International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 8– No.4, February 2015 – www.ijais.org

Plagiarism Index Estimation Algorithm: A Quantitative Approach

Monday O. Eze
Department of Computer Science/Maths/Informatics
Federal Univ. Ndufu-Alike, Ikwo (FUNAI), Ebonyi State, Nigeria

Shakeel A. Kamboh
Dept. of Maths & Statistics
Quaid-e-Awam University of Engineering, Science and Technology, Pakistan

ABSTRACT

Plagiarism has remained a serious setback especially in the academia. It is a major source of intellectual theft since it gives credits for scientific innovations to those who do not merit them. A number of efforts have been made by researchers to tackle plagiarism. However, one perceived research gap is the need to evolve verifiable computational techniques for detecting and quantifying the degree of plagiarism in digitized documents. This current research tackles this problem through a specialized plagiarism detection and quantification algorithm. It begins with a bi-partitioned search operation known as F-Search. This is followed by a purge operation which excludes the plagiarized sections discovered during the initial pass, thus giving rise to a fresh search space. The resulting search space is passed through a more thorough search operation known as T-Search. At this stage, the algorithm deals with specific plagiarism hiding tricks termed as whitespace flooding. The final output is a statistic known as the Plagiarism Index, which is a numeric value in the range [0, 1] for estimating the degree of plagiarism. The scope of this research covers the text domain. Each experimental dataset is made up of a set of two documents designed in such a way that one is assumed as the original document, while the second as a plagiarized copy. The system is designed and implemented in MATLAB.

General Terms

Pattern Recognition, Search Algorithms, String Manipulation, System Workflow, Software Implementation.

Keywords

Plagiarism Index, Cell Array, Plagiarism Quantification, Bi-Partition.

1. INTRODUCTION

The term plagiarism refers to the act of reproducing other people’s intellectual work, without appropriate citations. In other words, plagiarism is a form of intellectual crime [1], through which people steal or withhold the credit, which ought to have gone to the actual owner(s) of an intellectual work [2]. The Cambridge Advanced Learners Dictionary [3] defines plagiarism as the use of other persons’ idea or part of their work, while pretending that it is one’s own. Because of the importance of plagiarism to the academia and the productive society at large, a lot of research interests have been devoted to plagiarism detection in the recent time [4]. Unfortunately, manual plagiarism detection has been described as very difficult and time consuming [5], thus the need to develop appropriate computational techniques to deal with the anomaly.

Studies on plagiarism in schools [6] have revealed two important categories of students’ plagiarism. The first one is known as *deliberate plagiarism*, where someone is very much aware of the meaning and implications of plagiarism, but decided to indulge in it. The second one is known as *accidental plagiarism* [7], where people are ignorant of the fact that they were committing the act. Thus there is an urgent need to educate the researchers on how to avoid accidental plagiarism. It has been widely suggested that the advent of Internet has fueled widespread occurrences of plagiarism. Some of the authors that share this view are [8], [9] and [10]. Thus, cyber-plagiarism [11] refers to the practice of cutting and pasting information from the Internet into a formal writing without appropriate citations. The aim of this research is to develop a computational strategy for detecting and quantifying the degree of plagiarism in the text domain. A new statistic known as *Plagiarism Index* is used to achieve this objective. Some of the related works, the proposed algorithm and the experimental results will be presented in further details.

2. RELATED RESEARCH

A survey on plagiarism detection [12] outlined the application of attribute counting techniques based on average line length, file size, and average number of commas per line to derive file fingerprints. The resulting fingerprint is then used to estimate the degree of plagiarism. However, [13] has described this method as unreliable. A broad-based introduction to plagiarism detection, and its application, especially in the field of online journalism has been done by [14]. The research classified plagiarism detection into five categories namely external, intrinsic, cross-lingual, near-duplicate and partial-duplicate plagiarism detection. The earlier works such as [15] and [16] were cited as building blocks to this classification system. A natural language processing-based plagiarism detection technique using set theory has been proposed by [17]. It involves a series of steps, such as tokenization, string normalization and chunking. One of the weaknesses of this algorithm is that it fails considerably when the input documents contain texts in more than one language. Source code plagiarism detection [18] is an evolving application area aimed at preventing incidences of software theft at the coding level. A research in this regard by [19] attempts to detect source code plagiarism using an information retrieval technique known as latent semantic analysis by deriving semantic information from source-code files. The application of plagiarism detection as a fundamental part of conference paper review has been proposed by [20]. The work focused on the use of similarity algorithms such as Cosine, Manhattan and Euclidian techniques to search for evidences of plagiarism. The major challenge with this method is that it is characterized by serious tradeoffs between system accuracy and performance. A comparative research [21] has applied practical tests on four variants of a plagiarism detection technique known as chunking. The aim was to scientifically determine the application areas that best fits each of these variants. It was reported that the first variant is best suited for database plagiarism solutions, and the second for quoted texts plagiarism solutions. Furthermore, the third variant which is known as hashed breakpoint chunking is best suited for large set of documents, while the fourth is said be the most appropriate when high degree of reliability is of utmost priority. Apart from several English Language-based implementations, researchers have also attempted to build plagiarism detectors in a number of other languages. For instance, the Hangul Plagiarism Detection System (HPDS) detects plagiarism in the Korean Language [22], while the APlag System [23] detects plagiarism in Arabic documents. Plagiarism detection has also been attempted in multi-media. For instance, a typical research on plagiarism detection in music is the work done by [24]. Based on an extensive literature review, there are a number of issues that impede plagiarism solutions. As corroborated by [25], [26] & [27], three of such thorny issues are *plagiarism detection, plagiarism quantification and multi-lingual plagiarism*. The aim of the current research is to tackle the problem of plagiarism detection and quantification. The proposed algorithm attempts to detect cases of copying lines of a given document into another one without appropriate citations. It also detects cases of attempts to fool the plagiarism detector through the use of whitespaces, a technique which has been termed in this research as *whitespace flooding*. A major outcome of this research is its ability to estimate the degree of plagiarism using a numeric statistic termed as *plagiarism index.*

3. THE PROPOSED ALGORITHM

The aim of this section is to present the methodology of the current research. Fig. 1F is the workflow of the proposed plagiarism detection and quantification algorithm. It is made up of five compartments namely the Input, F-Search, Bi-Partition, T-Search, and the Index Quantifier. As shown in the workflow, the Input compartment accepts the two input documents represented with generic names Doc1 and Doc2. During the actual implementation, a standard nomenclature is adopted for these inputs. The aim of the algorithm is to computationally detect the existence of plagiarism between the input datasets, and to quantify the level of plagiarism involved. The F-Search engine performs a first level plagiarism search while the T-Search engine does a second level search. The letter 'T' stands for '*thorough*'. Thus at the T-Search stage, the system performs a more thorough plagiarism search on the suspected search space. As shown in the workflow, the formation of the 'VAB' and 'SUS' partitions precedes the T-Search operation. The VAB-Partition represents the search space confirmed to be 100% plagiarized, line by line. The prefix 'VAB' therefore stands for *'verbatim'*. In other words, this partition contains the area of the original document computationally confirmed to have been copied (plagiarized) verbatim into the second document. The SUS-Partition on the other hand, is the suspected partition. The prefix 'SUS' therefore stands for *'suspect'*. It is made up the content of Doc 1 which is suspected to have been plagiarized into Doc2, but requires further proof or confirmation. It is the SUS-Partition that is usually passed through a thorough search. Finally, the results of the two search operations are combined to obtain the cumulative plagiarism index. This is done at the plagiarism index quantification stage.

4. SYSTEM IMPLEMENTATION

The system workflow in Fig. 1F was implemented using a number of steps as will be outlined in this section. Fig. 2F is the component diagram for the program modules. MATLAB was used for system implementation, thus, each module ends with the file extension '*m*'. The data modules are listed in Fig. 3F. As shown in the diagram, the implementation data can be classified into three - the text files (with extension *txt*), the program output file (with extension *out*) and the implementation tables (with no extension). These will be explained in more details.

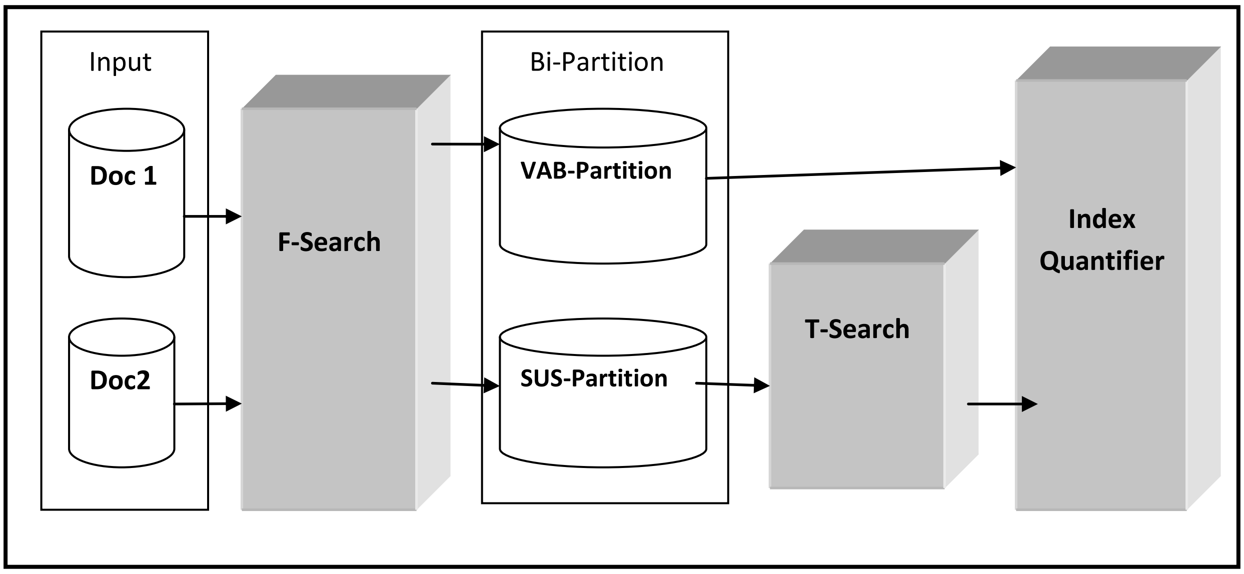


Fig. 1F: Plagiarism Detection & Quantification Algorithm

4.1 The Pre-Processing Operation

The preparation of the dataset involves the extraction of the input documents to be passed through the plagiarism detection and quantification system. Though the system can analyze as many files as possible, it however accepts only two at a given time. The input files are in text format with standard names *Origf.txt* and *Plagf.txt* for Doc1 and Doc2 respectively. The choice of the standard nomenclature stems from the fact that the first file is assumed as the original while the second as the plagiarized copy. The aim of the current system is to computationally analyze the two files and search for traces of plagiarism. The input files are placed in the same directory of the computer system housing the executable program. The program module implemented to handle preprocessing is known as *preprocess.m*. It takes the two input files, and gives the output known as '*Twopartif.out'*. The sample of this output file resulting from the experimental run is shown in Fig. 4F

4.2 The F-Search Operation

As the name suggests, this is the *first level plagiarism* search section of the system. As shown in the main workflow, it follows the data acquisition stage. The *F-Search* is implemented using the *PlagSearch1.m* program module, which was designed using the flowcharts in Fig. 11F and Fig. 12F in Appendix A. The following set of transformation equations summarizes one of significant steps in the search operation,

$$\begin{bmatrix} Origf.txt \rightarrow aatab \\ Plagf.txt \rightarrow bbtab \\ Twopart.txt \rightarrow ggtab \end{bmatrix} \qquad (1)$$

where the symbol '➔.' represents a line by line data extraction of the right hand side (RHS) file to populate the left hand side (LHS) table.

The six output tables shown in the data module diagram in Fig. 3F are *cell arrays*. A cell array is an array of diverse data structures. The choice of cell arrays in the system implementation is based on the fact that they support fast. string search operations, thus making them appropriate for plagiarism search. The program module *popabtab.m* populates the system tables *aatab* and *bbtab* with the contents of the two input files, while the program module *popggtab.m* populates the global table *ggtab* with the content of the partition file shown in Fig.4F.

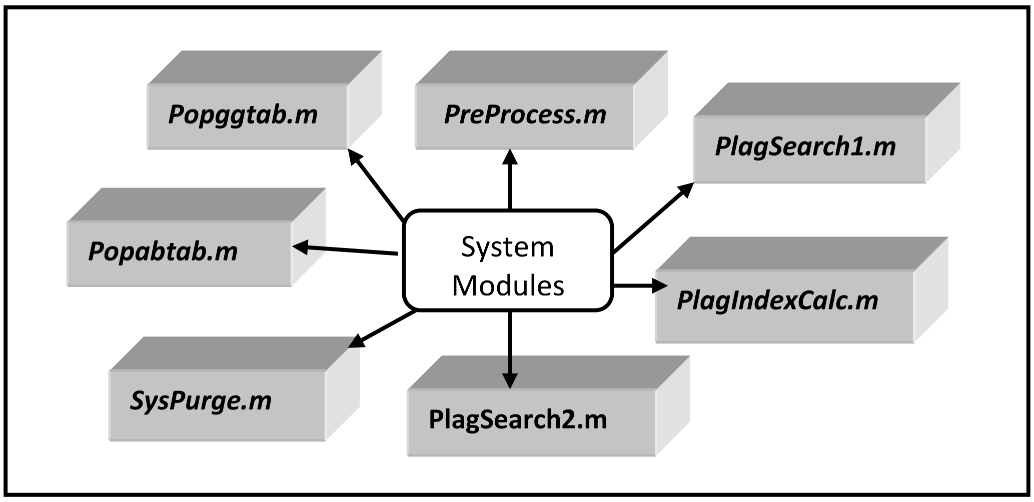


Fig. 2F: System Program Modules Diagram.

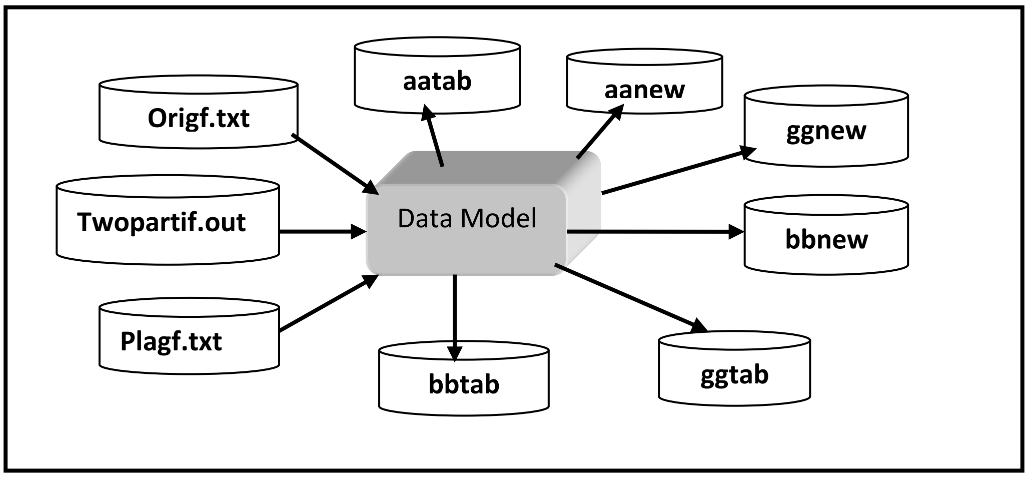


Fig. 3F: System Data Modules Diagram.

During the F-Search, each record in *ggtab* is computationally examined by the plagiarism search algorithm. The record structure of the global table is shown in Table T1.

Table T1: Standard Structure of ggtab

LLN	**LeftSide Data**	**CentralSymbol**	**RighSide Data**	**RLN**
🡐=================131=================🡒				

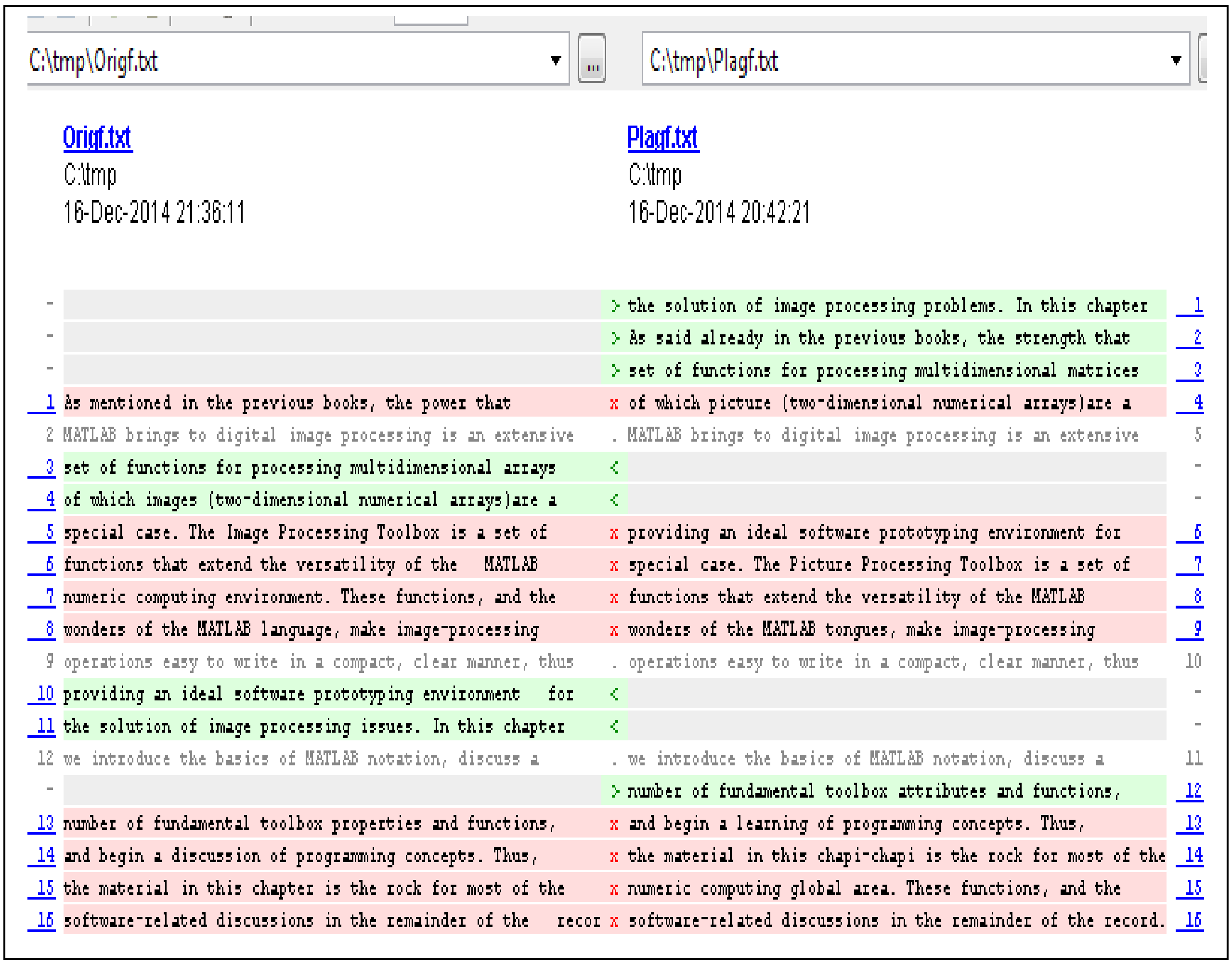
In the *ggtab* table structure, *LLN* and *RLN* are line numbers known as 'Left Line Number' and 'Right Line Number' respectively. The *LLN* keeps track of the position of the data acquired from *aatab*, known as '*Left Side Data'* while *RLN* keeps the corresponding position of '*Right Side Data'* acquired from *bbtab*. Thus, the significance of *ggtab* is that it matches the contents of *aatab* and *bbtab* side by side in an initial plagiarism search. In the implementation dataset, the size of each *ggtab* record is 131. The 66th character which marks the center of *ggtab* records forms what is termed the *Central Symbol*. There are four characters that constitute the central symbols. These are '<', '>', 'x', and '.' respectively. The central symbols are very important in the plagiarism F-Search, especially in the formation of bi-partitions.

4.3 Plagiarism Search Bi-Partition

As already explained at the workflow section, the plagiarism search space records are broken into two logical partitions. These are the VAB and SUS partitions respectively as shown in Fig. 5F. The formation of the logical partitions stems from a physical partition based on the four central symbols as demonstrated in Table T1. Thus, the search space records with central symbol 'dot' make up the VAB-Partition. These are records confirmed to be 100% plagiarized by copying the contents of one input document (Origf.txt) into the second one (Plagf.txt). Similarly, the search space records having the other three central symbols form the SUS-Partition. It is the SUS-Partition that undergoes a further search process called T-Search (or thorough search). The formation of the SUS-Partition is implemented through a process termed *as system purge* using a program module known as '*Syspurge.m'* as indicated in Fig. 2F. The following set of transformation equations summarizes the purge operation.

$$\begin{bmatrix} aatab \rightarrow aanew \\ bbtab \rightarrow bbnew \\ ggtab \rightarrow ggnew \end{bmatrix} \qquad (2)$$

where the symbol '➔' represents a line by line data extraction of the right hand side (RHS) table to populate the left hand table (LHS) table.



```
Origf.txt                                                   Plagf.txt
C:\tmp                                                      C:\tmp
16-Dec-2014 21:36:11                                        16-Dec-2014 20:42:21

 -                                                       > the solution of image processing problems. In this chapter   1
 -                                                       > As said already in the previous books, the strength that     2
 -                                                       > set of functions for processing multidimensional matrices    3
 1 As mentioned in the previous books, the power that    x of which picture (two-dimensional numerical arrays)are a     4
 2 MATLAB brings to digital image processing is an extensive . MATLAB brings to digital image processing is an extensive  5
 3 set of functions for processing multidimensional arrays <                                                            -
 4 of which images (two-dimensional numerical arrays)are a <                                                            -
 5 special case. The Image Processing Toolbox is a set of  x providing an ideal software prototyping environment for       6
 6 functions that extend the versatility of the   MATLAB   x special case. The Picture Processing Toolbox is a set of     7
 7 numeric computing environment. These functions, and the x functions that extend the versatility of the MATLAB         8
 8 wonders of the MATLAB language, make image-processing   x wonders of the MATLAB tongues, make image-processing         9
 9 operations easy to write in a compact, clear manner, thus . operations easy to write in a compact, clear manner, thus 10
10 providing an ideal software prototyping environment  for <                                                           -
11 the solution of image processing issues. In this chapter <                                                           -
12 we introduce the basics of MATLAB notation, discuss a    . we introduce the basics of MATLAB notation, discuss a      11
 -                                                       > number of fundamental toolbox attributes and functions,      12
13 number of fundamental toolbox properties and functions, x and begin a learning of programming concepts. Thus,         13
14 and begin a discussion of programming concepts. Thus,   x the material in this chapi-chapi is the rock for most of the 14
15 the material in this chapter is the rock for most of the x numeric computing global area. These functions, and the    15
16 software-related discussions in the remainder of the  recor x software-related discussions in the remainder of the record. 16
```

Fig. 4F: The Preprocessing Output File known as TwoPartif.out

In summary, the purge operation ensures that the VAB-Partition data which are in the three tables *aatab*, *bbtab* and *ggtab* are excluded in the search space used for the T-Search operation. A precaution is taken to preserve the original data. Thus, instead of totally wiping out the unwanted data, a work-around step is taken to preserve the original data. This is by creating a set of mirror tables *aanew*, *bbnew* and *ggnew* based on equation (2).

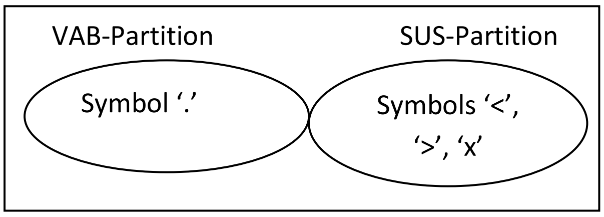
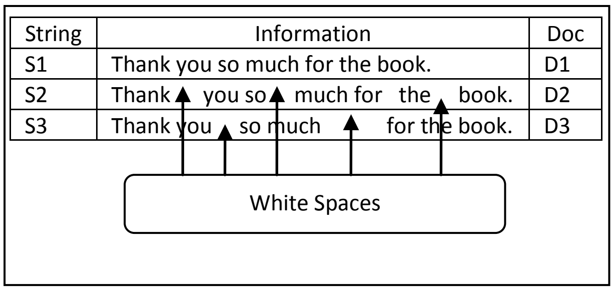


Fig. 5F: Central Symbols for Bi-Partition Formation

4.4 The T-Search Operation

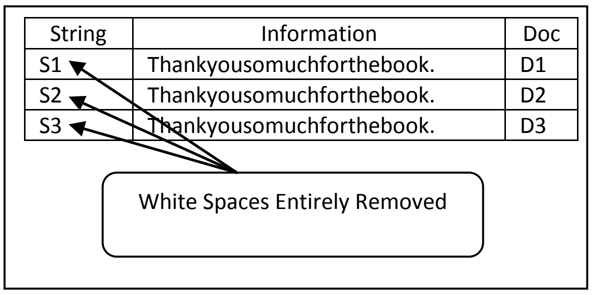
The two tables - *aanew* and *bbnew* resulting from the purge operation are used as inputs into the T-Search (thorough search) operations. One of the discoveries made during systems implementation is that a plagiarism detection algorithm could be '*fooled*' through the use of whitespaces. For instance, suppose S1 is a typical string in a digital document D1. Suppose S1 is plagiarized verbatim to form the new strings S2 and S3 in the documents D2 and D3 respectively as shown in Fig. 6F.



String	Information	Doc
S1	Thank you so much for the book.	D1
S2	Thank you so much for the book.	D2
S3	Thank you so much for the book.	D3

Fig. 6F: White Space Flooding in Documents

It is possible for a plagiarism detection system to erroneously assume that the strings S1, S2 and S3 are entirely different, even when they are exactly the same except for differences in the positions of whitespaces. In order to deal with this anomaly termed in this research as *whitespace flooding*, the algorithm performs whitespaces exclusion operation. Thus before performing strings comparisons with S1, S2 and S3, the algorithm first transforms them into the formats shown in Fig. 7F



String	Information	Doc
S1	Thankyousomuchforthebook.	D1
S2	Thankyousomuchforthebook.	D2
S3	Thankyousomuchforthebook.	D3

Fig. 7F: Outcome of Whitespace Exclusion Operation

The T-Search searches for plagiarism occurrences by comparing the contents of the two input files. The number of matches detected is then used to calculate the plagiarism index. The program module for the implementation of T-Search is known as *PlagSearch2.m*, with flowcharts shown in Fig. 13F and 14F in Appendix A.

4.5 The Plagiarism Index Quantifier

The plagiarism index of the system is the sum of the indices arising from both the F-Search and the T-Search operations. The Plagiarism Index for F-Search is given by equation (3).

$$PlagIndexF = \left\lfloor \frac{dtC}{2(dtC + xsC) + ldC + gtC)} \right\rfloor \qquad (3)$$

where *dtC* = number of lines having '.' as the central symbol, *xsC* = number of lines having 'x' as the central symbol, *ldC*= number of lines having '<' as the central symbol, and *gtC*= number of lines having '>' as the central symbol after bi-partition operation.

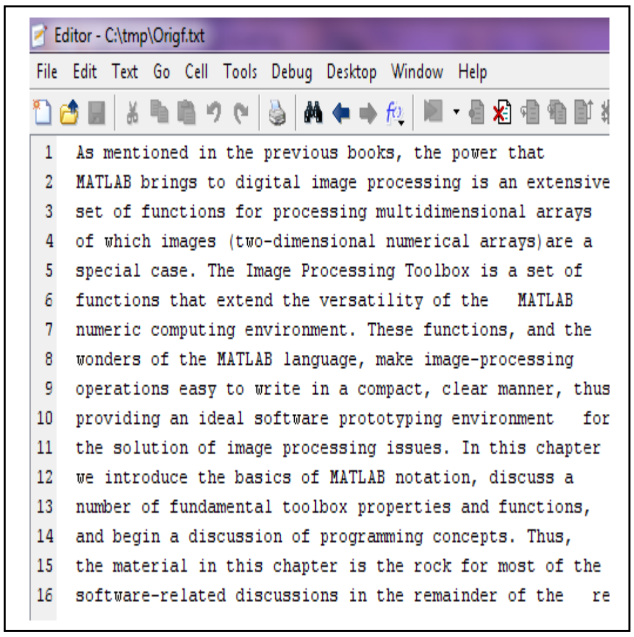
The Plagiarism Index for T-Search is given by equation (4).

$$PlagIndexT = \left\lfloor \frac{mtCount}{DocSize1 + DocSize2)} \right\rfloor \qquad (4)$$

where *mtCount* = number of plagiarism matches found during T-Search, *DocSize*1 and *DocSize2* are the sizes of the first and second documents in terms of number of lines.

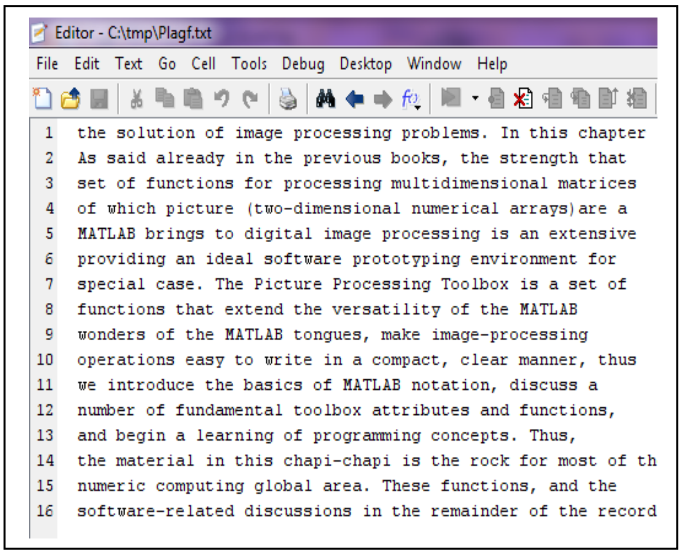
5. RESULTS AND DISCUSSION

Two documents were used as inputs to the plagiarism detecting system. The first one is a brief essay constructed in text format, while the second was generated by deliberately plagiarizing the first one. Thus, a number of lines of the original document were lifted verbatim and used to build the new document. The two input documents were given the standard input names. The first document (Origf.txt) and the second one (Plagf.txt) are shown in Fig. 8F and Fig 9F respectively.


Editor - C:\tmp\Origf.txt
File Edit Text Go Cell Tools Debug Desktop Window Help

```
As mentioned in the previous books, the power that
MATLAB brings to digital image processing is an extensive
set of functions for processing multidimensional arrays
of which images (two-dimensional numerical arrays)are a
special case. The Image Processing Toolbox is a set of
functions that extend the versatility of the  MATLAB
numeric computing environment. These functions, and the
wonders of the MATLAB language, make image-processing
operations easy to write in a compact, clear manner, thus
providing an ideal software prototyping environment  for
the solution of image processing issues. In this chapter
we introduce the basics of MATLAB notation, discuss a
number of fundamental toolbox properties and functions,
and begin a discussion of programming concepts. Thus,
the material in this chapter is the rock for most of the
software-related discussions in the remainder of the  re
```

Fig 8F: The First Input Dataset (Origf.txt)


Editor - C:\tmp\Plagf.txt
File Edit Text Go Cell Tools Debug Desktop Window Help

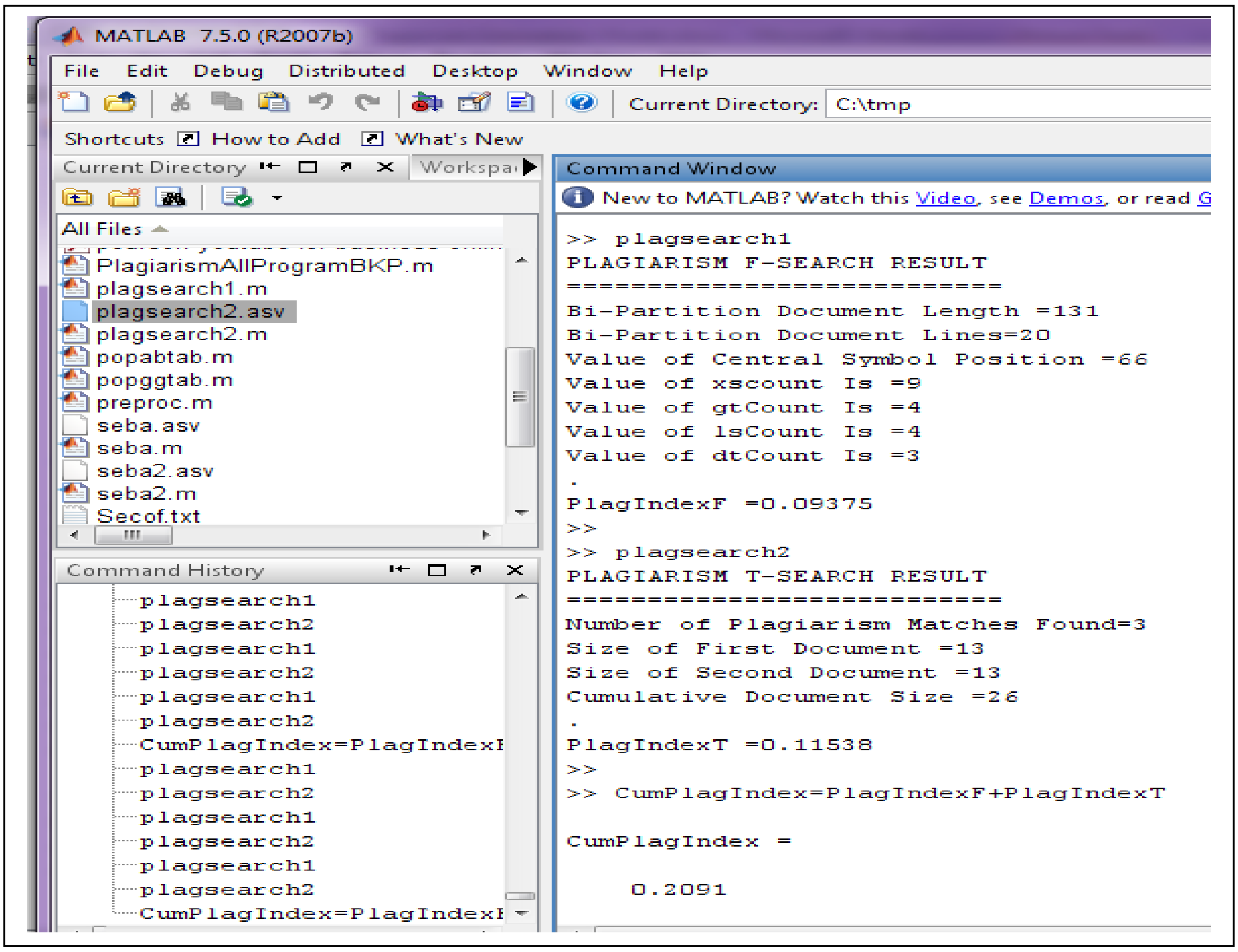
```
the solution of image processing problems. In this chapter
As said already in the previous books, the strength that
set of functions for processing multidimensional matrices
of which picture (two-dimensional numerical arrays)are a
MATLAB brings to digital image processing is an extensive
providing an ideal software prototyping environment for
special case. The Picture Processing Toolbox is a set of
functions that extend the versatility of the MATLAB
wonders of the MATLAB tongues, make image-processing
operations easy to write in a compact, clear manner, thus
we introduce the basics of MATLAB notation, discuss a
number of fundamental toolbox attributes and functions,
and begin a learning of programming concepts. Thus,
the material in this chapi-chapi is the rock for most of th
numeric computing global area. These functions, and the
software-related discussions in the remainder of the record
```

Fig 9F: The Second Input Dataset (Plagf.txt)

The experimental result is shown in Fig. 10F. As shown in the screen capture, the values calculated for *PlagIndexF* and *PlagIndexT* are 0.09375 and 0.11538 respectively. These are the plagiarism indices during the F-Search and T-Search operations respectively. The overall index is the sum of the two indices which is 0.2091. The major validation analysis done on the result is to perform a manual check to confirm that system detected all occurrences of plagiarisms, done by deliberately copying the contents of one input document into another. Thus, all the lines copied verbatim, as well as the ones carefully manipulated through whitespace flooding were all detected and listed by the system.

6. CONCLUSION/FUTURE RESEARCH

The importance of plagiarism detection cannot be overemphasized. As already stated in the research scope statement, this work is designed to operate in the text domain. Further efforts to extend the capability of this plagiarism detection and quantification algorithm will consider other specialized file formats such as binary files, image files, video files, and so on. Again, the system inputs are files accessed by the system from an archive location at run time. Further system improvements will consider the possibility of picking the input files from a network rather than a stand-alone system. Another area identified for future research is the development of a visualization algorithm which will display the results of the plagiarism search in a graphical spreadsheet format. This is expected to be a positive deviation from the current text-oriented output format.



```
>> plagsearch1
PLAGIARISM F-SEARCH RESULT
===========================
Bi-Partition Document Length =131
Bi-Partition Document Lines=20
Value of Central Symbol Position =66
Value of xscount Is =9
Value of gtCount Is =4
Value of lsCount Is =4
Value of dtCount Is =3
.
PlagIndexF =0.09375
>>
>> plagsearch2
PLAGIARISM T-SEARCH RESULT
===========================
Number of Plagiarism Matches Found=3
Size of First Document =13
Size of Second Document =13
Cumulative Document Size =26
.
PlagIndexT =0.11538
>>
>> CumPlagIndex=PlagIndexF+PlagIndexT

CumPlagIndex =

    0.2091
```

Fig 10F: Plagiarism Search Result

7. REFERENCES

[1] Green, S. P. 2002. Plagiarism, Norms, and the Limits of Theft Law: Some Observations on the Use of Criminal Sanctions in Enforcing Intellectual Property Rights. Hastings Law Journal, Vol. 54, No. 1.

[2] Ashish,M. & Sasmita,M. 2012. Student Plagiarism in Higher Education: An Enigma or An Intellectual Crime, Vol 1, Issue 1, p90-100.

[3] Cambridge Univ. Press. 2008. Cambridge Advanced Learner's Dictionary, Cambridge University Press, The Edinburgh Building, Cambridge UK.

[4] Tri Le, A.C., Judy, S., Margot, S., Michael, D. and Chris, J. 2013. Educating Computer Programming Students about Plagiarism through Use of a Code Similarity Detection Tool. LATICE, 2013, Learning and Teaching in Computing and Engineering (LaTiCE), pp. 98-105.

[5] Poongodi , D. and Tholkkappia, G.A. 2013. An Automatic Method for Statement Level Plagiarism Detection in Source Code Using Abstract Syntax Tree. Int. J. of Adv. Research in Comp. & Comm. Engineering., Vol. 2, Issue 4, pp 1923-1938

[6] Kashkur, M., Parshutin,S. & Arkady, B. 2010. Research into Plagiarism Cases and Plagiarism Detection Methods. Scientific Journal of Riga Tech. Univ., Vol 44, p139-144.

[7] Mason, P.R. 2009. Plagiarism in Scientific Publications, J Infect Developing Countries, Vol. 3(1), p1-4.

[8] Bradley,T. 2010. Student Plagiarism and the Use of Plagiarism Detection Tool by Community College Faculty (a PhD Dissertation), Department of Educational Leadership, Indiana State University, Indiana.

[9] Batane, T. 2010. Turning to Turnitin to Fight Plagiarism among University Students. Educational Technology & Society, 13 (2), p1-12

[10] Howard, R.M. 2007. Understanding Internet plagiarism, Computers and Composition, Vol. 24, p3–15.

[11] Melton, T.D., and Carmen, L.M. 2008. Plagiarism, Encyclopedia of the Social and Cultural Foundations of Education. Thousand Oaks, CA: SAGE 2008. p590-91

[12] Bin-Habtoor, A.S and Zaher, M.A. 2012. A Survey on Plagiarism Detection Systems. Int. Journal of Comp. Theory and Eng. Vol. 4, No. 2, p185-188

[13] Verco, K.L. and Wise, M.J. 2005. A comparison of automated systems for detecting suspected plagiarism, The Computer Journal.

[14] Efstathios, S. 2011. Plagiarism Detection Based on Structural Information, Dept. of Inf. and Communication Systems Eng., Univ. of the Aegean, Greece.

[15] Potthast, M., Barrón-Cedeño, A., Eiselt, A., Stein, B., and Rosso, P. 2010. Overview of the 2nd international competition on plagiarism detection. In Proceedings of the 4th Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse.

[16] Schleimer, S., Wilkerson, D.S., and Aiken, A. 2003. Winnowing: Local algorithms for document fingerprinting. In Proceedings of the ACM SIGMOD International Conference on Management of Data, 76-85.

[17] Kupers, R., & Conrad, S. 2012. A Set-Based Approach to Plagiarism Detection. Notebook for PAN at CLEF 2012

[18] Hage, J., Rademaker, P. and Vugt, N. 2010. A comparison of plagiarism detection tools. Technical Report UU-CS-2010-015, Department of Information and Computing Sciences Utrecht University, Utrecht, The Netherlands.

[19] Cosma, G. 2008. An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis (a PhD Thesis), University of Warwick, Department of Computer Science.

[20] Izzat, A., and Zakaria, I.S. 2012. Documents Similarities Algorithms for Research Papers Authenticity. Proceedings of Int. Conf. on Com. & Info Tech (ICCIT 2012) Hammamet, Tunisia. June 26-28, 2012, p210-214

[21] Pataki, M. 2003. Plagiarism Detection and Document Chunking Methods. Computer and Automation Research Institute, Hungarian Academy of Sciences

[22] Won, K.J., Choi, K., Yo, S. and Kim. J. 2013. A Study of Design and Implementation of Korean Plagiarism Detection System. International Journal of Software Eng. & Its Appl., Vol.7, No. 1, p211-220

[23] Mohamed, E.B.M. 2012. Detection of Plagiarism in Arabic Documents, Int.J. Inf Tech & Computer Science, p80-89,

[24] Jeong-II, P., Sang-Wook, K. and Miyoung, S. 2005. Music Plagiarism Detection Using Melody Databases., Knowledge- based intelligent info and eng systems lecture notes in comp science, Vol 3683, p684-693

[25] Asako, O. and Hajime, M. 2011. A Two-Step In-Class Source Code Plagiarism Detection Method Utilizing Improved CM Algorithm and SIM. International Journal of Innov. Computing, Info. & Control, Vol 7, No 8, Aug 2011, p4729-4739

[26] Salha, A., Naomie, S., and Ajith, A. 2012. Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods., IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, Vol. 42, No. 2, p133-149.

[27] Leilei, K., Zhimao, L., Haoliang, Q., and Zhongyuan, H. 2014. Detecting High Obfuscation Plagiarism: Exploring Multi-Features Fusion via Machine Learning., Int. Journal of U & E-Service, Sc & Tech., Vol.7, No.4. pp.385-396.

8. APPENDIX A

Flowchart for Program Module PlagSearch1.m

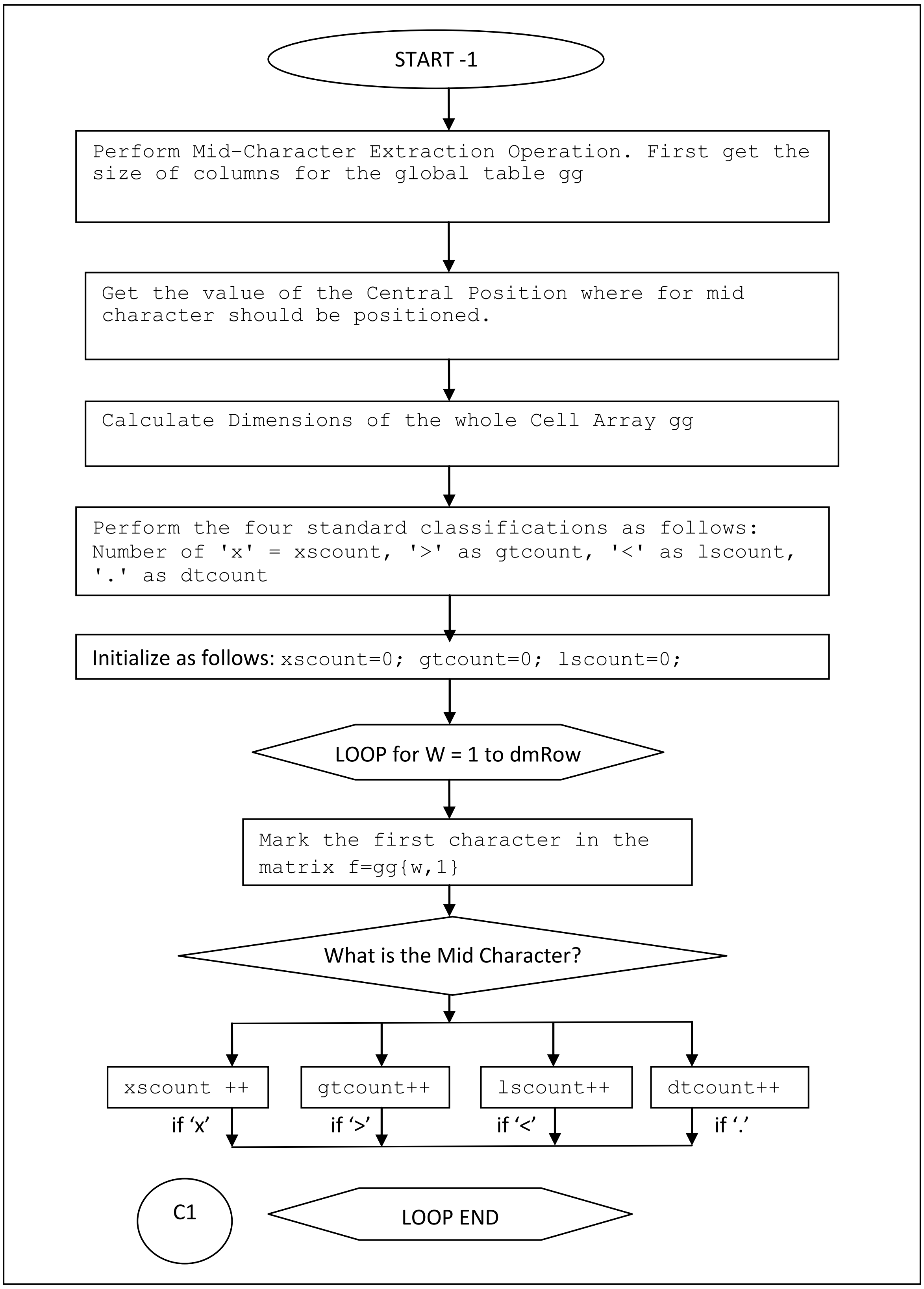

START -1
Perform Mid-Character Extraction Operation. First get the size of columns for the global table gg
Get the value of the Central Position where for mid character should be positioned.
Calculate Dimensions of the whole Cell Array gg
Perform the four standard classifications as follows: Number of 'x' = xscount, '>' as gtcount, '<' as lscount, '.' as dtcount
Initialize as follows: xscount=0; gtcount=0; lscount=0;
LOOP for W = 1 to dmRow
Mark the first character in the matrix f=gg{w,1}
What is the Mid Character?
xscount ++
gtcount++
lscount++
dtcount++
if 'x'
if '>'
if '<'
if '.'
C1
LOOP END

Fig 11F: Flowchart for F-Search -1

Flowchart Continuation

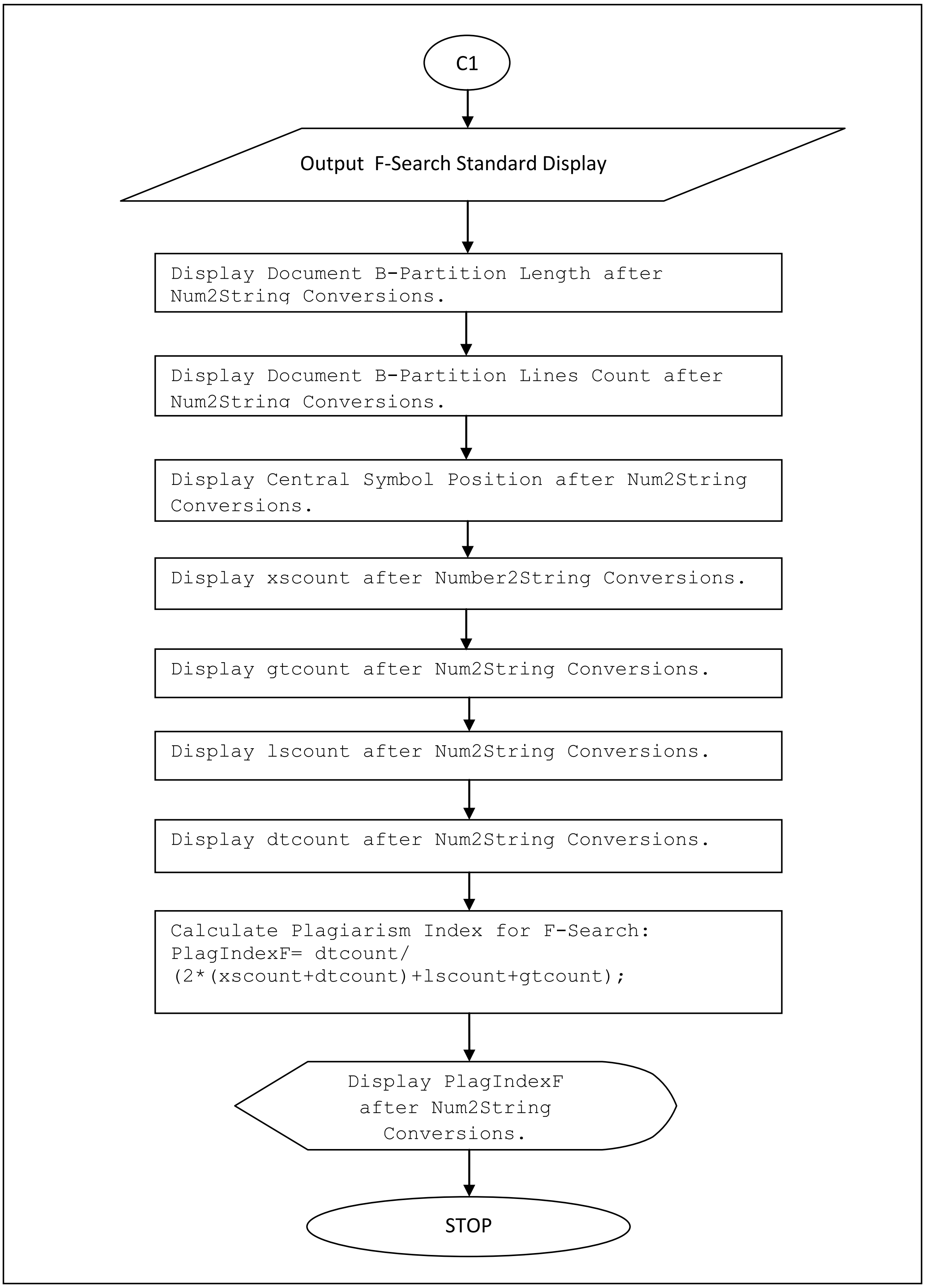

C1
Output F-Search Standard Display
Display Document B-Partition Length after Num2String Conversions.
Display Document B-Partition Lines Count after Num2String Conversions.
Display Central Symbol Position after Num2String Conversions.
Display xscount after Number2String Conversions.
Display gtcount after Num2String Conversions.
Display lscount after Num2String Conversions.
Display dtcount after Num2String Conversions.
Calculate Plagiarism Index for F-Search: PlagIndexF= dtcount/ (2*(xscount+dtcount)+lscount+gtcount);
Display PlagIndexF after Num2String Conversions.
STOP

Fig 12F: Flowchart for F-Search -2

Flowchart for Program Module PlagSearch2.m

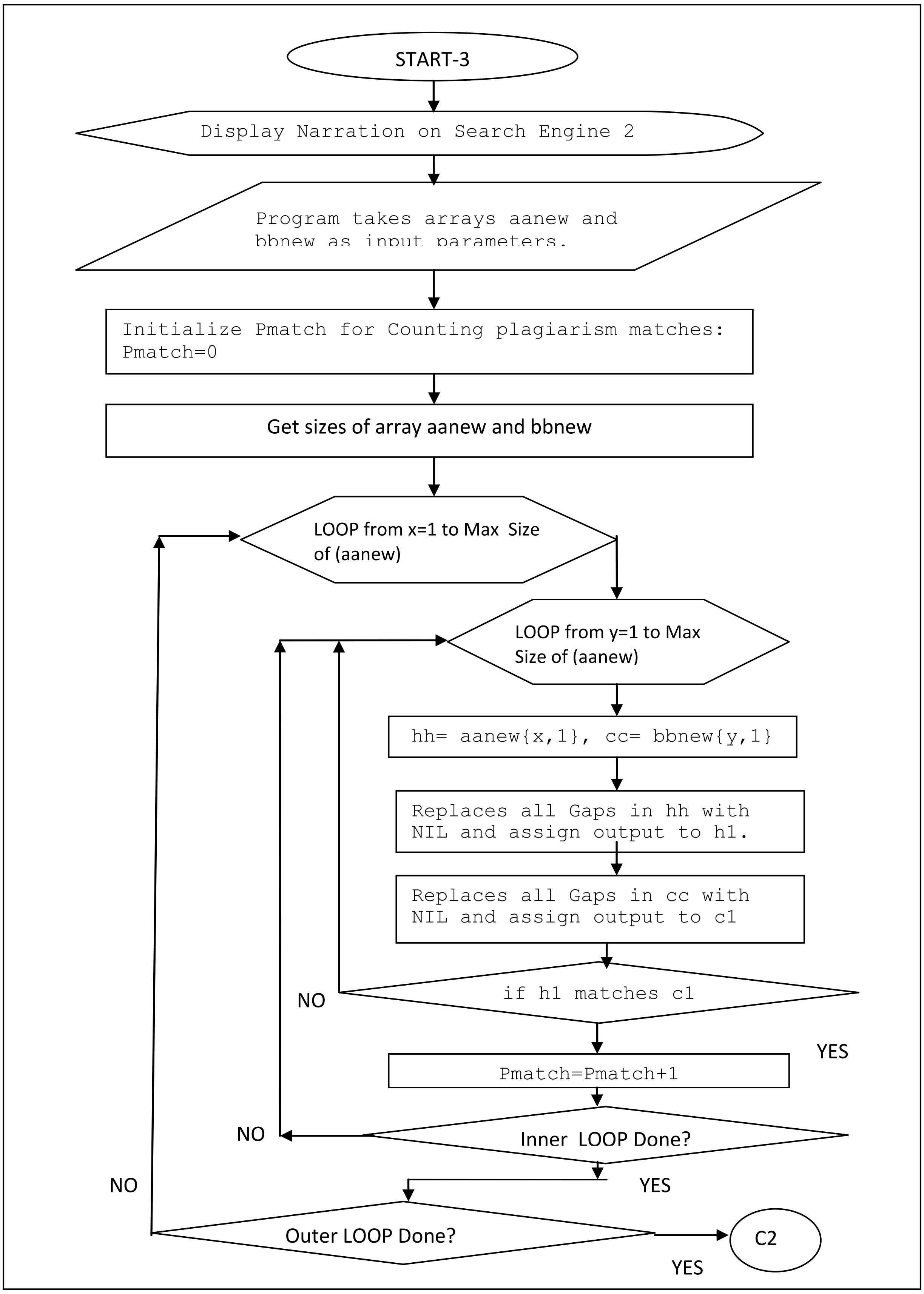

START-3
Display Narration on Search Engine 2
Program takes arrays aanew and bbnew as input parameters.
Initialize Pmatch for Counting plagiarism matches: Pmatch=0
Get sizes of array aanew and bbnew
LOOP from x=1 to Max Size of (aanew)
LOOP from y=1 to Max Size of (aanew)
hh= aanew{x,1}, cc= bbnew{y,1}
Replaces all Gaps in hh with NIL and assign output to h1.
Replaces all Gaps in cc with NIL and assign output to c1
if h1 matches c1
NO
YES
Pmatch=Pmatch+1
Inner LOOP Done?
NO
YES
NO
Outer LOOP Done?
C2
YES

Fig 13F: Flowchart for T-Search -1

Flowchart Continuation

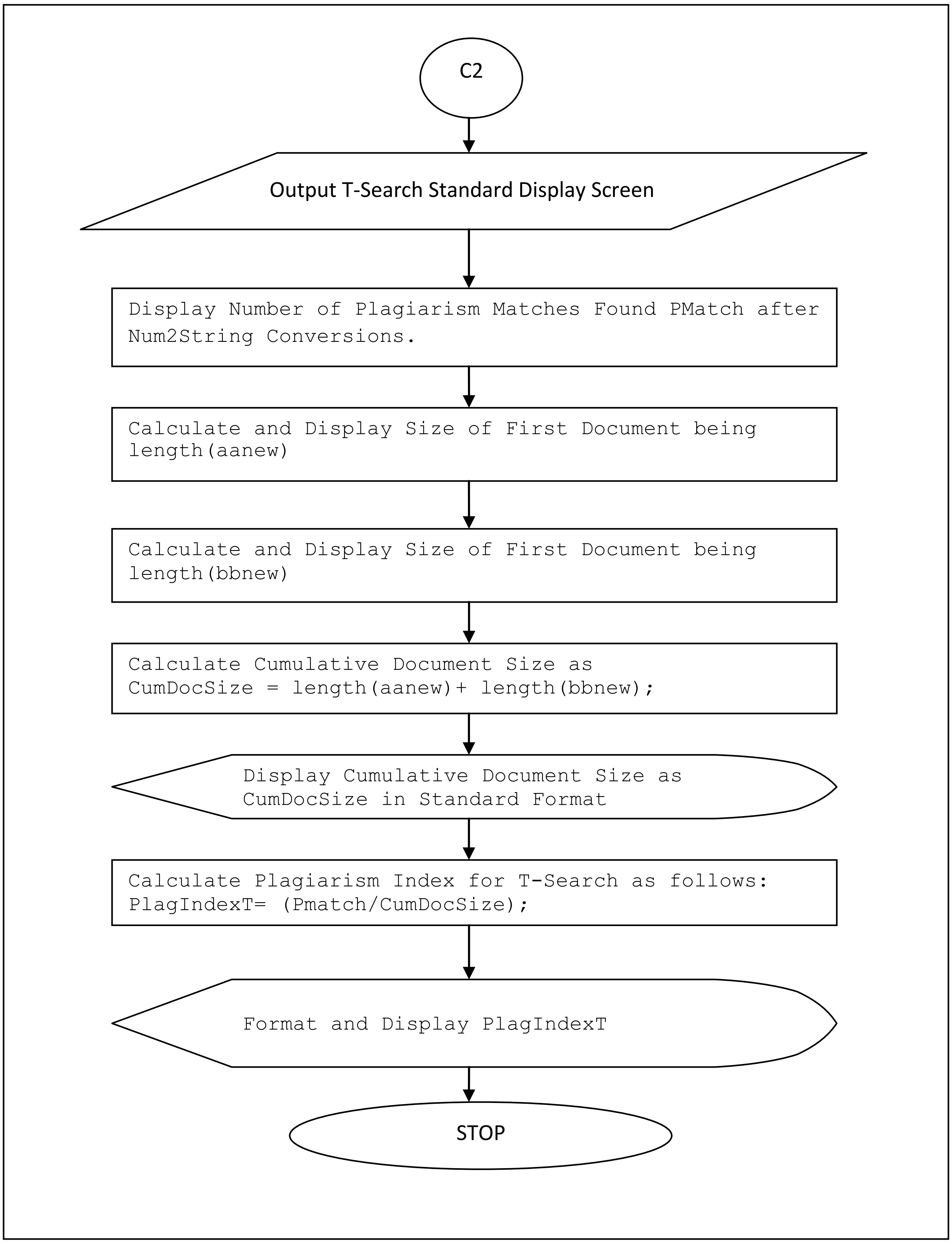

C2
Output T-Search Standard Display Screen
Display Number of Plagiarism Matches Found PMatch after Num2String Conversions.
Calculate and Display Size of First Document being length(aanew)
Calculate and Display Size of First Document being length(bbnew)
Calculate Cumulative Document Size as CumDocSize = length(aanew)+ length(bbnew);
Display Cumulative Document Size as CumDocSize in Standard Format
Calculate Plagiarism Index for T-Search as follows: PlagIndexT= (Pmatch/CumDocSize);
Format and Display PlagIndexT
STOP

Fig 14F: Flowchart for T-Search -2