Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2021, 8(4):99-105**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Plagiarism Detection System using an Enhanced String Matching Algorithm

Nkue Dumka, Taylor Onate

Department of Computer Science, Rivers State University, Port Harcourt, Nigeria
dumkaforgod21@gmail.com, taylor.onate@ust.edu.ng

Abstract The issue of plagiarism in our academic institution has risen greatly over the years. There has been a very lukewarm approach to this malady by institutions. There arises a need to address this situation. Plagiarism detection techniques have been discussed extensively. In this project we designed a new model (The Knowledge-Base String Matching Algorithm) which adopts the Knuth Morris Pratt Algorithm for detecting plagiarism in text document. The methodology used was the Dynamic System Development Method and the plagiarism detection software was implemented using the Java programming language. The developed plagiarism detection system was deployed on large document samples for validation and verification. The developed plagiarism detection system has been compared to the Knuth Morris Pratt Algorithm and has been proven to detect plagiarized words in situations where the use of the Knuth Morris Pratt Algorithm fails. The results of the test showed that the developed system is optimal for both small and large documents.

Keywords Plagiarism Detection System, Enhanced String Matching Algorithm

1. Introduction

In the academic field, the nation and the world at large, the advancement of information technology (IT) and, in particular, the Internet have significantly expanded the availability of information. One type of undue use of knowledge that IT has made much simpler is academic plagiarism [1-2]. Many resource documents are also accessible on the internet and are easy to view. Users can quickly build a new document by copying and pasting from this resource because of this availability. Users will also re-phrase the plagiarized section or replacing certain terms with their synonyms, where conventional plagiarism detection systems can hardly detect this sort of plagiarism. With the amount of information available, it is scarcely possible to identify plagiarism by manual examination [3]. Methods and systems capable of partly automating the identification of plagiarism (PD) are also an active area of study. This research proposes an efficient method of detecting plagiarism through string matching algorithms.

2. Literature Review

The Aho-Corasick algorithm is an algorithm developed by Alfred V. Aho and Margaret J. Corasick for string search. It is a dictionary-matching algorithm which locates elements within an input text of a finite set of strings (the "dictionary"). It concurrently fits all strings. By constructing an automaton. The algorithm's first step is to create a tree that contains the dictionary (a search tree). The second step is to add edges that transform this simple tree into an automaton that fits linear time. These edges allow quick transitions to other branches of the tree that share a similar prefix between failed string matches, allowing the automaton to transit between string matches without the need for backtracking.

Informally, the algorithm produces a computer with a finite state that resembles a tree with additional relations between the various internal nodes. These additional internal ties allow easy transfers to other tree nodes that share a similar prefix between failed string matches (for example, a search for a cat in a tree that does not contain a cat but contains a cart and thus does not prefix the node with ca) (e.g., in the previous case, a branch for attribute might be the best lateral transition). This allows the automaton to turn without backtracking between string matches. If a string dictionary (such as a computer virus database) is known in advance, it is possible to create the automaton while off-line and save the compiled automaton for subsequent use. In this case, in its run time, the input length plus the number of matched entries is linear. The Aho–Corasick string matching algorithm was the foundation of the original Unix command.

Ahmed [4] implemented plagiarism identification Using graph based representation. This approach was only viewed as a concept without tests or findings, so it is impossible to test this method without results and experiments. Pablo [5] suggested a method based on the gap between Lempel and Ziv that was used to derive structural knowledge from documents. The method looks for the outliers between each text fragment in the vector of distances. Thomas [6] implemented a system based on traditional techniques of extraction of information by choosing efficient data structures between the original and suspicious text for thorough analysis.

3. Materials and Methods

The proposed method is a Knowledge-based String Matching Algorithm (KSMA), an enhancement of the Knuth Morris Pratt Algorithm, which enhances the Knuth Morris Pratt Algorithm by ensuring that no more than N character comparison is needed for a string search (once some pre-computation is performed).By using the observation inside a key "text strings" S, It looks for the instances of a "word" W and its "synonyms" W[i...n] that the word itself provides sample details to decide when the next match will begin, passing re-examination of previously matched characters when a mismatch arises. When matching patterns with the text from left to right and using automata to find all the instances of a sequence in a text, the KSMA adopts features of the Knuth Morris Pratt Algorithm.

The advantages of the proposed system include:

i. Unlike the Knuth Morris Pratt Algorithm the proposed system solves the problem of Shake & Paste Plagiarism [such as replacing words with their synonyms to avoid plagiarism detection].

ii. In the input text T, the algorithm never has to step backwards. This makes the algorithm suitable for very big files being stored.

iii. The implementation of the proposed system algorithm is efficient because with its knowledge based containing synonyms of several words it becomes much more difficult to bypass.

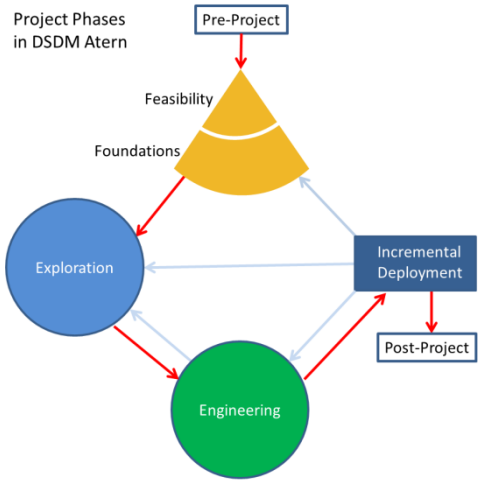

Figure 1: The Project Phases in DSDM

A technique for software Design refers to the mechanism used to coordinate, plan and monitor the information system implementation process. A large variety of such systems have evolved over the years, each with its own

established strengths and constraints. The Dynamic System Architecture Approach was used to construct this system. Dynamic System Development Method (DSDM) provides a structure for designing and managing systems controls that not only follow tight time limits and offer a repeatable RAD formula. The DSDM methodology addresses the RAD view of the developer, but also that of all other parties, including customers, project managers and quality assurance staff, who are not only involved in successful system development.

The Dynamic System Development Method (DSDM) is complicated since it uses gradual prototyping as a Fast Program Development method. This approach is especially useful for applications to be built in a brief amount of time and where at the outset of the application development the criteria can not be frozen. Whatever specifications are understood at a time, design is designed for them and design is created and implemented into the system.

To store synonyms of terms, the MYSQL (My Structured Query Language) Relational Database Management System (RDBMS) will be used. The different database tables to be used in the framework implementation are listed below.

Table 1: Word_Table Database Table Description

S/N	COLUMN NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1.	Word_id	VARCHAR	20	Primary Key	It stores Word id
2.	Word	VARCHAR	50	Unique	The specific name of each word is stored here.

TABLE NAME :Word_Table.

PRIMARY KEY: Word_id

DESCRIPTION: It store the Words in the knowledge base

Table 2: Synonyms Database Table Description

S/N	COLUMN NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1.	W_id	VARCHAR	20	Foreign Key	It stores ID for each unique word
2.	S_id	VARCHAR	20	Foreign Key	It stores ID for each unique word

Table Name: Synonym_Table.

Description: It store pairs of words in the knowledge base (database) establishing their relationships.

Table 3: System log Database Table Description

S/n	Column name	Data type	Size	Constraint	Description
1.	Log_id	VARCHAR	20	primary Key	It stores ID for each unique log
2.	Document	TEXT			It stores ID for each unique word
3.	KMP_PlagiarisedWords	INTEGER			It stores the total number of plagiarized words detected by the existing system
4.	KMP_Similarity	DECIMAL			It stores percentage of similarity analyzed by existing the system
5.	KSMA__PlagiarisedWords	INTEGER			It stores the total number of plagiarized words detected by the proposed system
6.	KSMA_Similarity	DECIMAL			It stores percentage of similarity analyzed by proposed the system
7.	Total_Words	INTEGER			The cumulative number of terms found in the text is retained for comparison with other documents.

Table name: Logs Description: it store details of the analysis report of both the proposed system and existing system.

4. Result and Discussion

The performance evaluation for Plagiarism Detection System Using An Enhanced String Matching Algorithm has been obtained via Java and MySQL programming languages.

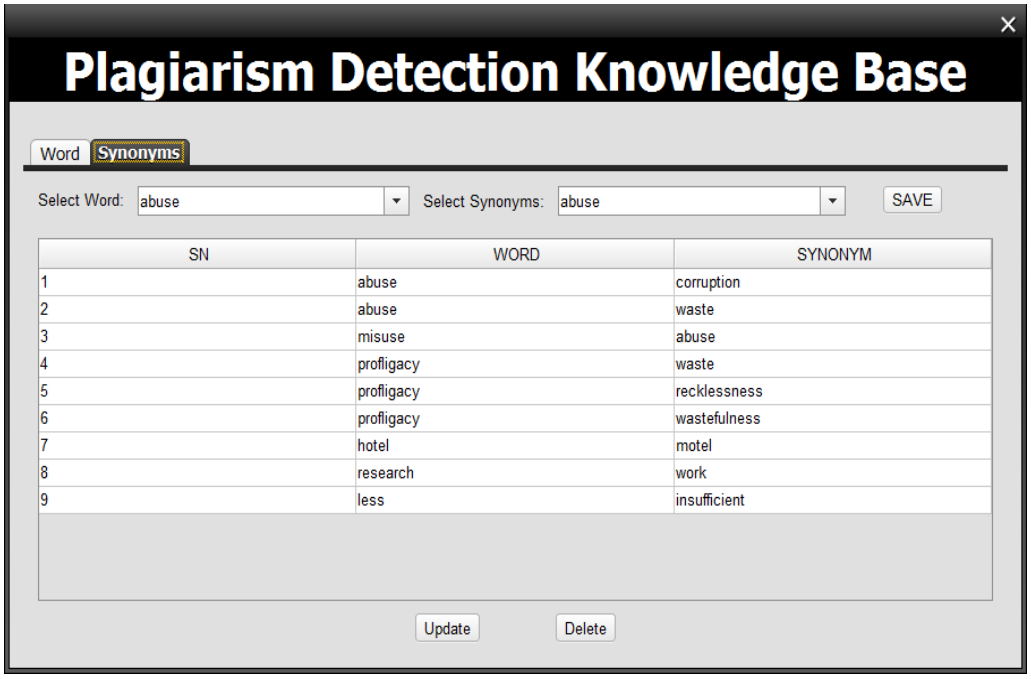

Figure 4.1

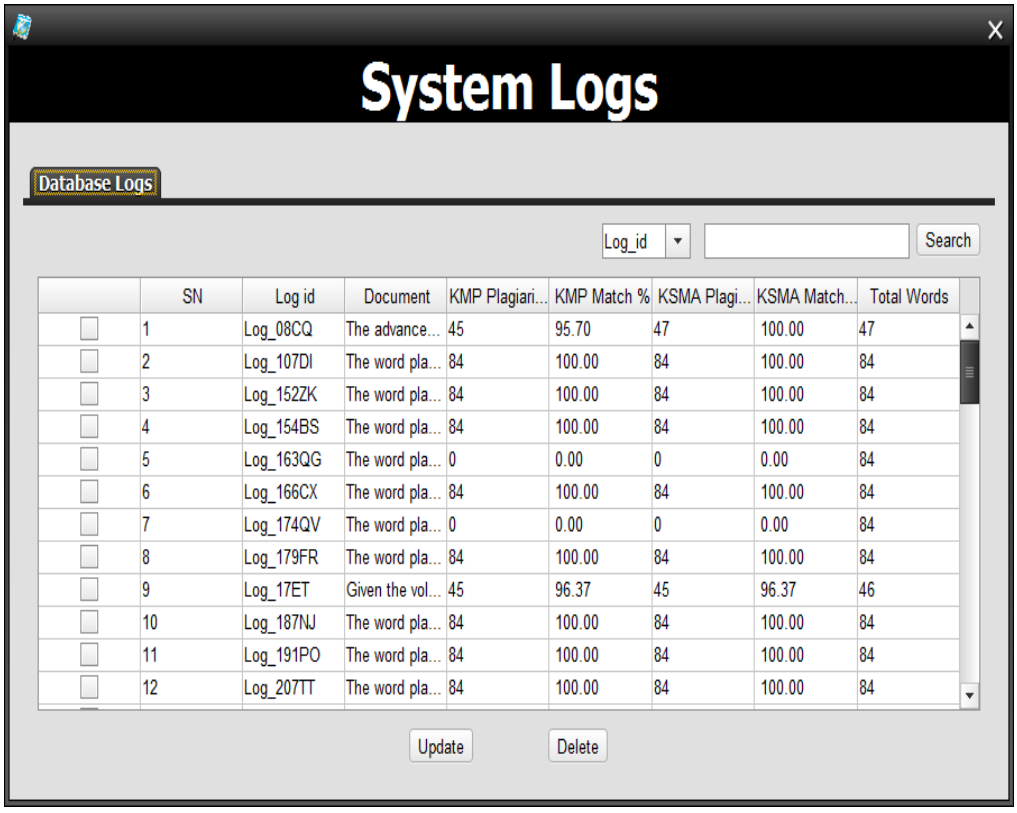

Figure 4.2

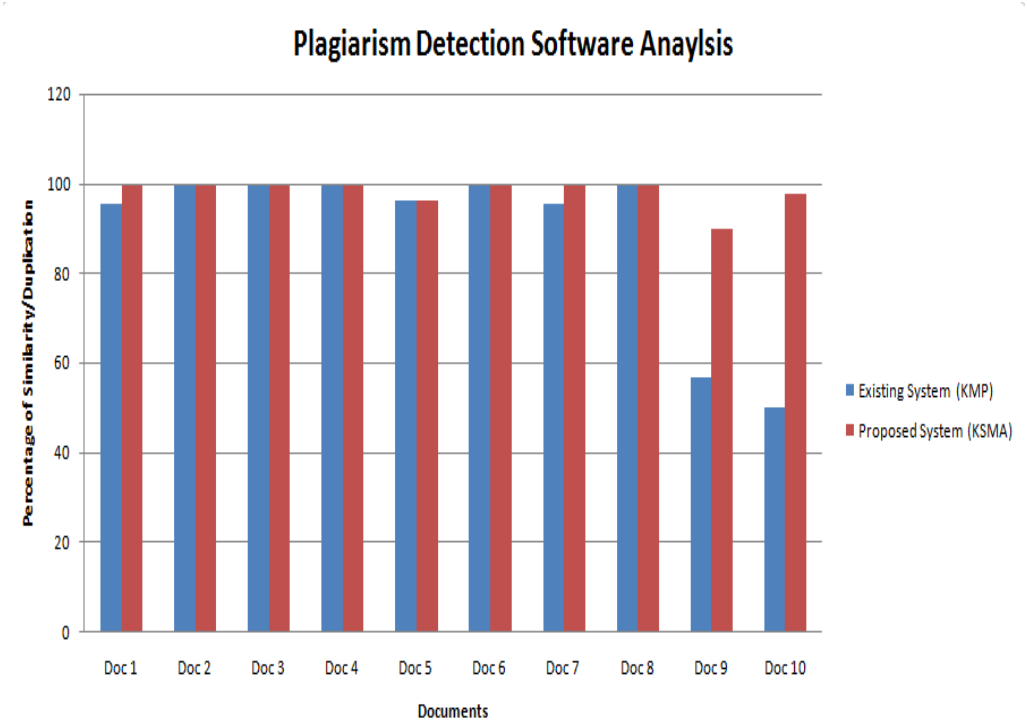

Figure 4.3

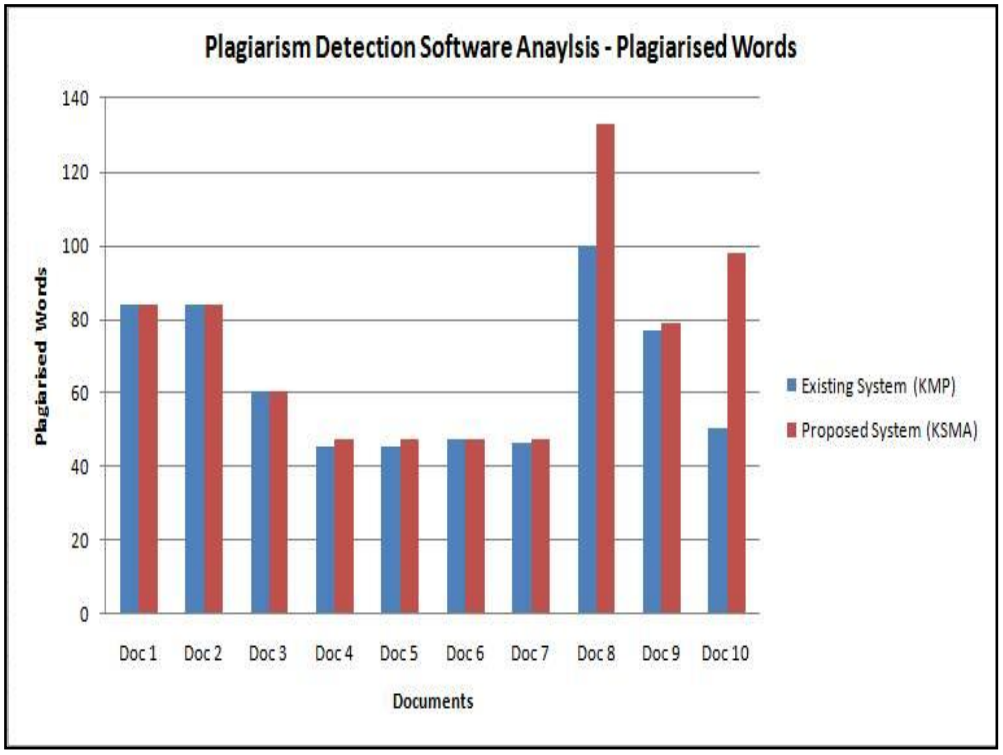

Figure 4.4

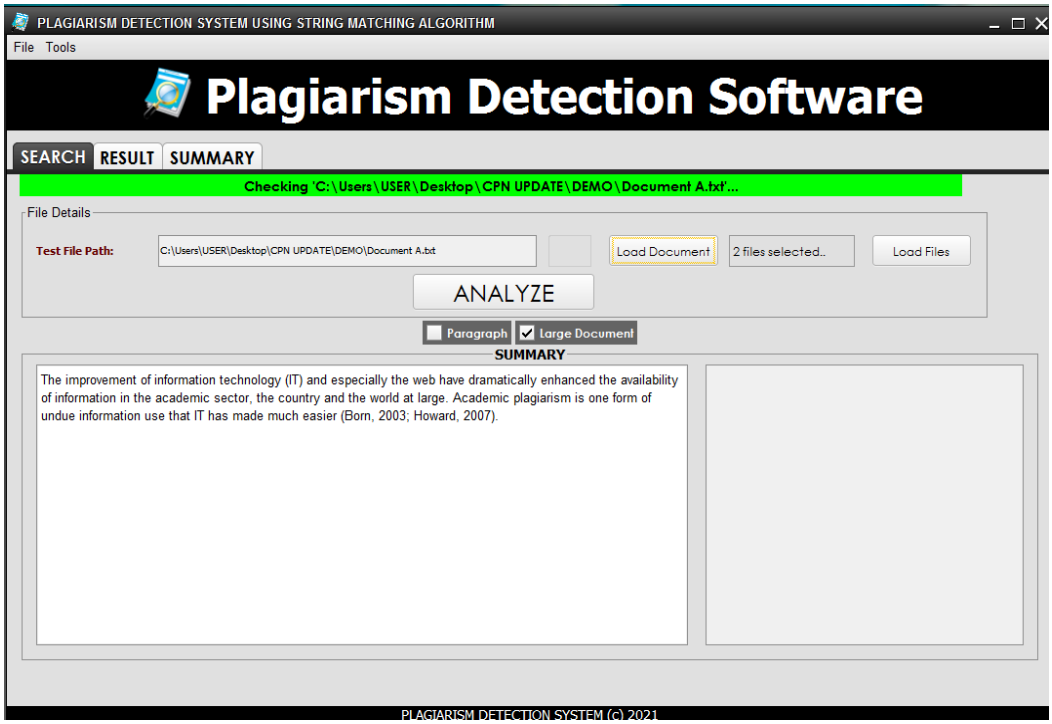

Figure 4.5

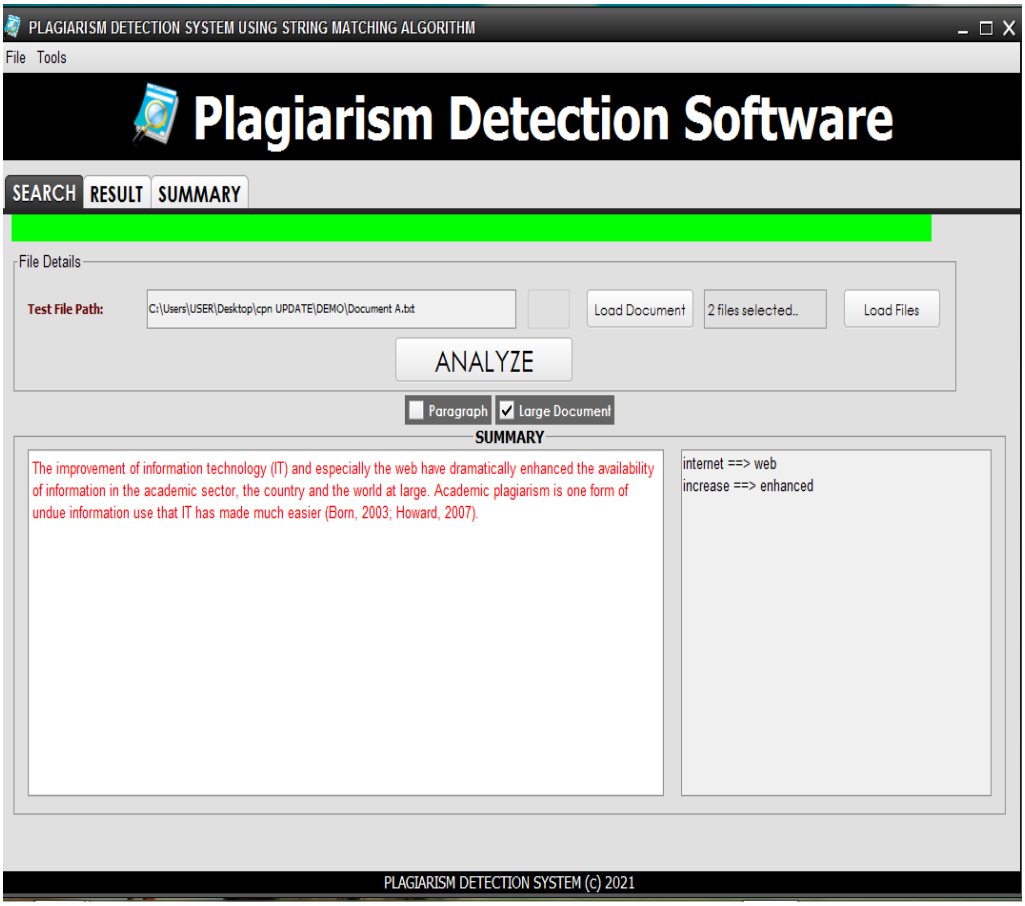

Figure 4.6

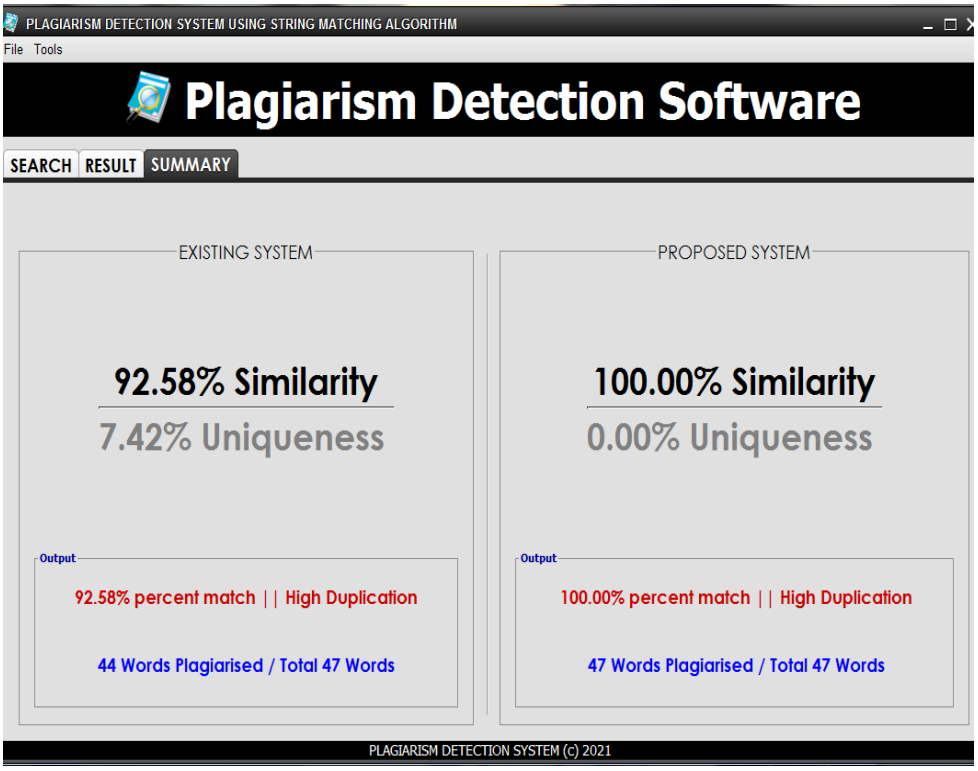

Figure 4.7

5. Conclusion

This work was centered on the development of a knowledge-based string matching algorithm adopted from The Algorithm of Knuth-Morris-Pratt (KMP) to detect plagiarism. The system is designed to overcome the shortcomings Of the Algorithm for Knuth-Morris-Pratt (KMP) by increasing the efficiency using a knowledge-base which detects words even when replaced with other words synonymous to them such as Shake & Paste (S&P). Excellent support for the efficient development of the framework was provided by the implementation setting and the instruments used. The outcome of this project leads to the fact that, a more powerful algorithm can be accomplished if this knowledge base is paired with the effective string matching of the Knuth-Morris-Pratt Algorithm (KMP).

References

[1]. Born, A. D. (2003). Teaching tip: How to reduce plagiarism. *Journal of information systems education*, *14*(3), 223.

[2]. Howard, R. M. (2007). Understanding "internet plagiarism". *Computers and composition*, *24*(1), 3-15.

[3]. Clough, P., & Stevenson, M. (2011). Developing a corpus of plagiarised short answers. Language resources and evaluation, 45(1), 5-24.

[4]. Ahmed, H., Osman, Salim N. and Binwahian, S. S. (2010). Plagiarism detection using graph based representation. *Journal of Computing*, 2(4): 2151-9617.

[5]. Pablo, S., José, C., Gonzalez. F & Villena-Román, J. (2010). Foundations and trend information. In Proceedings of the Uncovering Plagiarism Authorship and Social Software Misuse PAN 2010 Workshop.

[6]. Thomas, E. E. (2010). "We're Saying the Same Thing': How English Teachers Negotiated Solidarity, Identity, and Ethics through Talk and Interaction."