International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 2, Issue 5, Nov 2013, 147-154

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

STUDYING DIFFERENT METHODS FOR PLAGIARISM DETECTION

ALAA M. RIAD[1], FARAHAT F. FARAHAT[2], AZIZA S. ASEM[3] & MAHMOUD A. ZAHER[4]

[1]Professor & Dean of Faculty of Computers and Information, Mansoura University, Egypt

[2]Professor of Information System, Sadat Academy, Egypt

[3]IS Department, Faculty of Computers and Information, Mansoura University, Egypt

[4]Instructor at Faculty of Sciences and Human Studies at Al-Aflaj, Salman Bin Abdul-Aziz University, KSA

ABSTRACT

With the hug of the information on WWW and digital libraries, Plagiarism became one of the most important issues for universities, schools and researcher's fields. It is so easy through the internet and due to using advanced search engine to find documents or journals by students. So plagiarism is a global problem, which occurs in many different areas of our life. Also detecting the plagiarism is a challenging task. This paper presents a study of different systems of plagiarism detection, a summary of several plagiarism detection types, and techniques, are provided. Common feature of different detection systems are described. This paper presents a web enabled system to detect plagiarism in documents, and code, also this system could be used in E-Learning, E-Journal, and E-Business.

KEYWORDS: Plagiarism Detection, Plagiarism Types, Similarity Detection Techniques

INTRODUCTION

According to www.plagiarism.org, plagiarism is any of the following activities: Turning in someone else's work as your own, copying words or ideas from someone else without giving credit, failing to put a quotation in quotation marks, giving incorrect information about the source of a quotation.

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 involved: using sources without properly citing them, paraphrasing text, reusing ideas with/without citing references, and others. [1]

Plagiarized document detection plays important roles in many applications, such as file management, copyright protection, and plagiarism prevention. [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 [4]

A plagiarized code (also called code clone) 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, 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. [5]

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.

In this paper, section 2 classifies the plagiarism detection systems, reviews the plagiarism in documents and the plagiarism in source code and sec.3 presents the proposed system (ZPlag), final section , Sec.4 conclude this paper.

REVIEW

Plagiarism detection systems are classified into two major categories; Plagiarism in documents and Plagiarism in source code.

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. Most of these systems use plagiarism techniques known as similarity detection techniques, which 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.

These systems will be discussed as follows and Table1 summarize there features.

- Turnitin is the global leader in evaluating and improving student writing. The company's cloud-based service for originality checking, online grading and peer review saves instructors time and provides rich feedback to students. One of the most widely distributed educational applications in the world, Turnitin is used by more than 10,000 institutions in 126 countries to manage the submission, tracking and evaluation of student papers online.

Institutions license Turnitin on an annual basis. The institutions are encouraged to communicate with students about their use of Turnitin and how their academic integrity policies work. An instructor sets up a class and an assignment in the Turnitin service. Students or instructors then submit papers to Turnitin via file upload or cut-and-paste.

Turnitin currently offers interfaces and content matching for Originality Check and Grade Mark in English (Americas and UK), Spanish, French, German, Chinese (Traditional and Simplified), Japanese, Korean, Turkish, Portuguese, Italian , Arabic, Swedish and Dutch.

In Turnitin, we cannot have both the documents in one window to see the similarities of the compared texts. The only document that is shown is the suspicious text; the parts similar to the other document appear in red the distinct parts are in black color. [6] As stated on the web site, Turnitin processed over 80 million papers in 2012.

- APlag A new plagiarism detection tool for Arabic texts, based on a logical representation of a document as paragraphs, sentences, and words, and new heuristics for text comparison, presents the results of some experiments conducted on a dummy test set. It is also built around a content-based method consisting mainly in fingerprinting the texts according to Arabic language specificity and comparing their logical representations by

using heuristic methods. The preprocessing phase consists mainly in: Tokenization, stop-word removal, rooting and synonym replacement. [7]

The tool (APlag) uses three original heuristics on document trees to detect similarities at different logical levels (document, paragraph, and sentence) and LCS (longest common substring) as a similarity metric. Moreover the user interface implemented in Java. It uses a data set of 12 different Arabic texts to generate plagiarized texts with different levels of modification [7]:

- Copy/paste of randomly selected sentences and/or
- Paragraphs in the same or different order.
- Synonym replacement of randomly selected words.
- Inclusion of English words.

- EVE (The Essay Verification Engine) is a desktop application and 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. EVE checks student documents against available sources for purposes of identifying overly similar (plagiarized) textual patterns. Also EVE measures linguistic patterns contained in whole submitted papers against others found on the Internet. As stated on the EVE2 web site, EVE is very "powerful tool" can be used "at all levels of the education system to determine if students have plagiarized from the World Wide Web." And accepts essays in plain text, Microsoft Word, or Corel Word Perfect format and returns links to web pages from which a student may have plagiarized. Also Eve performs a large number of complex searches to find material from any Internet site and then does a direct comparison of the submitted essay to the text appearing on the suspect site. If it finds evidence of plagiarism, the URL is recorded. Once the search has completed, the teacher is given a full report on each paper that contained plagiarism, including the percent of the essay plagiarized, and an annotated copy of the paper showing all plagiarism highlighted in red. [6,8]
- APD (Arabic Plagiarism Detection) tool use the Internet to help professors and teachers in e-learning systems identify stolen intellectual property by utilizing Google API to find similar documents on the web [9]. The typical workflow in APD paradigm has two major steps. The first step, students submit their assignments in Arabic to the system, which in turn will be stored into reports database. The second step, the teacher triggers APD tool via a user interface to check the assignments for plagiarism. Then, the tool will compare the documents against the intra corpus collection which probably contains the previous assignments. Moreover, APD tool searches the web to give similar resources as well. An automatic report will be generated that contains highlighted plagiarized parts and a list of similar resources ranked from highest to lowest. APD tool has the framework illustrated in Figure 2. As shown, students, teachers and administrators are the main users. They can register, login after registration using email and password, use tools designed for each, and logout the system. The system was designed as web-enabled tool to be accessible for a wide spectrum of users and to avoid machines compatibility problems.

Table 1: Features of Existing Documents Plagiarism Detection Systems

	Turnitin	**APlag**	**EVE**	**APD**
Algorithms Used	Similarity technique	Similarity technique	Similarity technique	Similarity Technique
Document Language support	Multi language	Arabic	English	Arabic
Platforms for Client	Web-based	Desktop	Web-based	Web-based

Table 1: Contd.,

Main Users	Student / instructor / Teaching assistant	Teachers	Teachers / Students	Admin / Teachers / Students
File submission Mode	Web form	Desktop application	Web form	Web form
File format for submission	Microsoft word (doc)	Microsoft word (doc), Arabic text	Microsoft word , plain text , Corel word perfect	Microsoft word (doc),
Results	Web site and instructor report	Desktop report	Teacher report	Classroom report

Plagiarism in Source Code

Various plagiarism approaches have been proposed for detecting source code written with C, C++ or JAVA [11]. 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. And Tokenization [12,13] 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 [14].

Some of existing plagiarism detectors that employ such structure-based methods are **YAP** [15] , **JPlag** [16], **SID** [17], and **MOSS** [18] and Table2 summarize there features.

- **YAP**, which stands for Yet Another Plague, tries to find a maximal set of common contiguous substrings to detect plagiarism. It has three different versions - YAP1, YAP2 and YAP3. All three versions of YAP work as follows. In the first phase, source texts are used to generate token sequences. This phase involves several operations, such as, removal of comments and string-constants, translation from upper-case letters to lower case, mapping of synonyms to a common form, reordering the function into their calling order, and removal of all tokens that are not from the lexicon of the target language. In the second phase, which is a comparison phase, different versions of YAP use different algorithms. The original version of YAP is based on the UNIX utility "sdiff". YAP2, which was implemented in Perl, uses Heckel's algorithm. Yap3 is the latest version in YAP series, and uses an algorithm called Running Karp-Rabin, Greedy String Tiling (RKR_GST).

- **JPlag** JPlag can find plagiarism in source code written in Java, C, C++ and Scheme. JPlag, also, works in two phases. In the first phase programs to be compared are parsed, depending on the input language and converted into token strings. In the second phase, these token strings are compared in pairs for determining the similarity of each pair. During each such comparison, JPlag attempts to cover one token stream with substrings ("tiles") taken from the other as well as possible. The percentage of the token streams that can be covered is the similarity value. The matching step (phase 2) consists of two more phases. In phase 1, the strings are searched for biggest contiguous matches using three nested loops. The first one iterates over all the tokens in the first string. These nested loops collect the set of all longest common substrings. The second one compares this token with

every token in the second string. If they are identical, the innermost loop tries to extend the match as far as possible. In phase 2, all matches of maximal length found in phase 1 are marked. This means that all the tokens are marked and thus may not be used for further matches in phase 1 of subsequent iteration. The two phases of the matching step are repeated until no further matches are found or a lower bound for length, called "Minimum Match Length" is met. JPlag requires download of a Java program to the client. It requires the Java virtual machine (runtime environment) to be present for the client application to work. The application is a Java applet that provides a Graphical User Interface (GUI), which allows the users to browse their file system to submit the files. JPlag supports files written in C, C++ and Java. and the 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 [19]. 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.

- **SID**, which stands for Shared Information Distance or Software Integrity Detection, detects similarity between programs by computing the shared information between them. It was originally an algorithm developed for comparing how similar or dissimilar genomes are. It was later extended to other applications like finding plagiarism. SID finds plagiarism by computing the amount of shared information between two programs as follows:

$$D(x,y) = \frac{1 - K(x) - K(x/y)}{K(xy)}$$

Where K(x/y) is the Kolmogorov complexity of x given y. However, since the Kolmogorov complexity is not computable, SID uses a compression algorithm to approximate Kolmogorov complexity. SID also works in two phases. The first phase involves parsing the source programs to generate tokens. In the second phase an algorithm named Token Compress is used which computes heuristically the shared information metric D(x,y) between each program pair submitted. Then, all the program pairs are ranked by their similarity distances. SID can detect plagiarism in source code written in Java and C/C++. SID is a web based service. Users can submit the files in a compressed (zip) format. The user has to make separate zip files for source code files written in different programming languages. After processing the files, SID sends an email to the users to inform them that the results are ready to be viewed on the internet. Users need to log in to the SID web site to see the results.

- **MOSS** MOSS stands for "Measure Of Software Similarity" and was developed by Alex Aiken I at UC Berkeley in 1994. It is accessible on the Internet at http://theory.stanford.edu/~aiken/moss/. MOSS employs a document fingerprinting technique to detect textual similarity. It first extracts significant words or phrases from the documents under scrutiny, by applying whitespace sensitivity and noise suppression. This is done by ignoring noise data such as comments, whitespaces, capitalization and punctuation marks. Noise suppression also removes short or common words that are likely to complicate the comparison, such as "the", "a". Whitespace characters are hidden control characters, such as blanks, tabs, newline, carriage-return. Whitespace sensitivity and noise suppression leaves the strings that are used for comparison unaffected. After the documents are clean of noise, MOSS combines all text in the document together and divides them into small sub-strings, or k-grams. The length of k-gram is the number of alphabets in each sub-string and is individually defined by each user. Next an index number representing each sub-string is added to each document using a hashing function. Finally, the sequences of index numbers of the two documents are compared to find similarity between the two documents.

To use MOSS, a Perl script needs to be downloaded by the user. This script is used for submitting files to the server and for displaying some server responses at command prompt. MOSS displays results via a web page. Files are submitted by listing them as parameters to the command line. After submission of the files is complete, MOSS gives a response at the command prompt giving the URL of the web page where results can be viewed. MOSS presents the results as a list of ordered pairs with the matching percentage of each file in the pair and the number of lines matched. The matched results are followed by a list of errors that the program encountered during processing. For both measures, higher numbers mean more overlap and a higher probability of plagiarism.

Table 2: Features of Existing Source Code Plagiarism Detection Systems

	MOSS	JPlag	SID	YAP
Algorithms Used	Fingerprinting technique	Greedy String Tiling	Computes shared amount of information using Kolmogorov's complexity	Running-Karp_Rabin Greedy-String-Tiling
Language Supported	C, C++, Java. C#, Fortran, etc.	C, C++, Java. C#, scheme	C, C++, Java	Pascal, C, LISP
Platforms for Client	Unix	Java program. Requires Java run time applicable to the platform	Web-based	Unix
File Submission Mode	Command line	Java application	Web form	Command line
File format for submission	Files as parameter to Perl executable	Files by specifying folder in UI	Zip file	By specifying folder as parameter to Perl executable
Language Specification	Required	Required	Required	Use appropriate to kenizer
Results	Email with URL	HTML page	SID site	Written to file

PROPOSED SYSTEM

According to what has been discussed in the study above, we propose ZPlag which is a system for plagiarism detection in electronic resources. In other words, ZPlag is 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. The tokenization technique will be used for detecting plagiarism in code. Also, the simple algorithm will be used for comparing documents and code. The image vector features representation will be considered as the main issue when detecting plagiarism in images. The framework will be published in another paper.

CONCLUSIONS

In this paper, a study of 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 for plagiarism detection systems become very important issues and using them in E-Learning improves academic integrity.

Also, the instances of plagiarism can be greatly reduced, if not eliminated, with the use of plagiarism detection systems. Authors propose a system (ZPlag) that is able to detect many plagiarism attempts in different fields (E-Learning, E-Business, and E-Journals) and can be used to evaluate programs and papers with images included, and, therefore, increasing the quality of its design and maintaining high educational standards.

REFERENCES

1. M. Ali, H. M. Abdulla, and V. Snasel "Survey of Plagiarism Detection Methods", Fifth Asia Modelling Symposium, 2011.
2. F. Sanchez-Vega, E. Villatoro-Tello, M. Montes-y, L. Villase, P. Rosso, "Determining and characterizing the reused text for plagiarism detection" , Contents lists available at SciVerse ScienceDirect , Expert Systems with Applications 40 (2013) 1804–1813.
3. G. Oberreuter, and J. D. Velȗsquez, "Text mining applied to plagiarism detection: The use of words for detecting deviations in the writing style", Contents lists available at SciVerse ScienceDirect, Expert Systems with Applications 40 (2013) 3756–3763.
4. *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.*
5. U. Bandara, and G. Wijayarathna, "Source code author identification with unsupervised feature learning", Contents lists available at SciVerse Science Direct, Pattern Recognition Letters 34 (2013) 330–334.
6. *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.*
7. *M. Menai,and M. Bagais, "APlag: A Plagiarism Checker for Arabic Texts" The 6th International Conference on Computer Science & Education (ICCSE 2011), IEEE 2011.*
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. *C. J. Neill, and G. Shanmuganthan, "A Web-enabled plagiarism detection tool", IT Professional, 2004. 6(5): p. 19-23.*
10. *S. M. Alzahrani, N. Salim, "Statement-based fuzzy-set IR versus fingerprints matching for plagiarism detection in Arabic documents," In Proc. of the 5th Postgraduate Annual Research Seminar (PARS09), Johor Bahru, Malaysia, 2009.*
11. J. Y. Kuo, F. C. Huang, C. Hung, and L. H. Z. Yang, "The Study of Plagiarism Detection for Object-oriented Programming" Sixth International Conference on Genetic and Evolutionary Computing IEEE 2012.
12. *A. H. Osman, N. Salim, M. S. Binwahlan, S. Twaha, Y. J. Kumar and A. Abuobieda, "Plagiarism Detection Scheme Based on Semantic Role Labeling" , 978-1-4673-1090-1/12/ IEEE 2012.,*
13. *M. Joy and M. Luck, "Plagiarism in Programming Assignments," IEEE Transactions of Education, 2009*
14. F. Jian "Design and Implementation of a Kind of Word Document Plagiarism Detection Method ", International Conference on Computer Science and Information Processing (CSIP), IEEE 2012.
15. *M. J. Wise, "Detection of Similarities in Student Programs: YAP'ing may be Preferable to Plague'ing," ACM SIGSCE Bulletin (proc. Of 23rd SIGCSE Technical Symp.), 2002.*

16. *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.*

17. *X. Chen, B. Francia, M. Li, B. McKinnon, and A. Seker, "Shared Information and Program Plagiarism Detection", IEEE Transactions on Information Theory, vol. 50, pp.1545-1551, 2004*

18. *Schleimer, Saul, Daniel Shawcross Wilkerson, and Alexander Aiken, "Winnowing: Local Algorithms for Document Fingerprinting," SIGMOD 2002, 76-85.*

19. *M. J. Wise, "String Similarity via Greedy String Tiling and Running Karp-Rabin Matching," Department of Computer Science, University of Sydney, Australia, 2003.*