity. The Fig 1 shows the proposed model for similarity computation.

![Proposed similarity computation model](image)

Fig 1. Proposed similarity computation model

A. Preprocessing the codes

In preprocessing stage, the removal of white spaces, comment lines from the repository containing source codes. In similar manner the process is also performed on the test source code. A source code is collection of statements. Each statement is a collection of tokens. In this approach we have constructed a table for representing keywords with intermediate form like int is replaced as KI ,float is represented as KF etc.. The comment lines are used to increase the readability of the program but it doesn’t involve in achieving the task. After removing white spaces and comments, the keywords are replaced with intermediate form used as in the table. The identifiers are represented by <idk> where value of k={0,1,2,3,......}. All the control statements are converted into intermediate form.

Fig 2. Shows an example of preprocessing applied for two source codes.

//program for sum of 3 numbers.
int a, b, c, x;
x=a+b+c;  \rightarrow  \text{KI id1 id2 id3 id4 id5=id1 + id2 + id}

Source code 1 preprocessed code (in form of strings)
int a, b, c, x1 ,x2;
x1=a+b;
x2=x1+c;  \rightarrow  \text{KI id1 id2 id3 id4 id5=id1 + id2 id5 = id4 + id3}

Source code 2 preprocessed code (in form of strings)
After pre processing the codes is converted as strings which consist tokens. The run length is computed for the generated string.
B. Generating run length
A run in a composition is a maximal string of identical parts\cite{11}. The run length is the numerical value of the occurrence of the token consecutively. For example, for the string aaabbacccbbbbd, the run is 3, 2, 1, 3, 4, 1. The run is computed for the generated strings and represented as vectors. The derived vectors must be of equal number of digits otherwise ‘0’ will be padded at the end of the vector with less number of digits.
The run for above preprocessed codes are 1 1 1 2 1 1 1 1 and 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1. The similarity between the generated vectors is computed.

C. similarity computation
The similarity between two vectors is computed using cosine similarity calculating their dot product. The cosine similarity between two vectors is computed using the formula
\[
\cos(\theta) = \frac{A \cdot B}{|A| |B|}
\]
The cosine similarity lies between the range [0, 1]. If the two vectors are similar than cosine value is 1 or if value is 0 implies vectors are not similar. The value in between specifies the similarity value.

IV. EXPERIMENTAL ANALYSIS
The submitted student assignments for a class in the programming language in C are considered. To compare the similarity we have considered our data base of 300 programs where the changes like removing of comment lines, changing of identifiers, changing of control statements. The following table shows similarity percentage among the source codes. Table-1 shows a part of the results obtained from the original set.

<table>
<thead>
<tr>
<th>Assignment Number</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D1</td>
</tr>
<tr>
<td>T1</td>
<td>0.512</td>
</tr>
<tr>
<td>T2</td>
<td>0.4822</td>
</tr>
<tr>
<td>T3</td>
<td>0.654</td>
</tr>
<tr>
<td>T4</td>
<td>0.735</td>
</tr>
</tbody>
</table>

By considering 0.7 as threshold, the above representation clearly shows that T1 is similar with D3, T2 similar with D3 and T3 with D4 and so on. Precision, recall, and F-measure are considered as metrics for evaluating the system and 0.84, 0.76, 0.79 respectively.

The proposed approach is compared with available standard tool JPlag and difference of F-measure is 0.19.

V. CONCLUSION AND FUTURE ENHANCEMENT
We were successful in identifying similarity among the submitted assignments and source codes that are available in repository using run length distribution.
Our approach transforms the source codes to pseudo code after removing white spaces, comments etc. The run length is calculated on derived pseudo code and it is represented in the form of vector. Using cosine similarity, the proximities between submitted assignments and codes in repository are calculated. The designed approach works efficiently even on large data base.
Our proposed approach outperformed with the existing software like Jplag. In particular, the assignments with various sizes are considered and the proposed approach worked well regardless of the size of source code. The proposed approach can be used with other programming languages like C++, Java. The approach can identify the similarity among the source codes with changes like alteration of comments, white spaces, renaming of identifiers and loops. We have left computing similarity for source codes where reordering of statements and functions as our future work.
REFERENCES


