A Survey on Plagiarism Detection Systems

A. S. Bin-Habtoor and M. A. Zaher

***Abstract*--Being a growing problem, plagiarism is generally defined as "literary theft" and "academic dishonesty" in the literature, and it is really has to be well-informed on this topic to prevent the problem and stick to the ethical principles. This paper presents a survey on plagiarism detection systems, a summary of several plagiarism types, techniques, and algorithms is provided. Common feature of deferent detection systems are described. At the end of this paper authors propose a web enabled system to detect plagiarism in documents , code and images, also this system could be used in E-Learning, E-Journal, and E-Business.**

***Index Terms*—Plagiarism detection, plagiarism types, plagiarism techniques, plagiarism algorithms.**

I. Introduction

The term "plagiarize" is defined as to take (ideas, documents, code, image, etc) from another and pass them off as one's own without citation.

So plagiarism is a global problem, which occurs in many different areas of our life. There are many different forms of plagiarism, Plagiarism at schools can be a highly de-motivating factor for teachers and also for students. If plagiarism is not addressed sufficiently, plagiarists could gain undeserved advantage, e.g. more marks for their assignments with less effort.

There are various types of plagiarism [1] involved: using sources without properly citing them, paraphrasing text, reusing ideas with/without citing references, and others.

A plagiarized document detection plays important roles in many applications, such as file management, copyright protection, and plagiarism prevention. Existing protocols assume that the contents of files stored on a server are directly accessible. This assumption limits more practical applications, e.g., detecting plagiarized documents between two conferences, where submissions are confidential [2]. Plagiarism can take one of the popular types such as copying of the whole or some parts of the document, rewording same content in different words, using others' ideas or referencing the work to incorrect or non-existing sources [3]. Other ways of plagiarism include translated plagiarism wherein the content is translated and used without referencing the original work, artistic plagiarism in which different media such as images and videos are used to present other's work without proper citation [3]

Manuscript received February 18, 2012; revised March 31, 2012.

A. S. Bin-Habtoor was with the Electronics and Communication Engineering. From University of Technology, Baghdad (e-mail: ashabtoor@nu.edu.sa, dr_asyaboaymen@yahoo.com).

M. A. Zaher is with the current job is instructor at Faculty of Sciences and Arts at Al-Aflaj, Salman Bin Abul-Aziz University, KSA (e-mail: zilog2003@yahoo.com,).

A plagiarized code (also called code clone) [4] which can be defined as the reuse of the source code without permission or citation. So a plagiarized program can be defined as a program which has been produced from another program with a small number of routine transformations, routine transformations, typically text substitutions, do not require a detailed understanding of the program. Unfortunately, plagiarism of programming assignments has been made easier by large class sizes.

Plagiarism of computer programs can become quite common in large undergraduate classes. With a few simple editor operations it is possible to produce a plagiarized program with a different visual appearance. This makes the manual detection of plagiarized program difficult in large classes.

All these practices of plagiarism have negative impact on the learning process. Thus, how can we ensure dealing with plagiarism systems and how is plagiarism going to be detected and dealt with. It is a critical issue that needs solutions by computer scientists.

II. Review

We classified the survey into four categories:

1- Plagiarism in documents.
2- Plagiarism in code.
3- Plagiarism techniques.
4- Plagiarism algorithms.

These categories explained as follows:

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 regard, numerous plagiarism detection systems have been developed. These systems can be classified into two main categories, web-enabled systems and stand-alone systems.

1) Web-enabled systems: Developing web systems for plagiarism detection overcomes machine capability problems, facilitate the availability of the system to many users and extend the search of plagiarized resources to the World Wide Web easily. Here is discussion of two: First Turnitin [5, 6] is the most well-known commercial plagiarism detection system to which many universities from UK and USA subscribe. It uses an enormous database from the Internet and previous student works to be compared with the query document. Second SafeAssign [7] checks all submitted papers against the following databases: (i) the Internet. (ii) ProQuest database. (iii) Institutional document

archives containing all documents submitted to SafeAssign. (iv) Global Reference Database containing documents that were volunteered by students to help prevent cross-institutional plagiarism.

2) Stand-alone systems: Stand-alone software is developed to be installed on computers. Two systems will be explored here, EVE [6, 8, 9] and WCopyFind [6, 9, 10]. First EVE (The Essay Verification Engine) is a desktop application but it has the capability to make large number of searches on the Internet to locate matches between sentences in the query document and suspected websites. Thus, in order for EVE to work, the machine should be connected to the Internet. Second WCopyFind developed by University of Virginia, finds plagiarism between two or more assignments. The user can set or change some of the parameters that may influence the detection process such as the number of words used for detecting similarity among statements.

Several other tools have been developed for plagiarism detection such as Diff [11], SCAM [12], COPS [13], KOALA [14], SSK [15], CHECK [16], MDR [17, 18, 19], PPChecker [20], SNITCH [21], and Ferret [15, 22, 23]. They use variety of document characteristics that need different plagiarism detection approaches such as fingerprinting and fuzzy information retrieval [24].

B. *Plagiarism in Code*

Various plagiarism approaches have been proposed for detecting source code written with C, C++ or JAVA [25]. Each of these approaches focuses on certain characteristics of code plagiarism. For example, there are approaches which are designed mainly to compare source codes written in different programming languages. There are also approaches which are designed to handle complicated code modification but require longer detection time compared to common approaches. One of the approaches that we considered suitable for detecting plagiarism in programming course is the structure-based method, which mostly use tokenization and string matching algorithm to measure similarity. Some of existing plagiarism detectors that employ such structure-based methods are Plague [26], YAP [27] and JPlag [28].

1) Plague is one of the earliest structure-based detectors. Plague works in several steps. First, structure profiles of each source code are created. Then, those structure profiles are compared using Heckel algorithm. Suggested by Paul Heckel, the algorithm is designed to handle text files. Plague’s detection results are returned in the form of lists. By using a corresponding interpreter, the results can be processed further to make it easier to comprehend for common users. Plague is able to detect plagiarism for source code written in C.

2) YAP was developed based on Plague with some enhancements. The first version was created by Michael Wise. Then it was optimized into YAP2. The final version YAP3, which can also be used to detect text plagiarism [29]. All three versions of YAP have two phases in their processes. The first phase is the generation phase, where a token file is created for each source code. The second phase is comparison of every token files. The result of each comparison is a value called percent match, a value between 0 as minimum and 100. If the percent match of a pair of token files is larger than this minimum value, then the corresponding pair will be judged as a case of suspected plagiarism. YAP’s detection result is presented in the form of a text file.

JPlag is a system that can be used to detect plagiarism for source code written in Java, C, C++ and Scheme. It is available as a free web service. Its input is a directory containing programs that will be detected. Every source code in the directory are parsed and transformed to token strings.

3) These token strings will be compared to each other using Running Karp-Rabin Greedy String Tiling algorithm. JPlag’s detection result is displayed as a group of HTML files that can be opened using a standard browser. Detection statistics, similarity distribution, and pairs of programs suspected as plagiarism instances are shown on the main page[30,31]. The user can also choose a certain pair of program to be shown side-by-side. Similar segments of the code will be marked with different font colors.

C. *Plagiarism Techniques*

Plagiarism techniques known as similarity detection techniques [32]. A good example is found in the formerly popular attribute counting_techniques. Attribute counting techniques (such as [33] and [34]) create special “fingerprints” for collection files, including metrics, such as average line length, file size, average number of commas per line. The files with close fingerprints are treated as similar. Clearly, small fingerprint records can be compared rapidly, but this technique is now considered unreliable, and rarely used nowadays [35]. Modern plagiarism detection systems usually implemented using certain content-comparison techniques. The most popular techniques include string tiling, finding the joint coverage for a pair of files [36, 37] and parse trees comparison [38, 39]. Usually these techniques work for file pairs, so the comparison routine should be called for each possible file pair found in the input collection.

Also Fast Plagiarism Detection technique (FPDS) [40] tries to improve the algorithmic performance of plagiarism detection by utilizing a special indexed data structure to store input collection files.

And Tokenization [41] is a commonly-used technique that fights against renaming variables and changing loop types in computer programs. Simple tokenization algorithms substitute the elements of program code with single tokens. For example, all identifiers can be substituted with <IDT>, and all values with <VALUE> tokens. So, a line $a = b + 45$; will be replaced by <IDT>=<IDT>+<VALUE>; Therefore, renaming variables will not help the plagiarizer [42].

D. *Plagiarism Algorithms*

A number of algorithms to detect plagiarism are discussed. The simple algorithm based on string comparisons will explain as shown below:

1) Remove all comments.
2) Ignore all blanks and extra lines, except when needed as delimiters.
3) Perform a character string compare between the two

files.

4) Maintain a count of percentages of character correlation.

This algorithm is run for all possible program pairs. This simple algorithm will detect many cases of plagiarism. For code plagiarism detection, Faidhi and Robinson [43] characterize sex levels of program modification in a plagiarism spectrum. Level 0 is the original program without modifications. In level 1, only comments are changed. Level 2 changes the identifier names. Level 3 changes position of variables. Level 4 changes constants and procedures. In level 5 program loops are changed. In level 6 control structures are changed to an equivalent form using a different control structure (i.e. "for" changed to "if").

Several algorithms for plagiarism detection are based on software metrics [41]. Theses algorithms extract several software metrics features from a program and use this set of features to compare programs for plagiarism.

III. Proposed System

According to what has been discussed in the survey above (the plagiarism types, techniques, and algorithms), we propose a system for detection plagiarism in electronic resources. Another words a web enabled system to detect plagiarism in documents, code and images. For detection of plagiarism in documents we can use and develop similarity technique between the documents. And the tokenization technique will be used for detecting plagiarism in code. Also the simple algorithm will be used for comparing documents and code. And the image vector representation will be considered as the main issue when detecting plagiarism in images.

Finally, we propose an information system for detecting plagiarism in electronic resources used for detecting plagiarism in documents, code, and images, where the framework will be publish in another paper.

IV. Conclusion

A survey on plagiarism detection systems has been introduced.

With the evolution of the internet and the need for information the plagiarism continues to be a concern problem to universities, teachers, policy-makers and students. So authors conclude that the need of plagiarism detection systems become very important issues and the use of plagiarism detection systems in E-Learning improve academic integrity, and also instances of plagiarism can be greatly reduced, if not eliminated, with the use of a plagiarism detection systems. Authors propose a system that is able to detect many plagiarism attempts in deferent fields (E-Learning, E-Business, and E-Journals) and can be used to evaluate programs, papers with images included, and therefore, increasing the quality of its design.

References

[1] P. OGR, "What is Plagiarism?", [On Line] http://www.plagiarism.org/,Retrieved Nov. 15, 2010

[2] C. Lyon, R. Barrett, and J. Malcolm, "Plagiarism is Easy, but also easy to detect." *Cross-Disciplinary Studies in Plagiarism, Fabrication, and Falsification*, 2006.

[3] L. Romans, G. Vita, and G. Janis, "Computer-based plagiarism detection methods and tools: an overview," *the 2007 international conference on Computer systems and technologies*. 2007, ACM: Bulgaria.

[4] S. Mann and Z. Frew, "Similarity and originality in code: plagiarism and normal variation in student assignments," *the 8th Australian conference on computing education*, 2006.

[5] L. Chao, L., et al., "GPLAG: detection of software plagiarism by program dependence graph analysis," *the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, ACM: Philadelphia, PA, USA.

[6] C. J. Neill and G. Shanmuganthan, "A Web-enabled plagiarism detection tool." *IT Professional*, 2004.

[7] M. Ginger and C. Christian, "K-gram based software birthmarks," *the 2005 ACM symposium on applied computing*. 2005, ACM: Santa Fe, New Mexico.

[8] C. Hung-Chi, W. Jenq-Haur, and C. Chih-Yi, "Finding Event-Relevant Content from the Web Using a Near- Duplicate Detection Approach," *the IEEE/ACM International Conference on Web Intelligence*. 2007, IEEE Computer Society.

[9] H. Dreher, "Automatic Conceptual Analysis for Plagiarism Detection." *Issues in Informing Science and Information Technology*, 2007.

[10] L. J. Edward, "Metrics based plagarism monitoring." *Consortium for Computing Sciences in Colleges*, 2001.

[11] R. Yerra, "A Sentence-Based Copy Detection Approach for Web Documents," in *Fuzzy Systems and Knowledge Discovery*. 2005.

[12] S. Narayanan and G. Hector, "Building a scalable and accurate copy detection mechanism," *the first ACM international conference on Digital libraries*. 2006, ACM: Bethesda, Maryland, United States.

[13] B. Sergey, "Copy detection mechanisms for digital documents." *ACM international conference*, 2005.

[14] N. Heintze, "Scalable document fingerprinting." *the Second USENIX Workshop on Electronic Commerce*. 2006.

[15] J. P. Bao, "Semantic Sequence Kin: A Method of Document Copy Detection," in *Advances in Knowledge Discovery and Data Mining*. 2004.

[16] S. Antonio, L. Hong Va, and W. H. L. Rynson, "CHECK: a document plagiarism detection system," in *Proceedings of the 2007 ACM symposium on Applied computing*. ACM: San Jose, California, United States.

[17] W. Kienreich, "Plagiarism Detection in Large Sets of Press Agency News Articles." In *17th International Conference on Database and Expert Systems Applications*, 2006. DEXA '06. 2006.

[18] Kriszti, "Document overlap detection system for distributed digital libraries," *the fifth ACM conference on Digital libraries*. 2000, ACM: Texas, USA.

[19] A. F. Raphael, "Signature extraction for overlap detection in documents," *the twenty-fifth Australasian conference on Computer science*, 2002, Australian Computer Society, Inc.: Melbourne, Victoria, Australia.

[20] N. Kang, A. Gelbukh, and S. Han, "PPChecker: Plagiarism Pattern Checker in Document Copy Detection, in Text," *Speech and Dialogue*. 2006.

[21] N. Sebastian and P.W. Thomas, "SNITCH: a software tool for detecting cut and paste plagiarism." 2006, ACM.

[22] J. Bao, C. Lyon, and P. Lane, "Copy detection in Chinese documents using Ferret." *Language Resources and Evaluation*, 2006.

[23] J. Bao, "A fast document copy detection model." *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 2006.

[24] S. M. Alzahrani, "Plagiarism auto-detection in Arabic scripts using statement-based fingerprints matching and fuzzy-set information retrieval approaches." 2008, University of Technology Malaysia: Johor.

[25] A. Christian and S. M. M. Tahaghoghi, "Plagiarism detection across programming languages," *the 29th Australasian Computer Science Conference*, 2006.

[26] G. Whale, "Plague : plagiarism detection using program structure," *Dept. of Computer Science Technical Report 8805*, University of NSW, Kensington, Australia, 2008

[27] M. J. Wise, "Detection of Similarities in Student Programs: YAP'ing may be Preferable to Plague'ing," *ACM SIGSCE Bulletin (proc. Of 23rd SIGSCE Technical Symp.)*, 2002.

[28] P. Lutz, M. Guido, and M. Phlippsen, "JPlag: Finding plagiarisms among a set of programs," *Fakultät für Informatik Technical Report 2000-1*, Universität Kalrsruhe, Karlsruhe, Germany, 2000.

[29] M. J. Wise, “YAP3: Improved Detection of Similarities in Computer Programs and Other Texts,” SIGCSE’06, 2006.

[30] M. J. Wise, “Neweyes: A System for Comparing Biological Sequences Using the Running Karp-Rabin Greedy String Tiling Algorithm,” *Department of Computer Science, University of Sydney,* Australia, Technical Report 463, 2003 ,

[31] M. J. Wise, “String Similarity via Greedy String Tiling and Running Karp-Rabin Matching,” *Department of Computer Science, University of Sydney,* Australia, 2003.

[32] R. Karp and M. Rabin, “Efficient Randomized Pattern-Matching Algorithms,” *IBM Journal of Research and Development*, 2007.

[33] S. Grier, “A tool that detects plagiarism in Pascal programs,” *ACM SIGCSE Bulletin*, 2001.

[34] J. A. Faidhi and S. K. Robison, “An empirical approach for detecting program similarity within a university programming environment,” *Computers and Education*, 2008.

[35] K. L. Verco and M. J. Wise, “a comparison of automated systems for detecting suspected plagiarism,” *The Computer Journal*, 2005.

[36] M. J. Wise, “YAP3: improved detection of similarities in computer program and other texts,” *Proc. of SIGCSE’96 Technical Symposium*, 2006.

[37] L. Prechelt, G. Malpohl, and M. Philippsen, “Finding plagiarisms among a set of programs with JPlag,” *Journal of Universal Computer Science*, 2008.

[38] D. Gitchell and N. Tran, “Sim: a utility for detecting similarity in computer programs,” *the 30th SIGCSE Technical Symposium on Computer Science Education*, 2006.

[39] B. Belkhouche, A. Nix, and J. Hassell, “Plagiarism detection in software designs,” *Proc. of the 42nd Annual Southeast Regional Conference*, 2004.

[40] M. Mozgovoy, K. Fredriksson, and D. White, “Fast plagiarism detection system,” *Lecture Notes in Computer Science*, 2005.

[41] M. Joy and M. Luck, “Plagiarism in Programming Assignments,” *IEEE Transactions of Education*, 2009

[42] B.S. Baker, “On finding duplication and near-duplication in large software systems,” *Proc. of the 2nd IEEE Working Conference on Reverse Engineering*, 2005

[43] J. A. Faidhi and S. K. Robinson, “An empirical approach for detecting program similarity and plagiarism within a university programming environment,” *computer education*, 2007.

A. S. Bin-Habtoor Shabwah, Yemen, Dec. 1st, 1963, PhD in Electronics and Communication Engineering. From University of Technology, Baghdad, 2004. He gain work experiences as follows: the current job is asses. prof. at Faculty of Sciences and Arts at Sharourah, Najran University, KSA since 2010 General Administrator of information networks in Hadhramout University of Science and Technology from 2008 to 2010. Coordinator of Hadhramout University with ministry of Higher education especially in the information networks from 2008 to 2010. E-Mail ashabtoor@nu.edu.sa dr_asyaboaymen@yahoo.com

M. A. Zaher Dakahlia, Egypt, Jan.31st, 1971, MSc in information systems. From mansoura University faculty of information and computer sciences, Egypt, 2010. He gain work experiences as follows: instructor in computers science at PCNET, NY, NY from 1997 to 2001. Faculty of Sciences and Arts at Sharourah, Najran University, KSA, 2010. The current job is instructor at Faculty of Sciences and Arts at Al-Aflaj, Salman Bin Abul-Aziz University, KSA, since 2011. E-Mail zilog2003@yahoo.com.