Plagiarism Detection by using Karp-Rabin and String Matching Algorithm Together

Sonawane Kiran Shivaji
Master of Engineering, Computer Engineering,
Ahmednagar, Maharashtra,

Prabhudeva S
Asst. Professor,
VACOE, Ahmednagar, Maharashtra

ABSTRACT

In today word copying something from other sources and claiming it as an own contribution is a crime.

We have also seen it is major problem in academic where students of UG, PG or even at PhD level copying some part of original documents and publishing on own name without taking proper permission from author or developer.

Many software tools in exist to find out and assist the monotonous and time consuming task of tracing plagiarism, because identifying the owner of that whole text is practically difficult and impossible for markers. In our presentation we have focused on practical assignments (projects) as well as written document which is to be submitted by students in to college or university.

Because of this crucial task and day by day increasing research in different fields, industry, academy people demanding such software to detect whether submitted articles, books, national or international papers are genuine or not. In this paper, our algorithm divides submitted articles in small pieces and scans it to compare with connected databases to the server on internet. Some existing work compares submitted articles with previously submitted articles i.e. with existing database.

Keywords

Document retrieval; Plagiarism, Algorithm Karp-Rabin; plagiarism detection String mtching.)

1. INTRODUCTION

Since last few decades, it is a challenge to find out similarity between two documents and coping somebody's work in our paper's tendency is rocking. It is serious problem in all area. This challenge encourages us to take a efforts to provide practical approach for detecting plagiarism in two sequence.

A special issue is published by IEEE transaction on plagiarism. It is confirmed in Guest Editorial Plagiarism that [1], "Plagiarism is a deplorable and increasing threat to educational organizations and it is a risk for function of academic. This threat is especially true in a world where Information Technology has made copying information easier. Plagiarism is an act of fraud. It involves both stealing someone else's work and lying about it afterward".

We have also observed that often unclear margin between plagiarism and research.

Plagiarism Problem:

There are different kinds of plagiarism such as;

Replica: Copying exact another work as a own work.

Fusion: Copying text from multiple sources and creating new fusion of it without citation.

Borrows find similar word and replace it.

Aggregator: Papers having citation but that is not original work.

Paraphrasing: Borrows changed some word but not whole statement.

Copying Idea: Concept is taken without any coping text.

There are many reasons for plagiarism among students like laziness, fear of failure, high expectation, poor time management etc... It found that there is very less awareness about plagiarism and their respective action.

From this we can divide plagiarism in mainly 4 categories i.e. Singular, Paired, Multidimensional and Carpal. It is again spit into 2 parts, one is external and another is external plagiarism. External detection use list of reference document for detecting plagiarism in doubtful document.

In internal detection, it does not refer any reference document for plagiarism detection.

Plagiarism in Software Assignment: Most of the students copying content without prior permission or acknowledgment. In academic courses different programming languages are there and a student has performed many assignments on it in their academic years. Oftenly Students never pay much concentration and instead performing assignments they copy and paste programs inadvertently. Since we have focused on such tool to prevent student's tendency of copying contents of others and if they do such then teacher may be able to detect and punish to students.

Many academic courses have programming languages as subject or in many subject assignments have to written using different programming languages. Same problem or similar problem is assigned to the entire class. Students never pay much attention in their practical assignments. They copy and paste programs unintentionally. This tendency of student must be changed. Any tool which can detect such copy would support teacher to punish such cases.

To find out similarity between two sequences is a plagiarism or whether similarity is resulting from an analogous working method based on the hypothetical knowledge is difficult because what is cause of similarity is difficult to understand. And we are considering it precisely.

Since we are using Karp-Rabin algorithm along with String matching algorithm. Karp-Rabin used in many existing tools like Jplag, MOSS (Measure of software similarity), CPD (Copy/Paste Detector) etc….

2. RELATED WORK

Many program plagiarism detection tools are developed which are based on programming language keywords or logical statements of program [e.g.3-6]. To hide original code plagiarist adds unwanted program lines or change position of statements.

These changes can be detected by structure metric systems but exact plagiarism percentage cannot be measured. PK2 is a structure metric tool developed by Technical University, Madrid, Spain. The only situation that pk2 cannot detect is an assignment compounded by several very small fragments of source code. These tools may give falls result in case of shuffling of statements or addition of unnecessary statements. Comparison of tools based on these two concepts is done in detailed in paper [10].

Data dependency matrix method [8, 9] developed by us is a new concept which based on data assignment statements of program. Both this methods are elaborated and compared in following chapters.

The PK2 tool has been developed in the Computer Architecture Department of the Technical University of Madrid, Spain [1]. Students are asked to developed small project using system programming in C, assembly language programming, input/ output system and microprogramming. Students copied or plagiarized program by same method. Each programming language has its own keywords (reserved words). These keywords can be used to catch the cheater. Plagiarists are people who do not know enough to do the assignment on their own. They usually do aesthetic changes without significantly altering the underlying program structure, randomly changing identifier names, comments, punctuation, and indentation [1].

The PK2 tool is based on structure metric system. While comparing a programming language such as Java, only its reserved words and the most used library function names are considered. The PK2 processes each given program file, transforming it into an internal representation.

This process translated the occurrence of each key word by a corresponding internal symbol. This process generates signature string. The tool compares only the underlying program structure. As this tool is based on structure metric system it uses four similarity criteria. These are as follows:

1. Length of longest common substring.
2. Cumulative value of the length of common sequences

   of reserved words.
3. Normalized value of cumulative value
4. Percentage of reserve words common to both files.

The PK2 tool gives teachers hints about which pairs to inspect for plagiarism, but the final decision in a case of plagiarism is very difficult to make. This tool has proved to be flexible. It has been successfully used to detect partial and total copies in very different environments.

James A McCart and Jay Jarman, both were working on project of Microsoft Access and they proposed and developed tool as a Cheater Cheater Pumkin Eater (CCPE) in 2008. CCPE was written using Visual Basic for Applications within a Microsoft Access database. To determine if Microsoft Access projects were duplicates, properties such as the read-only creation date of the database and its objects of tables, queries, forms, reports, etc.. were compared. When a database or an object within a database is created, a document object (DO) is created which stores properties of the newly created database or object.

Each DO contains standard properties such as creation date, last updated date, and name. It also provides built-in summary properties of the database, such as the database title, are stored in a separate DO. The last updated properties date is associated with changes to the built-in summary properties. If database is copied then it is an exact duplicate of the original and all of the creation date, last updated date, and name properties for all of the objects within the copied database are the same as in the original.

The CCPE is very effective technological tool which is implemented to detect plagiarism in Database Access projects. This tool has given positive result by reducing percentage of plagiarized projects [3].

Tommy W. S. Chow and M. K. M. Rahman has developed a approach of "Multilayer SOM With Tree-Structured Data for Efficient Document Retrieval and Plagiarism Detection" in 2009 [2].

They have proposed data retrieval (DR) and plagiarism detection (PD) using tree-structured document demonstration and multilayer self-organizing map (MLSOM).

It evaluates a full input document or program as a query for executing retrieval of data and PD. Tree-structured representation of documents increases accuracy of DR and PD by including local with traditional global characteristics. Hierarchical presentation of documents of global and local variables enables the MLSOM to be used for PD. Tommy W. S. Chow and M. K. M. Rahman has proposed two methods of PD. First is an additional room to DR method along with additional local sorting. The second method is document association on the bottom layer SOM. Computational cost of DR and PD is very high due to capacity of huge document scanning at a once since it is useful for large database [2].

The SID tool is developed by Chen, Brent Francia, Ming Li, Brian Mckinnon, Amit Seker in the University of California[2]. They have defined the Information Distance between two sequences to be roughly the minimum amount of energy to convert one sequence to another sequence, and vice versa.

The SID is based on compression algorithm. Authors has created improved Lempel Ziv algorithm to obtain proper compression technique. It has included different steps of SID are as follows:

1. It breaks program string into small segments that is token.

2. Lempel Ziv algorithm Compresses tokens.

3. It finds out percentage of plagiarism by using Kolmogorov complexity formulas.

SID has been widely examined and then used. Users have used SID positively to catch plagiarism cases. It has checked UCSB programming assignments, JAVA assignments and many projects SID system uses a special compression program to heuristically approximate Kolmogorov complexity. It also shows similar part in two programs which reduces teachers work to search copied part. it also detects subtler similarities.

3. SYSTEM ALOGORITHM

Karp-Robin Algorithm:

We are taking a help of Karp-Rabin Algorithm. It uses fingerprints to find occurrences of one string into another string. Karp-Rabin Algorithm reduces time of comparison of two sequences by assigning hash value to each string and

word. Without hash value, it takes too much time for comparison like if there is a word W and input string is S then word is compared with every string and sub string in program and hence it consumes more time. Karp-Rabin has introduced concept of Hash value to avoid time complexity O(m2). It assigns hash value by calculating to both word and string/substring. So hash of substring (S) matches with hash value of W then only we can say exact comparison is done.

At the comparison process there are four categories [4]:

1. Right to left

2. Left to right-

3. In specific order

4. In any order

Karp-Rabin algorithm preferred category from left to right comparison. Function of hash must able to find has value efficiently. When first time name would be hashing with the same hash it save the data causing yields a value which will be compared to at data is index with the value.

It can deal with multiple pattern matching that's why people preferred this Karp-Rabin algorithm. Otherwise behavior of other algorithm is to perform basic pattern matching.

Its having O(nm) complexity. Where n is length of text and m is length of pattern. It is little bit slow also due to we have to check every single character from the text.

But we can overcome this by having hash function which is efficient as well as easy to implement.

Suppose a k-grams c1…ck is consider as K- digit number by considering base b, then hash value H(c1….ck) will be;

$$c_1*b_{k-1} + c_2*b_{k-2}+\ldots.+c_{k-1}*b+c_k$$

We are using dependency matrix for comparison of same size matrices [9]. It is assumed that plagiarist will change text, position or name of variables but total variables in a function would remain same. Such a plagiarized code can be detected using the algorithm. The expression list algorithm compares all lists of functions with another function this is advantageous in various ways. In this case matrix method gives false detection as it compares only same size matrices. This is drawback of matrix system.

String matching Algorithm:

It is used to compute similar strings. It performs character by character matching.

4. SYSTEM ARCHITECTURE

Block Diagram shown in Fig 1 which gives outline of Plagiarism detection by using Hash function and string matching algorithms. It gives overall idea about processing of string matching as well as creating similarity matrix for finding out percentage of plagirism.

Block Diagram of Plagiarism detection by using these algorithms:

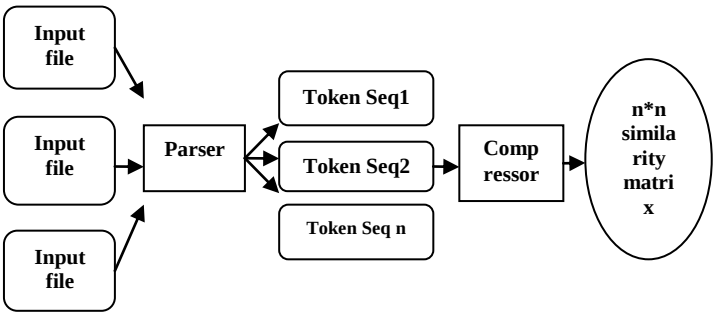

Fig 1: System Architecture

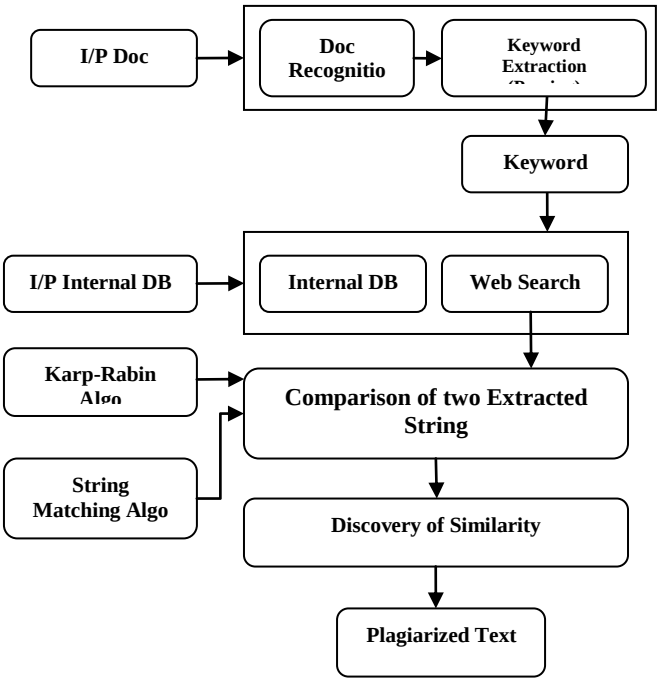

Fig 2: System Architecture of Plagiarism Detection.

Here system architecture shows in Figure 2. We have given input document for parsing process.

It is also known as Syntactic Analysis which performs analysis of given i/p, it can be a natural language or machine understandable language. It checks heuristics rules which are predefined in system. It also confirms grammar rules while matching string. It breaks sentence into tokens known as segmentation.

In keyword extraction, it find and extract keyword from whole i/p document, removes stop Word and stored as a stemmed words in Keyword list.

Then Keyword is given as input to the Search Engine in System. In Web Search engine, it contains internal dataset which are provided by internal user/candidates from college/university etc…

We can also map external dataset with existing system dataset.

Search Engine forward two set of string for matching towards Comparison Engine module in which input dataset and internal dataset query present.

Here we are using Karp-Rabin algorithm along with String matching algorithm for detecting suspicious material in given documents. Karp-Rabin is string searching and comparison algorithm by using hash function. Karp-Rabin also speeds up the processing of string comparison by matching given pattern with different i/p document's string/substring by using hash values. It uses hash function for assigning hash value to every string/substring in text.

If hash value of given pattern and substring matches, it means two string are similar.

Eg. WordString [Plagiarism]=6[hash value],

If search engine got similar hash value in internal dataset then both strings are matching strings.

There is problem with Karp-Rabin algorithm is that, for keeping minimum value of hashed word, it assign similar hash values for different string in documents. Since it creates confusion even hash value are same but string are non similar. Sometimes it cannot detect similarity.

For removing this drawback and improving efficiency of system, we are using String Matching algorithm along with Karp-Rabin Algorithm when it detects similar hash value in system. It keeps string in arrays and checks it character by character in array. It checks for similarity, if it is match then and then it is similar. Since it improves accuracy of plagiarism detection which are not getting in existing tools.

Then result of similarity is generated and highlights that plagiarized text. It also shows the source of plagiarized text in document.

In analysis part we have seen different tools performance and our proposed system performance which we are claiming here.

We have computer Plagiarism detection (PD) accuracy by using Precision value and Recall value as a follows; [13]

$$\text{Precision Value (P)} = \frac{\text{Number of Correct Doc recovers for PD}}{\text{Number of total doc recovers for PD}}$$

$$\text{Recall Value (R)} = \frac{\text{Number of Correct Doc recovers for PD}}{\text{Number of Total relevant Doc for PD}}$$

Table 1: Performance Analysis

Sr. No	Approach	Precision Val %	Recall Val %
1	MLSOM	64%	60%
2	LSI	63%	66%
3	Proposed System	80% and Above	80% and Above

5. CONCLUSION

This paper proposes new plagiarism detection techniques by using Karp-Robin algorithm and String Matching algorithm. Here data dependency expression list, extracted keyword and using dual algorithm approach which overcomes all problems of matrix, similar hash value as well as string matching, which detects plagiarized programs or documents by using hash function.

Experiments have well verified its efficiency over existing tools and its applicability in practice. Our proposed system will give result of precision value up to 85% and above as well as recall value. It is also able to minimize failed detection percentage around 10%.

6. ACKNOWLEDGMENT

I am very thankful to the people those who have provided me continuous encouragement and support to all the stages and ideas visualize. I am very much grateful to the entire BVDU group for giving me all facilities and work environment which enable me to complete my task. I express my sincere thanks to M.K.Shirsagar Sir, Prabhudeva Sir, Head of the Computer Department, VB College of Engineering, Ahmednagar who gave me their valuable and rich guidance and help in presentation of this research paper.

7. REFERENCES

[1] Manuel Cebrián, Manuel Alfonseca, and Alfonso Ortega "Towards the Validation of Plagiarism Detection Tools by Means of Grammar Evolution" IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 13, NO. 3, JUNE 2009.

[2] Tommy W. S. Chow and M. K. M. Rahman has developed a approach of "Multilayer SOM With Tree-Structured Data for Efficient Document Retrieval and Plagiarism Detection" IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 20, NO. 9, SEPTEMBER 2009

[3] Francisco Rosales, Antonio García, Santiago Rodríguez, José L. Pedraza, Rafael Méndez, and Manuel M. Nieto," Detection of Plagiarism in Programming Assignments" , IEEE Transactions on Education, vol.51, no.2, May 2008, pp.174-183.

[4] Arliadinda D, Yuliuskhris Bintoro, R Denny Prasetyadi Utomo, Pencocokan String dengan Menggunakan Algoritma Karp-Rabin dan Algoritma Shift Or. Jurnal STT Telkom.

[5] Manuel Cebrián, Manuel Alfonseca, and Alfonso Ortega "Towards the Validation of Plagiarism Detection Tools by Means of Grammar Evolution" IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 13, NO. 3, JUNE 2009.

[6] X. Chen, B. Francia, M. Li, B. McKinnon, and A. Seker, "Shared information and program plagiarism detection," IEEE Trans. Inf. Theory, vol. 50, no. 7, pp. 1545–1551, Jul. 2004.

[7] A. Parker and J. O. Hamblen, "Computer algorithm for plagiarism detection," IEEE Trans. Educ., vol. 32, no. 2, pp. 94–99, May 1989.

[8] S. Schleimer, D. Wilkerson, and A. Aiken, "Winnowing: Local algorithms for document fingerprinting," in Proc. 22nd Association for Computing Machinery Special

Interest Group Management of Data Int. Conf., San Diego, CA, Jun. 2003, pp. 76–85.

[9] Seema Kolkur , Madhavi Naik (Samant) "Program plagiarism detection using data dependency matrix method" , in proceedings of International Conference on Computer Applications 2010 , Pondicherry , India December 24-27, 2010, pp 215-220.

[10] Seema Kolkur, Madhavi Naik (Samant) "Comparative study of two different aspects of program plagiarism detection", presented and published in proceedings of International Conference On Sunrise Technologies 2011, Dhule, India January 13-15, 2011.

[11] Xin Chen, Brent Francia, Ming Li, Member, IEEE, Brian McKinnon, and Amit Seker "Shared Information and Program Plagiarism Detection" IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 50, NO. 7, JULY 2004.

[12] Chao Liu, Chen Chen, Jiawei Han , Philip S. Yu "GPLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis", KDD'06, Philadelphia, Pennsylvania, USA. August 20–23, 2006.

[13] Tommy W. S. Chow and M. K. M. Rahman " Multilayer SOM With Tree-Structured Data for Efficient Document Retrieval and Plagiarism Detection"IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 20, NO. 9, SEPTEMBER 2009.

[14] K. L. Verco and M. J. Wise, "Software for detecting suspected plagiarism: Comparing structure and attribute-counting systems," in *Proc. 1st SIGCSE Australasian Conf. Computer Science Education*, J. Rosenberg, Ed., New York, Jul. 1996, pp. 81–88.

[15] M. Joy and M. Luck, "Plagiarism in programming assignments," *IEEE Trans. Educ.*, vol. 42, no. 2, pp. 129–133, May 1999.

[16] X. Chen, B. Francia, M. Li, B. McKinnon, and A. Seker, "Shared information and program plagiarism detection," *IEEE Trans. Inf. Theory*, vol. 50, no. 7, pp. 1545–1551, Jul. 2004.

[17] C. Liu, C. Chen, J. Han, and P. S. Yu, "Gplag: Detection of software plagiarism by program dependence graph analysis," in *Proc. 12th Int. Conf. Special Interest Group Knowledge Discovery and Data Mining*, New York, 2006, pp. 872–881.

[18] A. Apostolico, "String editing and longest common subsequences," in *Handbook of Formal Languages, Volume 2 Linear Modeling: Background and Application*. Berlin, Germany: Springer-Verlag, 1997, pp. 361–398.

[19] L. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms," in *Proc. 7th Int. Symp. String Processing Information Retrieval*, Los Alamitos, CA, 2000, pp. 39–48.

[20] V. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Sov. Phys.—Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.

[21] C. Daly and J. Horgan, "Patterns of plagiarism," in *Proc. 36th SIGCSE Tech. Symp. Computer Science Education*, New York, 2005, pp. 383–387.

[22] *"MC88110: Second Generation RISC Microprocessor User's Manual,"* Motorola Inc., Schaumburg, IL, 1991.

[23] A. S. Tanenbaum, *Modern Operating Systems*, 2nd ed. Engelewood Cliffs, NJ: Prentice-Hall, 2001.

[24] M. Córdoba and M. Nieto, Jul. 2007, Technical Univ. Madrid, Spain http://www.datsi.fi.upm.es/docencia/Estructura/U_Control

[25] C. Bell and A. Newell, *Computer Structures: Readings and Examples*. New York: McGraw-Hill, 1971.

[26] F. Culwin and T. Lancaster, 2001, Plagiarism, Prevention, Deterrence and Detection [Online]. Available: http://www.ilt.ac.uk/resources/ Culwin-Lancaster.htm