Student Paper Comparison System using Kolmogorov Complexity and Diff Algorithm

Marco DEL ROSARIO Jr*

Laguna State Polytechnic University, San Pablo City, Laguna, Philippines

(*Corresponding author's e-mail: macky.delrosario@lspu.edu.ph)

Abstract

In the academe, students sometimes rely on copying each other's works or previous works in order to meet requirements. This action leads to plagiarism, which is becoming part of academic institutions concern in reducing growing academic dishonesty. Though commercialized plagiarism checkers are available, some are not designed to compare submitted papers while some only allows to compare two up to documents. With regards to such issue, this study aims to design and develop a comparison system for students' papers that is capable of uploading multiple documents and calculating the similarity between the documents. Moreover, the application was created using HTML, CSS, JavaScript, PHP, and MySQL. The developed system consists of two main modules: The Document Upload which allows users to upload multiple documents for comparison and manage the stored documents, and the Document Comparison, which serves as the mechanism of the system for detecting possible plagiarism. An algorithm based on Kolmogorov Complexity was utilized to measure the similarity between the documents while the Diff Algorithm was used to highlight the parts of the document that is suspected to be plagiarized. Furthermore, tests were performed to verify if the system is functioning as expected and to measure the accuracy of the output from the system. Eighty-one respondents evaluated the developed system using the ISO 25010 software quality model in terms of the Product Quality. An overall mean of 4.63 which can be interpreted as "Excellent" in descriptive terms validates that the objectives of the study were met and achieved, and further indicates that the system was developed according to its desired functions and requirements.

Keywords: Plagiarism; Diff Algorithm; Kolmogorov Complexity Algorithm; Normalized Compression Distance; Comparison System

Introduction

Reynolds (2010) expressed that Plagiarism is the deed of stealing somebody's concepts or words and claim them as one's own. As agreed by Helgesson and Eriksson (2015), the most common definition of plagiarism is made out of two sections. First of it is that to suit the work of another person. Second is taking it as one's personal work and not giving appropriate recognition. In a different description, plagiarism can be defined as stealing. Plagiarism is the act of utilizing someone else's words or thoughts without offering credit to that particular individual. In other words, the deed of taking another person's ideas and passing them off as one's personal idea denotes the concept of "plagiarism". As it appears in the growing educational concerns, plagiarism has now turned into an indispensable character of an individual digital life (Del Rosario, 2018). Helgesson and Eriksson (2015) also concluded that in all academic institutions, plagiarism is a well-acknowledged and a rising issue. Del Rosario (2018) stated that technology, specifically the internet, contributes greatly to sharing resources and information around the world. Access to the information available on the internet leads to the increase of threat of plagiarism. Regrettably, current technologies are not being responsible for the proper protection of intellectual properties.

In an educational institution, one of the most common requirements submitted by the students are papers, in forms of research papers, narrative reports or term papers. These papers contain the ideas of a student in a certain topic. Most students tend to copy prepared solution available on the internet. Most students are unaware that their actions of copying and duplicating other person's intellectual property is not just unethical but actually a crime. Conveniently, there are plagiarism detection tools available. However, these tools allow a user to compare their document to the document saved in the system that is mostly available online (Del Rosario, 2018). Having said that, there are situations wherein students tend to submit papers copied from their classmates. According to the study conducted by Eya (2007), even academic institutions suffer from the effects of plagiarism. She stated that it is proven that course instructors tend to be frustrated with the issues of plagiarism. Students submit their copied papers to their faculty without knowing that they are plagiarizing. At the same time, most faculty members fall short in screening these research papers. They should consider if these are the students' original ideas. In relation to this, the developed system will be used to identify similarities between submitted documents from the students. Though, there are plagiarism checkers available commercially, these software limits its function by comparing documents against their own database. One example is Turnitin. This is an electronic text matching system that is utilized to calculate the similarities between students' submitted work and existing electronic sources (Holy Spirit University of Kaslik). However, Turnitin (and iThenticate) are not intended to compare two submitted papers (Pinto, 2017). Nonetheless, there are other software available online that compare two papers such as Copyleak which is described itself as "*advanced cloud platform for tracking and*

tracing content online" (Copyleaks, 2019). The software can compare two text documents, but this software allows user to compare only two documents at a time. Comparing multiple documents will be exhausting and time consuming.

The study aims to develop a student paper Comparison System. Moreover, the study specifically aims to determine if Kolmogorov Complexity and Diff algorithm will be efficient to the Comparison System and it also intends to design and construct the Comparison system. The system should be able to upload multiple student paper/file; calculate the similarity of a document to discover any act of possible plagiarism; identify parts of the document that are detected with possible plagiarism. The system will be tested with regards to the functionality and accuracy of the developed system. It will be evaluated adopting the ISO 25010 software quality using the product quality composition as the evaluation tool.

Materials and methods

The system will be designed using two algorithms. One is the Diff Algorithm and one based on the Kolmogorov Complexity.

Kolmogorov Complexity. In the study entitled "Text comparison using data compression" (2013), the algorithm used to detect similarity between documents is the Kolmogorov Complexity. Kolmogorov Complexity is one of the perfect measures for computation of the similarity of two strings in a defined alphabet. However, the algorithm is incomputable as stated by Platos, Prilepok, and Snasel (2013). As what can be seen in the study of Wortel (2005), who also developed a system that is capable of plagiarism detection, he applied the Normalized Compression Distance as the measure. Wortel's method is centered on the theoretical concept of Kolmogorov Complexity.

The Kolmogorov Complexity is represented in this discussion as K(x). Kolmogorov Complexity is a measure for the amount of absolute information in string x. K(x) is identified as the length of the shortest binary program with no input that outputs x. While, K(x|y) represents the conditional Kolmogorov Complexity or the Kolmogorov Complexity of x given y is the length of the shortest binary program that on input y outputs x. According to Platos, Prilepok, and Snasel (2013), Information Distance is derived using the Kolmogorov Complexity. The Information Distance represented as E(x, y) is defined as the shortest binary program that, with input x computes y, and with input y computes x. The normalized version of E(x, y), called the Normalized Information Distance (NID), is defined in Figure 1 as:

$$NID(x,y) = \frac{\max\{K(x|y), K(y,x)\}}{\max\{K(x), K(y)\}}$$

Figure 1. *Normalized information distance.*

This NID is a metric and has the property that if two files are similar according to the particular feature described by a normalized computable distance, then they are also similar in the sense of the NID. Unfortunately, the Kolmogorov Complexity is not computable, which makes it impossible to use the NID in practice (Platos, Prilepok, and Snasel, 2013). Vitanyi et al (2008) agreed that NID is based on Kolmogorov Complexity and thus incomputable. However, it can be utilized if the objects have a string representation by using compression algorithms to estimate the Kolmogorov Complexity. As agreed by Platos (2013), the approximation of Kolmogorov Complexity can be done by using compression. The idea is that a general purpose compressor detects as much regularity in as wide a range of files as reasonably possible.

$$NCD(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

Figure 2. *Normalized compression distance.*

As shown in Figure 2, an approximation of the NID using a real compressor C is called the Normalized Compression Distance (NCD). Where C(x) signifies the compressed size of string x, and xy signifies the concatenated strings x and y. This NCD is the real-world version of the ideal notion of the NID, with K(x|y) approximated as C(xy)−C(y). The NCD was proved to be a dissimilarity metric under some reasonable assumptions on the compressor.

Diff Algorithm. A string matching algorithm is considered to be important to all and each computer user. For an instance of altering a text, the user is likely to processes the text. These instances require an algorithm in order to achieve the desired objective. Despite that fact, the longer the string that is being matched with a pattern, consideration of the efficiency of a searching algorithm becomes very important (Drozdek, 2010).

According to Butler (2007), in the most basic form of a diff algorithm takes two strings, and returns the changes needed to make in the old string to the new one. They are useful in comparing different versions of a document or file, to see at a glance what the differences are between the two. Wikipedia, for example, uses diffs to compare the changes between two revisions of the same article.

Butler (2007) stated in his article that it is not as simple as it seems in solving the problem. He managed to formulate his algorithm using PHP, 18 lines of code algorithm. It is not the most efficient way to do a diff, but it is probably the easiest to understand. The algorithm works by identifying the longest sequence of words common to both strings, and recursively finding the longest sequences of the remainders of the string until the substrings have no words in common. At this point, it adds the remaining new words as an insertion and the remaining old words as a deletion.

Project Design. The project entitled "Student Paper Comparison System using Kolmogorov Complexity and Diff Algorithm" is a system that will help to resolve student related plagiarism issues/problems in an academic institution. The context diagram for the system is shown in Figure 3. It shows the entity of the user and the main process which corresponds to the developed system. The data flowing in the process are the Students' Paper or documents. After the main process, the data flowing out will be the Analyzed result or simply the similarity report generated by the system.

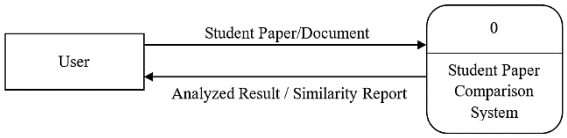

Figure 3. *Context diagram.*

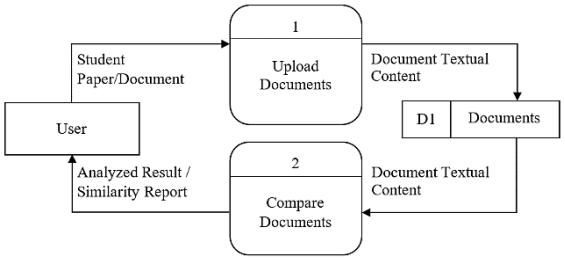

Figure 4. *Level 0 DFD.*

As mentioned in the earlier part, the system has two main modules, the Document Upload and Document Comparison module. Level 0 DFD shown in Figure 4. The user must upload the student paper/documents into the system. It allows multiple upload of documents in both Word Document (.doc or .docx) or Portable Document Format (.pdf). After the documents where uploaded, the system will remove non-textual content and unnecessary text content on the documents. Non-textual contents are the images or tables while the unnecessary texts are commonly used words like punctuation marks and articles (a, an, the). The stripped contents will then be stored in the system's database. As seen in Level 1 DFD for the Upload Process in Figure 5.

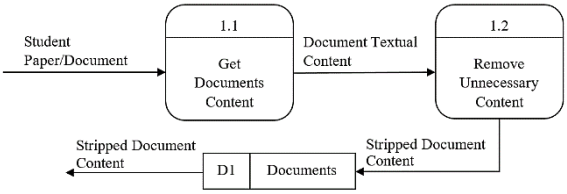

Figure 5. *Level 1 DFD for the upload process.*

After the contents were stored in the system's database, the Comparison process will be executed. The comparison process was divided into two sub-processes, namely: Compute Similarity process and Identify part process. As shown in Level 1 DFD in Figure 6.

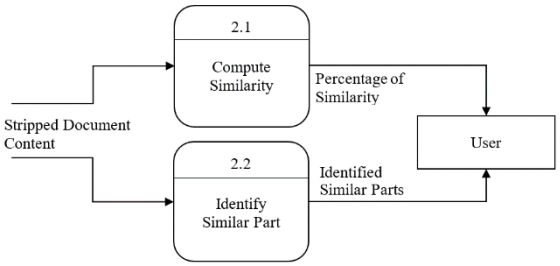

Figure 6. *Level 1 DFD for the comparison process.*

The Kolmogorov Complexity algorithm used to calculate the similarities between documents. An output of the computed percentage of similarity will be given to the user. On the other hand, the system used the Diff Algorithm to identify parts of the document that are similar.

As shown in Figure 7, the pseudocode for identifying the similarity between documents, the system will accept the submitted document and document from the database and store the string in an array variable old and new respectively. The array variable will separate every word in the document. It will search for the identical words from the variable old in the variable new. Once a match is found, it will be marked as identical and will be included in the matrix variable, which contains both the matched and the mismatched words.

```
function diff($old, $new)
    $matrix = array();
    $maxlen = 0;
    foreach($old as $oindex => $ovalue)
        $nkeys = array_keys($new, $ovalue);
        foreach($nkeys as $nindex)
                $matrix[$oindex][$nindex] = isset($matrix[$oindex - 1][$nindex - 1]) ?
                    $matrix[$oindex - 1][$nindex - 1] + 1 : 1;
            if($matrix[$oindex][$nindex] > $maxlen)
                $maxlen = $matrix[$oindex][$nindex];
                $omax = $oindex + 1 - $maxlen;
                $nmax = $nindex + 1 - $maxlen;
```

Figure 7. *Pseudocode for identifying similarity.*

```
function htmlDiff($old, $new)
    $ret = '';      # will hold the output string
    $x = 0;         # variable counter for number of words
    $y = "";        # temporary storage
    $diff = diff(preg_split("/[\s]+/", $old), preg_split("/[\s]+/", $new));

    foreach($diff as $k)
        if(is_array($k))
            $ret .= "<mark>". $y . ' '."</mark>";   # highlight the return words
            $x = 0;
            $y = "";
            $ret .= (!empty($k['d'])?"".implode(' ',$k['d'])." ":'');
        else
            $x++;
            $y .= $k.' ';
    return $ret;    # output the words
```

Figure 8. *Pseudocode for highlighting similarity.*

After that process, the system will generate an analyzed result that will be presented to the user. The Kolmogorov Complexity Algorithm using the Normalized Compression Distance (NCD) is used to measure the percentage of similarities between the documents. NCD formula is illustrated in Figure 9. The variables sx and sy stands for the content of the uploaded documents. Both documents were compressed. At the same time, variables min and max were initialized using the value of variable x and y, respectively. The combination of both strings will also be compressed and stored in variable xy.

```
function NCD($sx, $sy)      # sx and sy are the strings to compare

    $x = $min = strlen(compress($sx));      # compress document 1

    $y = $max = strlen(compress($sy));      # compress document 2

    $xy = strlen(compress($sx.$sy));        # concatenate strings

    if ($x > $y )           # if x is shorter that y, swap min/max
        $min = $y;
        $max = $y;

    $res = ($xy - $min) / $max;             # NCD Fomula

    return 100 * $res;      # transform result to percentage
```

Figure 9. *Pseudocode for similarity percentage.*

There is a condition that determines which string is shorter. If the content of y is shorter than x the value of min and max will be swapped else, the value of min and max is retained. Then, the formula for getting the NCD will be performed. The combination of x and y will be deducted by the value of the shorter string divided by the value of the longer string. Then, the result will be multiplied by 100 to get the percentage of similarity between documents using the Normalized Compression Distance.

Results and discussion

Test Results

The developed system underwent the process of testing to ensure its quality in accordance with functionality and accuracy. The first testing conducted was the ***Functionality Test***. This testing checks if all the functions of the system are working and giving correct output. Multiple test scenarios and test cases were performed. Table 1 shows the list of all test scenarios performed and the number of test cases for each scenario.

Table 1 *Summary of result of functionality test*

Functionality	Number of Test Cases	Remarks
1. Login / Logout	5	5 Passed, 0 Failed
2. Document Upload	4	4 Passed, 0 Failed
3. Document Comparison	4	4 Passed, 0 Failed
4. User Management / User Logs	9	9 Passed, 0 Failed
5. System Settings	5	5 Passed, 0 Failed

Legend:

Passed – indicates that the actual result of the test meets the expected result.

Failed – indicates that the actual result of the test was different from the expected result.

There were 5 test scenarios performed in the conduct of the testing. Correspondingly, a total of 27 test cases were performed to determine the functionality of the system. Each test case was conducted aiming to verify if every function

that the system is performing is conforming to the goals and requirements of the system. The last column in the table title “Remarks” indicates the number of passed and failed test cases.

In this discussion, the testing to be discussed are the two main functions or modules of the system, namely, the document upload, and document comparison.

Table 2 *Document upload test case*

Test Scenario	Document Upload	**Remarks**	Passed
Test Case	Select a file with valid file format		
Test Steps			
1. Click document compare tab			
2. Enter topic			
3. Choose a document to upload			
4. Click upload			
Expected Result		**Actual Result**	
Document upload must be successful		Document upload successful	
Postcondition			
Time and Date of upload is stored in the database. Display that the document upload was successful			

Table 2 shows the test case table of the document upload module by selecting valid file/s. This test case was performed to verify if the system was able to produce the expected result which is to upload the all selected files successfully and store it and its information in the database. As seen in Table 2, the document upload test case has 4 test steps. These steps are click Document Compare tab, enter a topic, and select the document file to be uploaded then click the register button. This test case expects valid information and valid document with correct file format as inputs. A result of successful upload of a document is anticipated. After the execution of the test step, the system was able to successfully upload the submitted document. For that reason, the test reflects that the function was working properly and that the system passed the test.

Table 3 *Document comparison test case*

Test Scenario	Document Comparison	**Remarks**	Passed
Test Case	Select A File With A Valid File Format		
Test Steps			
1. Click document compare tab			
2. Enter topic			
3. Choose a document to upload			
4. Click upload			
Expected Result		**Actual Result**	
Document comparison process must be performed successfully		Document comparison was successfully	
Postcondition			
Time and Date of Comparison is stored in the Database. Display document comparison result.			

The test case for the Document Comparison module of the system is shown in Table 3. This test was performed in order to identify if the developed system was able to perform the plagiarism mechanism and generate a result. The accuracy of the result shall be discussed in the subsequent section. In this test, successful execution of the document comparison process was the anticipated result of the testing. The following steps were performed in order to determine if the system meets the desired outcome. Click the document comparison tab, enter a topic, and select a file to be uploaded then click the compare button. After conducting the steps, an actual outcome of successfully performing the document comparison arrived. This means the test in using document comparison module passed. In addition, a postcondition of storing the time and date of comparison was successfully performed as well as displaying the document comparison result. This further indicates that the system was able to perform the comparison by using the Normalized Compression Distance Algorithm which is based on Kolmogorov Complexity and Diff Algorithm.

Accuracy Testing. This test assessed the system on the basis of the correctness of the outputs. Series of tests were conducted to measure the accuracy of the outputs of the developed system. The purpose of these series of tests was to measure the system's capability in detecting similarities between the submitted documents. There was a total of 9 documents used as specimen samples in three set of tests. For each set of tests, 3 documents were used. Table 4 shows each documents' name and its characteristics. The first set of documents are the original documents in which the other documents were compared upon. The second set is the document combining the original documents. Then the third set is the document copied from the original documents.

Table 4 *Accuracy test document distribution*

Document Name	Remarks
A	Original Document
B	Original Document
C	Original Document
AB	Combined A and B
AC	Combined A and C
BC	Combined B and C
D	Copied from A
E	Copied from B
F	Copied from C

Three separate tests were administered, matching documents A, B, and C against itself in every test. The documents were uploaded to the system. The first set of tests aims to identify the system's capability to distinguish identical documents. The expected similarity result is 100%. As shown in Table 5, the average of the actual similarity results for all three tests is 100%. This means that the system was able to detect identical documents.

Table 5 *Accuracy test set 1*

Registered Document	Compared Document	Expected Result *	Actual Result *
A	A	100%	100%
B	B	100%	100%
C	C	100%	100%
* - refers to the similarity percentage generated by the system.			

Determining if the system could identify a document that was combined with the original document is the purpose of the second set of tests. Six independent tests were conducted, comparing the non-combined documents against the documents AB, AC, and BC in every test. The result of comparing the combined documents may be easily assumed of 50%. However, the similarity percentage of 50% was not the expected result for this test since the exact number of words or the exact number of plagiarized parts in the document is not known. A result falling within a similarity threshold between 30% to 70% as an approximate range was expected instead of 50%.

Table 6 shows the result of the test in determining the accuracy of the system if two documents are combined. The average similarity result of all three tests falls between indicated similarity threshold of 30%-70%. This means that the system was able to detect a similarity between the combined documents.

Table 6 *Accuracy test set 2*

Registered Document	Compared Document	Expected Result	Actual Result
A	AB	30% - 70%	45%
A	AC	30% - 70%	53%
B	AB	30% - 70%	65%
B	BC	30% - 70%	63%
C	BC	30% - 70%	47%
C	AC	30% - 70%	58%
* - refers to the similarity percentage of the system.			

The last set of test intends to determine if the system may possibly recognize actual copying which is the primary aim of the system. Three independent tests were conducted, comparing the document A against D, B against E, and C against F. In this test, the expected result may change depending on the value of the similarity threshold of the documents. Higher similarity threshold is expected if there is a higher number of plagiarized words or phrases in the document. In this test, the anticipated similarity threshold should be greater than 30%. A result having greater that 30% specifies that the documents being compared against has the possibility of actual copying or plagiarism. As shown in Table 7, the similarity result for all three tests is greater than 30% (assumed similarity threshold). This means that the system was able to detect actual copying activities in the uploaded document.

Table 7 *Accuracy test set 3*

Registered Document	**Compared Document**	**Expected Result ***	**Actual Result ***
A	D	> 30%	74%
B	E	> 30%	44%
C	F	> 30%	59%
* - refers to the similarity percentage of the system.			

Series of tests have been described to demonstrate the accuracy of the system. Even though the results of the tests are particular with respect to the testing set of documents, it could explained the general behavior of the system. It shows that the system was able to produce an acceptable and accurate measurement of similarity between the test documents.

Evaluation Result

This system was subjected to evaluation with 81 respondents consisted of IT practitioners and students from the Laguna State Polytechnic University. This study adopted the ISO 25010 software quality model in terms of the Product Quality composition as an evaluation tool with the following criteria: Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability. All the respondents' ratings were consolidated and computed to get its quantitative and qualitative interpretation. The interpretation was based on the range of scale value for interpreting the evaluation result that can be seen in Table 8.

Table 8 *Range of scale value for interpreting the evaluation result*

Range	Interpretation
4.21 – 5.00	Excellent
3.41 – 4.20	Very Good
2.61 – 3.40	Good
1.81 – 2.60	Fair
1.00 – 1.80	Poor

Table 9 summarizes the evaluation results from the respondents' ratings showing the mean per criterion and the corresponding qualitative interpretation. The table also presents the overall mean by getting the average of all the means of the eight criteria.

Table 9 *Results of respondents' ratings of the system*

Criteria	Mean	Qualitative Interpretation
Functional Suitability	4.68	Excellent
Performance Efficiency	4.66	Excellent
Compatibility	4.65	Excellent
Usability	4.59	Excellent
Reliability	4.53	Excellent
Security	4.63	Excellent
Maintainability	4.64	Excellent
Portability	4.62	Excellent
Overall Mean	**4.63**	**Excellent**

The respondents of the study agreed that the system has an excellent rating in terms of functionality suitability which indicates that the set of functions covers all the specified tasks and user objectives. It obtained a mean of 4.68 which indicates the it has “Excellent” interpretation in terms of quality.

In terms of the Performance Efficiency, the system attained a mean of 4.66 which is equivalent to "Excellent" in the descriptive term. The respondents also agreed that it is performing its functions to meet the required response, processing times and throughput rates. Correspondingly, the system meets the required types of resources to be used as well as meeting the maximum limit of the system.

While in terms of compatibility, it was deemed “Excellent” in terms of qualitative interpretation for its capability to run in different web browsers. This points out that the developed system can perform its required functions efficiently while sharing a common environment and resources with other system, without inconvenient effect on other system. In addition, the system or its components can exchange information and use the information that has been exchanged. It garnered a mean of 4.72 from the respondents.

It was also agreed that the system was “Excellent” with regards to its Usability criterion wherein users were able to recognize that it was appropriate for their needs and was easy to operate. A mean of 4.59 indicates that the system protects the users against errors by giving instructions and warnings. It also gives a pleasing user interface and satisfying interaction for the user.

An “Excellent” rating in terms of qualitative interpretation was also given in the reliability criterion achieving a mean of 4.53. This indicates that under normal operation, it was operational and accessible. The system operates as intended despite the presence of hardware or software faults, and it can recover the data directly affected and re-establish the desired state of the system whenever there is an interruption or a failure.

Moreover, the system ensures that there are appropriate authentication, authorization and access control in the system’s Security. This means that the developed system ensures that the data are accessible to those authorized with access, as well as preventing others access to, modification of data and storing records. Traceability of the user’s action is also present in the system. It attained a mean of 4.63.

The respondents further agreed that the system was maintainable and can easily be treated in case of failure. This means that the system is composed of discrete components such that a change to one component has minimal impact on other components. The asset can be used in more than one system. It is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified. The system can be effectively and efficiently modified without introducing defects or degrading existing product quality and test criteria can be established for a system, product or component and tests can

be performed to determine whether those criteria have been met. It obtained a mean of 4.64 which is interpreted as "Excellent".

Lastly, in terms of Portability, they agreed that the developed system adapts to a different platform, hence, it can be easily installed in a specified environment. The developed system attained a mean of 4.62 which can be interpreted as "Excellent" in descriptive term.

Among the different criteria, the Functional Suitability criterion obtained the highest mean, while reliability got the lowest mean but still falls within the range of the scale value of "Excellent". The overall mean generated for all the criteria contained in ISO 25010 software quality model in terms of the Product Quality composition yielded an average of 4.63 which validates that the system has attained its anticipated functions according to the requirements, and indicates that the system is "Excellent".

Description of the Software Developed

This study led to the development of the project which is an information system that functions as a comparison system of submitted student papers (Figure 10-a). The project caters the user the basic features and functionalities of an information system allowing them to upload, view, and maintain the document records (student papers) kept in the database (Figure 10-b).

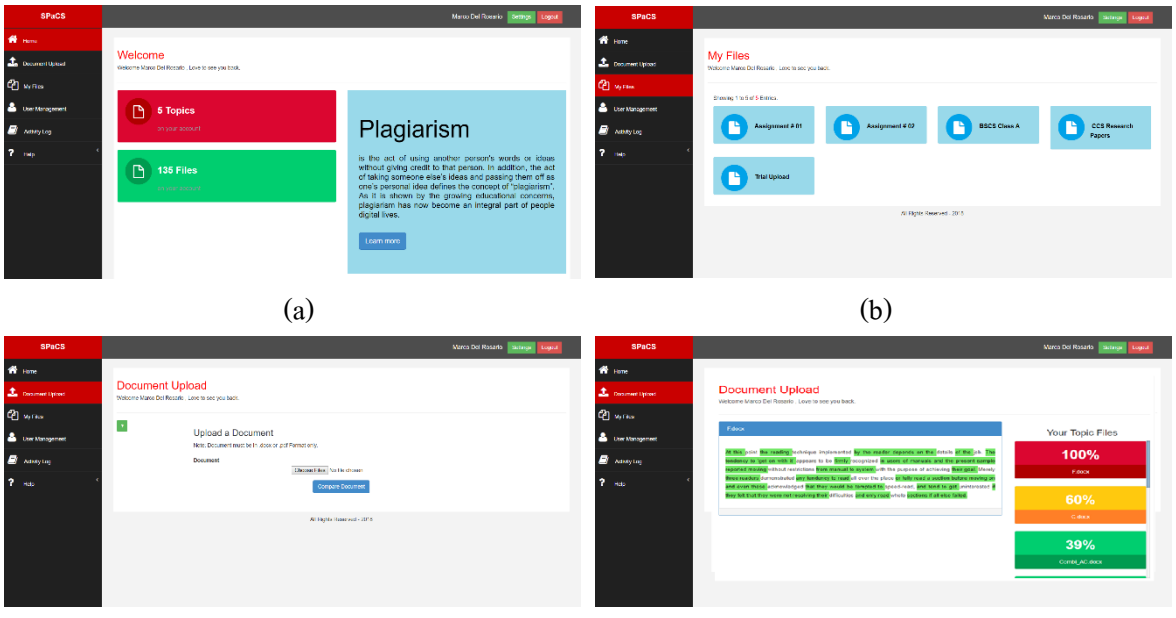

Figure 10. *(a) home screen (b) document topics (c) document comparison (d) document comparison result*

The system has the capability to detect similarity between documents using the document comparison module (Figure 10-c). This feature prompts the user to submit multiple documents which will be compared against each other. After comparing, the user will be given a similarity result. This result indicates the percentage of similarity measured using the Normalized Compression Distance or Kolmogorov Complexity algorithm. Furthermore, a user can view the similarity between documents by prompting the system to highlight the words, phrases or portions of the document that are found identical using the Diff Algorithm. However, the system was not able to compare document images, tables, and other non-textual content.

In addition, the system has the provision of showing the log that records a user activity. The system also enables the administrator to manage the user information. It permits to add a new user and edit their information as well as reset a password.

Conclusions

On the basis of the findings presented in the earlier section, the developed system was created according to the intended design, algorithm, and functionalities. Therefore, it was concluded that this study was able to successfully develop a student paper Comparison System using Kolmogorov Complexity and Diff algorithm. This indicates that the algorithm used was able to perform the necessary requirement of identifying similarities between student papers/document. It allows a user to upload multiple file document and then calculate the similarity and identify suspected copied parts of a document. The developed system was tested and successfully meets the expected outcome in terms of system functionality and accuracy. The functions and features of the system were assessed using 27 test cases covering all the functions present in the system. The accuracy of the system was measured using series of test to prove that it was able to produce an acceptable and accurate measurement The performance of the developed system was evaluated using the ISO 25010 by 81 respondents and was rated as “Excellent” which proves that the system is beneficial to an academic institution. The current study provided evidence that a system can be developed to perform a comparison between submitted documents from the students and proves its significance and usefulness to an academic institution. However, there are certain limitations that the current study did not take into consideration. It is unclear whether there is a better or more efficient string matching algorithm to be utilized in detecting similarities. Moreover, it is unknown if the system will be able to perform effectively and efficiently if there are more users using it simultaneously.

Several suggestions and further modifications were determined based on the respondents’ evaluation; 1) make the system compare documents image, tables, and other non-textual content and allow it to accept different file formats, 2)

enable it to highlight the similarity of multiple documents in one viewing, 3) allow the downloading of the highlighted document, 4) provide a graphical representation of the data as well as produce analysis of the data in the system and 5) include the functionality to detect grammatical errors and suggests the correction to the documents.

Acknowledgments

The author is indebted to the College of Computer Studies, to the respondents of the study, and to the student and faculty of the Laguna State Polytechnic University, San Pablo City, Laguna, Philippines.

References

[1] G. Helgesson and S. Eriksson, "Plagiarism in Research", Springer Science & Business Media (2015).

[2] W. Badke, "Training Plagiarism Detectives: The Law and Other Order Approach", onlinemag.net, 31, 6, 50-52 (2007).

[3] M. Del Rosario Jr, "Development of Capstone and Theses Plagiarism using Kolmogorov Complexity Algorithm", Technological University of the Philippines, Manila (2018).

[4] G. W. Reynolds, "Principles of Ethics in Information Technology", Giuani Prints House, Manila (2010).

[5] J. D. C. Eya, "Development of Plagiarism Detector for the Family of C Program Source Codes", Technological University of the Philippines, Manila (2007).

[6] P. C. R. Lane, C. Lyon and J. A. Malcolm, "Demonstration of the Ferret plagiarism detect", In: Proceedings of the 2nd International Plagiarism Conference (2006).

[7] C. Lyon, P. Lane, W Ji, J. A. Malcolm and J. Bao, "Copy Detection in Chinese Documents using the Ferret: A Report on Experiments" (n.d.).

[8] A. Bugarín, M. Carreira, M. Lama and X. M Pardo, "Plagiarism Detection using Software Tools: A Study in a Computer Science Degree", University of Santiago de Compostela (n.d.).

[9] Holy Spirit University of Kaslik Library, "Turnitin@USEK: Detecting and Deterring Plagiarism in Students' Work Instructor's Manual" Drafted (2014).

[10] Pinto, G. Kenneth, Can I compare two papers directly to each other? (Instructor), Retrieved from https://wiki.nus.edu.sg/pages/viewpage.action?pageId=101817481 (2019).

[11] Copyleaks: Compare Documents, Retrieved from https://copyleaks.com/compare (2019).

[12] A. Patel, K. Bakhtiyari and M. Taghavi, "Evaluation of cheating detection methods in academic writings", Library Hi Tech, 29, 4, 623-640 (2011).

[13] Interpreting Turnitin Originality Reports, Retrieved from

https://eat.scm.tees.ac.uk/bb8content/resources/recipes/interpretTurnitin.pdf (2017).

[14] The University of the West Indies at St. Augustine, "Guidelines for Staff and Students on Plagiarism", Trinidad and Tobago (2010).

[15] L. Prechelt, G. Malphol and M. Philippsen, "JPlag: Finding Plagiarisms among Set of Programs", Technical Report 2000-1, University of Karishuhe (2000).

[16] Application Programming Interface, Retrieved from http://www.basicknowledge101.com/pdf/km/Application%20programming%20interface.pdf (2017).

[17] J. E. Estebal, "Development of Tracer Study and Management System for LSPU-SPC", Technological University of the Philippines, Manila (2012).

[18] K. Mehlhorn and P. Sanders, "Algorithms and Data Structures: The Basic Toolbox", Springer (2007).

[19] M. A. Weiss, "Data Structures and Algorithm Analysis in C++", 4th ed., Pearson Education, New Jersey (2014).

[20] G. Barnett and L. Del Tongo, "Data Structures and Algorithms: Annotated Reference with Examples" DotNetSlackers (2007).

[21] Md. Khairullah, "Enhancing worst sorting algorithms" Int. J. Adv. Sci. Technol., 56, 13-26 (2013).

[22] M. Alam and A. Chugh, "Sorting algorithm: An empirical analysis", Int. J. Eng. Sci. Innov. Technol., 3, 2, 118-126 (2014).

[23] Searching Algorithm, Retrieved from http://www.idc-online.com/technical_references/pdfs/.../Searching_ Algorithms.pdf (2017).

[24] A. Drozdek, "Data Structures and Algorithms in Java", 2nd ed., Thomson Learning, Boston (2010).

[25] B. Langmead, Retrieved from http://www.langmead-lab.org/teaching-materials (2016).

[26] J. Platos, M. Prilepok and V. Snasel, "Text comparison using data compression", VSB-Technical University of Ostrava (n.d.).

[27] L. L. Wortel, "Plagiarism Detection Using the NCD", Universiteit de Amsterdam (2005).

[28] P. Vitanyi, F. Balbach, R. Cilibrasi and M. Li, Normalized Information Distance, Retrieved from https://arxiv.org/abs/0809.2553 (2017).

[29] P. Butler, A Simple Diff Algorithm in PHP. Retrieved from https://paulbutler.org/archives/a-simple-diff-algorithm-in-php/ (2018).

[30] ISO/EIC 25010: Quality Model, Retrieved from https://edisciplinas.usp.br/pluginfile.php/294901/mod_resource/content/1/ISO%2025010%20-%20Quality%20Model.pdf (2017).

[31] ISO 25010 Software Quality Model, Retrieved from Codacy Web site: https://blog.codacy.com/enterprise-software-a-summary-of-iso-25010-software-quality-model-7100575d6f6 (2017).