AN IMPROVED ALGORITHM FOR PLAGIARISM DETECTION USING N-GRAM AND STRING MATCHING

BY

EGONU JOSIAH CHIMEBUKA

20134869258

A THESIS

SUBMITTED TO THE POSTGRADUATE SCHOOL,
FEDERAL UNIVERSITY OF TECHNOLOGY, OWERRI.

IN PARTIAL FULFILMENT FOR THE AWARD OF THE DEGREE
MASTER OF SCIENCE (M.Sc.) DEGREE IN INFORMATION TECHNOLOGY

October, 2016
This is to certify that this work “An Improved Algorithm for Plagiarism Detection using N-gram and String Matching” was carried out by Egonu, Josiah Chimebuka (Reg. No: 20134869258) in partial fulfilment of the requirements for the Award of degree Master of Science (M.Sc.) in Information Management Technology, School of Management Technology, Federal University of Technology, Owerri.

Dr. Ahaiwe Josiah
(Supervisor)

Prof. B.C. Asiegbe
(HOD, IMT)

Prof. C. C. Ibe
(Dean, SMAT)

Prof. (Mrs.) N.N. Oti
(Dean, Postgraduate School)

Prof. (Mrs.) F.N. Ogwueleka
(External Examiner)

09/01/17
Date

09/01/17
Date

Date

Date

12/01/17
Date
DEDICATION

This Project is dedicated to Almighty God for the safe journeys he granted me all through the course of my post-graduate study in FUTO.
ACKNOWLEDGEMENT

But there is a spirit in man: and the inspiration of the Almighty gives him understanding; ~ Job. 32 vs 8.

To sail through the academic waters of the Federal University of Technology Owerri without mentioning the various sailors that manned my Academic ship would be wrong. My very sincere thanks go to the head of the Department of Information Management Technology, Prof. B.C. Asiegbu, and more especially my supervisor, Dr. Josiah Ahaiwe and also to Dr. Charles Ikerionwu for their invaluable contributions to the development of the algorithm used in this work. Big thanks too to Dr. Chidiebere Ugwu of the University of Port-Harcourt for the insight in artificial Intelligence. Finally a big thank you to my Boss, Mr. Robinson Mbato for his understanding and support during the period of my study.
# TABLE OF CONTENT

List of Tables .......................................................................................................................... viii

Chapter 1 .................................................................................................................................. 1

Introduction .............................................................................................................................. 1

1.1 Background of Study ...................................................................................................... 1

1.2 Statement of the Problem .............................................................................................. 2

1.3 Aim and Objective of the Study .................................................................................... 4

1.4 Significance of the Study ............................................................................................... 5

1.5 Research Questions ......................................................................................................... 6

1.6 Scope of the Study ........................................................................................................... 6

1.7 Limitations of the Study .................................................................................................. 7

1.8 Organization of the Study ............................................................................................... 7

Chapter 2 .................................................................................................................................. 9

Literature Review ..................................................................................................................... 9

2.1 Historical Development of Plagiarism Detection. .......................................................... 9

2.2 Theoretical Framework of the Study ............................................................................. 13

2.3 Concepts, Theories, Model approach and Technology .................................................. 22

2.3.1 Document Fingerprinting ......................................................................................... 25

2.3.2 String Matching ......................................................................................................... 32

2.3.3 Stylometry ................................................................................................................ 33

2.3.4 Semantic Analysis ...................................................................................................... 38

2.4 Features Employed in Plagiarism Detection .................................................................. 40

2.5 Plagiarism Detection Softwares: An analysis ................................................................ 42

2.5.1 PlagAware .................................................................................................................. 43

2.5.2 PlagScan .................................................................................................................... 44

2.5.3 ChekForPlagiarism.net ............................................................................................ 45

2.5.8 Empirical Framework of the Study ............................................................................ 47

2.9.9 Research Gap ............................................................................................................. 58

CHAPTER 3 ................................................................................................................................ 60

RESEARCH METHODOLOGY ................................................................................................. 60
3.1 Methodology Adopted..............................................................................................60
3.3 Data collection ........................................................................................................63
  3.3.1 Characteristics of the Population ......................................................................63
  3.3.2 Sampling Design and Procedure ......................................................................66
  3.3.3 Data Collection Instrument .............................................................................67
3.4 System Analysis Procedures ..................................................................................68
  3.4.1 Financial Feasibility Study of the Proposed System ........................................70
    3.4.1 Performance Indicator of the Existing System ..............................................73
    3.4.2 Performance Indicator of the Proposed System .............................................77
CHAPTER 4 ...................................................................................................................79
Design and Implementation ..........................................................................................79
4.1 Data Presentation and System Analysis .................................................................79
  4.1.1 Data Preparation ..............................................................................................79
4.2 System Design .........................................................................................................84
  4.2.1 Logical System Design of Proposed System ..................................................84
    4.2.1.1 High Level Models ....................................................................................85
    1. Admin.mdf .........................................................................................................91
    2. Ngrams.mdf .......................................................................................................91
    3. References.mdf ................................................................................................92
    4. Rank.mdf ............................................................................................................92
    4.2.1.8 Entity Relationship Diagram ......................................................................93
    4.2.1.9 System Algorithm, Flowchart and String Matching Method ......................94
      1. System Algorithm .............................................................................................94
        a. Ngram Splitting and String Matching Algorithm ...........................................94
        b. Paragraph Splitting Algorithm .....................................................................95
        c. Word Splitting Algorithm .............................................................................95
        d. SubAlgorithm for Matching string ................................................................96
    4.2.2 Program Flowchart .........................................................................................96
    4.2.3 Physical System Design ..................................................................................98
    4.2.4 Design Considerations ...................................................................................98
List of Tables

Table 2.0: Educational level difference in awareness of plagiarism among postgraduate students .................................................................................................................................................. 21

Table 2.1: Plagiarism Awareness level among Post-graduate students in Nigerian universities .................................................................................................................................................. 22

Table 2.2: Reason why students Plagiarize ................................................................................................................................................................................................. 22

Table 2.3: Chunk sizes for fingerprinting method .................................................................................................................................................. 28

Table 2.4: Overview of Chunk selection methods for fingerprinting method. ........................................................................................................... 30

Table 2.6: Comparative analysis of Plagiarism Detection Softwares ........................................................................................................... 42

Table 2.7 Comparison of various plagiarism detection software ranked on a scale of 1 - 5 .................................................................................................................................................. 47

Table 2.8. Common n-grams in different documents .......................................................................................................................... 57

Table 3.0 Results for performance measures for various algorithms developed at PAN-10 .................................................................................................................................................. 76

Table 2.6: Comparative analysis/attributes of Plagiarism Detection Softwares. \textbf{Error! Bookmark not defined.}

\textbf{Table 3.0} Comparison of various plagiarism detection software ranked 1 - 5 (Asim et al., 2011) .................................................................................................................................................. \textbf{Error! Bookmark not defined.}

Table 3.1: analysis and comparison of existing systems and their heuristic retrieval models .................................................................................................................................................. 83

(Potthast, et.al, 2010) ........................................................................................................................................................................................................ 83

Table 3.2 Performance results for the external plagiarism detection task ........................................................................................................... 84

Table 4.0: Table showing the entities for storing the reference text ........................................................................................................... 92

Table 4.1: Table showing the entities for storing and ranking the plagiarized text ........................................................................................................... 92

Table 4.2. N-gram = 1 for 30 iterations using 392 documents from the PAN-09 Corpora .................................................................................................................................................. 110

Table 4.3. N-gram = 2 for 30 iterations using 392 documents from the PAN-09 Corpora .................................................................................................................................................. 111

Table 4.3. N-gram = 3 for 30 iterations using 392 documents from the PAN-09 Corpora .................................................................................................................................................. 111

Table 4.4. N-gram = 4 for 30 iterations using 392 documents from the PAN-09 Corpora .................................................................................................................................................. 113
Table 4.5. N-gram = 5 for 30 iterations using 392 documents from the PAN-09 Corpora.................................................................................................................................114
Table 4.6. N-gram = 6 for 30 iterations using 392 documents from the PAN-09 Corpora.................................................................................................................................115
Table 4.7. N-gram = 7 for 30 iterations using 392 documents from the PAN-09 Corpora.................................................................................................................................116
Table 4.5 Summary of Highest and Lowest Values of Precision and Recall of Result ........................................................................................................................................117
Table 4.5: Best three values presented by participants of PAN-09 .......................118
List of Figures

Fig 2.0, Levels of Plagiarism and their sources .................................................................14
Fig 2.1: Graphical representation of Student’s in Nigerian Universities and Academic writing requirements .................................................................19
Fig 2.2: Percentage of Nigerian students who may have copied or plagiarized at one time or the other .................................................................20
Fig 2.3: Reasons why Nigerian Students plagiarize. Error! Bookmark not defined.
Fig 2.4 Classification of Plagiarism detection systems .....................................................24
Fig 2.5 Classification of Plagiarism detection systems .....................................................26
Fig 2.6 Concept of Outliers in Plagiarism detection ......................................................37
Fig 2.7: Common features exploited in plagiarism detection ........................................40
Figure 2.8: The style change function of document solid line ........................................52
Figure 2.9: The style change function of the plagiarism-free document (a false positive). The dashed line indicates the threshold of the plagiarized passage criterion .................................................................52
Figure 2.10: The style change function of document a plagiarism-free document ....53
Figure 2.11: Detected similarity for the non-plagiarised pairs of documents.
Parameter n is the n-gram length and the o parameter is the number of letters used as o set .................................................................55
Fig 3.0 Rapid application development methodology Error! Bookmark not defined.
Fig 3.1 Phases of Agile Software development (Mihai, 2014) .........................................61
Fig 3.3: System analysis Procedure ..............................................................................69
Fig 3.3 precision, recall & F-measure for MOSS ..............................................................75
Fig 3.5 Precision, recall & F-measure for Code Match ................................................75
Fig 3.7 Highest F-measures recorded by existing systems ...........................................76
Fig 4.0 graphical plot of granularity from PAN-09 .........................................................82
Fig 4.1 Logical system design showing various modules in the plagiarism detection system .........................................................................................85
Fig 4.2 Flowchart of the plagiarism detection system ....................................................97
Fig 4.3 Plagiarism selection setting for different n-gram levels ..................................100
Fig 4.4 Input space for the suspicious document, to be entered as text or uploaded 100
Fig 4.5 Output Report on Plagiarised sections. .......... Error! Bookmark not defined.

Fig 4.6 A sample plagiarism detection case from the PAN-09 Corpora. ............ Error! Bookmark not defined.

Fig 4.7 Detections and highlight of plagiarised sections based on n-gram = 2 .. Error! Bookmark not defined.

Fig 4.8 Detections and highlight of plagiarised sections based on ngram = 2 ....... 101

Fig 4.9 Original Text fed into the plagiarism detection engine. ..... Error! Bookmark not defined.

Fig 4.10 Detection Density using n-gram = 2 .......... Error! Bookmark not defined.

Fig 4.11 Detection Density using n-gram = 3 ....................................................... 102

Fig 4.12 Detection Density using n-gram = 4 ....................................................... 102

Fig 4.13 Detection Density using n-gram = 5 ....................................................... 103

Fig 4.14 Detection Density using n-gram = 6 ....................................................... 103

Fig 4.15 Detection Density using n-gram = 7 ....................................................... 104

Fig 4.16 Precision values for n=1,2,3,4,5,6 and 7 .............................................. 121

Fig 4.17 Recall values for n=1,2,3,4,5,6 and 7 ..................................................... 122
ABSTRACT

To Plagiarise is to include another person’s work, idea, method or results without due acknowledgement. In this work, we developed an innovative algorithm to detect external plagiarism in documents, using a combination of Word-n-gram and string matching technique. The suspicious document was split into sentences and further split into word n-gram and each n-gram was used to perform a pattern match on a standard PAN-09 corpora. We discovered that the best plagiarism detection occurred when we used the word-2-gram and word-3-gram for heavily disguised plagiarised passages. The detected passages are then tagged according to the number of the retrieved texts in the documents. To check for the level of similarity, we score the plagiarised passages according to the number of the passages where the plagiarised texts occur. A high precision value range of 0.93 – 0.95 was achieved, but there were indications that with a low recall score, many documents were left out of the retrieval process. But the ones retrieved were very relevant to the subject being investigated. This approach is a novel approach to detecting plagiarised passages that have been re-worded and returned very close match to the passages that were plagiarised in the document. It was also found to be useful in monolingual plagiarism detection and mild to medium obfuscation but very weak in heavily obfuscated documents and cross lingual plagiarism detection unless a language translator was integrated to help translate.

Keywords: Plagiarism Detection, N-grams, String Matching, Algorithm
Chapter 1

Introduction

1.1 Background of Study

Information has increased exponentially, today; there is a geometric increase in knowledge acquisition, information gathering, dissemination and sharing. Many authors are becoming aware of the need for taking responsibility for their ideas and the implications of lack of originality in academic and professional research work. Lawsuits are being filed almost every day against stealing of copyright work, Public office holders are losing their offices because of lack of Ingenuity and integrity in their previous academic works (Weber, 2010). Academic integrity involves a number of undeniable values some of these values include, but are not limited to truth, trust, respect, fairness, honesty and responsibility and ideals that should be upheld by all educational stakeholders. “Academic integrity involves ensuring that in research, teaching and learning, both staff and students act in an honest way (Bretag et al., 2013). They need to acknowledge the intellectual contributions of others, be open and accountable for their actions, and exhibit fairness and transparency in all aspects of scholarly endeavor” (Exemplary Academic Integrity, 2013). Academic integrity ensures public trust in the credibility of scholarship at all levels of education including the research process and its outcomes (Bretag, et. al., 2010). Today, there is a general call for sanity in the educational system as the Academia is beginning to call for
original works from both the student and the Lecturers. The invention of software programs and algorithms that help detect copied or plagiarized text from a wide variety of data sources are making it increasingly difficult for students to Copy. There are many plagiarism detection software with improved algorithms that are available for use on the web, many of them have their shortcomings, some of them are time consuming; others use a technique which is memory consuming or which does not integrate well into other Plagiarism detection software to improve the detection method. This research work seeks to develop a concept for plagiarism detection by using fragmented word sizes known as n-grams and matching them over standardized text corpora for plagiarism detection. This combined technique will produce a pattern matching algorithm that will improve the accuracy of detection of plagiarized documents from a pool of data under investigation. The algorithm can be integrated and applied to the existing plagiarism detection systems in Nigerian universities to detect plagiarism in assignments and academic work submitted by Nigerian Students and also to check for plagiarism in Journals run by Nigerian Editors and professional.

1.2 Statement of the Problem

Plagiarism is a problem that has dire consequences for the academic community, whether in the arts, sciences, or technology (Weber, 2010). The consequences are by far greater than the plagiarist can think of. This calls for a renewed effort in applying all research and development effort in developing appropriate tools to stop or reduce plagiarism. (Gipp, 2014) noted that Plagiarism detection systems today only detect at
most copy and paste plagiarism, but leave off other forms of plagiarism such as idea plagiarism and disguised plagiarism.

Today, many papers, journals and academic papers are being published, as a result of research efforts. These works are being added to an existing pool of academic research work which has been done previously by various individuals in years past. The new findings and improvements are published daily, the publications must be adequately referenced and the authors must acknowledge the owners of the Ideas they reference as the original idea owners. When proper acknowledgement is done, the original owner of the idea gets the credit, but when an idea or work that has been painstakingly done is copied word for word and represented as the copier’s ideas, we term it plagiarism. This could lead to several problems both in the industry and the academia. The problems associated with plagiarism are numerous and must be tackled quickly and immediately.

In the event of plagiarism, professional and academic integrity are at stake. The plagiarist is blacklisted as an impostor and could even lose public office or have his degree revoked. One case was that of Germany’s former minister of defense, Guttenberg, whose Doctorate degree was revoked in 2010 because of plagiarizing his doctoral thesis (German Defence Minister, 2015). In February 2011, it was discovered that Guttenberg’s thesis contained texts of other authors without citation. This led to plagiarism accusations and charges against him, which incited the public and his political opponents against him, and aroused a public debate as to the genuineness of
his Doctorate degree and his Political Office. (German Plagiarism Minister, 2011). Guttenberg denied any charges of plagiarism, but asked the university to withdraw his thesis and revoke his Ph.D., admitting “severe errors in workmanship”. The university finally revoked his Ph.D. title on February, 23 2011, announcing that there were going to be investigations to be done. On March 1, 2011, Guttenberg resigned as the minister of defense (German Plagiarism Minister, 2011).

Plagiarism has many far reaching consequences that have been widely reported. When a person is accused of plagiarism, this ruins his career if he is investigated and found guilty. If he is into medical sciences, his medical license can be revoked and his certificate withdrawn. For those in the academia, to lose the high honor of publishing means death of one’s career if there is a discovery that the individual is busy plagiarizing articles and presenting them as his. To curb plagiarism, we have to be equipped with the right tools and techniques for plagiarism detection.

Some of the problems that led to this research design work are:

1. Develop a proper plagiarism detection software for archival and detection of plagiarism in Nigeria.
2. Develop and indigenous and proven algorithm for Plagiarism detection
3. To manage make available plagiarism detection software free of subscriptions and rental payments.

1.3 Aim and Objective of the Study
This study’s aim is to propose an improved way of detecting plagiarism in Nigerian Institutions using the n-gram and pattern matching algorithm in strings. Many traditional Plagiarism detection tools detect plagiarism only in Global documents, using various means of detection and do not allow you an access to your own customized corpora of documents. This makes it very difficult for students and researchers alike to have access to an improved and fast plagiarism detection system. The summarized Objective of this study is to design, develop, test, implement, and evaluate an improved plagiarism detection approach that is capable of detecting exact and disguised forms of plagiarism in local corpora, using the n-gram and string pattern matching algorithm.

To achieve the objective above, the following research sub-objectives have to be met:

1: Analyze the existing plagiarism detection techniques, algorithms and software and identify their strengths and weaknesses.
2: Develop an improved plagiarism detection concept based on the n-gram and string matching technique.
3: Design an improved detection algorithm that employ the theoretical concept introduced in 2 and are fitted to detect the plagiarism forms currently understudy.

1.4 Significance of the Study

Plagiarism is ravaging the Academia. There would be a decay of knowledge institutions if plagiarism is not nipped in the bud (McCabe, 2005). This research
work seeks to improve on existing algorithms for the intent of detecting and reducing Plagiarism in Nigerian Universities.

1.5 Research Questions

To be able to achieve the objectives of the study, the following research questions must be asked and answered.

**Research Question 1:** Is it possible to split a document into words and sentences and identify plagiarism from the split words?

**Research Question 2:** Are there available string matching algorithms that can be implemented and applied Research Question 1, to detect and retrieve information quickly?

**Research Question 3:** Can we cluster the passages where the plagiarized text was copied from into groups of relevant sentences, thereby making it possible to easily reference the texts copied from.

**Research Question 4:** What is the best number of words that will allow the algorithm perform at maximum speed and have the highest precision?

**Research Question 5:** What is the performance index for measuring the success rate of data retrieval/ matching for very large documents?

1.6 Scope of the Study
The study will be limited to the identification of plagiarism in documents with texts in one language. It will seek to attempt plagiarism detection in heavily obfuscated documents, but will not be extended to include a search corpora over the internet, or to include a multi-lingual plagiarism detection approach. The study will be limited to text segmentation using n-gram approach, string matching and result classification or result clustering for improved analysis. This work does not include data-minning methodologies and processes.

1.7 Limitations of the Study.

This research work takes into consideration text-based plagiarism detection. The work is limited in that it cannot be applied to non-machine readable forms, such as Idea plagiarism, research result plagiarism and graph data.

During the implementation of the software, one limitation faced was the limitation of connecting to a corpora of data archived by search engines, restrictions exist on how to use these search results, it may be helpful to develop a custom WebCrawler to archive webpages for pattern matching purposes.

1.8 Organization of the Study

This work is organized into 5 chapters.

Chapter one presents the introduction, which consists of the background study, the scope of the work, the relevance of the study, the premise for the research and other information necessary in the successful understanding of the study.
In Chapter Two, we present the literature review, which includes the relevant theories and model of the problem under study.

Chapter Three discusses the various research methods applied in the Study and the formation of the algorithm.

The Chapter Four presents the test of the algorithm, the analysis of the precision, recall and comparison with previous plagiarism detection systems.

We finally conclude by presenting the conclusion, where the formulated algorithm can be applied, how it can be integrated into existing plagiarism detection software and opportunities for further research based on the study. All these are presented in the fifth chapter which concludes the work.
Chapter 2

Literature Review

2.1 Historical Development of Plagiarism Detection.

"Plagiarism is the act of taking the writings of another person and passing them off as one’s own. The fraudulence is closely related to forgery and piracy-practices generally in violation of copyright laws” (Britannica, 2016).

Fishman (2009) defines plagiarism as the use of words, ideas, or work products attributable to another identifiable person or source, without attributing the work to the source. In a situation where there is a legitimate expectation of original authorship, in order to obtain some benefit, credit, or gain which need not be monetary. Since plagiarism entails copying verbatim, someone else’s work and passing them off as one’s own, it has become necessary that all academic journals and documents that are submitted be tested for plagiarized content with a reliable plagiarism detection tool.

In the early 70’s when computer science began to be a thing of research and interest, (Halstead, 1977) described the earliest automatic plagiarism detectors. These plagiarism detectors leveraged on software science tools and technologies and software processes that were developed by Morris Halstead, to detect the degree of similarity that existed in text and software programs. Some quantities were suggested by Halstead as the quantities to be measured. These quantities are:
\[ \eta_1 \text{= number of unique operators} \]
\[ \eta_2 \text{= number of unique operands} \]
\[ N_1 \text{= number of operator occurrences} \]
\[ N_2 \text{= number of operand occurrences} \]

These four metrics were used by Ottenstein (1988) to develop the first automated plagiarism detection system. Ottenstein observed pairs of programs with identical values for each of \( \eta_1, \eta_2, N_1 \) and \( N_2 \) are assumed to be similar and therefore deserving closer inspection.

Berghel and Sallach (1992) experimented further and discovered that other metrics such as the number of loops and the number of procedures could be combined to find more accurate measures of similarity between pairs of programs. The systems developed by Ottenstein and Berghel have been referred to as attribute-counting-metric systems. Another type of automatic plagiarism detection system used attribute counting metrics, but had an additional feature which was comparing the representations of the program structure, so as to generate a better similarity score. (Donaldson et al., 1981) developed a system that uses eight attribute counting metrics and also generates a representation of the software program. Each letter in the string was a representative of an occurrence, whether once or many times, and these can be procedure calls, methods, variable declarations or even assignments. If the string representation matches or if the metric counts are similar, we can conclude that the pair was plagiarised.
Today, most of the common plagiarism detection systems use structure metrics; this is achieved by comparing the string representation of program structures. An exact match is not needed all that is needed is just to check how similar the token string are.

In (Hermann et al., 2006), Robinson and Soffa developed a system that was designed to assess the code quality of FORTRAN programs. It does this by verifying the quality of the code, detecting possible plagiarism and also makes suggestions where necessary on how improvements can be done.

An improvement on the work of Robinson and Soffa was what Donaldson, Lancaster and Sposato did in 1981. They created the first plagiarism detection system that leveraged on structure-metrics. Using attribute-counting metrics, the software was able to scan source files to store information about various lines of code and statement and later analysed for similarity.

Whale (1990) developed a software called “Plague”, which generates structure profiles from the input programs composed of structural information and then transforms the code into tokens that are sequenced. Another algorithm called Running-Karp-Rabin Greedy-String-Tiling was developed by wise in 1991 to match sequences of tokens. The algorithm was a breakthrough in string matching and is still used in search engines and software systems today. In Hermann et al., (2006), Prechelt developed a software for plagiarism detection which he termed JPlag. JPlag works by transforming the programs into a sequence of tokens and compares them with a greedy-string-tiling algorithm, to determine similarity.
Today, various documents are accessible over the internet and as such can be copied, passed off as an original work without attributing the idea to the original author. Plagiarism can be considered as one of the numerous heinous electronic crimes, like computer hacking, computer viruses, spamming, phishing, copyrights violation and others crimes. It is also defined as the act of taking or attempting to take or to use (whole or parts) of another person’s works, without referencing or citing him as the owner of this work (Snasel et al., 2011). It may be directly copying and pasting the selected or plagiarized text, modifying and removing some selected filler words or keywords from the source of the information which may come from the internet, academic books, magazine, newspaper, research, journal, personal information or ideas jotted down in a diary.

The Merriam-Webster Online Dictionary defines plagiarism as to steal and pass off (the ideas or words of another) as one’s own. To use another’s person’s production without crediting the source. To commit literary theft to present as new and original an idea or product derived from an existing source. 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 (Gipp, 2011).

Many times students are unsure about whether or not they are plagiarizing and failure to correctly give credit to your sources can result in academic sanctions.
2.2 Theoretical Framework of the Study

Plagiarism can take on various forms, it can include disguising one’s work, copying the part or whole, deleting some part of the work and presenting it as yours, using some other people’s research data or findings, or simply not citing resources properly (Gipp, 2011).

Plagiarism does not need to occur intentionally to be labelled as intentional. Most plagiarists do so out of ignorance or because they think it does not matter really matter.

According to McCabe et al (2015), a research carried out among about 82,000 students, recorded 40% of undergraduates and approximately 25% of graduates were engaged in plagiarism within 12 months prior to the study. Results of other studies range as high as ~90% of the respondents self-reporting that they plagiarized at one time or the other.

In the academia, there are numerous cases of plagiarism that have become widely publicized in the media. A plagiarism check on 285,000 scientific texts from arXiv.org, gave results of more than 500 plagiarized texts papers very likely to have been plagiarized by various academic authors. In addition, 30,000 documents (about 19.9% of the collection) were found to be duplicates or containing: “…excessive self-plagiarism” (Sorokina et al., 2005).
Fig 2.1, Levels of Plagiarism and their sources. (Ali et al., 2014)

Ali Bukar (2014) in Figure 2.1 states the various sources of student plagiarism, with the internet being the largest source of plagiarized content. Though the practice of Plagiarism which have been noticed to appear in many forms have revealed quite a great number of illegal means of using uncited text, some others cite, but do not do so properly. Some of the types of plagiarism identified are

A. **Literal Plagiarism:** Weber (2010) describes it as the undue copying of text with very little or no disguise. Gipp (2014) categorized this form of plagiarism into two:

a. **Copy & Paste Plagiarism:** This form of plagiarism specifies the act of taking over text verbatim from another author (*Weber*, 2010).
b. **Shake & Paste**  This form of plagiarism is done by copying and merging sentences or paragraphs from different sources with slight adjustments necessary for forming a coherent text (Weber, 2010).

B. **Disguised plagiarism** denotes practices intended to mask copied segments of text and diagrams (Lancaster, 2003). There are various masking techniques that are easily identifiable. These are:

a. **Expansive plagiarism** refers to the insertion of additional text into or in addition to copied segments (Lancaster, 2003).

b. **Contractive plagiarism** describes the summarization or tailoring of copied material to a specific number of words (Lancaster, 2003).

c. **Mosaic plagiarism** entails merging various text parts from various sources and thus confusing and muddling up the whole text, causing obfuscation with the intention of hiding the act of Plagiarism from traditional plagiarism detection softwares. Other time, the act of changing the order and position of the words are done by the plagiarists or substituting words with synonyms or entering/deleting filling words (Weber, 2010)

d. **Technical disguise** summarizes techniques for hiding plagiarized content from being automatically detected by exploiting weaknesses of current text-based analysis methods. This is done by substituting characters with graphically identical symbols from foreign alphabets or inserting letters in white font color (Bretag and Mahmud, 2009).
e. **Undue paraphrasing** defines the intentional rewriting of foreign thoughts in the vocabulary and style of the plagiarist without giving due credit for concealing the original source (Lancaster, 2003).

f. **Translated plagiarism** is defined as the manual or automated conversion of content from one language to another intended to cover its origin (Weber, 2010).

g. **Idea plagiarism** encompasses the usage of a broader foreign concept without due source acknowledgement. Examples are the appropriation of research approaches and methods, experimental setups, argumentative structures, background sources, without due source acknowledgement. (Maurer et al., 2006).

h. **Self-plagiarism** characterizes the partial or complete reuse of one’s own previous writings not being justified by scientific goals, e.g. for presenting updates or providing access to a larger community, but primarily serving the author for artificially increasing citation counts or receiving grants in a academic environment. (Bretag et al. 2013).

Academic plagiarism has been around for centuries, but became very profound in the 20th Century (Gipp, 2014). “Since 1920, researchers in academic behaviour have critically analyzed the topic and found out that there is a great deal of plagiarism going on in the academia” (Gipp, 2014).

A research using self-report surveys showed that several students engage in collegiate cheating behaviors and also extensive plagiarism (Crown, 1998). Plagiarism
counts and charges have increased against various public office holders in the west and European countries and Nigeria is not an exception.

The various cases that have been leveled against the plagiarists in recent times are a warning to future plagiarists or those considering the idea. One of the cases was that of Germany’s former Minister of defense Guttenberg whose Ph.D. was revoked in 2010 for plagiarizing his doctoral thesis. In February 2011, it was discovered that Guttenberg’s thesis contained texts of other authors without citation. This led to plagiarism accusations against Guttenberg, which in turn incited the public and his political opponents to commence a heated debate. Guttenberg denied any plagiarism (Guttenberg, 2010), but asked the University of Bayreuth to revoke his title, admitting severe errors in workmanship. The university revoked his title on 23 February 2011, announcing further investigations. Due to public and political pressure, Guttenberg resigned on 1 March 2011 as Germany’s Minister of Defense (Guttenberg, 2010).

Unfortunately, in Nigeria, the case of plagiarism is unpleasing. Many Nigerian academics are being found guilty of plagiarism on a daily basis. Sanusi, Lamido Sanusi, the former Central Bank governor was accused of plagiarizing his thesis by one Professor Dike Njoku of National University Sacramento, USA (Sanusi, 2012).

Professor Dike, an Adjunct Professor at the School of Engineering and Technology, National University, Sacramento, US, in a statement of claims filed by his lawyer, Mr. E.U. Chinedum, said that Sanusi breached his copyrights on two different occasions on November 26, 2010 and December 10, 2010 when he presented lectures he delivered at the 8th convocation ceremony of Igbinedion University, Okada, Edo
State, on November 26, 2010, where Sanusi presented a paper titled “Growth Prospects for the Nigerian Economy” (Sanusi, 2012). The second occasion was on December 10, 2010 at the Convocation Square, Abubakar Tafawa Balewa University, Bauchi, where Sanusi again, presented a paper titled “Global Financial Meltdown and the Reforms in the Nigerian Banking Sector (Sanusi, 2012).

Orim (2013) agrees that most of Nigerian students have an incomplete comprehension of what plagiarism is and what forms it takes. Surprisingly, a number of lecturers at Nigerian universities perceived plagiarism in a similar way to their students and had a mostly incomplete understanding of it. There was also very little agreement among lecturers on an institutional definition of plagiarism. It is hardly surprising then that the students could not understand why some ways of writing are considered as plagiarism. As a result, a number of students claimed they might have plagiarized in the past as they were unaware about the concept or its ethical implications. Lack of awareness about plagiarism among participating Nigerian students was of the order of 81% and 67% in the two UK universities, respectively, and 40% in the Nigerian universities. A number of the students said they had only heard about plagiarism when the researcher asked the question. Hence, the main types of plagiarism practiced by the students were mostly due to a lack of acquisition of the relevant skills for the appropriate use of sources. Since the predominant form of assessment in Nigeria is examinations, many of the students are not given opportunities to develop academic writing skills (Orim, 2014).
“…a lot of students are not aware of the problem of plagiarism, because majority of the work done in most of these institutions is mainly exam based and some of these exams are even open book exams. So they are not totally aware of the issue of plagiarism until they come to the point that they are writing up their theses” (Orim, 2014)

Many Nigerian students are unaware of the proper writing requirements of a well-tailored research work; hence they find themselves entangled in the yoke of plagiarism.

![Graphical representation of Student’s in various universities and Academic writing requirements](image)

<table>
<thead>
<tr>
<th>Reason for Plagiarism</th>
<th>UK university Case 1 (%)</th>
<th>UK university Case 2 (%)</th>
<th>Nigerian universities (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree (%)</td>
<td>2</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>Disagree (%)</td>
<td>13</td>
<td>17</td>
<td>39</td>
</tr>
<tr>
<td>Agree Somewhat (%)</td>
<td>4</td>
<td>6</td>
<td>17</td>
</tr>
<tr>
<td>Agree (%)</td>
<td>76</td>
<td>61</td>
<td>34</td>
</tr>
<tr>
<td>Strongly Agree (%)</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Fig 2.1: Graphical representation of Student’s in various universities and Academic writing requirements (Orim, 2014)

Reasons for plagiarism given by some students included no clearly stated academic writing requirements, no efficient checks being in place and discouragement of creativity (Orim, 2014) and (Idiegbeyan-ose et al., 2016). In most cases, the
Nigerian universities were found to still use the traditional approach to teaching and learning. A number of senior management staff according to (Orim, 2014) stated that corruption in the society in which they studied was also a factor. Another type of student plagiarism was the re-submission of work previously submitted by students studying at other universities. Additionally, it was observed that most of the Nigerian students studying in Nigerian universities were more likely to plagiarize than Nigerian students studying in the UK.

![Pie chart showing percentage of Nigerian students who may have plagiarized at one time or the other](image)

Fig 2.2: Percentage of Nigerian students who may have plagiarized at one time or the other (Orim, 2014)

In Idiegbeyan-ose et. al (2016), the level of awareness of what plagiarism IS and its consequences among Post-graduate students increased as the educational level of Nigerian students increased. Doctoral students showed that they had the highest level of awareness about the subject, while Master of Science (M.Sc.) and Master of
Philosophy (M.Phil.) students had considerable amount of awareness, Post Graduate Diploma students were completely unaware of what plagiarism meant and its consequences to their academic papers.

Table 2.1: Educational level difference in awareness of plagiarism among postgraduate students. (Idiegbeyan-ose et al., 2016)

<table>
<thead>
<tr>
<th>Degree</th>
<th>High</th>
<th>Average</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGD</td>
<td>--</td>
<td>--</td>
<td>8</td>
</tr>
<tr>
<td>Masters</td>
<td>10</td>
<td>42.4</td>
<td>4.4</td>
</tr>
<tr>
<td>M.Phil./Ph.D.</td>
<td>8.1</td>
<td>4.6</td>
<td>--</td>
</tr>
<tr>
<td>Ph.D.</td>
<td>22.3</td>
<td>0.2</td>
<td>--</td>
</tr>
</tbody>
</table>

In answering the question, are Post-graduate students aware of plagiarism and its consequences in Nigerian universities? The findings revealed as presented in table 3 that almost half of the students or 46.4% reported average level of awareness. However, 38.8% reported high level of awareness and only 14.8% reported low level of awareness. This implies an average awareness of plagiarism issues among postgraduate students in Nigerian universities.
Table 2.1: Plagiarism Awareness level among Post-graduate students in Nigerian universities (Idiegbeyan-ose, 2016).

<table>
<thead>
<tr>
<th>Variation</th>
<th>Frequency</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Level of Awareness</td>
<td>140</td>
<td>41.4</td>
</tr>
<tr>
<td>Average Level of Awareness</td>
<td>156</td>
<td>46.2</td>
</tr>
<tr>
<td>Low Level of Awareness</td>
<td>42</td>
<td>12.4</td>
</tr>
</tbody>
</table>

Some of the identifiable reasons why Nigerian Students Plagiarise according to (Idiegbeyan-ose et al., 2016) are pressure to meet submission deadlines, Inadequate or poor writing skills and lack of knowledge on what constitutes plagiarism.

Table 2.2: Reason why students Plagiarize (Idiegbeyan-ose et al., 2016)

<table>
<thead>
<tr>
<th>Reason</th>
<th>Frequency</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure to meet deadlines</td>
<td>116</td>
<td>35</td>
</tr>
<tr>
<td>Inadequate writing skills</td>
<td>106</td>
<td>32</td>
</tr>
<tr>
<td>Lack of knowledge on what constitute plagiarism</td>
<td>108</td>
<td>33</td>
</tr>
</tbody>
</table>

2.3 Concepts, Theories, Model approach and Technology.

There are various theories, concepts and algorithms that can be employed in detecting plagiarism in suspicious texts. Some of these methods could be manual or automatic (Stein, 2011). The manual plagiarism detection approaches in a quoted text is quite cumbersome and painstaking, prone to errors and usually takes time to accomplish.
The Automatic Plagiarism detection approach to finding and detecting plagiarism in quoted texts can be applied by employing relevant plagiarism detection engines for text comparison and plagiarism detection. There are many software systems that employ various algorithms in finding plagiarism; some of these systems make use of various algorithms.

The current approaches that are being used for plagiarism detection, though efficient, have their various weaknesses. (Weber, 2010) agrees that Plagiarism Detection softwares find copies of the copied text, not plagiarism. Of the Many types of software that are available for plagiarism detection in the software market, many of them cease to exist after a few years because they do not offer comprehensive or near comprehensive software solution to the problem of plagiarism detection, or to the problem of heavy obfuscation in documents. Most of the available plagiarism detection systems perform external plagiarism detection, there are no systems really that perform intrinsic plagiarism detection, or that are available for detection of graphic or flowchart plagiarism (Weber, 2010).

There are various means of detecting plagiarism in documents; these means help us classify available approaches by the type of similarity assessment they most prominently apply.
Fig 2.4 Classification of plagiarism detection systems (Gipp, 2014)

All the approaches in fig 2.5 need a document base to be able to perform document analysis on the various documents, apart from stylometric approach, which does not really need any document base, but performs the plagiarism check on the documents itself (Gipp, 2014), checking for stylometric differences in various parts of the document.

Plagiarism Detection System applies vector space models or string-matching procedures to the entire document to be able to detect plagiarism in documents. This method helps us to tag suspected documents as either suspicious or pass them as original. The documents are tagged suspicious if they exceed a stipulated fraction of the entire document length. Plagiarism detection in various languages require a more sophisticated combinatorial of Natural Language processing technique, language translation technique and other language and text processing techniques to be able to detect plagiarism that has been passed on in another language (Stein et al., 2011).
2.3.1 Document Fingerprinting

As we know from forensic science, fingerprints are unique to an individual, so also, every document has its own unique characteristics called its fingerprint. Fingerprints are the most commonly used external plagiarism detection feature.

The fingerprinting technique typically represents a document by splitting it into various substrings and selecting a subset of all the substrings formed. The set of substrings is called the fingerprint. The elements in the fingerprint are called the minutiae. In fingerprint detection, the minutiae are what make a fingerprint either unique or compromised. To further make the fingerprint usable, and workable, a hash is often applied to the fingerprint to transform the minutiae from long strings into space efficient byte strings (Gipp, 2014). A document is compared by computing the documents fingerprint and then matching or querying each of the minutiae with a pre-compiled array of fingerprint indexes of fingerprints for all documents in a particular reference collection.

Stein (2005) identifies the fingerprinting method as the approach to plagiarism detection that has very wide usage. This method forms representative digests of documents by selecting a set of multiple substrings (n-grams) from them. Computational resources and time are limiting factors to fingerprinting, which is why this method typically only compares a subset of minutiae to speed up the computation and allow for checks in very large collection, such as the Internet.
When a document’s selected minutiae match with that of another document, it’s indicative of a strong presence of plagiarism in the text being checked. But there must be a similarity threshold for benchmarking documents that are to be classified as plagiarized and those that should be classified as original. For example if two documents contain the phrase “but in the beginning it was not so”, and the Plagiarism detection software segments this as a subset of a substring in a sentence and hashes this as a fingerprint. What it means is that there will be a false positive, if an appropriate benchmark is not set, as this phrase may be found in any other document and contains some joining and stop-words.

Some of the factors that affect the fingerprinting technique when used in plagiarism detection software are

- The chunking unit,
- Size of the chunks,
- The resolution of the fingerprint,
- The strategy for the selection of the chunks,
- The way the similarity is detected/discerned.

A chunk is the relative number of small segments that a document or text is subdivided into.

1. **The Chunk Unit**: This is the number of segments that the plagiarism detection software using the fingerprinting method divides a text into. It also defines whether the text divided is combined into bigger compositied called chunks (Gipp, 2014). We may employ n-gram chunking unit, or employ sentence level chunking unit or phrasal level chunking unit or better still paragraph level chunking depending on the prevailing methodology used.

2. **The chunk size** determines how granular the fingerprint will be. The probability that documents share substrings decreases with increasing substring length. Larger chunks are better at computational efficiency, because fewer chunks must be stored for each document and thus require less system resources to evaluate, but they have the disadvantage of exposing the document and making it vulnerable to plagiarism detection, this is because when one character is changed, it makes the fingerprint characteristics of the document to be altered thereby for a long chunk of text, we may assume that there is no plagiarism present, whereas all it took to hide the plagiarism was just shaking
off some characters and introducing a few new words that made the document look genuine. Small chunks sizes promise a better deal with plagiarism detection, but require a more dedicated computer resource to be able to detect and match common substrings that are in an examined document (Heintze, 1996). Generally, chunk sizes differ in the fingerprinting technique; there are arrays of chunk sizes that have been standardized by researchers. Some of these are

Table 2.3: Chunk sizes for fingerprinting method (Gipp, 2014)

<table>
<thead>
<tr>
<th>S/N</th>
<th>Chunk Size</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>3-4 characters</td>
<td>• Stay,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• char,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• acte,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• rist,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ics</td>
</tr>
<tr>
<td>2.</td>
<td>single content words</td>
<td>• Animal,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Behavior,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fresh,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Rape</td>
</tr>
<tr>
<td>3.</td>
<td>3-5 content words</td>
<td>• From all the surrounding,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• another animalistic behavior,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• gruesome encoding appareled in lasciviousness</td>
</tr>
<tr>
<td>4.</td>
<td>7-10 content words</td>
<td>• The plagiarism detection approaches are all prevalent in the situations,</td>
</tr>
<tr>
<td></td>
<td>(Broder et.al,1997)</td>
<td>• this makes it possible for us to evaluate published text,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• On the basis that there is no plagiarism present</td>
</tr>
<tr>
<td>5.</td>
<td>8-11 stop words</td>
<td>• It is of an on for from where while with,</td>
</tr>
</tbody>
</table>

3. The resolution is defined as the number of minutiae present in a fingerprinted text document, that is the sum total of the substrings that have been hashed in a fingerprint. This can either be fixed or it can vary that is., the number of hashed
substrings a fingerprint contains and can be either fixed or variable. The number of minutiae is directly proportional to the detection accuracy. The more the number of minutiae, the more the detection accuracy of the text. Fingerprint resolution is also dependent on the computing resources. The more we have the resolution, the more the computing resources that will be used (Broder et al., 1997). Fixed resolution fingerprints are computation friendly, but are also prone to leaks in the detection process as this means that no matter the number of texts or the length of the paragraph, the document plagiarism checker will not vary the resolution but will use the same fingerprint for all the paragraphs being checked for plagiarism. This means that there will be more computer resources to evaluate the document, but in cases of disguised plagiarism, the document may be passed. They generally yield lower detection accuracy especially for very long documents (Heintze, 1996). When we use fixed-resolution fingerprints, a book may not share enough minutiae with a paragraph copied from it thus making the copied text escape detection.

Variable-resolution fingerprinting methods compute more minutiae for a given word length of document. (Heintze, 1996) agrees that the Longer the document, the higher the percentage of the encoding of the text. This increases detection accuracy, but requires higher computational effort. It should be noted that full fingerprinting considers all available minutiae in the substring. However, the fingerprint index for a full resolution fingerprinting Plagiarism detection
System requires eight or more times the disk space of the original document collection and also higher processing time.

4. **Chunk selection strategy** determines which text sections the fingerprint should encode and which section it should skip, the encoded section will then be made comparable to other documents. A selection of reasonable chunks is necessary, because the computational requirements of full-resolution fingerprinting are too high for practical computing. There are various fingerprint chunking strategies that are employed in the fingerprinting approach these are as outlined below

Table 2.4: Overview of Chunk selection methods for fingerprinting method.
(Gipp, 2014)

<table>
<thead>
<tr>
<th>S/N</th>
<th>Chunking Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>Probability-based selection (Broder et al., 1997)</td>
</tr>
<tr>
<td>3.</td>
<td>Frequency-based selection (Heintze, 1996)</td>
</tr>
</tbody>
</table>

5. **Similarity function** helps us to look at the minutiae that the suspected document shares with a reference document. Thus generating a similarity score. The similarity score is the measure of similarity as discovered in the text under investigation. When Documents exceed a certain threshold score, then the document is suspected as containing plagiarism. To calculate a basic similarity function, we can employ the formula as used by Kasprzak and Brandejs (2010), where they defined a fixed number of matching minutiae as the threshold.
Another very excellent method for calculating the similarity function is to take the fraction of all minutiae which will be represented by \((d)\) of a suspicious document \(ds\) that overlap with the minutiae of a genuine document \(dg\). The overlap represents a share to some degree of plagiarism in the suspected plagiarized document. This approach was invented by Broder & Glassman (Broder, 1997), who proposed using containment together with a measure they termed resemblance, \(r(ds, dg)\). Resemblance is the Jacquard coefficient \([J]\) for the sets of minutiae in a document subset, and is an expression of the global similarity of the two sets. Resemblance and containment have been used frequently in plagiarism detection research in research institutes and universities, along with other sophisticated similarity detection and measurement tools and methodologies, such as the length of the document (Bernstein et al., 2004), the frequency distribution of the minutiae in both documents (Scherbinin, 2009), or the maximum difference in the vector component of the minutiae. (Zou et al., 2010).

\[
\mathbb{C}(ds, dg) = \frac{|M(ds) \cap M(dg)|}{|M(ds)|}
\]  
Equation 1

\[
\mathcal{R}(ds, dg) = \frac{|M(ds) \cap M(dg)|}{|M(ds) \cup M(dg)|}
\]  
Equation 2

\(\mathcal{R}(ds, dg)\) is also known as the Jaccard Coefficient and it gives us the global similarity of the two sets \(ds\), and \(dg\). The major problem with fingerprinting is that though it’s a great tool and method for detecting plagiarism, it incurs a considerable amount of resource overhead in computation. To be able to get acceptable detection accuracy, we
have to tune the factors down, and this may have an overall impact on the ability of the plagiarism detection software to detect plagiarism. The parameter choice of fingerprinting methods reflects this challenge. The combinations of parameters that perform best will definitely depend on the size of the collection of both the reference and suspected document, and on the expected amount and form of plagiarism present.

2.3.2 String Matching

String matching refers to searching for a given character sequence, or "pattern", in a text. Plagiarism Detection Systems employing string-matching approaches commonly use suffix document models. Suffix data structures store each substring of a text and allow for efficient comparisons. Using string matching for Plagiarism Detection requires the computation of suffix document models for the suspicious document and for all documents in the reference collection. Because the pattern to search for is initially unknown in a Plagiarism Detection setting, the detection procedure must select portions of the suspicious text and check them against all other suffix models (Gipp et al., 2011). Baker was among the first to employ suffix trees for Plagiarism Detection (Baker, 1993). She augmented the trees’ vertices with positional information that allowed detecting all matching strings of maximum length. Baker defined a heuristic similarity threshold and tailored her procedure to check source code for plagiarism. She suggested an adaption of the algorithms to text plagiarism detection, but did not pursue further development in this area. The Match Detect Reveal (MDR) system also employed string matching for Plagiarism Detection. The Match-Detect-Reveal adopted Ukkonen’s algorithm, which only considers suffixes of
full words for constructing the tree. Match-Detect-Reveal used the matching statistics algorithm of Chang and Lawler for overlap computation. Khmelev et al. (2000) constructed a Plagiarism Detection System using suffix arrays for document representation and the "R-measure", i.e. the normalized sum of repeated substrings, for similarity calculation. Goan et al. (2014) used String B-Trees and similarity assessments leveraging “knowledge of common text patterns” (Goan et al., 2011) for Plagiarism Detection.

The strength of substring-matching in Plagiarism Detection approaches is their accuracy in detecting verbatim text overlaps (Gipp, 2011). Suffix document models encode the complete character information of a text, which distinguishes them from the document models that most fingerprinting methods employ. If two documents share substrings, suffix document models enable the detection of this overlap through string matching. The major drawbacks of string matching in a plagiarism Detection Software context are the difficulty of detecting disguised plagiarism. Another disadvantage is that a very high computing power is needed in the execution of the algorithms for exact-string matching. Today, the most space-efficient suffix tree, suffix array and suffix vector implementations allow searching in linear time and require on average approximately $8n$ of storage space, with $n$ being the number of characters in the original document. String B-Trees allow searching in $(\log C)$.

2.3.3 Stylometry

Holmes (1998) defines Stylometry as the use of statistical methods to check, evaluate and analyze a writer’s method of writing. A very important aspect of Stylometry is
Authorship attributions (AA). Authorship verification has dominated the Stylometry landscape for a while; it is a problem area in the research field of Stylometry as it relates to intrinsic plagiarism detection. In Authorship verification, we answer the question of whether a particular author wrote a given text or not. Conducting stylometric comparisons in any of these three possible categories can solve this problem and answer the question accordingly. In (Koppel et. Al., 2011) these categories include:

1. Comparing a document from an author that has authorship is being doubted suspected author with a corpus that is from the same author. This scenario represents the authorship verification problem.
2. Comparing a document written by an author that his work is being suspected with other documents written by various other different authors in order to identify the sections that are similar or that look alike. This is a classic case of external plagiarism detection.
3. Comparing a document with various text segments that were alleged to have been written by an author in question with other text segments within the same document for variation in the writing style of the author. This represents a typical intrinsic plagiarism detection approach; because there are no external reference documents to check the work against.

Stylometry in plagiarism detection is usually applied to intrinsic plagiarism detection. There are no identifiable application of stylometry in extrinsic plagiarism detection because other methods have been applied to detect plagiarism (Gipp, 2014). Intrinsic plagiarism detection approaches apply the construction of models that help in comparing and discerning an author’s style of writing within individual segments of a
text. The goal generally is to identify various text segments that are different from other segments stylistically. By this we mean identifying the sections that are different in writing style from other sections. When these sections are identified, they become markers to lack of originality and thus may contain plagiarism (Meyer, 2006). Intrinsic Plagiarism Detection approaches solve a one-class classification problem. The Genuine text segments that share the same attribute represent the class that is being targeted, while the plagiarized text segments form outliers with divergent attributes. A classification method/algorithm must learn the characteristics of the class that is genuine that is the target class and use their knowledge learned for either accepting or rejecting outliers. In this case outliers are rejected as they represent the text segment that are suspicious, and that are inconsistent with the general writing style and may therefore contain plagiarism (Potthast, 2009). Stein et al. (2006) posited that intrinsic plagiarism detection procedures must contain the following parts that are key in the detection of plagiarism in documents. Style model defines the collection of features in the lingual set that will be analyzed by the algorithm. Most style sets use a unique combination of features from the combination of features that have been proposed for stylometric analysis (Stein et al., 2011). Some of the features have been categorized as below.

i. Lexical features: They appear on the character level e.g. n-gram Frequency (i.e. the number n-related words that occur in the original and suspect text segments), or on the word level, (i.e. the average word length or syllables per word in the original and suspect text segments.)
ii. Syntactic features: This is textual analysis by application of statistics to the parts of speech/other syntactic elements.

iii. Structural features: Every author has a signature editorial structure. This structural feature includes average paragraph length or punctuation frequency. An outlier detection procedure operates on the features vectors of the text segment under question and the segments contained in the overall document to identify significantly different text segments in the document. Many classifiers for one-class classification problems are available. Intrinsic plagiarism detection approaches always employ classic measures of dispersion, such as standard deviation or median absolute deviation, and vector-based using cosine similarity (Muhr et al., 2009).

(Meyer et al., 2007) shows various machine-learning approaches that had the capacity of learning the relative differences in feature vectors. Stein et al. (2006) applied methods using estimated feature distributions in the target and outlier class. An outlier post-processing procedure determines whether when we have multiple outliers, if they form a larger section and whether they are suspicious enough to be reported as deviant. This problem can be solved using two approaches mainly heuristic voting or meta-learning (Stein, 2011). Intrinsic plagiarism detection has the advantage over other plagiarism detection systems because it does not depend on any external collection of references. Intrinsic plagiarism detection could be a pointer text segments that need a more thorough analysis by other plagiarism detection systems. There are multiple factors that affect the accuracy and reliability of automated stylometry analyses, some
of which are the genre of the text under investigation, the volume of the analyzed text, the purity of the analyzed text. For example, text quoted from other sources, table facts and data, figures and other meta-data could directly or indirectly affect the statistical analysis on the style (Stamatatos, 2009).

Figure 2.6 shows the concept of outliers in intrinsic plagiarism detection. The regions A and B are from the same document, after the document was analyzed, the various segments were separated by a support vector classifier and the classification is done according to the analysis of the terms in the document.

An obstacle to the purity of text in intrinsic plagiarism detection is joint publications, where various ideas and writing styles are fused into one. This also is a problem when concentrating on stylometric analysis for plagiarism detection, detecting differences in writing style and note multiple authorship is a challenge for
this kind of tasks. Some of the available systems have their fair share of failure and successes when it comes to plagiarism detection. There are various means of detecting plagiarism in documents, these means help us classify available approaches by the type of similarity assessment they most prominently apply.

2.3.4 Semantic Analysis

The semantic method analyses and compares text based on the semantic allocation for each term inside the sentence (Osman et al., 2012). The technique is used to take up a document and access the document by segmenting the various texts that make up the document. A semantic role in the sentence that was broken down is then labelled by a semantic labeller. A semantic role labeller is used to detect the semantic-term annotation of the text to detect plagiarism.

The grammar-based method identifies plagiarised texts by considering the grammatical structure of a document with the help of context free grammar concepts to identify plagiarism from paraphrased text (Adam, 2014). This method contains a string-based matching model which analyses the grammatical structures or patterns of sentences in order to determine whether the sentences have been significantly altered. It finally tells whether the document is plagiarised or not by computing the extent to which the plagiarised document has altered the structure and grammar of the sentences in the original documents.

The syntax-based technique identifies plagiarism with the aid of syntactical part-of-speech (POS) tags which are utilised in representing text structures to support comparison and analysis (Elhadi et al., 2008). Similarity scores are calculated based
on the number of corresponding POS tags existing between the suspicious and original documents. Other techniques that employ the syntax-based approach to detect plagiarism are

- syntactic and semantic-based techniques such as duplication-gram,
- re-ordering and
- alignment of words,
- POS,
- phrase tags and
- Semantic similarities of sentences.

The semantic technology-based technique measures the semantic similarity between words and their meanings to confirm plagiarised text through matching the various keywords of the texts between the documents (Agarwal et al., 2013). Figure 2.7 shows the features exploited in plagiarism detection in documents. This ranges from part-of-Speech Tagging to Paraphrasing and Syntax and semantic structures.
2.4 Features Employed in Plagiarism Detection

Features are the pointers or parameters used in detecting plagiarism of all types. Various features were identified in the literature and are listed below.

Fig 2.7: Common features exploited in plagiarism detection (Alzahrani et al., 2012)
1. **Highest ranking keywords feature**, similarity of first sentence, phrasal query and the subsequence with the longest common term. The Highest keyword feature is executed by removing the stop-words in a document (Taiseer et. al, 2015), otherwise known as the stemming process, where the words that appeared most are singled out in the document, and considered keywords that capture the ideas in the papers (Kent and Salim, 2010). However, finding some of the frequently appearing words in the papers does not necessarily mean that the entire idea of a paper has been captured. The first sentence of a document comprises crucial ideas that capture the entire concept of the work. To use this feature to detect plagiarism, the first sentence of the document perceived to be original is matched with the first sentence of the suspected document.

2. **Lexical features**. These features focus on using the lexical structure of the text or documents, which operate at the character or the word level of the document in order to trace the plagiarism in the suspicious document (Alzahrani et al., 2012). This type of approach tries to enhance the standard string matching comparison in order to detect plagiarism. The processing techniques that this approach relies upon include tokenisation, lowercasing, punctuation removal and stemming. However, it can vary across techniques in terms of words, sentences, passages, human defined sliding window or an n-gram.

3. **Structural features**. Structural features (or tree-structured features), conversely, take into account the way the words are distributed throughout the document. Very few plagiarism detection approaches have been developed to
handle structural or tree features (Chow and Rahman, 2009), (Zhang and Chow, 2011), (Alzahrani et al., 2011).

2.5 Plagiarism Detection Softwares: An analysis

Table 2.6: Comparative analysis of Plagiarism Detection Softwares

(Taiseer et. al, 2015)

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Detection tools</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tumlin</td>
<td>Eve2</td>
<td>CopyCatchGold</td>
<td>WordCheck</td>
<td>Glatt</td>
<td>Moss</td>
<td>Jplag</td>
<td>Google</td>
<td>Yahoo</td>
</tr>
<tr>
<td>Checks source code?</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Checks free text?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td><strong>Type of text tool operates on</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operates intra-corpally?</td>
<td>Y</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Operates extra-corpally?</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td><strong>Type of corpus tool operates on</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designed for use by students?</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>Designed for use by teachers?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Instant response?</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>Free?</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Comparing the detection accuracy and performance of plagiarism detection systems is quite difficult. Many authors have proposed various plagiarism detection prototypes, but they often use non-standardized evaluation methods in their comparison. Potthast et al.(2010) in his review which contained 139 publications, discovered that 80% of the papers used their own text corpora and that about 38 of the 139 authors did not offer comparisons among plagiarism detection systems in the their research. (Potthast et al, 2010).
Some of the common Plagiarism detection softwares in use globally are

### 2.5.1 PlagAware

Plagaware is a web-service that’s commonly employed for textual plagiarism detection (Asim et al., 2011). It allows the users to perform a textual and topical search on specific topics. Plagaware is basically a search engine, and it’s strong in detecting plagiarism on texts fed into its search space. It also allows you to search, find analyze and trace plagiarism on the specified topic similar to the topics under investigation. It uses the classical Google/Bing search engine algorithm for detecting and scanning plagiarism, and also provides various reporting formats which helps the user to decide whether his document was plagiarized or not. Primarily, Plagaware is used in monitoring web contents for content theft and also for assessment of transmitted text. It can be accessed online at http://www.plagaware.com

Typical applications of PlagAware are

- **Tracing content theft:** webmaster can use PlagAware for detecting and tracing plagiarisms of websites, in order to find out the plagiarized or the copied contents (Asim et al., 2011).

- **Plagaware is used in search for plagiarisms of student’s academic documents** and analyze them (Asim et al., 2011).

- **Proof of authorship is also provided by PlagAware** (Asim et al., 2011).

Some of the features of PlagAware are
• Database checking: PlagAware allows submission of word and checking them against a wide range of databases online (Asim et al., 2011).

• Internet checking: PlagAware as an online web application allows students check their academic documents against plagiarism over World Wide Web (Asim et al., 2011).

• Synonym and sentence structure checking: PlagAware does not support synonym and sentence structure checking (Asim et al., 2011).

• Multiple document comparison: Plagaware offers comparison of multiple documents (Asim et al., 2011).

2.5.2 PlagScan

PlagScan is a software service employed for use in textual plagiarism check. It is often used by colleges, Universities and schools and provides various accounts with various performance features. PlagScan uses very complex algorithms for checking and analyzing documents for plagiarism detection, based on current research in the field of Natural Language processing, syntax and semantic lexical structure. A Unique document fingerprint signature is extracted from the lexical structure of the document and then a comparison is done on the software’s database or on other online documents available on the internet. PlagScan can detect various forms of plagiarism including shake and paste plagiarism, copy and paste plagiarism, and disguised plagiarism. PlagScan provides measure of the accurate level of plagiarism in any document after
scanning. It can be accessed at http://www.plagascan.com. Some of the features of PlagScan are listed below

- Database Checking: PlagScan has its own database that includes a collection of papers, articles and journals. From all over the World Wide Web (Asim et al., 2011).

- Internet checking: PlagScan provides internet checking of all submitted documents (Asim et al., 2011).

- Publications checking: PlagScan is mainly used in the academia to find and validate submitted articles against plagiarism (Asim et al., 2011).

- Syntax and semantics check: Though it does not perform a syntax and semantic check on the lexical structure of the words supplied, it does offer an API that allows developers to integrate the functionality of PlagScan into their web applications (Asim et al., 2011).

- Multiple Document comparison: It also allows for comparison of various documents in corpora (Asim et al., 2011).

2.5.3 ChekForPlagiarism.net

ChekForPlagiarism.net is a web application that was developed by a team of academics (Asim et al., 2011). It has the reputation of being one of the best plagiarism detection systems online. It’s employed in the detection of plagiarism and its stoppage and helps in the prevention of plagiarism of academic papers by performing a fingerprinting of the papers and storing it in its database. This fingerprint-based
approach is used to analyze and summarize the collection of documents, thereby making it unique in its own corpora. By creating fingerprint characteristics for each document in the collection, we can quickly find the articles that match or those that are similar in a large collection of articles (Asim et al., 2011).

This singular feature of checkForPlagiarism.net has increased the efficiency of the detection algorithm. It’s available from http://www.checkForPlagiarism.net.

Other plagiarism detection tools are ithenticate, plagiarismdetection.org and Turnitin. These are all available online and are useful in the plagiarism detection process of string matching and disguised plagiarism checking. A feature comparison of the tools will reveal their strengths and weaknesses. (Asim et al., 2011).
Table 2.7 Comparison of various plagiarism detection software ranked on a scale of 1 - 5

<table>
<thead>
<tr>
<th>Features</th>
<th>PlagAware</th>
<th>PlagScan</th>
<th>iThenticate</th>
<th>ChekForPlagiarism</th>
<th>Plagiarismdetecting.org</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Checking Online</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Checking on the Internet</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Checking of Publications</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Comparing Multiple Documents</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Support of Multiple Languages</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Syntax and Semantic Check (Shake and paste, Disguised Plagiarism, Obfuscation)</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

2.6 Empirical Framework of the Study

In Stamatatos, (2011), plagiarism detection was carried out on a document using n-gram profiles of documents by removing stop words. (Meyer et al., 2007) and (Stein et al., 2007) did a work on intrinsic plagiarism Detection, and defined it as being a more ambitious task, since no reference corpus is given. Intrinsic plagiarism detection
is the task of determining plagiarism on a document when the corpus is not defined or when it is impossible or impractical to have a reference data available. Also, comparing a suspicious text with all the texts of a large document may be impossible and impractical due to computational and space requirements.

In plagiarism analysis and detection, whenever an author or a plagiarist tries to insert text or obfuscate the document, the newly inserted text is seen as noise and may affect the performance of the plagiarism detector (Stamatatos, 2011). The first people to use the character n-gram to detect plagiarism and inconsistencies in text documents was Graham et al. (2005). Though their results were poor, yet their work proved innovative. They failed for some obvious reasons, one being they only used character bigrams, they also measured the similarity between texts using cosine distance, which is an unrealistic and unreliable tool for measuring distance between very short text characters. In Graham et al, (2005), plagiarism detection was done at the paragraph level based on predefined text and their task was to identify whether two consecutive paragraphs differ in style or not.

Stamatatos, E., (2011) proposed an approach to define a sliding window over the text length and compare the text in the window with the whole document. This will give a function that will define the change in the writing style in the document, this approach was applied to intrinsic plagiarism detection in Stamatatos, (2011). The anomalies in the document were then used to detect the plagiarised sections. The peaks of the anomaly detection function will give us the text sections that have very glaring dissimilarity. Each text is considered as a cluster-of-n-gram characters. Assume we
are given a predefined $n$ that denotes the length of strings; a vector of frequencies of all the character n-grams that appear at least once in the text are built. The vector of frequencies is called the text profile and its size is dependent on the text length. The longer the text is, the bigger the profile will be. What will the value $n$ be? $N$ denoting the number of words or characters to be used in the n-grams for the detection to be done successfully. A high value of $n$ will mean that we will have long texts to deal with, and capture better information from the words, based on their relationship with one another and with themselves. But there is a downside to increasing the value of $n$. On the other hand; a high $n$ considerably increases the dimensionality of the profile, thereby increasing computing power and time for the match to take place. To keep a low dimensionality profile, to keep dimensionality character 3-grams was used in (Stamatatos, 2007; Koppel et al., 2009). In table 2.7, we find the complete set of parameter settings for the proposed method used in Stamatatos, (2011). A few of the documents in the evaluation corpus was used to generate the settings
Table 2.7: Parameters used in n-gram Profiles Intrinsic Plagiarism Detection (Stamatatos, 2011)

<table>
<thead>
<tr>
<th>Description</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character n-gram length</td>
<td>$n$</td>
<td>3</td>
</tr>
<tr>
<td>Sliding window length</td>
<td>$l$</td>
<td>1,000</td>
</tr>
<tr>
<td>Sliding window step</td>
<td>$s$</td>
<td>200</td>
</tr>
<tr>
<td>Threshold of plagiarism-free criterion</td>
<td>$t_1$</td>
<td>0.02</td>
</tr>
<tr>
<td>Real window length threshold</td>
<td>$t_2$</td>
<td>1,500</td>
</tr>
<tr>
<td>Sensitivity of plagiarism detection</td>
<td>$a$</td>
<td>2</td>
</tr>
</tbody>
</table>

Stamatatos (2007) studied the performance of various distance measures that determine the similarity or dissimilarity between two character $n$-gram profiles, and came up with the following distance measure.

\[
d_1(A, B) = \sum_{g \in P(A)} \left( \frac{2(f_A(g) - f_B(g))}{f_A(g) + f_B(g)} \right)^2 \quad \text{.................. (3)}
\]

\[
nd_1(A, B) = \sum_{g \in P(A)} \left( \frac{2(f_A(g) - f_B(g))}{f_A(g) + f_B(g)} \right)^2 \frac{4|P(A)|}{4|P(A)|} \quad \text{.................. (4)}
\]

In (3) above, $f_A(g)$ and $f_B(g)$ are the frequency of occurrence of the $n$-gram words $g$ in the two documents $A$ and $B$ respectively. Stamatatos, (2011) claims that $d_1$ is not a distance function and was only designed to be used when text $A$ is shorter than text $B$. 
In Stamatatos (2011), (3) was modified to (4) so that it was used no more for comparing only two text segments, but for comparing a text segment against the whole document.

In (3), \(|P(A)|\) is the size of the profile of text A. to ensure that the values of dissimilarity function lie between the highest similarity of 0 and lowest similarity score of 1, Stamatatos, (2011) introduced the denominator. \(nd1\) is also called the normalized \(d1\). \(w\) is a sliding window of length \(l\) with a step \(s\), each time the window is moved to the right by \(s\) characters the profile of the next \(l\) characters is extracted. Whenever \(l\) is greater than \(s\), then we assume the windows are overlapping and a style change has occurred. The style change can be mathematically defined as:

\[
sc(i,D) = nd1(w_i, D), \ i=1...|w| \\
\]

In the method employed by Stamatatos, (2011), the task of plagiarism detection is done by detecting peaks of that function corresponding to text sections that significantly differ from the rest of the document.
Figure 2.8: The style change function of document solid line. The dashed line indicates the threshold of the plagiarized passage criterion. The binary function above indicates real plagiarized passages (high values). (Stamatatos, 2011)

Figure 2.9: The style change function of the plagiarism-free document (a false positive). The dashed line indicates the threshold of the plagiarized passage criterion. (Stamatatos, E., 2011).
Kucecka (2011) developed a Plagiarism Detection method in Obfuscated Documents using an N-gram Technique. The N-gram comparison technique developed by Kucecka, (2011) was based on splitting the text into strings of length $n$ starting at a given position in the text. To calculate the position of the next n-gram, Kucecka, (2011), obtained it from the position of an actual n-gram shifted by a given $o$ set. The parameter $n$, being the number of the words to include in the n-gram is given on the input by the user. The $o$ set value depends on the used $n$-gram division. If we got our n-grams by joining words together, the value of our $o$ set will be the number of words that will be skipped when building the next n-gram. If on the other hand, we got our n-gram by joining various letters, and we didn’t consider the end of the word, the $o$ set will represent the number of letters that we skipped. There are various
approaches to division of the n-grams at the either the word level or the character level. Kucecka, (2011) identifies some of these methods as

1. Overlapping n-grams – definitely, all n-grams must have a starting position. When the division is done in such a way that it some common substring with the previous n-gram(s) that were before it, we say they have overlapped. The position of the overlapping substring will also be something to note. For instance, splitting the string CHUKWUEMEKA into n-grams with a parameter of \( n=3 \) and \( o=1 \), will create the set of the following n-grams:

   CHU, HUK, UKW, KWU, WUE, UEM, EME, MEK, EKA

2. Non-overlapping n-grams - none of the n-grams created from the text share the exactly same substring on the same position.

   An example of this will be splitting CHUKWUEMEKA into n-grams of \( n=3 \) and \( o=1 \). CHU, KWU, EME, KA

Kucecka (2011), implemented the n-gram comparison technique into an existing plagiarism detection system. Equation (6) was used to measure similarity between documents A and B, where function Gram (A) returns set of n-grams in document A (analogically for B). Document B represents in our case the plagiarised and obfuscated document.

\[
sim(A, B) = \frac{\text{Gram}(A) \cap \text{Gram}(B)}{\text{Gram}(B)} \times 100,
\]

\( \ldots \ldots \ldots \ldots \ ) (6)
Where functions Gram (A) returns set of n-grams in document A. Document B represents in equation (4) represents the plagiarised document.

In fig 2.11, Kucecka (2011) discovered that when the value of \( o \) is increased the number of similarities in the document corpora decreases, \( o \) is the number of words or grams that are to be skipped before realising the next n-grams.

![Detected similarity for non-plagiarised documents](image)

Figure 2.11: Detected similarity for the non-plagiarized pairs of documents. Parameter \( n \) is the n-gram length and the \( o \) parameter is the number of letters used as \( o \) set (Kucecka, 2011).

Kucecka, (2011) focused on determining similarity between the two documents, the plagiarised parts of document A and the obfuscated ones. Various plagiarised pairs for each obfuscation were created where the plagiarised document had about 30% of the text from its original source. Obfuscation was carried out string obfuscation, and experiments with various parameters of n-gram technique were carried out, and showed its performance from the view of similarity detection. Overall, the detection based on letter division showed to be much more effective on discovering the
The detected similarity was about 0% when 3-grams were used, about 33% similarity was achieved using the 15-grams divided by letters with the o set value equal 1.

In Alberto et al., (2009), if the original and suspicious text fragments are matched and found to be close enough; it can be assumed that they are a potential plagiarism case that needs to be checked further. An option for the improvement of the algorithm was to carry out a comparison of text chunks based on word-level n-grams, and see the number of text chunks that would return the best precision value. In Lyon (2004) the reference and suspicious texts are split into trigrams, composing two sets that are compared later. To flag a passage as plagiarised or not, the number of the common trigrams that are shared by a passage must meet a certain threshold before we can say the passage was plagiarised. Another option that was carried out was to split the documents into sentences. Alberto et al., (2009) method, was able to detect plagiarised sentences, based on the intersection and complement of the vocabulary of the reference section and the vocabulary of the suspicious text.

In Alberto et al, (2009), the task of discovering a document if it was a suspicious document $s$ when given a reference corpus $D$, The objective was to answer the question.

“Is a sentence $s_i \in s$ plagiarised from a document $d \in D$?”

We consider here that the plagiarised text fragments may appear mixed or modified. The n-gram based text comparison attempts to tackle this problem of whether the text
was modified or is still in its real state. Alberto et al. (2009) considered n-grams because independent texts have a small amount of common n-grams. One may ask why n-grams can’t we see the same n-grams in various documents? Table 2.8 shows the probability of finding common n-grams in different documents decreases as n increases, this means that no matter how large the corpora, we will see very few of the n-grams occurring if the value of n is reducing.

Table 2.8. Common n-grams in different documents (avg. words per document: 3,700)

<table>
<thead>
<tr>
<th>Documents</th>
<th>1-grams</th>
<th>2-grams</th>
<th>3-grams</th>
<th>4-grams</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1692</td>
<td>0.1125</td>
<td>0.0574</td>
<td>0.0312</td>
</tr>
<tr>
<td>2</td>
<td>0.0720</td>
<td>0.0302</td>
<td>0.0093</td>
<td>0.0027</td>
</tr>
<tr>
<td>3</td>
<td>0.0739</td>
<td>0.0166</td>
<td>0.0031</td>
<td>0.0004</td>
</tr>
</tbody>
</table>

Since a plagiarised sentence could be made of small texts from various sections in an original document, Alberto et al. (2009) suggests that the reference documents should not be split into sentences, but simply into n-grams. The method is based on the next four considerations:

1. The suspicious document s is split into sentences (s_i);
2. s_i is split into word n-grams. The set of n-grams represents the sentence;
3. A document d is not split into sentences, but simply into word n-grams; and
4. Each sentence s_i ∈ s is searched singleton over the reference documents.

To know if s_i was plagiarised from d ∈ D, (Alberto et al, 2009) compared the corresponding sets of n-grams asymmetrically, due to the differences in the size of the sets.
Where \( N(s_i) \) is the set of n-grams in \((s_i)\) and \( N(d_i) \) is the set of n-grams in \((d_i)\). If the maximum \( C(s_i|d) \), after considering every \( d \in D \), is greater than a given threshold, \( S_i \) is said to be a plagiarised text from \( d_i \).

2.7 Research Gap

Plagiarism detection software and approaches have for many years been an active area of research which has brought about an advancement in the field of science, engineering, Medicine and the Arts. Plagiarism detection algorithms have leveraged on various statistical and mathematical advances to develop a reliable procedure for the detection of plagiarism detection.

(Kucecka, 2011) focused on determining similarity between the two documents, the plagiarised parts of document A and the obfuscated ones. (Stamatatos, 2009) implemented an approach that evaluated plagiarism based on sliding window approach. This was useful for plagiarism detection intrinsically, but was not efficient in external plagiarism detection.

In this work, we developed a plagiarism detection approach that was an improvement of Alberto et al. (2009). The approach will seek to reduce the time taken in finding text overlaps in plagiarized documents, by reducing the number of n-grams tokens used, we also hope to improve the precision score by performing a one-on-one
string matching on the plagiarized document from tokenized n-grams that are stored in the database. An Improved precision score would mean an improvement in the algorithm presented.
CHAPTER 3

RESEARCH METHODOLOGY

3.1 Methodology Adopted

The methodology adopted in this study is the agile methodology of software development. This was adopted because Agile methodology requires less planning and it divides the tasks into small increments. (Mihai, 2014) defines agile as an umbrella term for several iterative and incremental software development methodologies. Agile process is meant for short term projects with an effort of team work that follows the software development life cycle. Software development life cycle includes the following phases.

1. Requirements gathering,
2. Analysis,
3. Design,
4. Coding,
5. Testing,
Fig 3.1 Phases of Agile Software development (Mihai, 2014)
The agile process is an iterative process in which changes can be made according to the customer satisfaction. In agile process new features can be added easily by using multiple iterations. Agile software development set of methodologies have the following advantages over the other methodologies. (Mihai, 2014) outlines some of these advantages as

1. **Iterative**: the main objective of agile software processes is satisfaction of customers, so it focuses on single requirement with multiple iterations.

2. **Modular**: Agile process decomposes the complete system into manageable pieces called modules. Modularity plays a major role in software development processes. Time boxing as agile process is iterative in nature; it requires the time limits on each module with respective cycle.

3. **Parsimonious**: In agile processes parsimony is required to mitigate risks and achieve the goals by minimal number of modules.

4. **Incremental**: As the agile process is iterative in nature, it requires the system to be developed in increments, each increment is independent of others, and at last all increments are integrated into complete system.

5. **Adaptive**: Due to the iterative nature of agile process new risks may occurs. The adaptive characteristic of agile process allows adapting the processes to attack the new risks and allows changes in the real time requirements.
7. **Convergent**: All the risks associated with each increment are convergent in agile process by using iterative and incremental approach.

8. **Collaborative**: As agile process is modular in nature, it needs a good communication among software development team. Different modules need to be integrated at the end of the software development process.

9. **People Oriented**: In the agile processes customer satisfaction is the first priority over the technology and process. A good software development team increases the performance and productivity of the software.

### 3.3 Data collection

Knatterud, (1998) defines data collection as the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes. The concept of data collection is almost the same in the fields of human academic study, while the methods vary as we traverse disciplines, the thrust of data collection remains an accurate and honest information collection methodology. In this study the data used in the evaluation of the n-gram plagiarism detection Algorithm is PAN-PC-09 Plagiarism Corpus available at **PAN-PC-09**, (2009).

### 3.3.1 Characteristics of the Population

The PAN Plagiarism Corpus (PAN-PC-09) is a collection of 41,223 documents in which 94,202 artificial plagiarism cases have been inserted (Potthast, 2010). It is the
first corpus which allows for large-scale evaluations of both intrinsic and external plagiarism detection methods. The corpus contains documents that has artificial plagiarism has been inserted automatically by mechanical and electronic means, by the use of a word generation algorithm. The corpus can be used to evaluate two kinds of plagiarism detection tasks: (i) External plagiarism detection; and (ii) Intrinsic plagiarism detection. During its construction, various parameters were varied so that it would allow for a wide range of cases in plagiarism detection. Some of the parameters that were varied in the construction according to Potthast (2009) are

- **Document Length.** Out of the 41,223 documents, about 20,611.5 (50%) of the documents have between 1-10 pages, 14,428.05 documents have between 10 pages to 100 Pages this represents 35% of the whole population having medium length pages, and 15% of the 41,223 documents have large (100-1,000) pages.

- **Suspicious-to-Source Ratio.** About 20,611.5 documents out of the 41,223 are designated as plagiarised documents Dq, and 50% are still kept as source documents D.

- **Plagiarism Percentage.** The percentage of plagiarism per suspicious document \( d_q \in D_q \) ranges from 0% to 100% a seen in fig 3.2. In order to compose a realistic framework, 50% of the suspicious documents contain no plagiarism at all.

- **Plagiarism Length.** The length of a plagiarism case is uniformly distributed between 50 and 5,000 words.

Figure 3.2 is the distribution of the plagiarism per document found in the 41,223 Documents. The distribution follows the normal distribution curve.
• **Plagiarism Languages.** 90% of the cases are monolingual English plagiarism. The remaining 10% are cross-language plagiarism, i.e., the source document is written either in German or in Spanish, and the plagiarism is translated into English.

• **Plagiarism Obfuscation.** The monolingual portion of the plagiarism in the external test corpus is obfuscated. The degree of obfuscation ranges uniformly from 0 as none to 1 as high. To see how obfuscation is done we refer to the work of (Kucecka, 2011). It is common knowledge that Plagiarists often paraphrase or “shake” the text that they have copied, so as to obfuscate it, and conceal their wrong doing.
3.3.2 Sampling Design and Procedure

In the generation of the plagiarised passages, an artificial plagiarizing synthesizer, that simulates text obfuscation $s_x$ in order to generate a different text section $s_q$ to be inserted into $d_q$, was developed based on the following characteristics:

- **Random Text Operations.** Given $s_x$, $s_q$ is created by shuffling, removing, inserting, or replacing words or short phrases at random.

- **Semantic Word Variation.** Given $s_x$, $s_q$ is created by replacing each word by one of its synonyms, hypernyms, hyponyms, or even antonyms.

- **POS-preserving Word Shuffling.** $s_q$ is created by shuffling words while maintaining the original sequence of parts of speech in $s_x$. These operations do not guarantee the generation of human-readable text. However, automatic text generation is still a largely unsolved problem which is why we have approached the task from the basic understanding of content similarity in information retrieval, namely the bag-of-words model.

In the development of this work, the choice of the number of documents to choose to test the algorithm was selected using Yamane (1964) formula for finite population. This was done because we could not test all the documents in the corpora because of computing time and the limited power resources available.

$$n = \frac{N}{1+N(e)^2} \quad \text{................. (3.1)}$$
N = Total population size

e = co-efficient of confidence or margin of error or allowable error or level of significance

n = sample size to be determined

\[ n = \frac{41223}{1 + 41223(0.05)^2} \]

\( n \) in this case is given as 392.316 \( \sim \) 392 documents. We therefore choose to perform the plagiarism detection using about 392 documents. We could not test all the documents in the corpora because of computing time and the limited power resources available. Out of the 392 documents, 50% will represent documents without plagiarism and 50% will represent documents with plagiarism.

### 3.3.3 Data Collection Instrument

There was no additional data collection done, as the data needed to test the system was already available as prepared by the PAN-PC-09 Corpora and obtained from their website (PAN-PC, 2009). The data has already been analyzed earlier to contain 41,223 documents and have about 50 to 5,000 words in each document. The data needed to analyze the performance of the algorithm was collected from the system during test and the values for the precision, recall and f-measure were calculated from the retrieved documents and the relevant documents available according to the formula for precision and recall.
3.4 System Analysis Procedures

Systems analysis is a process of collecting factual data, understand the processes involved, identifying problems and recommending feasible suggestions for improving the system functioning (Beimborn, 2003). This involves studying the business processes, gathering operational data, understanding the information flow, finding out bottlenecks and evolving solutions for overcoming the weaknesses of the system so as to achieve the organizational goals. System Analysis also includes subdividing of complex process involving the entire system, identification of data store and manual processes. The major objectives of systems analysis are to find answers for each business process:

- What is being done,
- How is it being done,
- Who is doing it,
- When is he doing it,
- Why is it being done
- How can it be improved?

It is more of a thinking process and involves the creative skills of the System Analyst. It attempts to give birth to a new efficient system that satisfies the current needs of the user and has scope for future growth within the organizational constraints. The result of this process is a logical system design. Systems analysis
is an iterative process that continues until a preferred and acceptable solution emerges.

The present study adopted system analysis procedure of structural system analysis and design methodology (Gangolly, 1997).

Fig 3.3: System analysis Procedure (Gangolly, 1997)

The steps of system analysis and procedure of Gangolly (1997) used in this study are presented below.

**STEP 1:** Documentation of how the existing system works.

**STEP 2:** Documentation of what the existing system does.

**STEP 3:** Documentation of what the proposed system will do.

**STEP 4:** Documentation of how the proposed system will work.
3.4.1 Financial Feasibility Study of the Proposed System

Fonollera, (2009) defines a feasibility study (FS) as an evaluation tool used to determine the viability/profitability of a certain idea. It is a tool that systematically explores whether a given idea will work and whether it should be pursued further for implementation. The financial feasibility study determines the amount of money required in realization of the project, which includes the source of finance and cost involved. According to Schwalbe (2012), financial considerations are often an important aspect of the project’s selection process, especially during tough economic times. There are three primary methods for determining projected financial value of projects Schwalbe, (2012):

1. Net present value analysis (NPV)
2. Return on investment (ROI)
3. Payback analysis

**Net present value analysis:** is a method of calculating the expected net monetary gain or loss from a project by discounting all expected future cash inflows and outflows to the present point in time. The NPV is the sum of all the present value of the money that gets into the business (considered positive) and the present value of all the money that gets out of the business (considered negative) at a certain discount rate Fonollera (2009). The discount rate is set by the goals of the business. In general a higher positive NPV is preferable. If the NPV is negative it indicates that it is not worth pursuing after all. If NPV is zero, it suggests that you should better off putting the money in a bank.
Steps involved in performing the calculation manually for NPV are as follows (Schwalbe, 2012):

1. Determine the estimated costs and benefits for the life of the project and the projects it produces.

2. Determine the discount rate. A discount rate is the rate used in discounting future cash flow. It is also called the capitalization rate or opportunity cost of capital.

3. Calculate the net present value. There are several ways to calculate NPV. The mathematical formula for calculating NPV is:

   \[
   \text{NPV}(r) = \sum_{t=0}^{N} \frac{R_t}{(1 + r)^t}
   \]

   Where \( t \) equals the year of cash flow. \( N \) is the last year of the cash flow. \( R \) is the amount of cash flow each year and \( r \) is the discount rate.

The formula for the discount factor is \( t/(1 + r)^t \)

Where \( r \) or \( i \) is the discount rate.

**Return on Investment:** According to Schwalbe (2012) ROI is the result of subtracting the project costs from benefits and then dividing by costs. ROI is always a percentage. It can be positive or negative. In calculating ROI it better to consider discounted costs and benefits for multi-year projects. Mathematical formula for ROI.

\[
\text{ROI} = \frac{\text{total discounted benefits} - \text{total discounted}}{\text{discounted costs}}
\]
The higher the ROI, the better.

**Payback Analysis:** According to Schwalbe (2012) payback period is the amount of time take to recoup, in the form of net cash inflows, the total invested amount in the project. Payback analysis determines how much time will lapse before accrued benefits overtake accrued costs. It occurs when the net cumulative benefits equals the net cumulative benefits cost equals zero. As stated by (Fonollera, 2009) the payback period is computed by cumulating the estimated annual cash inflows and determining the point in time at which they equal the investment outlay provided the periodic cash flows are not uniform.

**The Cash Flow Statement:** according to (Fonollera, 2009) a cash flow statement is a financial statement that provides information about cash inflows (receipts or sources of cash) and cash outflows (payments or uses of cash) of the business for the given period of time. Cash inflows are money transactions that go into the business while cash outflows are cash transaction that goes out of the business. The net cash flow formula is stated below (Schwalbe, 2012):

\[
\text{NET CASH FLOW} = \text{INFLOWS} - \text{OUTFLOWS} \\
\text{OR} \\
\text{CASH FLOW} = \text{BENEFITS} - \text{COSTS}
\]

**Cost estimate:** according to Investor world (2016) cost estimate is an appropriate value of the total cost of a service, product, resources or project used for planning, sales quotes or resource allocation. Cost estimate are generally prepared as accurately
as possible to prevent misallocation of resources or negative perceptions from clients, managers or potential customers. Following the steps involves in the ground rule and assumptions for the cost estimates according to (Schwalbe, 2012) the steps listed below will be used for cost estimate of the proposed system. The steps are as follows:

1. This project was preceded by a detailed study and proof of concept to show that it is possible to develop a software for plagiarism detection to assist students and academic staff in detecting plagiarism in copies of submitted thesis and assignments in Nigerian Universities.

2. The main goal of this project is to design an enhanced system of data mining in telecommunication service provider to support decision making, continuing developing the software (especially the user interfaces), test the proposed system in telecommunication companies and train 110 data analyst in selected regions to use the proposed system.

3. The project has the following working breakdown structure (WBS):
   a. Project management
   b. Hardware : laptop devices and Servers
   c. Software development
   d. Testing
   e. Training and Support
   f. Reserves

3.4.1 Performance Indicator of the Existing System
In Vedran (2011), test cases used were written in C# programming language and all of them were created by the authors. Test cases were placed in 6 different categories and their total number is 50.

Categories were constructed considering variable names, types, properties, methods and classes. All the variable test categories included checking the behaviour of the algorithm when the variable names, types, assigned constant values and location of their declaration varied. Various property definition styles, their type, name and returning values were tested. Although it tests some of the most common variations with variables, including usage of various existing methods for converting one variable type to another, the syntax itself contains a relatively small number of test cases. Cases that test changing method name, returning type and various parameter reordering, insertions and deletions are also significant parts of the method in question.

The loops category contains test cases that check various loop replacements and definitions while the class category deals with cases that test changing class name, namespace, and reordering and renaming of class members. Authors conducted manual inspection of all test cases used in the research. The results of the aforementioned manual comparison are shown in a 50x50 matrix. The rows and columns of which are correspondent to the test cases used. Each cell represents the value between two test cases. This matrix represents a reference matrix to which all results obtained by plagiarism detection systems are compared. The two used evaluation methods were precision and recall including their harmonic mean, the F measure. Those methods evaluate the algorithms’ behaviour and sensitivity to various
code modification techniques. Precision is defined as a fraction of correctly
categorized test cases divided by the number of test cases claimed to be similar. Recall
is defined as fraction of correctly categorized test cases divided by the number of test
cases manually categorized as similar. F measure is defined as a harmonic mean of
precision and recall, so that both measures are equally represented Vedran, (2011).

As it is expected, all plagiarism detection systems have very high recall when the
threshold is low, and it decreases as the threshold increases. ILMatch is the only
system that does not reach recall of 100%; its highest value is 95%. On the other hand,
precision rises with a threshold, and reaches 100% on the high threshold, except for
the CodeMatch system, whose maximal precision is 81%. The best identified F-measures for tested plagiarism detection systems are shown in the Figure 3.7. Performance of MOSS and JPlag is almost the same: the best F-measure for those systems is about 73%, while CodeMatch and ILMatch show the best performance, their best F-measure is about 85%.

![Fig 3.7 Highest F-measures recorded by existing systems (Vedran, 2011).](image)

The performance indicators identified in this study for evaluation and assessment are qualities such as Precision, Recall, F-measure and Plagdet Score (Vedran, 2011).

The calculations for the F-measure, Precision, recall and granularity are given below. Let $S$ be the set of all plagiarized passages and $R$ the set of all the detections made by the plagiarism detection algorithm. Let $S_R$ be a subset of $S$, whose detections are present in $R$. The evaluation metrics Obbereuter et al. (2011) proposed for plagiarism detection is

Recall = \[ \frac{1}{|S|} \sum_{i=1}^{|S|} \left( \frac{\# \text{ of detected chars of } s_i}{|s_i|} \right) \] .......................... (3.3)

Precision = \[ \frac{1}{|R|} \sum_{i=1}^{|R|} \left( \frac{\# \text{ of Plagiarised chars of } r_i}{|r_i|} \right) \] .......................... (3.4)

Granularity = \[ \frac{1}{|S_R|} \sum_{i=1}^{|S_R|} \left( \# \text{ detections of } s_i \in R \right) \] .......................... (3.5)
Overall Score = \frac{F-Measure}{\log_2(1+\text{granularity})} \hspace{1cm} \text{(3.6)}

F-Measure = 2 \times (\text{recall} \times \text{precision}) / (\text{recall} + \text{precision}) \hspace{1cm} \text{(3.7)}

Table 3.0 Results for performance measures for various algorithms developed at PAN-10. (Obbereuter et al., 2011)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Overall Score</th>
<th>F-Measure</th>
<th>Precision</th>
<th>Recall</th>
<th>Granularity</th>
<th>Lead Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.80</td>
<td>0.80</td>
<td>0.94</td>
<td>0.69</td>
<td>0.69</td>
<td>Kasprzak et al</td>
</tr>
<tr>
<td>2.</td>
<td>0.71</td>
<td>0.74</td>
<td>0.91</td>
<td>0.63</td>
<td>0.63</td>
<td>Zou et al</td>
</tr>
<tr>
<td>3.</td>
<td>0.69</td>
<td>0.77</td>
<td>0.84</td>
<td>0.71</td>
<td>0.71</td>
<td>Muhr et al</td>
</tr>
<tr>
<td>4.</td>
<td>0.62</td>
<td>0.63</td>
<td>0.91</td>
<td>0.48</td>
<td>0.48</td>
<td>Grozea et al</td>
</tr>
<tr>
<td>5.</td>
<td>0.61</td>
<td>0.61</td>
<td>0.85</td>
<td>0.48</td>
<td>0.48</td>
<td>Obbereuter et al</td>
</tr>
<tr>
<td>6.</td>
<td>0.59</td>
<td>0.59</td>
<td>0.85</td>
<td>0.45</td>
<td>0.41</td>
<td>Torrejon et al</td>
</tr>
<tr>
<td>7.</td>
<td>0.52</td>
<td>0.53</td>
<td>0.73</td>
<td>0.41</td>
<td>0.39</td>
<td>Pereira et al</td>
</tr>
<tr>
<td>8.</td>
<td>0.51</td>
<td>0.52</td>
<td>0.78</td>
<td>0.39</td>
<td>0.29</td>
<td>Palkovskii et al</td>
</tr>
<tr>
<td>9.</td>
<td>0.44</td>
<td>0.45</td>
<td>0.96</td>
<td>0.29</td>
<td>0.32</td>
<td>Sobha et al</td>
</tr>
<tr>
<td>10.</td>
<td>0.26</td>
<td>0.39</td>
<td>0.51</td>
<td>0.32</td>
<td>0.24</td>
<td>Gottron et al</td>
</tr>
<tr>
<td>11.</td>
<td>0.22</td>
<td>0.38</td>
<td>0.93</td>
<td>0.24</td>
<td>0.30</td>
<td>Micol et al</td>
</tr>
<tr>
<td>12.</td>
<td>0.21</td>
<td>0.23</td>
<td>0.18</td>
<td>0.30</td>
<td>0.17</td>
<td>Costa-jussa et al</td>
</tr>
<tr>
<td>13.</td>
<td>0.21</td>
<td>0.24</td>
<td>0.40</td>
<td>0.17</td>
<td>0.14</td>
<td>Nawab et al</td>
</tr>
</tbody>
</table>

F-measure is the harmonic mean of the precision and recall values. The harmonic mean of a list of numbers tends strongly toward the least elements of the list, it tends (compared to the arithmetic mean) to mitigate the impact of large numbers and aggravate the impact of small ones. Granularity was introduced to quantify the number of the detections of a given plagiarized passage. If the algorithmic model detects the passage more than once, it gets penalized. The lower the granularity score, the better the algorithm.

3.4.2 Performance Indicator of the Proposed System
In assessing the performance of the proposed system, the performance indicator identified is the same of that of existing system. In this work, we are interested in knowing at what n-gram value we’ll get the highest precision for retrieved results. This will be discussed later in the result in Chapter 4.
CHAPTER 4

Design and Implementation

4.1 Data Presentation and System Analysis

This chapter presents the physical/logical design, system analysis of the existing and proposed system, financial analysis, implementation, testing and results achieved were also discussed.

4.1.1 Data Preparation

This is the process of gathering, combining, structuring and organizing data so it can be analyzed and checked for similarities against the plagiarism corpus. The components of data preparation include data generation, profiling, injection, matching and grouping; it often also involves comparing n-grams of data pooled from various sources or generated electronically to test for plagiarism in the source document. In this study data collected for testing the plagiarism detection process falls into two categories: secondary data and primary data.

Secondary data: in this study a corpus of already generated data from the PAN-09 plagiarism detection task was used as a training data set for the model. The data was prepared fed into the detection database and formed the source data for the test. The database design is presented in section (4.2.1) of this chapter.

Primary data: in this study the primary data formed the plagiarized documents itself. The data was generated by obfuscating some of the text in the primary data, changing
some of the synonyms and stop words and trying to change syntax and semantic of the information presented in the source document.

4.1.2 System Analysis

The present study adopted system analysis and procedure of structural system and design methodology (Gangolly, 1997). The steps involved in the analysis was enumerated in chapter three (section 3.4).

Analysis of Data Collected from the Existing System.

The features of the existing plagiarism detection softwares are ranked on a scale of 1 to 5. The various plagiarism detection algorithms from the PAN-09 Plagiarism test challenge were summarized and analyzed and the various precision, recall and Granularity scores recorded as recorded in Potthast (2010). The test was carried out on the same text corpora used in this work.
Table 4.1 Performance results for the external plagiarism detection task (Potthast, et.al, 2010).

<table>
<thead>
<tr>
<th>Rank</th>
<th>Overall F</th>
<th>Precision</th>
<th>Recall</th>
<th>Granularity</th>
<th>Participant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.4871</td>
<td>0.4884</td>
<td>0.5193</td>
<td>0.4610</td>
<td>Grozea, Gehl and popescu (2009)</td>
</tr>
<tr>
<td>2.</td>
<td>0.4265</td>
<td>0.4335</td>
<td>0.3901</td>
<td>0.4877</td>
<td>Kasprzak, Brandejs and Kripac (2009)</td>
</tr>
<tr>
<td>3.</td>
<td>0.4229</td>
<td>0.4544</td>
<td>0.4709</td>
<td>0.4390</td>
<td>Basile et al (2009)</td>
</tr>
<tr>
<td>4.</td>
<td>0.2131</td>
<td>0.3700</td>
<td>0.4682</td>
<td>0.3059</td>
<td>Palkovskii, Belov, and Muzika (2009)</td>
</tr>
<tr>
<td>5.</td>
<td>0.1833</td>
<td>0.4001</td>
<td>0.4826</td>
<td>0.3417</td>
<td>Muhr et al. (2009)</td>
</tr>
<tr>
<td>6.</td>
<td>0.9966</td>
<td>0.4333</td>
<td>0.5231</td>
<td>0.3699</td>
<td>Scherbinin and Butakov (2009)</td>
</tr>
<tr>
<td>7.</td>
<td>0.0739</td>
<td>0.0926</td>
<td>0.0696</td>
<td>0.1382</td>
<td>Stamatatos (2009)</td>
</tr>
<tr>
<td>8.</td>
<td>0.0586</td>
<td>0.0587</td>
<td>0.0327</td>
<td>0.2831</td>
<td>Hagbi and Koppel (2009)</td>
</tr>
<tr>
<td>9.</td>
<td>0.0454</td>
<td>0.1216</td>
<td>0.4586</td>
<td>0.0701</td>
<td>Pereira, Moreira, and Galante (2009)</td>
</tr>
<tr>
<td>10.</td>
<td>0.0366</td>
<td>0.0525</td>
<td>0.0311</td>
<td>0.1689</td>
<td>Seaward and Matwin (2009)</td>
</tr>
<tr>
<td>11.</td>
<td>0.0184</td>
<td>0.0185</td>
<td>0.0095</td>
<td>0.3210</td>
<td>Vallés Balaguer (2009)</td>
</tr>
<tr>
<td>12.</td>
<td>0.0131</td>
<td>0.0387</td>
<td>0.0203</td>
<td>0.4234</td>
<td>Malcolm and Lane (2009)</td>
</tr>
<tr>
<td>13.</td>
<td>0.0081</td>
<td>0.0157</td>
<td>0.2579</td>
<td>0.0081</td>
<td>Allen (2009)</td>
</tr>
</tbody>
</table>

Some striking results in table 3.2 that call for analysis are the precision score, Recall and the Granularity.

Figure 4.1 is a graphical plot of the various algorithms implemented in the PAN-09 plagiarism detection exercise and the corresponding granularity scores. A high granularity score (Scherbinin et. al, 2009) in table 3.2 tells us that the algorithm detects already detected passages and thus may report a high precision, and a corresponding low recall value, but this is a false positive because it does not give the true position of the detection process.

Grozea, et.al (2009), had a high precision score of 0.5193, which was higher than all the other precision values. A precision value that tends towards 1 means the algorithm is performing well. Other performance issue that may be of interest is the running time of the algorithm, the time and space complexity of the algorithm.
None of the algorithms in table 3.2 have a precision score less than 0.8. This means that the level of accuracy of the detection algorithm is less than 80%.

The Corpus used for the analysis of the algorithm in this work is the PAN-PC-09 corpus. We present below the analysis of other algorithms presented by participants of the competitive conference. For the competition, 13 different participants developed plagiarism detection systems to tackle one or both of the tasks external plagiarism detection and intrinsic plagiarism detection. The questions that naturally arise: how do they work and how well? We survey the approaches in a unified way and report on their detection quality in the competition. Table 3.1 gives the analysis and comparison of the various existing algorithms and their heuristic retrieval models.
Table 4.2: analysis and comparison of existing systems and their heuristic retrieval models. (Potthast et.al, 2010)

<table>
<thead>
<tr>
<th>Heuristic Retrieval</th>
<th>Detailed Analysis</th>
<th>Participant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieval Model.</td>
<td>Exact Matches of dq and dx ∈ Dx. Character-16-grams</td>
<td>Grozea, Gehl, and Popescu (2009)</td>
</tr>
<tr>
<td>Character-16-gram Vector Space Model</td>
<td>Match Merging Heuristic to get (sq, sx). Computation of the distances of adjacent matches. Joining of the matches based on a Monte Carlo optimization. Refinement of the obtained section pairs, e.g., by discarding too small sections.</td>
<td></td>
</tr>
<tr>
<td>(frequency weights, cosine similarity)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comparison of Dq and D.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exhaustive Candidates Dx ⊂ D for a dq</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The 51 documents most similar to dq</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrieval Model.</td>
<td>Exact Matches of Dq and dx ∈ Dx. Word-5-grams Match Merging Heuristic to get (sq, sx). Extraction of the pairs of sections (sq, sx) of maximal size which share at least 20 matches, including the first and the last n-gram of sq and sx, and for which 2 adjacent matches are at most 49 Not-matching n-grams apart.</td>
<td>Kasprzak, Brandejs, and K˚ripa˚c (2009)</td>
</tr>
<tr>
<td>Word-5-gram VSM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Boolean weights, Jaccard similarity)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comparison of Dq and D.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exhaustive Candidates Dx ⊂ D for a dq</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documents which share at least 20 n-grams with dq.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrieval Model.</td>
<td>Exact Matches of dq and dx ∈ Dx. Word-8-grams Match Merging Heuristic to get (sq, sx). Extraction of the pairs of sections (sq, sx) which are obtained by greedily joining consecutive matches if their distance is not too high.</td>
<td>Basile et al. (2009)</td>
</tr>
<tr>
<td>Word-8-gram VSM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(frequency weights, custom distance)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comparison of Dq and D.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exhaustive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Candidates Dx ⊂ D for a dq.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The 10 documents nearest to dq.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrieval Model.</td>
<td>Exact Matches of dq and dx ∈ Dx. Sentences Match Merging Heuristic to get (sq, sx). Extraction of the pairs of sections (sq, sx) which are obtained by greedily joining consecutive sentences. Gaps are allowed if the respective sentences are similar to the corresponding sentences in the other document.</td>
<td>Muhr et al. (2009)</td>
</tr>
<tr>
<td>Word-1-gram VSM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(frequency weights, cosine similarity).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comparison of Dq and D.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clustering-based data-partitioning of D’s sentences. Comparison of Dq’s sentences with each partitions’ centroid.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

83
Candidates $D_x \subseteq D$ for a dq. For each sentence of dq, the documents from the 2 most similar partitions which share similar sentences.

Retrieval Model. Winnowing fingerprinting 50 char chunks with 30 char overlap. Comparison of Dq and D. Exhaustive Candidates $D_x \subseteq D$ for a dq. Documents whose fingerprints share at least one value with dq’s fingerprint.

Exact Matches of dq and $d_x \in D_x$. Fingerprint chunks Match Merging Heuristic to get (sq, sx). Extraction of the pairs of sections (sq, sx) which are obtained by enlarging matches and joining adjacent matches. Gaps must be below a certain Levenshtein distance.

Scherbinin and Butakov (2009)

### 4.2 System Design

System design is the process of defining the components, modules, interfaces, and data for a system to satisfy specified requirements (Blanchard and Fabrycky, 2010).

#### 4.2.1 Logical System Design of Proposed System

Logical system design pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modelling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems, designs are included (Ulrich and Eppinger, 2000). The proposed system will split words in a suspicious text into various n-grams and match them against a dictionary of existing articles and documents which have been archived and stored in a database.

Figure 4.1 shows a logical design showing various modules in the plagiarism detection system that implements an n-gram string matching technique.
Fig 4.1 Logical system design showing various modules in the plagiarism detection system.

The input to the system will be suspicious documents in document or text formats. These documents will go through the preprocessing stage, where the text will be split into passages and the number of n-grams, sentences, words and punctuation marks will be taken note of.

4.2.1.1 High Level Models

System Architecture with Server Technology

To test the algorithm, this work imitates a typical four tier system and the components have its meaning as follows:

1. **Server Suite**: this comprises of three component Network server which manages the entire network processes in the system; HTTP server which manages all http requests in the system and also all processes linking the system to another network; Data storage and processing server which creates and processes complex data comparison activities/procedures
2. **Process Server**: This comprises of the two components DBMS (Database Management System) which is responsible for the retrieval, inserting, operation and maintenance of Data in the data repository while data repository which store the actual data in the system.

3. **Process Node**: This module is responsible for channeling users request to the proper channel for effective and efficient responses.

4. **User Interface**: This is the Graphical User Interface (GUI) component of the system which includes where the administrator can input data can carry out other operations as required. This is also includes screens and forms for users to view the profiles and carry out other operations as deemed fit. Tools for developing this component include HTML, CSS and JavaScript.

Figure 4.2: Architecture of the Proposed System with Server Technology
Framework of the Proposed System

Also, figure 4.3 illustrates the operational 3–tier Framework of the proposed system. This diagram shows the communication existing between the users (from the interface), Data Processing and storage and string Matching and the Output.

**Graphical User Interface:** this module is responsible for the interaction/communication between the system and humans.

**Sever Infrastructure:** Enabling component communication and access to privilege users to access the dynamic web document.

**Database Server:** This module is responsible for storage of data within the system to be used and sent to/from the system to other devices.

**String Matching Approach:** involves the application of string matching techniques such as Brute force, Knuth-Morris-Pratt etc. for text matching and plagiarism detection.
4.2.1.2 Input/output Design

The Input module is the interface through which the user communicate with the system by supplying an input. The Output module provides scripts (documents, results based on the operations performed in the system.

Input Design

The system uses this module to capture information from external environment, the input into the system is the source documents and the suspicious documents which is used to train the algorithm and detect plagiarism.

(a) New User Registration: the template captures the details of an individual that want to use the system for plagiarism detection. This is shown in figure 4.10

Figure 4.4: New User Registration Template
(b) **Algorithm Settings:** this template captures the settings for the algorithm for training the plagiarism algorithm, and also to select the n-gram suitable for string matching against the matching server. Figure 4.5 shows the screen for selection of the desired algorithm for the string Matching.

![Figure 4.5: Settings for the Plagiarism Detection Interface.](image)

(c) **Input for the Algorithm Training:** this template captures the input of the text corpora for training the algorithm, and for detecting plagiarism in suspicious Documents.

![Figure 4.6: Input screen for the Entering of Suspicious documents and training Data.](image)
Output Design

The system uses this module to convey information such as results, acknowledgement receipt to people. Thus, the output templates are presented as follows.

(a) Sample Output for Plagiarism Detection Results: Figure 4.16 display the output of the new user registration.

Figure 4.7: Sample output for Plagiarism Detection result.

4.2.1.7 Database Design

The database designs illustrate the classification details of the data records used for the corpus saving and string matching analysis. This is done in order to establish the
relationships within the entities making up the data set in the database. The tables below show the operations required to perform analysis using the data mining techniques.

1. **Admin.mdf**

This database file structure captures and stores the admin/users login details with respect to their levels of permission and the Table 4.8 present it.

Table 4.3: Admin.dbf Structure

<table>
<thead>
<tr>
<th>S/N</th>
<th>FIELD NAME</th>
<th>DATA TYPE</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>USERNAME</td>
<td>VARCHAR</td>
<td>122</td>
</tr>
<tr>
<td>2</td>
<td>PASSWORD</td>
<td>VARCHAR</td>
<td>122</td>
</tr>
<tr>
<td>3</td>
<td>LEVEL</td>
<td>VARCHAR</td>
<td>122</td>
</tr>
<tr>
<td>4</td>
<td>CREATOR</td>
<td>VARCHAR</td>
<td>122</td>
</tr>
</tbody>
</table>

2. **Ngrams.mdf**

This database file structure captures the n-gram groups in documents according to the setting selected. This it to avoid a waste of computing resources while the string matching is being performed.

Table 4.4: Word Ngram.dbf Structure

<table>
<thead>
<tr>
<th>S/N</th>
<th>FIELD NAME</th>
<th>DATA TYPE</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DOCUMENT NAME</td>
<td>Varchar</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>AUTHOR</td>
<td>Varchar</td>
<td>122</td>
</tr>
<tr>
<td>3</td>
<td>NUMBER OF WORDS</td>
<td>Varchar</td>
<td>122</td>
</tr>
<tr>
<td>4</td>
<td>N-GRAMGROUP</td>
<td>Text</td>
<td>max</td>
</tr>
<tr>
<td>6</td>
<td>DATE</td>
<td>DateTime</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>YEAR</td>
<td>Varchar</td>
<td>30</td>
</tr>
</tbody>
</table>
3. **References.mdf**

This database file structure captures the reference documents that were used to check against the suspicious documents.

**Table 4.5: Table showing the entities for storing the reference text**

<table>
<thead>
<tr>
<th>DOI</th>
<th>int</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHOR_NAME</td>
<td>varchar(50)</td>
</tr>
<tr>
<td>DOCUMENT_NAME</td>
<td>varchar(50)</td>
</tr>
<tr>
<td>NO_OF_WORDS</td>
<td>int</td>
</tr>
<tr>
<td>NO_OF_PARAGRAPHS</td>
<td>int</td>
</tr>
<tr>
<td>NO_OF_SENTENCES</td>
<td>int</td>
</tr>
<tr>
<td>TEXT</td>
<td>varchar(MAX)</td>
</tr>
<tr>
<td>DATE_ARCHIVED</td>
<td>date</td>
</tr>
</tbody>
</table>

4. **Rank.mdf**

This database file structure stores the documents that have been ranked temporarily. This is to avoid over clogging the memory of the device if the suspicious document to be checked is very large.

**Table 4.6: Table showing the entities for storing and ranking the plagiarized text**

<table>
<thead>
<tr>
<th>RANK_ID</th>
<th>INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOI</td>
<td>INT</td>
</tr>
<tr>
<td>AUTHOR_NAME</td>
<td>VARCHAR(50)</td>
</tr>
<tr>
<td>DOCUMENT_NAME</td>
<td>VARCHAR(50)</td>
</tr>
<tr>
<td>TEXT</td>
<td>VARCHAR(MAX)</td>
</tr>
<tr>
<td>DATE_ARCHIVED</td>
<td>DATE</td>
</tr>
<tr>
<td>RANK</td>
<td>INT</td>
</tr>
<tr>
<td>DOCUMENT_TAG</td>
<td>INT</td>
</tr>
</tbody>
</table>
4.2.1.8 Entity Relationship Diagram

An entity–relationship model is a diagrammatic data model for showing the relationship between various aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database, thereby making it relational. The main components of Entity Relationship models are entities and the relationships that can exist among them. The entities in this work are the ADMIN.MDF, NGRAM.MDF, REFERENCES.MDF, RANK.MDF. Thus, this is illustrated in figure 4.8
4.2.1.9 System Algorithm, Flowchart and String Matching Method

Algorithm design is of utmost importance in software development; it simplifies the job of a programmer; inform him of his next step in the conception of a program and guide him towards the realization of the entire program.

1. System Algorithm

An algorithm is a logical sequence of discrete steps used to solve a problem in a finite amount of time. It is therefore the breakdown of a particular task into several specific sub-tasks. It’s important to a system analyst and a programmer is to understand the concept of algorithm in details so as to be able to come up with simple and efficient systems (Rouse, 2014)

a. Ngram Splitting and String Matching Algorithm

The algorithm is for splitting the strings into n-grams and matching the texts with the reference document.

\[ \text{N-gram + string matching algorithm (Ebuka 2016)} \]

Let the string to be matched be \( S_m \) and let the Plagiarised String be \( S_p \)

1. Input Plagiarised text and Source Text \((S_m, S_p)\)
2. Split plagiarised text \( S_p \) into passages by their paragraphs \( \text{goto split_plag}(S_p) \)
3. Split sentences in \( S_p \) into words \( \text{goto split_words}(S_w) \)
4. Foreach word in \( S_w \)
Begin
Get the n-gram word neighbour
Start at index i on $S_p$
Goto StringMatch(ngram,$S_p$)
End

5. Cluster words according to tag given
6. Display results and calculated precision
7. Calculate rank and f-measure

b. Paragraph Splitting Algorithm

The algorithm is for splitting the strings into paragraphs.

// subroutine to split into paragraphs
split_plag($S_p$)
I. begin
   a. foreach string sentence in $S_p$
   b. if _sentence begins with a paragraph
      i. return _sentence
   c. else
      i. continue
II. end
// subroutine to split paragraphs into words

c. Word Splitting Algorithm

The algorithm is for splitting the Paragraphs into words.

split_words ($S_w$)
I. begin
   a. foreach word in $S_w$
   b. if word begins with a space
      i. return word
   c. else
      i. continue
end
d. SubAlgorithm for Matching string

// subroutine to match ngram in source Document
StringMatch(ngram, S_p)
int tag = 0
Begin
Foreach gram in ngrams
    Begin
        Foreach n-words in S_p
            If gram == n-words in S_p
                tag n-words
            end
        end
    tag++
end

The running time of the algorithm is given as O(n^2) for StringMatch Algorithm, O(n) for the split_words, O(n) for the split_plag

The whole time complexity for the application will be given as O(n) x O(n^3) x O(n)

Since the worst time that will be taken to run the algorithm is the time taken by the StringMatch algorithm, the worst time for the execution of the algorithm will be O(n^3)

4.2.2 Program Flowchart

Program flowchart is the diagrammatic representation of how the entire system works.

The system flowcharts of the model is presented figure 4.2.2
Fig 4.2: Flowchart of the plagiarism detection system.
4.2.3 Physical System Design

The physical design is followed by a physical design or coding. Physical design produces the working system by defining the design specifications, which tell the programmers exactly what the candidate system must do. The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data through call and produce the required report on a hard copy or display it on the screen.

4.2.4 Design Considerations

In developing the software many factors were considered in order to make the software conform to the standard of accuracy required for deployment purposes and also coding standards set by Microsoft in developing standard applications. Some of these factors include.

4.2.5 Interactivity

The application must be as interactive as possible, and make the user feel very comfortable navigating through the various Pages. Throughout the course of using the application the end user should be guided through by a series of steps in other to make the software usage as easy as possible.
4.2.6 Simplicity

Various Standard Menus and toolbars belonging to the application were strategically placed on the user interface of the program; this is to ensure the least possible complexity in the day to day usage of the application.

4.2.7 Security

Application Security is considered in the software in the use of system resources, this disallows unauthorized use of the systems resources when the application is being run by another program in the kernel. Security is very important in Windows applications as this ensures the calls made to the processor thread emanate from the application and the services started by the application are stopped accordingly, so as to avoid unnecessary overloading of system resources.

4.2.8 Output design Screenshot.

The sample of the output displayed here is the screenshot of the Output of the result of the various analysis done on the text corpora.

Figure 4.3 shows the settings for the plagiarism detection software and the various algorithm selectors. There is also an option for selecting to match the suspicious document on the local corpora or to match it against the archived documents on the web.

Figure 4.4 is the input space for entering the text documents or for selecting them from a file system. The text extracted from the document can then be used to train the string matching model and also to perform plagiarism detection tasks.
Fig 4.3: Plagiarism selection setting for different n-gram levels

Fig 4.4: Input space for the suspicious document, to be entered as text or uploaded.

Figure 4.5 shows the detections on a document that was plagiarized. The n-gram value of 2 was used to check the document and the plagiarized sections were highlighted.
Figure 4.7 shows the detections on a document that was plagiarized. The n-gram value of 3 was used to check the document and the plagiarized sections were highlighted.

Figure 4.8 shows the detections on a document that was plagiarized. The n-gram value of 4 was used to check the document and the plagiarized sections were highlighted.

Figure 4.9 shows the detections on a document that was plagiarized. The n-gram value of 5 was used to check the document and the plagiarized sections were highlighted.

Figure 4.10 shows the detections on a document that was plagiarized. The n-gram value of 6 was used to check the document and the plagiarized sections were highlighted.

**Fig 4.5:** Detections and highlight of plagiarised sections based on ngram = 2
PageRank is a link analysis algorithm used by the Google Internet search engine that assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. Google assigns a numeric weighting from 0-10 for each webpage on the Internet; this PageRank denotes a site's importance in the eyes of Google.

"Bayes' Theorem" or "Bayes' Rule", or something called Bayesian reasoning.

The Bayesian Conspiracy is a multinational, interdisciplinary, and shadowy group of scientists that controls publication, grants, tenure, and the illicit traffic in grad students. The best way to be accepted into the Bayesian Conspiracy is to join the Campus Crusade for Bayes in high school or college, and gradually work your way up to the inner circles.

Bayes' Theorem

Let and be sets. Conditional probability requires that

Fig 4.6: Detection Density using n-gram = 3

Fig 4.7: Detection Density using n-gram = 4
Figure 4.10 shows the detections on a document that was plagiarized. The n-gram value of 6 was used to check the document and the plagiarized sections were highlighted.
4.3 System Requirements

The proposed system is a typical web-based application. Therefore, the basic system requirement consists of hardware, software and database component requirements. These are explained in section (4.3.1, 4.3.2, and 4.3.3) respectively.

4.3.1 Hardware Requirements

The new system is designed to be used over the network with the following minimum hardware specification requirements:

1. Pentium II series processor
2. RAM (Random Access Memory) 312 MB (Mega Byte)
3. CD-ROM drive, 48x (speed)
4. 10 GB (Giga Byte)
5. 56kbps full duplex fax modem
6. Any external drive for backup
7. 15” color monitor, preferably a flat screen monitor for space management and to avoid excess radiation on the eye of the user.

4.3.2 Software Requirements

The new system is designed to be implemented with the following minimum software requirements:

1. Operating System: any version of Microsoft Windows operating system (windows NT/2000/ME/XP/Vista/7/8), any version of Linux, and any version of Macintosh will enable one to run this system comfortably.

2. Anti-virus/Anti-spyware Software: Any good anti-virus software consisting of Norton’s, Avira, among others, that will protect the system against malicious attacks is useful.

3. System Browser Software: Internet Explorer (any version)/ Mozilla Firefox browser/Opera browser/Safari browser or any other system browser.

4.3.3 Database Requirements

The database requirements needed to implement the new system consist of the following:

1. Microsoft Windows Server 2014
2. Microsoft .NET Framework 5.0
4.4 Implementation and Testing

4.4.1 Implementation

The proposed programming language chosen C#, it is well specified in the software requirement under database specification in the section 4.3.2.

**Coding:** This is the actual programming for implementing the designs in this work. It consists of the major modules such as:

1. Database Module: This is to establish the connection between the front end and back end of the system development.
2. CSS module: These codes are used for interface layout design and arrangement.
3. Java script module: This is used for validation and events automation that links the functional programs.
4. Image module: This contents the images or pictures that displays on the system interface.
5. Login and logout module: It concerns with designing the login and logout interfaces.
6. Index page: These codes enables is used to realize the homepage of the system.
7. Program Listing: Program listing of the modules 1 to 6 and other files can be seen at the appendix

4.4.2 Testing

Testing is an important part of every software development. This is done to verify if the system achieves the goal set for it. It involves the execution of a software component or system to evaluate one or more properties of interest. There are various phases of testing done in this proposed system. However, the tests performed include:

1. Unit Testing

This testing method checks individual units of source code for errors or repeated entities. It also used to test sets of one or more program modules together with associated control data, usage procedures, and operating procedures to determine if they are fit for use.

The unit testing on the proposed application was performed on the various algorithms and the performance of the various algorithms noted. The units in the proposed system returned the expected results as discussed in section 3.4.1

2. Integration Testing

Integration testing is a software testing in which individual software modules are combined and tested as a group. It is usually done after unit testing and before validation testing. Integration testing takes modules that have been unit tested, groups them in larger aggregates and applies tests defined in an integration test plan to those aggregates. This serves as its input. The output is the integrated system
ready for system testing. The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware.

3. **System Testing.**

System Testing is testing conducted on a complete, integrated system to evaluate the system’s compliance with its specified requirements. System testing is a more limited type of testing; it seeks to detect defects both within the code and also within the system as a whole. System testing was carried out on the proposed system and it was found to return the desired results as discussed in section 3.4.1.

The test performed was on each source document in the PAN-09 Corpora. This involved feeding the proposed system with each of the source documents and using it to train the algorithm. All the documents were saved in the database and for each obfuscated document fed into the system, the various corresponding documents where plagiarism occurred were noted and tagged. The precision and recall values were then calculated from there using the standard procedures for calculating precision and recall. This was done and repeated severally until the iteration was completed.
4.5 Test and Test Results. The identified weak performance indicators are the major indicators considered in evaluation of the new system. After running various PAN-09 corpora on the system, various precision values were generated for various results at different n-gram values of 1,2,3,4,5,6,7.

For the combined string matching and n-gram plagiarism detection algorithm, at an N-gram value of n=1, the highest precision value recorded was 0.75.

For the combined string matching and n-gram plagiarism detection algorithm, at an n-gram value of n=2, the highest precision value recorded was 0.98 and the lowest precision value recorded was 0.45. In this work we were more concerned with the values of precision because this gives you the number of relevant documents that were plagiarized in a corpora with respect to the suspicious document. The highest recall value was 0.89 and the lowest value was 0.2.
### Table 4.2. N-gram = 1 for 30 iterations using 392 documents from the PAN-09 Corpora

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>2.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>3.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>4.</td>
<td>0.46</td>
<td>0.76</td>
<td>0.57311475</td>
</tr>
<tr>
<td>5.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>6.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>7.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>8.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>9.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>10.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>11.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>12.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>13.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>14.</td>
<td>0.45</td>
<td>0.75</td>
<td>0.5625</td>
</tr>
<tr>
<td>15.</td>
<td>0.46</td>
<td>0.76</td>
<td>0.57311475</td>
</tr>
<tr>
<td>16.</td>
<td>0.46</td>
<td>0.76</td>
<td>0.57311475</td>
</tr>
<tr>
<td>17.</td>
<td>0.46</td>
<td>0.76</td>
<td>0.57311475</td>
</tr>
<tr>
<td>18.</td>
<td>0.47</td>
<td>0.77</td>
<td>0.58370968</td>
</tr>
<tr>
<td>19.</td>
<td>0.48</td>
<td>0.7</td>
<td>0.56949153</td>
</tr>
<tr>
<td>20.</td>
<td>0.48</td>
<td>0.7</td>
<td>0.56949153</td>
</tr>
<tr>
<td>21.</td>
<td>0.48</td>
<td>0.68</td>
<td>0.56275862</td>
</tr>
<tr>
<td>22.</td>
<td>0.48</td>
<td>0.68</td>
<td>0.56275862</td>
</tr>
<tr>
<td>23.</td>
<td>0.48</td>
<td>0.68</td>
<td>0.56275862</td>
</tr>
<tr>
<td>24.</td>
<td>0.48</td>
<td>0.68</td>
<td>0.56275862</td>
</tr>
<tr>
<td>25.</td>
<td>0.48</td>
<td>0.6</td>
<td>0.53333333</td>
</tr>
<tr>
<td>26.</td>
<td>0.48</td>
<td>0.58</td>
<td>0.52528302</td>
</tr>
<tr>
<td>27.</td>
<td>0.48</td>
<td>0.53</td>
<td>0.50376238</td>
</tr>
<tr>
<td>28.</td>
<td>0.58</td>
<td>0.48</td>
<td>0.52528302</td>
</tr>
<tr>
<td>29.</td>
<td>0.63</td>
<td>0.42</td>
<td>0.504</td>
</tr>
<tr>
<td>30.</td>
<td>0.75</td>
<td>0.35</td>
<td>0.47727273</td>
</tr>
</tbody>
</table>
Table 4.3. N-gram = 2 for 30 iterations using 392 documents from the PAN-09 Corpora

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.45</td>
<td>0.8</td>
<td>0.576</td>
</tr>
<tr>
<td>2.</td>
<td>0.45</td>
<td>0.8</td>
<td>0.576</td>
</tr>
<tr>
<td>3.</td>
<td>0.43</td>
<td>0.8</td>
<td>0.55934959</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.42</td>
<td>0.8</td>
<td>0.55081967</td>
</tr>
<tr>
<td>2.</td>
<td>0.43</td>
<td>0.41</td>
<td>0.65</td>
</tr>
<tr>
<td>3.</td>
<td>0.51</td>
<td>0.49</td>
<td>0.65</td>
</tr>
<tr>
<td>4.</td>
<td>0.62</td>
<td>0.6</td>
<td>0.63</td>
</tr>
<tr>
<td>5.</td>
<td>0.67</td>
<td>0.69</td>
<td>0.62</td>
</tr>
<tr>
<td>6.</td>
<td>0.68</td>
<td>0.68</td>
<td>0.61</td>
</tr>
<tr>
<td>7.</td>
<td>0.66</td>
<td>0.69</td>
<td>0.59</td>
</tr>
<tr>
<td>8.</td>
<td>0.68</td>
<td>0.7</td>
<td>0.55</td>
</tr>
<tr>
<td>9.</td>
<td>0.7</td>
<td>0.72</td>
<td>0.52</td>
</tr>
<tr>
<td>10.</td>
<td>0.71</td>
<td>0.73</td>
<td>0.51</td>
</tr>
<tr>
<td>11.</td>
<td>0.74</td>
<td>0.79</td>
<td>0.49</td>
</tr>
<tr>
<td>12.</td>
<td>0.75</td>
<td>0.8</td>
<td>0.47</td>
</tr>
<tr>
<td>13.</td>
<td>0.77</td>
<td>0.81</td>
<td>0.45</td>
</tr>
<tr>
<td>14.</td>
<td>0.79</td>
<td>0.82</td>
<td>0.45</td>
</tr>
<tr>
<td>15.</td>
<td>0.8</td>
<td>0.83</td>
<td>0.4</td>
</tr>
<tr>
<td>16.</td>
<td>0.87</td>
<td>0.84</td>
<td>0.48</td>
</tr>
<tr>
<td>17.</td>
<td>0.82</td>
<td>0.84</td>
<td>0.45</td>
</tr>
<tr>
<td>18.</td>
<td>0.87</td>
<td>0.81</td>
<td>0.4</td>
</tr>
<tr>
<td>19.</td>
<td>0.91</td>
<td>0.82</td>
<td>0.38</td>
</tr>
<tr>
<td>20.</td>
<td>0.9</td>
<td>0.87</td>
<td>0.37</td>
</tr>
<tr>
<td>21.</td>
<td>0.91</td>
<td>0.84</td>
<td>0.33</td>
</tr>
<tr>
<td>22.</td>
<td>0.93</td>
<td>0.81</td>
<td>0.3</td>
</tr>
<tr>
<td>23.</td>
<td>0.95</td>
<td>0.84</td>
<td>0.3</td>
</tr>
<tr>
<td>24.</td>
<td>0.98</td>
<td>0.79</td>
<td>0.25</td>
</tr>
<tr>
<td>25.</td>
<td>0.92</td>
<td>0.79</td>
<td>0.23</td>
</tr>
<tr>
<td>26.</td>
<td>0.93</td>
<td>0.78</td>
<td>0.21</td>
</tr>
<tr>
<td>27.</td>
<td>0.95</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>28.</td>
<td>0.8</td>
<td>0.22</td>
<td>0.34509804</td>
</tr>
</tbody>
</table>

Table 4.3. N-gram = 3 for 30 iterations using 392 documents from the PAN-09
<table>
<thead>
<tr>
<th>Iteration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.3</td>
<td>0.75</td>
<td>0.428571</td>
</tr>
<tr>
<td>2.</td>
<td>0.39</td>
<td>0.75</td>
<td>0.513158</td>
</tr>
<tr>
<td>3.</td>
<td>0.4</td>
<td>0.74</td>
<td>0.519298</td>
</tr>
<tr>
<td>4.</td>
<td>0.5</td>
<td>0.76</td>
<td>0.603175</td>
</tr>
<tr>
<td>5.</td>
<td>0.55</td>
<td>0.72</td>
<td>0.623622</td>
</tr>
<tr>
<td>6.</td>
<td>0.58</td>
<td>0.71</td>
<td>0.63845</td>
</tr>
<tr>
<td>7.</td>
<td>0.57</td>
<td>0.69</td>
<td>0.624286</td>
</tr>
<tr>
<td>8.</td>
<td>0.59</td>
<td>0.67</td>
<td>0.62746</td>
</tr>
<tr>
<td>9.</td>
<td>0.66</td>
<td>0.66</td>
<td>0.66</td>
</tr>
<tr>
<td>10.</td>
<td>0.62</td>
<td>0.64</td>
<td>0.629841</td>
</tr>
<tr>
<td>11.</td>
<td>0.61</td>
<td>0.51</td>
<td>0.555536</td>
</tr>
<tr>
<td>12.</td>
<td>0.68</td>
<td>0.55</td>
<td>0.60813</td>
</tr>
<tr>
<td>13.</td>
<td>0.78</td>
<td>0.57</td>
<td>0.658667</td>
</tr>
<tr>
<td>14.</td>
<td>0.71</td>
<td>0.54</td>
<td>0.61344</td>
</tr>
<tr>
<td>15.</td>
<td>0.72</td>
<td>0.53</td>
<td>0.61056</td>
</tr>
<tr>
<td>16.</td>
<td>0.7</td>
<td>0.51</td>
<td>0.590083</td>
</tr>
<tr>
<td>17.</td>
<td>0.74</td>
<td>0.58</td>
<td>0.650303</td>
</tr>
<tr>
<td>18.</td>
<td>0.73</td>
<td>0.55</td>
<td>0.627344</td>
</tr>
<tr>
<td>19.</td>
<td>0.81</td>
<td>0.54</td>
<td>0.648</td>
</tr>
<tr>
<td>20.</td>
<td>0.72</td>
<td>0.42</td>
<td>0.530526</td>
</tr>
<tr>
<td>21.</td>
<td>0.77</td>
<td>0.47</td>
<td>0.58371</td>
</tr>
<tr>
<td>22.</td>
<td>0.75</td>
<td>0.43</td>
<td>0.54661</td>
</tr>
<tr>
<td>23.</td>
<td>0.72</td>
<td>0.42</td>
<td>0.530526</td>
</tr>
<tr>
<td>24.</td>
<td>0.77</td>
<td>0.44</td>
<td>0.56</td>
</tr>
<tr>
<td>25.</td>
<td>0.71</td>
<td>0.41</td>
<td>0.519821</td>
</tr>
<tr>
<td>26.</td>
<td>0.69</td>
<td>0.35</td>
<td>0.464423</td>
</tr>
<tr>
<td>27.</td>
<td>0.63</td>
<td>0.33</td>
<td>0.433125</td>
</tr>
<tr>
<td>28.</td>
<td>0.68</td>
<td>0.32</td>
<td>0.4352</td>
</tr>
</tbody>
</table>
Table 4.4. N-gram = 4 for 30 iterations using 392 documents from the PAN-09 Corpora

<table>
<thead>
<tr>
<th></th>
<th>0.73</th>
<th>0.31</th>
<th>0.435192</th>
</tr>
</thead>
<tbody>
<tr>
<td>29.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.</td>
<td>0.7</td>
<td>0.34</td>
<td>0.457692</td>
</tr>
</tbody>
</table>
Table 4.5. N-gram = 5 for 30 iterations using 392 documents from the PAN-09 Corpora

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.21</td>
<td>0.46</td>
<td>0.288358</td>
</tr>
<tr>
<td>2.</td>
<td>0.29</td>
<td>0.46</td>
<td>0.355733</td>
</tr>
<tr>
<td>3.</td>
<td>0.3</td>
<td>0.45</td>
<td>0.36</td>
</tr>
<tr>
<td>4.</td>
<td>0.4</td>
<td>0.44</td>
<td>0.419048</td>
</tr>
<tr>
<td>5.</td>
<td>0.47</td>
<td>0.43</td>
<td>0.449111</td>
</tr>
<tr>
<td>6.</td>
<td>0.48</td>
<td>0.42</td>
<td>0.448</td>
</tr>
<tr>
<td>7.</td>
<td>0.48</td>
<td>0.4</td>
<td>0.436364</td>
</tr>
<tr>
<td>8.</td>
<td>0.49</td>
<td>0.38</td>
<td>0.428046</td>
</tr>
<tr>
<td>9.</td>
<td>0.5</td>
<td>0.36</td>
<td>0.418605</td>
</tr>
<tr>
<td>10.</td>
<td>0.52</td>
<td>0.33</td>
<td>0.403765</td>
</tr>
<tr>
<td>11.</td>
<td>0.53</td>
<td>0.32</td>
<td>0.399059</td>
</tr>
<tr>
<td>12.</td>
<td>0.59</td>
<td>0.3</td>
<td>0.397753</td>
</tr>
<tr>
<td>13.</td>
<td>0.6</td>
<td>0.28</td>
<td>0.381818</td>
</tr>
<tr>
<td>14.</td>
<td>0.61</td>
<td>0.26</td>
<td>0.364598</td>
</tr>
<tr>
<td>15.</td>
<td>0.62</td>
<td>0.26</td>
<td>0.366364</td>
</tr>
<tr>
<td>16.</td>
<td>0.63</td>
<td>0.21</td>
<td>0.315</td>
</tr>
<tr>
<td>17.</td>
<td>0.64</td>
<td>0.29</td>
<td>0.39914</td>
</tr>
<tr>
<td>18.</td>
<td>0.64</td>
<td>0.26</td>
<td>0.369778</td>
</tr>
<tr>
<td>19.</td>
<td>0.61</td>
<td>0.21</td>
<td>0.312439</td>
</tr>
<tr>
<td>20.</td>
<td>0.62</td>
<td>0.19</td>
<td>0.290864</td>
</tr>
<tr>
<td>21.</td>
<td>0.65</td>
<td>0.18</td>
<td>0.281928</td>
</tr>
<tr>
<td>22.</td>
<td>0.66</td>
<td>0.14</td>
<td>0.231</td>
</tr>
<tr>
<td>23.</td>
<td>0.61</td>
<td>0.11</td>
<td>0.186389</td>
</tr>
<tr>
<td>24.</td>
<td>0.64</td>
<td>0.11</td>
<td>0.187733</td>
</tr>
<tr>
<td>25.</td>
<td>0.61</td>
<td>0.1</td>
<td>0.171831</td>
</tr>
<tr>
<td>26.</td>
<td>0.59</td>
<td>0.06</td>
<td>0.108923</td>
</tr>
<tr>
<td>27.</td>
<td>0.58</td>
<td>0.04</td>
<td>0.074839</td>
</tr>
<tr>
<td>28.</td>
<td>0.58</td>
<td>0.03</td>
<td>0.057049</td>
</tr>
<tr>
<td>29.</td>
<td>0.58</td>
<td>0.01</td>
<td>0.019661</td>
</tr>
<tr>
<td>30.</td>
<td>0.6</td>
<td>0.03</td>
<td>0.057143</td>
</tr>
</tbody>
</table>
Table 4.6. N-gram = 6 for 30 iterations using 392 documents from the PAN-09 Corpora

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.32</td>
<td>0.59</td>
<td>0.414945</td>
</tr>
<tr>
<td>2.</td>
<td>0.4</td>
<td>0.59</td>
<td>0.476768</td>
</tr>
<tr>
<td>3.</td>
<td>0.41</td>
<td>0.58</td>
<td>0.480404</td>
</tr>
<tr>
<td>4.</td>
<td>0.51</td>
<td>0.57</td>
<td>0.538333</td>
</tr>
<tr>
<td>5.</td>
<td>0.58</td>
<td>0.56</td>
<td>0.569825</td>
</tr>
<tr>
<td>6.</td>
<td>0.59</td>
<td>0.55</td>
<td>0.569298</td>
</tr>
<tr>
<td>7.</td>
<td>0.59</td>
<td>0.53</td>
<td>0.558393</td>
</tr>
<tr>
<td>8.</td>
<td>0.6</td>
<td>0.51</td>
<td>0.551351</td>
</tr>
<tr>
<td>9.</td>
<td>0.61</td>
<td>0.49</td>
<td>0.543455</td>
</tr>
<tr>
<td>10.</td>
<td>0.63</td>
<td>0.46</td>
<td>0.531743</td>
</tr>
<tr>
<td>11.</td>
<td>0.64</td>
<td>0.45</td>
<td>0.52844</td>
</tr>
<tr>
<td>12.</td>
<td>0.7</td>
<td>0.43</td>
<td>0.532743</td>
</tr>
<tr>
<td>13.</td>
<td>0.71</td>
<td>0.41</td>
<td>0.519821</td>
</tr>
<tr>
<td>14.</td>
<td>0.72</td>
<td>0.39</td>
<td>0.505946</td>
</tr>
<tr>
<td>15.</td>
<td>0.73</td>
<td>0.39</td>
<td>0.508393</td>
</tr>
<tr>
<td>16.</td>
<td>0.74</td>
<td>0.34</td>
<td>0.465926</td>
</tr>
<tr>
<td>17.</td>
<td>0.75</td>
<td>0.42</td>
<td>0.538462</td>
</tr>
<tr>
<td>18.</td>
<td>0.75</td>
<td>0.39</td>
<td>0.513158</td>
</tr>
<tr>
<td>19.</td>
<td>0.71</td>
<td>0.34</td>
<td>0.45981</td>
</tr>
<tr>
<td>20.</td>
<td>0.71</td>
<td>0.32</td>
<td>0.441165</td>
</tr>
<tr>
<td>21.</td>
<td>0.71</td>
<td>0.31</td>
<td>0.431569</td>
</tr>
<tr>
<td>22.</td>
<td>0.71</td>
<td>0.27</td>
<td>0.391224</td>
</tr>
<tr>
<td>23.</td>
<td>0.71</td>
<td>0.24</td>
<td>0.358737</td>
</tr>
<tr>
<td>24.</td>
<td>0.71</td>
<td>0.24</td>
<td>0.358737</td>
</tr>
<tr>
<td>25.</td>
<td>0.72</td>
<td>0.23</td>
<td>0.348632</td>
</tr>
<tr>
<td>26.</td>
<td>0.7</td>
<td>0.19</td>
<td>0.298876</td>
</tr>
<tr>
<td>27.</td>
<td>0.7</td>
<td>0.17</td>
<td>0.273563</td>
</tr>
<tr>
<td>28.</td>
<td>0.69</td>
<td>0.16</td>
<td>0.259765</td>
</tr>
<tr>
<td>29.</td>
<td>0.71</td>
<td>0.14</td>
<td>0.233882</td>
</tr>
<tr>
<td>30.</td>
<td>0.71</td>
<td>0.16</td>
<td>0.261149</td>
</tr>
</tbody>
</table>
Table 4.7. N-gram = 7 for 30 iterations using 392 documents from the PAN-09 Corpora

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.27</td>
<td>0.72</td>
<td>0.392727</td>
</tr>
<tr>
<td>2.</td>
<td>0.36</td>
<td>0.72</td>
<td>0.48</td>
</tr>
<tr>
<td>3.</td>
<td>0.37</td>
<td>0.71</td>
<td>0.486481</td>
</tr>
<tr>
<td>4.</td>
<td>0.47</td>
<td>0.73</td>
<td>0.571833</td>
</tr>
<tr>
<td>5.</td>
<td>0.52</td>
<td>0.69</td>
<td>0.593058</td>
</tr>
<tr>
<td>6.</td>
<td>0.55</td>
<td>0.68</td>
<td>0.60813</td>
</tr>
<tr>
<td>7.</td>
<td>0.54</td>
<td>0.66</td>
<td>0.594</td>
</tr>
<tr>
<td>8.</td>
<td>0.56</td>
<td>0.64</td>
<td>0.597333</td>
</tr>
<tr>
<td>9.</td>
<td>0.63</td>
<td>0.63</td>
<td>0.63</td>
</tr>
<tr>
<td>10.</td>
<td>0.59</td>
<td>0.61</td>
<td>0.599833</td>
</tr>
<tr>
<td>11.</td>
<td>0.58</td>
<td>0.48</td>
<td>0.525283</td>
</tr>
<tr>
<td>12.</td>
<td>0.65</td>
<td>0.52</td>
<td>0.577778</td>
</tr>
<tr>
<td>13.</td>
<td>0.75</td>
<td>0.54</td>
<td>0.627907</td>
</tr>
<tr>
<td>14.</td>
<td>0.68</td>
<td>0.51</td>
<td>0.582857</td>
</tr>
<tr>
<td>15.</td>
<td>0.69</td>
<td>0.5</td>
<td>0.579832</td>
</tr>
<tr>
<td>16.</td>
<td>0.67</td>
<td>0.48</td>
<td>0.559304</td>
</tr>
<tr>
<td>17.</td>
<td>0.71</td>
<td>0.55</td>
<td>0.619841</td>
</tr>
<tr>
<td>18.</td>
<td>0.7</td>
<td>0.52</td>
<td>0.596721</td>
</tr>
<tr>
<td>19.</td>
<td>0.78</td>
<td>0.51</td>
<td>0.616744</td>
</tr>
<tr>
<td>20.</td>
<td>0.69</td>
<td>0.39</td>
<td>0.498333</td>
</tr>
<tr>
<td>21.</td>
<td>0.74</td>
<td>0.44</td>
<td>0.551864</td>
</tr>
<tr>
<td>22.</td>
<td>0.72</td>
<td>0.4</td>
<td>0.514286</td>
</tr>
<tr>
<td>23.</td>
<td>0.69</td>
<td>0.39</td>
<td>0.498333</td>
</tr>
<tr>
<td>24.</td>
<td>0.74</td>
<td>0.41</td>
<td>0.527652</td>
</tr>
<tr>
<td>25.</td>
<td>0.68</td>
<td>0.38</td>
<td>0.487547</td>
</tr>
<tr>
<td>26.</td>
<td>0.66</td>
<td>0.32</td>
<td>0.43102</td>
</tr>
<tr>
<td>27.</td>
<td>0.6</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>28.</td>
<td>0.65</td>
<td>0.29</td>
<td>0.401064</td>
</tr>
<tr>
<td>29.</td>
<td>0.7</td>
<td>0.28</td>
<td>0.4</td>
</tr>
<tr>
<td>30.</td>
<td>0.67</td>
<td>0.31</td>
<td>0.423878</td>
</tr>
</tbody>
</table>

4.5.1 Result Discussion
The best performance is seen at an n-gram value of 2 in table 4.3, with precision values of 0.95 and 0.93 and recall values of 0.01 and 0.21 respectively. No Other n-gram value gives us this value. Table 4.5 gives the summary of the highest values of the precision and recall from the 7 tests that were performed on the algorithm.

Table 4.8 Summary of Highest and Lowest Values of Precision and Recall of Result

<table>
<thead>
<tr>
<th>N-Gram</th>
<th>Highest Precision</th>
<th>Highest Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.75</td>
<td>0.35</td>
<td>0.5710</td>
</tr>
<tr>
<td>2</td>
<td>0.95</td>
<td>0.01</td>
<td>0.0190</td>
</tr>
<tr>
<td>2</td>
<td>0.93</td>
<td>0.21</td>
<td>0.3420</td>
</tr>
<tr>
<td>3</td>
<td>0.87</td>
<td>0.37</td>
<td>0.5152</td>
</tr>
<tr>
<td>4</td>
<td>0.81</td>
<td>0.54</td>
<td>0.6480</td>
</tr>
<tr>
<td>5</td>
<td>0.66</td>
<td>0.64</td>
<td>0.6200</td>
</tr>
<tr>
<td>6</td>
<td>0.75</td>
<td>0.42</td>
<td>0.5384</td>
</tr>
<tr>
<td>6</td>
<td>0.75</td>
<td>0.39</td>
<td>0.5131</td>
</tr>
<tr>
<td>7</td>
<td>0.78</td>
<td>0.51</td>
<td>0.6179</td>
</tr>
</tbody>
</table>

At an n-gram of n = 7, we had a precision value of 0.78 and a recall value of 0.51 which is higher than the results achieved at PAN-09 conference. Performance indicators suggest that plagiarism detection can be performed using string matching with n-gram values within {2-6}.

Comparing the results with the result of the PAN-09 competition, using the same corpora, in this task, we were able to achieve a higher precision value of 0.95 using the String Matching and N-gram algorithm. Grozea, Gehl and popescu (2009), achieved a precision value of 0.5193 and a recall value of 0.4610. The precision results achieved in this work was 0.93 with a recall of 0.21 and a granularity of 1.04.
Table 4.9: Best three values presented by participants of PAN-09

<table>
<thead>
<tr>
<th>Rank</th>
<th>Overall</th>
<th>F</th>
<th>Precision</th>
<th>Recall</th>
<th>Granularity</th>
<th>Participant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.4871</td>
<td>0.4884</td>
<td>0.5193</td>
<td>0.4610</td>
<td>1.0038</td>
<td>Grozea, Gehl and popescu (2009)</td>
</tr>
<tr>
<td>2.</td>
<td>0.4265</td>
<td>0.4335</td>
<td>0.3901</td>
<td>0.4877</td>
<td>1.0228</td>
<td>Kasprzak, Brandejs and Kripac (2009)</td>
</tr>
<tr>
<td>3.</td>
<td>0.4229</td>
<td>0.4544</td>
<td>0.4709</td>
<td>0.4390</td>
<td>1.1060</td>
<td>Basile et al (2009)</td>
</tr>
<tr>
<td>4.</td>
<td>0.4200</td>
<td>0.3420</td>
<td>0.93</td>
<td>0.21</td>
<td>1.04166667</td>
<td>Chimebuka Egonu (2016)</td>
</tr>
</tbody>
</table>

At n=2, we were able to achieve a high precision score of 0.93, but with a recall score of 0.21, it suggests that many documents were left out in the checking and classification. The recall score is the lowest among the three best participants. This suggests to us that the fraction of relevant instances that are retrieved is small, while the precision values at 0.93, suggests that the fraction of retrieved instances that are relevant are more.

So we can conclude that the algorithm performs well in retrieving documents that have exact similarity with the one in question, with a high precision score.

**Research Question 1:** Is it possible to split a document into words and sentences and identify plagiarism from the split words?

Yes. We were able to split the various suspicious documents into word-n-grams of 1, 2, 3, 4, 5, 6 and 7 respectively and tested them against the source corpora for plagiarism detection.

**Research Question 2:** Are there available string matching algorithms that can be implemented and applied Research Question 1, to detect and retrieve information quickly?
An Exact string matching algorithm was used with the n-gram split words. The algorithms used mimics brute force algorithm.

**Research Question 3:** Can we cluster the passages where the plagiarized text was copied from into groups of relevant sentences, thereby making it possible to easily reference the texts copied from.

In this work, we were able to tag the passages that were plagiarized to enable us easily detect the passages that were plagiarized.

**Research Question 4:** What is the best number of words that will allow the algorithm perform at maximum speed and have the highest precision?

For the Corpus used, PAN-09 corpus, we discovered that the highest precision value using the n-gram and string matching algorithm was achieved at n=2, similar values were also gotten for n=3. This was very useful for detecting plagiarism in shake and paste documents and also in “copy and paste” plagiarism cases, but useless in multilingual and idea plagiarism cases or for heavily obfuscated plagiarism cases.

**Research Question 5:** What is the performance index for measuring the success rate of data retrieval/matching for very large documents?

The performance indices used in determining the suitability of the algorithm was Precision, recall, granularity and F-Measure, which is a harmonic mean between the precision and recall values. Since the harmonic mean of a list of numbers tends
strongly toward the least elements of the list, it tends (compared to the arithmetic mean) to mitigate the impact of large outliers and aggravate the impact of small ones.

**Benefits of the New System**

1. The system presented in this work will be useful in checking plagiarism in student project works and other academic writings done in the university.

2. A user friendly system made for analyzing data, thus showing the sections that were plagiarized and tagging them with numbers to show where the information was retrieved from.

3. The system affords academic institutions the opportunity of owning a customized plagiarism detector, which can combine various detection algorithms.

4. The system also integrates an online search engine for searching global data pool in the even no match was seen in the Local database.

5. It allows the students archive their work as original work against the school database, hence preventing plagiarism.

6. The string matching algorithm used here can be introduced in various other matching tasks, such as intrusion detection matching, matching for clustering of data, matching in Biogenetics and Information Retrieval tasks.
Fig 4.11 Precision values for n=1, 2, 3, 4, 5, 6 and 7
### Comparison of Recall Values for n=1,2,3,4,5,6,7

<table>
<thead>
<tr>
<th>n</th>
<th>Recall at n=1</th>
<th>Recall at n=2</th>
<th>Recall at n=3</th>
<th>Recall at n=4</th>
<th>Recall at n=5</th>
<th>Recall at n=6</th>
<th>Recall at n=7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.65</td>
<td>0.8</td>
<td>0.65</td>
<td>0.75</td>
<td>0.46</td>
<td>0.59</td>
<td>0.72</td>
</tr>
<tr>
<td>2</td>
<td>0.65</td>
<td>0.9</td>
<td>0.8</td>
<td>0.75</td>
<td>0.46</td>
<td>0.59</td>
<td>0.72</td>
</tr>
<tr>
<td>3</td>
<td>0.64</td>
<td>0.8</td>
<td>0.8</td>
<td>0.74</td>
<td>0.46</td>
<td>0.59</td>
<td>0.72</td>
</tr>
<tr>
<td>4</td>
<td>0.63</td>
<td>0.8</td>
<td>0.76</td>
<td>0.74</td>
<td>0.46</td>
<td>0.59</td>
<td>0.72</td>
</tr>
<tr>
<td>5</td>
<td>0.62</td>
<td>0.76</td>
<td>0.74</td>
<td>0.7</td>
<td>0.46</td>
<td>0.59</td>
<td>0.72</td>
</tr>
<tr>
<td>6</td>
<td>0.61</td>
<td>0.74</td>
<td>0.7</td>
<td>0.7</td>
<td>0.46</td>
<td>0.59</td>
<td>0.72</td>
</tr>
<tr>
<td>7</td>
<td>0.6</td>
<td>0.7</td>
<td>0.69</td>
<td>0.7</td>
<td>0.46</td>
<td>0.59</td>
<td>0.72</td>
</tr>
</tbody>
</table>

**Fig 4.12** Recall values for n=1, 2, 3,4,5,6 and 7
CHAPTER 5

Summary, Conclusions and Recommendation

5.1 Summary of Findings

In this project work several literature works were reviewed in order to identify the performance of various algorithms employed in plagiarism detection. The various approaches employed by the participants of the PAN-09 competition were reviewed and their results compared with the result of this work. Plagiarism detection is a very resource-intensive task in terms of computing power, so the best algorithms for detecting and retrieving the source documents must be implemented for the retrieval process.

At n=2, we were able to achieve a high precision score of 0.93, but with a recall score of 0.21, it suggests that many documents were left out in the checking and classification. The recall score is the lowest among the three best participants. This suggests to us that the fraction of relevant instances that are retrieved is small, while the precision values at 0.93, suggests that the fraction of retrieved instances that are relevant are more.

So we can conclude that the algorithm performs well in retrieving documents that have exact similarity with the one in question, with a high precision score.
Some levels of obfuscations in the text corporal went undetected. This is a loophole in the algorithm and can be improved by addition of part-of-speech tagging using a wordnet database.

The speed of information retrieval of the proposed system is better than existing models and various other algorithms could be built into the system for improved performance.

Test against multi-lingual plagiarism proved inefficient as string matches done in multi-lingual scenarios could not yield any result.

5.2 Conclusions

Based on results of this study, we draw the following conclusions:

1. Improvements in the plagiarism detection process can be achieved using a combined string matching and word n-gram for information retrieval.
2. The retrieval model for the documents can be applied to other information retrieval problems for better precision in information retrieval.
3. To achieve an improved performance in the plagiarism detection process, using n-gram and string matching, the value of n has to be set at n= 2 for optimum performance.
4. The combination of string matching and n-grams cannot be applied to heavily disguised plagiarism detection cases, where heavy text obfuscation has occurred. To do this, other semantic and syntax features must be considered.
5. The speed of the information retrieval and detection process depends on the speed of the hardware components that host the application

5.3 Recommendations

Based on the summary of findings, the following recommendations were made:

1. As the Database grows, it may be necessary to have a document/NOSQL database structure which can handle large chunk of data / Big Data.

2. Syntax and semantic POS tagging algorithm could be added after the n-grams string matching for cases of idea plagiarism or plagiarized texts with heavy obfuscation

3. Multilingual translators can be integrated into the application for cross-lingual plagiarism detection.

4. Further splitting the words and using character n-grams could yield better results if researched and tested.
REFERENCES


Consequences of Plagiarism http://www.ithenticate.com/resources/6-consequences-of-plagiarism retrieved on February, 12, 2016


Educational Development Center Technical Article on Plagiarism and how to avoid it accessed from https://edc.carleton.ca/files/Repo/file_240/Plagiarism.pdf (Last accessed February 8, 2016)


Gipp, B., and Meuschke, N. (2011) Citation Pattern Matching Algorithms for Citation-based Plagiarism Detection: Greedy Citation Tiling, Citation Chunking and Longest Common Citation Sequence. In Proceedings of the 11th ACM symposium on Document engineering (DocEng '11), Mountainview, CA, USA, ACM. Doi: 10.1145/2034691.2034741.

Gipp B. (2014), Citation-based Plagiarism Detection - Detecting Disguised and Cross-language Plagiarism using Citation Pattern Analysis. Springer Vieweg Research, 2014.


Halstead, Maurice Howard, (1977) elements of software science, Elsevier publishing.


Middle Tennessee State University, Graduate School brochure on Plagiarism http://www.mtsu.edu/graduate/pdf/Plagiarism.pdf (Accessed 8, February 2016.)


Sanusi Lamido Sanusi: Global financial meltdown and the reforms in the Nigerian banking sector (Speech by Mr Sanusi Lamido Sanusi, Governor of the Central Bank of Nigeria, at a Public Lecture delivered at the Convocation Square, Abubakar Tafawa Balewa University, Bauchi, 10 December 2010) http://www.bis.org/review/r110124c.pdf (retrieved on February,12,2016).


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using BrainboxxClassLibrary;
using System.Data;
using System.Net.Mail;
using FreeTextBoxControls;
using System.IO;

public partial class Complaints : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            try
            {
                string UserCategory = Session["UserCategory"].ToString();
                string UserName = Session["UserName"].ToString();
                string StudentsID = Session["StudentsID"].ToString();
                string UserID = Session["UserID"].ToString();
                string SchoolID = Session["SchoolID"].ToString();
                //ADD TERM AND SESSION
                string SchoolSession = Session["Session"].ToString();
                string Term = Session["Term"].ToString();
                //string subject = Session["Subject"].ToString();
                Label Username = (Label)Page.Master.FindControl("UsernameLabel1");
                Username.Text = "Welcome " + Session["UserName"].ToString().ToLowerInvariant();
            }
            catch
            {
                Response.Redirect("~/LoginOneTime.aspx");
            }
            GetStudentRecords();
        }
        foreach(var text in complainerTextBox.Text.ToList())
        {
            MessageBox.Show(text.ToString());
        }
        /// textbox.Select(0, textbox.Text.Length)
String[] counts = complainerTextBox.Text.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);
List<string> words = new List<string>();

foreach (var word in counts)
{
    //if()
    //{

    //}
    Label14.Text = word.ToString();
}

int count = complainerTextBox.Text.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).Count();
Label13.Text = count.ToString() + " Words";

int sentences = complainerTextBox.Text.Split(new char[] { '.' }, StringSplitOptions.RemoveEmptyEntries).Count();
Label12.Text = sentences.ToString() + " Sentences";

///complainerTextBox.Text.Select()

try{
    string SchoolID = Session["SchoolID"].ToString();
}catch{
    Response.Redirect("~/LoginOneTime.aspx");
}

private void GetStudentRecords()
{
    BrainboxxClass dal = new BrainboxxClass();
    DataSet ds = new DataSet();
    string msg = "";
    ds = dal.LND_FetchStudentRecord(Session["StudentsID"].ToString(), Session["SchoolID"].ToString(), ref msg);
}
if (ds.Tables[0].Rows.Count == 0)
{
    MessageBox.Show("Your Records were not Found");
    return;
}
if (msg != "")
{
    MessageBox.Show("Unable to Complete because " + msg);
}

string EmailAddress = ds.Tables[0].Rows[0]["E_MAIL"].ToString();
string NameofChild = ds.Tables[0].Rows[0]["STUDENT_NAME"].ToString() + " " +
ds.Tables[0].Rows[0]["STUDENT_SURNAME"].ToString();
Session["EmailAddress"] = EmailAddress;
Session["NameofChild"] = NameofChild;
complainerTextBox.Text = NameofChild;

private void sendEmail(string Recipients, string Message, string MessageTitle, string ParentFromAddress, string ChildName)
{
    DataSet StaffAndStudent = new DataSet();
    BrainboxxClass dal = new BrainboxxClass();
    string SchoolCode = Session["SchoolCode"].ToString();
    DataSet server = new DataSet();
    string msg = "";
    DataSet ds = new DataSet();
    server = dal.FetchSchoolEmailDetails(SchoolCode, ref msg);
    MailMessage mail = new MailMessage();
    string SMTPMailServer = server.Tables[0].Rows[0]["EMAIL_SERVER"].ToString();
    string FromAddress = server.Tables[0].Rows[0]["EMAIL_FROM"].ToString();
    string EmailAddress = server.Tables[0].Rows[0]["EMAIL_ADDRESS"].ToString();
    string PasswordEmailAddress = server.Tables[0].Rows[0]["EMAIL_PASSWORD"].ToString();

    //use the From Name as the School Name
    mail.From = new MailAddress(ParentFromAddress, "Complaint from " + ChildName);
    mail.Subject = MessageTitle;
    mail.Bcc.Add(Recipients);
    mail.Body = Message;
    SmtpClient MailSMTPserver = new SmtpClient(SMTPMailServer);
    MailSMTPserver.Credentials = new System.Net.NetworkCredential(EmailAddress, PasswordEmailAddress);
    MailSMTPserver.Send(mail);
    FreeTextBoxControls.SubScript super = new FreeTextBoxControls.SubScript();
}
protected void ScanButton1_Click(object sender, EventArgs e)
{
    //complainerTextBox0.Text = Plagiarised( Bold(complainerTextBox.Text));
    BrainboxxClass dal = new BrainboxxClass();
    string msg = "";
    DataSet ds = new DataSet();

    string AuthorName = "",
    string DocumentTitle = "",
    string DocumentName = "",
    int NoOfWords = 0;
    int NoOfParagraphs = 0;
    int NoOfSentences = 0;
    string TextToArchive = "";

    //TextBox txtData; // assume this is your text box control
    //FileUpload fileUpload; //your fileupload control
    //StreamReader r = new StreamReader(FileUpload1.FileContent);
    //complainerTextBox.Text = r.ReadToEnd();

    var _sentences = complainerTextBox.Text.Split(new char[] { '.' }, StringSplitOptions.RemoveEmptyEntries).ToList();

    if (SavetoDBCheckBox1.Checked)
    {   //save the contents of COMplaintsTextbox to the Database
        AuthorName = "Chimebuka Josiah Egonu";

        if (FileUpload1.HasFile)
        {
            DocumentTitle = FileUpload1.FileName.ToString();

            DocumentName = FileUpload1.FileName.ToString() + DateTime.Now;
        }
        else
        {
            DocumentTitle = AuthorName + " " + DateTime.Now;
            DocumentName = AuthorName + "_" + DateTime.Now;
        }

        int _NoofWords = complainerTextBox.Text.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).Count();
        int _NoOfSentences = complainerTextBox.Text.Split(new char[] { '.' }, StringSplitOptions.RemoveEmptyEntries).Count();
        int _NoOfParagraph = complainerTextBox.Text.Split(new char[] { '
' }, StringSplitOptions.RemoveEmptyEntries).Count();


NoOfWords = _NoofWords;
NoOfSentences = _NoOfSentences;
NoOfParagraphs = _NoOfParagraph;
TextToArchive = complainerTextBox.Text;

dal.PlagiarismTextArchival(AuthorName, DocumentTitle, DocumentName, NoOfWords,
NoOfSentences, NoOfParagraphs, DateTime.Now, TextToArchive, ref msg);

if(msg != "")
{
    MessageBox.Show("Your work could not be archived because " + msg);
    return;
}

if(SearchLocalDBCheckBox2.Checked)
{
    //Perform a Check on the Local Database of all the papers and Journals that have been
Indexed in the database

    ListBox1.Items.Clear();

    string ngram = RadioButtonList1.SelectedItem.Value;

    GetN_gramWord(Convert.ToInt32(ngram));

    //List<string> RebuiltWords = new List<string>();
    //foreach (var text in _sentences)
    //{ //check if the database contains any thing like this text
    //    ds = dal.PlagiarismTextArchivalCheck(text, ref msg);
    //    if (msg != "")
    //    {
    //        MessageBox.Show("We Cannot Check the LocalDB now because " + msg);
    //        return;
    //    }
    //    if (ds.Tables[0].Rows.Count > 0)
    //    {
    //        //add the selected document Name into the array of Texts for the FOund Cases, and
colour that particular text segment.
    //        //Iterate through the DataSet and Put the Data into a listview
    //        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
    //        {
    //            //Add the Data to the Listview
    //            //... (Code continues here)
    //        }
    //    }
}
foreach item in the dataset, put the item in a ListBox with the attendant parameters needed

get the Name of the Document

get the string where the Document found

ListBox1.Items.Add(" MATCH FOR " + text + " WAS FOUND IN " + ds.Tables[0].Rows[i]["DOCUMENT_NAME"].ToString());

RebuiltWords.Add(" " + Plagiarised (text));

else

RebuiltWords.Add(" " + text);

//find the first 5 string of words before a citation

//find the next five string of words before a citation

dal.plagiarismLocalCheck();

//for a new Line
///string[] lines = complainerTextBox0.Text.Split(new string[] { Environment.NewLine }, StringSplitOptions.None);

//for a word
String[] lines = complainerTextBox.Text.Split(new string[] { "\\n", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "$", \"\"");

//List<String> BuiltUpWords = new List<string>();
//BuiltUpWords.Add(StopWords.ToList().ForEach(m=>m.l));

// String char[] = complainerTextBox.Text.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);
//foreach (String word in counts)
//{
//  if (word.Contains("of"))
//  {
//    //return a coloured Stop
//    complainerTextBox0.Text += Italics(" " + word);
//    //    Literal1.Text += (leagueTeamRow[i].teamName);
//    //BuiltUpWords.Add(Italics(word));
//  }
//  else
//  {
//    complainerTextBox0.Text += Italics(" " + word);
//  }
//}

///string startingString = this.complainerTextBox.Text;
///List<string> l = startingString.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).ToList();
///var combinations = l.Select(q => FindPairs(q, l))
///.ToList()
///.SelectMany(r => r);
///foreach (var combination in combinations)
///{
///    BuiltUpWords.Add(String.Join(" ", combination));
///    Console.WriteLine();
///}

//complainerTextBox0.Text = BuiltUpWords.ToString();

///
///for a Sentence
///string[] lines = ;

//foreach (String word in complainerTextBox.Text.Split(new string[] { "\n" }, StringSplitOptions.None))
//{
//  foreach(char Character in word)
//  {
//    MessageBox.Show(Character.ToString());
private void GetN_gramWord(int n_gram)
{
    int Count = 0;

    BrainboxxClass dal = new BrainboxxClass();
    string msg = "";
    DataSet ds = new DataSet();

    string[] counts = complainerTextBox.Text.Split(new char[] {' '}, StringSplitOptions.RemoveEmptyEntries);
    List<string> NgramWords = new List<string>();
    complainerTextBox0.Text = "";
    NgramWordsTextBox3.Text = "";
    int Annotation = 0;
    foreach (string word in counts)
    {
        Count++;
        Annotation++;

        //add the word to the Hidden text box
        NgramWordsTextBox3.Text += " " + word;

        string[] _counts = NgramWordsTextBox3.Text.Split(new char[] {' '}, StringSplitOptions.RemoveEmptyEntries);

        if (_counts.Count() == n_gram)
        {
            //get the words and use it to perform the Plagiarism Check in the Database.

            ds = dal.PlagiarismTextArchivalCheck(NgramWordsTextBox3.Text, ref msg);

            if (msg != "")
            {

            }
        }
    }
}
{ MessageBox.Show("We Cannot Check the LocalDB now because "+msg); return; }

if (ds.Tables[0].Rows.Count > 0) {
    // add the selected document Name into the array of Texts for the Found Cases, and colour that particular text segment.
    // Iterate through the DataSet and Put the Data into a listview
    //
    //
    for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
    {
        // foreach item in the dataset, put the item in a ListBox with the attendant parameters needed
        // get the Name of the Document
        // get the string where the Document found
        ListBox1.Items.Add( "\" + Annotation +\"\" + Match Found:" + NgramWordsTextBox3.Text + "WAS FOUND IN " + ds.Tables[0].Rows[i]["DOCUMENT_NAME"].ToString());

        complainerTextBox0.Text += ("\" + Plagiarised(NgramWordsTextBox3.Text) + Italics(superscript(Annotation.ToString())));
    }
}

else
{
    complainerTextBox0.Text += "\" + NgramWordsTextBox3.Text;
}

NgramWordsTextBox3.Text = string.Empty;
// else
//{
//    complainerTextBox0.Text += "\" + NgramWordsTextBox3.Text;
//}

// NgramWords.Add();
// if (word.Contains("of"))
//{
//    // return a coloured Stop

// complainerTextBox0.Text += Italics("\" + word);
// Literal1.Text += (leagueTeamRow[i].teamName);
// BuiltUpWords.Add(Italics(word));
//}
private static List<List<string>> FindPairs(string s, List<string> list)
{
    List<List<string>> result = new List<List<string>>();
    int index = list.IndexOf(s);
    for (int t = 2; t < list.Count; t++)
    {
        if (index + t <= list.Count)
        {
            var words = list.Skip(index).Take(t).ToList();
            if (words.Count() >= 2)
            {
                result.Add(words);
            }
        }
    }
    return result;
}

public string Bold(string imputText)
{
    string Text = "";
    Text = "<b>" + imputText + "</b>";
    return Text;
}

public string Italics(string imputText)
{
    string Text = "";
    Text = "<i>" + imputText + "</i>";
    return Text;
}

public string superscript(string imputText)
{
    string Text = "";
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace TextClustering.Lib
{
    /// <summary>
    /// Represents each document in n-dimensional vector space using Vector Space Model
    /// </summary>
    public class DocumentVector
    {
        //Content represents the document(or any other object) to be clustered
        public string Content { get; set; }
        //represents the tf*idf of each document
        public float[] VectorSpace { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace TextClustering.Lib
{
    /// <summary>
    /// Prepares cluster center using K-Means clustering algorithm in n-dimensional vector space.
    /// Document similarity is measured using cosine similarity.
    /// </summary>
    public static class DocumentClustering
    {
        private static int globalCounter = 0;
        private static int counter = 0;
        /// <summary>
        /// Prepares the document cluster, Grouping of similar type of text document is done here.
        /// </summary>
        /// <param name="k">initial cluster center</param>
        /// <param name="documentCollection">document corpus</param>
        /// <returns></returns>
        public static List<Centroid> PrepareDocumentCluster(int k, List<DocumentVector> documentCollection, ref int _counter)
        {
            globalCounter = 0;
            // Prepares k initial centroid and assign one object randomly to each centroid.
            List<Centroid> centroidCollection = new List<Centroid>();
            Centroid c;

            /*
             * Avoid repetition of random number, if same no is generated more than once same document is added to the next cluster.
             * so avoid it using HashSet collection.
             */
            HashSet<int> uniqRand = new HashSet<int>();
            GenerateRandomNumber(ref uniqRand, k, documentCollection.Count);

            foreach (int pos in uniqRand)
            {
                c = new Centroid();
                c.GroupedDocument = new List<DocumentVector>();
                c.GroupedDocument.Add(documentCollection[pos]);
                centroidCollection.Add(c);
            }

            Boolean stoppingCriteria;
            List<Centroid> resultSet;
            List<Centroid> prevClusterCenter;

            InitializeClusterCentroid(out resultSet, centroidCollection.Count);

            do
            {
            }
        }
    }
}
prevClusterCenter = centroidCollection;

foreach (DocumentVector obj in documentCollection)
{
    int index = FindClosestClusterCenter(centroidCollection, obj);
    resultSet[index].GroupedDocument.Add(obj);
}
InitializeClusterCentroid(out centroidCollection, centroidCollection.Count());
centroidCollection = CalculateMeanPoints(resultSet);
stoppingCriteria = CheckStoppingCriteria(prevClusterCenter, centroidCollection);
if (!stoppingCriteria)
{
    //initialize the result set for next iteration
    InitializeClusterCentroid(out resultSet, centroidCollection.Count);
}
}

/// <summary>
/// Generates unique random numbers and also ensures the generated random number
/// lies with in a range of total no. of document
/// </summary>
/// <param name="uniqRand"></param>
/// <param name="k"></param>
/// <param name="docCount"></param>
private static void GenerateRandomNumber(ref HashSet<int> uniqRand, int k, int docCount)
{
    Random r = new Random();
    if (k > docCount)
    {
        do
        {
            int pos = r.Next(0, docCount);
            uniqRand.Add(pos);
        } while (uniqRand.Count != docCount);
    }
    else
    {
        do
        {
            int pos = r.Next(0, docCount);
            uniqRand.Add(pos);
        } while (uniqRand.Count != k);
    }
}
/// <summary>
/// Initialize the result cluster centroid for the next iteration, that holds the result to be returned 
/// </summary>
/// <param name="centroid"></param>
/// <param name="count"></param>
private static void InitializeClusterCentroid(out List<Centroid> centroid, int count)
{
    Centroid c;
    centroid = new List<Centroid>();
    for (int i = 0; i < count; i++)
    {
        c = new Centroid();
        c.GroupedDocument = new List<DocumentVector>();
        centroid.Add(c);
    }
}
/// <summary>
/// Check the stopping criteria for the iteration, if centroid do not move their position it meets 
/// the criteria 
/// or if the global counter exist its predefined limit(minimum iteration threshold) than iteration 
/// terminates 
/// </summary>
/// <param name="prevClusterCenter"></param>
/// <param name="newClusterCenter"></param>
/// <returns></returns>
private static Boolean CheckStoppingCriteria(List<Centroid> prevClusterCenter,
List<Centroid> newClusterCenter)
{
    globalCounter++; 
counter = globalCounter; 
    if (globalCounter > 11000)
    {
        return true;
    }
    else
    {
        Boolean stoppingCriteria;
        int[] changeIndex = new int[newClusterCenter.Count()]; //1 = centroid has moved 0 == 
        centroid do not moved its position
        int index = 0;
        do
        {
            int count = 0;
            if (newClusterCenter[index].GroupedDocument.Count == 0
            &&
            prevClusterCenter[index].GroupedDocument.Count == 0)
            {
                index++;
            }
            else if (newClusterCenter[index].GroupedDocument.Count != 0
            &&
            prevClusterCenter[index].GroupedDocument.Count != 0)
            {
            }
            else if (newClusterCenter[index].GroupedDocument.Count != 0
            &&
            prevClusterCenter[index].GroupedDocument.Count != 0)
for (int j = 0; j <
newClusterCenter[index].GroupedDocument[0].VectorSpace.Count(); j++)
{
    //
    if (newClusterCenter[index].GroupedDocument[0].VectorSpace[j] ==
prevClusterCenter[index].GroupedDocument[0].VectorSpace[j])
    {
        count++;  
    }
}

if (count == newClusterCenter[index].GroupedDocument[0].VectorSpace.Count())
{
    changeIndex[index] = 0;
}
else
{
    changeIndex[index] = 1;
}
index++;  
else
{
    index++;  
    continue;

}
}

while (index < newClusterCenter.Count());

// if index list contains 1 stopping criteria is set to false
if (changeIndex.Where(s => (s != 0)).Select(r => r).Any())
{
    stoppingCriteria = false;  
}
else
    stoppingCriteria = true;

return stoppingCriteria;

}

//returns index of closest cluster centroid
private static int FindClosestClusterCenter(List<Centroid> clusterCenter,DocumentVector obj)
{
    float[] similarityMeasure = new float[clusterCenter.Count()];

    for (int i = 0; i < clusterCenter.Count(); i++)
    {

similarityMeasure[i] = 
SimilarityMatrics.FindCosineSimilarity(clusterCenter[i].GroupedDocument[0].VectorSpace, 
obj.VectorSpace);

} 

int index = 0; 
float maxValue = similarityMeasure[0]; 
for (int i = 0; i < similarityMeasure.Count(); i++) 
{
    // if document is similar assign the document to the lowest index cluster center to avoid 
    the long loop
    if (similarityMeasure[i] > maxValue)
    {
        maxValue = similarityMeasure[i];
        index = i;
    }
} 
return index;

} 

// Reposition the centroid
private static List<Centroid> CalculateMeanPoints(List<Centroid> _clusterCenter)
{
    for (int i = 0; i < _clusterCenter.Count(); i++)
    {
        if (_clusterCenter[i].GroupedDocument.Count() > 0)
        {
            for (int j = 0; j < _clusterCenter[i].GroupedDocument[0].VectorSpace.Count(); j++)
            {
                float total = 0;

                foreach (DocumentVector vSpace in _clusterCenter[i].GroupedDocument)
                {
                    total += vSpace.VectorSpace[j];
                }

                // reassign new calculated mean on each cluster center, It indicates the reposition of 
                centroid
                _clusterCenter[i].GroupedDocument[0].VectorSpace[j] = total / 
                _clusterCenter[i].GroupedDocument.Count();
            }
        }
    }
return _clusterCenter;
/// Find Residual sum of squares it measures how well a cluster centroid represents the
/// member of their cluster
/// We can use the RSS value as stopping criteria of k-means algorithm when decreases in RSS
/// value falls below a
/// threshold t for small t we can terminate the algorithm.
///
/// <summary>
private static void FindRSS(List<Centroid> newCentroid, List<Centroid> _clusterCenter)
{
    //TODO:
}

</summary>

DATABASE CONNECTIVITY USING WEB.CONFIG

<?xml version="1.0"?>
<configuration>
<configSections>
<section name="Infragistics.web" type="System.Configuration.SingleTagSectionHandler, System, Version=1.0.3300.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" />
</configSections>
<infragistics.web styleSetName="Default" styleSetPath="~/ig_res" />
<dataConfiguration defaultDatabase="BRAINBOX" />
<connectionStrings>
    <add name="BRAINBOX" connectionString="Data Source=MRSOFTEOGLOBALSERVER2014;Initial Catalog=BRAINBOX;Integrated Security=False;User ID=sa;Password=mrssoft" providerName="System.Data.SqlClient" />
    <add name="Excel03ConString" connectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source={0};Extended Properties='Excel 8.0;HDR={1}'" />
    <add name="Excel07ConString" connectionString="Provider=Microsoft.ACE.OLEDB.12.0;Data Source={0};Extended Properties='Excel 8.0;HDR={1}'" />
    <add name="BRAINBOXConnectionString" connectionString="Data Source=oak.arvixe.com;Initial Catalog=BRAINBOX;Persist Security Info=True;User ID=nexthost;Password=ebuka@123" providerName="System.Data.SqlClient" />
    <add name="BRAINBOXConnectionString1" connectionString="Data Source=EBUKA-PC\DATASERVER;Initial Catalog=BRAINBOX;Persist Security Info=True;User ID=SA;Password=ebu111" providerName="System.Data.SqlClient" />
</connectionStrings>
<appSettings>
    <add key="ChartImageHandler" value="storage=file;timeout=20;dir=c:\TempImageFiles;" />
    <add key="EmailServer" value="mail.gracelandintlschool.com" />
    <add key="fromAddress" value="admin@gracelandintlschool.com" />
    <add key="salesEmailAddress" value="info@gracelandintlschool.com" />
    <add key="FolderPath" value="_Documents/" />
</appSettings>
<system.web>
<machineKey
validationKey="EBF8D4721D0EDACC01DAD58383D5AD6D43B028F0652071278E379C535245862E9F
DEA8C3DC29D3C017A3C8BC07C5DF090994B21C309692E3E4335D958DA28"
decryptionKey="0DEA5C469961B4708BBB2B2005A1F6A3BC3A5216C955B1FB326B249AD3C
FB44" validation="SHA1" decryption="AES" />

<httpRuntime maxRequestLength="10245556" requestValidationMode="2.0"
executionTimeout="99900" />

<httpHandlers>
<add path="Reserved.ReportViewerWebControl.axd" verb="*"
Version=10.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" validate="false"/>
<add path="Telerik.ReportViewer.axd" verb="*"
Version=50.11.316, Culture=neutral, PublicKeyToken=a9d7983dfc261be" validate="true"/>
</httpHandlers>
</compilation debug="true" targetFramework="4.0">
<assemblies>
<add assembly="Infragistics4.WebUI.UltraWebGrid.v10.3, Version=10.3.20103.1013,
Culture=neutral, PublicKeyToken=7DD5C3163F2CD0CB"/>
<add assembly="Infragistics4.WebUI.Shared.v10.3, Version=10.3.20103.1013,
Culture=neutral, PublicKeyToken=7DD5C3163F2CD0CB"/>
<add assembly="System.Design, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/>
<add assembly="Microsoft.ReportViewer.WebForms, Version=10.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/>
<add assembly="Microsoft.ReportViewer.Common, Version=10.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/>
<add assembly="Microsoft.Build.Framework, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/>
<add assembly="System.Management, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/>
<add assembly="System.Web.Extensions.Design, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
<add assembly="System.Windows.Forms, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=561934E089"/>
<add assembly="Infragistics4.WebUI.WebDataInput.v10.3, Version=10.3.20103.1013,
Culture=neutral, PublicKeyToken=7DD5C3163F2CD0CB"/>
<add assembly="Infragistics4.WebUI.WebDateChooser.v10.3, Version=10.3.20103.1013,
Culture=neutral, PublicKeyToken=7DD5C3163F2CD0CB"/>
Culture=neutral, PublicKeyToken=7DD5C3163F2CD0CB"/>
<add assembly="Infragistics4.WebUI.Documents.v10.3, Version=10.3.20103.1013,
Culture=neutral, PublicKeyToken=7DD5C3163F2CD0CB"/>
<add assembly="Infragistics4.WebUI.Misc.v10.3, Version=10.3.20103.1013,
Culture=neutral, PublicKeyToken=7DD5C3163F2CD0CB"/>
</assemblies>
</compilation>
</system.web>
<buildProviders>
</compilation>
<authentication mode="Forms">
<forms loginUrl="~/Account/Login.aspx" timeout="36600" />
</authentication>
<membership>
<providers>
<clear />
<add name="AspNetSqlMembershipProvider" type="System.Web.Security.SqlMembershipProvider" connectionStringName="ApplicationServices" enablePasswordRetrieval="false" enablePasswordReset="true" requiresQuestionAndAnswer="false" requiresUniqueEmail="false" maxInvalidPasswordAttempts="5" minRequiredPasswordLength="6" minRequiredNonalphabet numericCharacters="0" passwordAttemptWindow="10" applicationName="/" />
</providers>
</membership>
<profile>
<providers>
<clear />
<add name="AspNetSqlProfileProvider" type="System.Web.Profile.SqlProfileProvider" connectionStringName="ApplicationServices" applicationName="/" />
</providers>
</profile>
<roleManager enabled="false">
<providers>
<clear />
<add name="AspNetSqlRoleProvider" type="System.Web.Security.SqlRoleProvider" connectionStringName="ApplicationServices" applicationName="/" />
</providers>
</roleManager>
<pages validateRequest="false" />
<customErrors mode="Off" />
</system.web>
</system.webServer>
<modules runAllManagedModulesForAllRequests="true" />
<validation validateIntegratedModeConfiguration="false" />
<directoryBrowse enabled="false" />
<rewrite>
<rules>
<rule name="RemoveASPX" enabled="true" stopProcessing="true">
<match url="([^\.]+\..*|[^\.]+)\..*" />
<action type="Redirect" url="{R:1}" />
</rule>
<rule name="AddASPX" enabled="true">
<match url="^[^\?]+\..*" negate="false" />
<conditions>
<add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
<add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true" />
</conditions>
<action type="Rewrite" url="{R:0}.aspx" />
</rule>
</rules>
</rewrite>
<handlers>
 FORM DESIGN HTML CODE
</configuration>
FUTOPlagScan ~ Plagiarism Detection Algorithm

No of Words: 0
No of Sentences: 0
PlagScore: 0%
Settings
Save to FUTOPlagDB
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithm Settings</strong></td>
<td></td>
</tr>
<tr>
<td>N-gram Citation Based Algorithm</td>
<td></td>
</tr>
<tr>
<td>N-Gram String Matching Algorithm</td>
<td></td>
</tr>
<tr>
<td>Syntax Recursion Algorithm</td>
<td></td>
</tr>
<tr>
<td>String Profiling Algorithm</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>N-Gram Settings</strong></td>
<td></td>
</tr>
<tr>
<td>2-Word-Gram</td>
<td></td>
</tr>
<tr>
<td>3-Word-Gram</td>
<td></td>
</tr>
<tr>
<td>4-Word-Gram</td>
<td></td>
</tr>
<tr>
<td>5-Word-Gram</td>
<td></td>
</tr>
<tr>
<td>6-Word-Gram</td>
<td></td>
</tr>
<tr>
<td>7-Word-Gram</td>
<td></td>
</tr>
<tr>
<td>Sentence-Gram</td>
<td></td>
</tr>
</tbody>
</table>

**Search Options**
- Search Local DB
- Search Online
- Highlight Plagiarised Sections
function SaveFUTODB()
{
    $('#Paper')
}

</script>