Text-Based Plagiarism Detection System

By

Hazliyana Binti Hussain

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons) in
Business Information System

December 2005

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

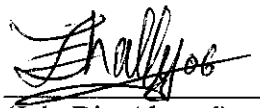
Text-Based Plagiarism Detection System

by

Hazliyana Binti Hussain

A project dissertation submitted to the
Business Information System Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for for the
BACHELOR OF TECHNOLOGY (Hons)
IN BUSINESS INFORMATION SYSTEM

Approved by,



(Jale Bin Ahmad)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

December 2005

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgments, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

HAZLIYANA BINTI HUSSAIN

ABSTRACT

Due to increasing of internet usage, students attempt to plagiarize the digital documents as their own work without acknowledging the sources as references. As this phenomenon becomes very common among students, a system that can detect plagiarism is most welcome to overcome the problem. The system is able to map out the words from the body of text files and then compare the strings between the text files. Besides, the system is also able to compare lines in the text files. The system is developed referring to the concept of Word Frequency Model which count the number words occurrence in the text files.

ACKNOWLEDGEMENT

First and foremost, I would like to express my gratitude to Allah S.W.T, because with His mercy and blessings had gave me the strengths to face challenges in completing this project for my Final Year Project.

I would like to express my profound appreciation, highest gratitude and sincere thanks to my supervisor, Mr. Jale Bin Ahmad for all the valuable guidance, positive and constructive criticism and advice that have been given to me while I was involved in the completion of this project.

I also would like to express my gratitude and thanks to all lecturers and tutors in IT and IS department who eventually helped me during the project and also in sharing their knowledge and information, which has made the project an unforgettable. Not to forget, special thank you to all my friends who helped and share their knowledge with me during the project development.

Lastly, I acknowledge with greatest appreciation to other personnel not mentioned above whom gave me such great support in completing this project successfully and to UTP for giving me a chance to gain knowledge and experiences during the final year project development. Last but not least, I sincerely apologize for all the problems involuntarily caused by myself. All of your kindness and cooperation are highly appreciated and will be fondly remembered.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF STUDY

Plagiarism refers to the use of another's ideas, information, language, or writing, when done without proper acknowledgment of the original source. Essential to an act of plagiarism is an element of dishonesty in attempting to pass off the plagiarized work as original. Plagiarism is not necessarily the same as copyright violation, which occurs when one violates copyright law. Like most terms from the area of intellectual property, *plagiarism is a concept of the modern age* and not really applicable to medieval or ancient works.

Through keyword-driven Internet research using search engines like Google and Yahoo, millions worldwide have easy, instant access to a vast and diverse amount of online information. Compared to encyclopedias and traditional libraries, the World Wide Web has enabled a sudden and extreme decentralization of information and data.

The existence and widespread use of the Internet has increased the occurrence of plagiarism. Students are able to use search engines to locate information on a wide range of topics. Once located, this information can be cut-and-pasted into new documents with minimal effort.

The students also tend to distribute the same information among themselves and have relatively same contents in the assignments without properly acknowledge the sources as reference. The size of the Internet makes it difficult for lecturers to trace the source of plagiarized material.

1.2 PROBLEM STATEMENT

Technological advances have made the plagiarism activities become common between the students in campuses and universities. As stated, plagiarism is using others' ideas and words without clearly acknowledging the source of that information as their own idea and works.

Although plagiarism is not a new issue, the recent use of the internet for information is increasingly making plagiarism more difficult for lecturers to recognize. In a 1999 survey of 2,100 students on 21 campuses across the country, about one-third of the participating students admitted to serious test cheating and half admitted to one or more instances of serious cheating on written assignments. On most campuses, over 75% of students admit to some cheating.

This shows that, most of the students produce non-original assignments and plagiarism activities day by days become overpowering. It happened because it is no longer possible for the lecturers to simply recognize the text from which the student may have copied or to detect that two students have a similar work. The lecturers must now only be able to detect work which may have been taken from any of potential web sites by noticing a change in the student's style of writing but it is still not effective to prevent plagiarism.

As the internet provides lots of free or paid information such as paper mills, journal and articles, the students attempted to copy the digital material as their own work. They are using the ideas of different persons as their own ideas to complete the assignments and projects. Students do likely not understand the content that they have copied. Thus, the qualities of their project or assignments do not meet the education quality standard. Besides, the lecturers tend to give wrong or non exact evaluation on their work.

There is also side impact caused by plagiarisms. Plagiarism may demoralize the honest student, successful plagiarism will encourages lifelong dishonesty, plagiarizers will undermine their own education and it will depress the faculty to encounter plagiarism.

As a student, they should have come out with their own ideas and solution to produce high quality of paper works. Thus, it is the academic responsibilities to handle the problem by preventing and detecting the plagiarism activities among the students.

A system which can be used to detect and prevent plagiarism is suggested to overcome the problem.

The project is significant to the lecturers in order to prevent and detect plagiarism among the students in campus especially in UTP. Besides, it will also help the organization, UTP to produce quality graduates and well independent student.

1.3 OBJECTIVE AND SCOPE

The objective of the project is to enable the lecturers to detect the similarities of students' assignments by using a Text-Based Plagiarism Detection System. The system is responsible to identify whether the submitted assignments have similar contents or not. By doing this, the lecturers can detect and prevent plagiarism among the students.

The scope of the study involves strings mining where the system is able to map out strings or text within the submitted digital assignments or project papers (softcopy). With the extracted strings, the system will compare the similarities between two or more text files and finally present the plagiarism status. The study is focusing on the UTP's current scenario where most of the students do plagiarism.

The relevancy of the project is obviously to give advantages to the lecturers and the faculty itself in order to detect or prevent plagiarisms. This can help faculty to achieve its objective to produce well-rounded graduates who are creative and innovative with the potential to become leaders of industry and the nation.

Operationally, the project is feasible because of the advantages that the faculty can achieve. The people in the organization especially lecturers, would think

the project will be very helpful in the future in order to prevent plagiarism. Besides, the availability of information, references, knowledge and skills will help the project to be technically feasible. Referring to the time frame or schedule feasibility, project scope and the budget, the project is practicable to be developed.

CHAPTER 2

LITERATURE REVIEW

The text feature extraction is a common issue in Information Retrieval, Text Mining, Web Mining [2], Text Classification/Clustering and Document Copy Detection. The most popular approach is word frequency based scheme [1].

There are several models that can be used for the text feature comparison or extraction. The models that available are Word Frequency Model (WFM) [1] and Semantic Sequence Model (SSM) [1]. Under Word Frequency Model (WFM) there are other two models which are Vector Space Model (VSM) [8] and Relative Frequency Model (RFM) [1].

According to Jun-Peng Rao, Jun-Yi Shen, Xiao-Dong Liu and Qin-Bao Song, the Word Frequency Model (WFM) is the most popular text feature extraction model which counts word appearance in documents and/or whole corpus to build the text feature vector and then measure the vectors similarity by dot product, cosine function or others like that to represent the similarity of documents. This model is fit to represent text similarity which is applied in text classification/clustering.

The Vector Space Model (VSM) [8] is a regular model to represent text documents. It is also used widely in Text Mining, Web Mining, Text Classification/Clustering and also Information Retrieval. The TFIDF algorithm is often combined with VSM.

The Relative Frequency Model (RFM) is presented by SCAM (Stanford Copy Analysis Method) so as to find out subset copies. SCAM was developed by Shivakumar and Garcia-Molina [4] to improve the previous copy detection system, COPS (Copy Protection System) [3].

The RFM [1] is the first asymmetric similarity measures in copy detection. The RFM is derived from VSM [1], where the both construct the text feature vector based on word frequency. The different between VSM and RFM is like mentioned before, RFM uses asymmetric similarity to measure in copy detection but VSM uses symmetric cosine function to do that.

Let $F(A)$ and $F(B)$ be document A and B word frequency vectors, then the similarity of A and B in VSM is $S_{VSM}(A,B)$:

$$S_{VSM}(A,B)=\frac{\sum_{i=1}^{n}\alpha_i^2 F_i(A)F_i(B)}{\sqrt{\sum_{i=1}^{n}\alpha_i^2 F_i^2(A)\times\sum_{i=1}^{n}\alpha_i^2 F_i^2(B)}}$$

where α_i is the word weight vector, $F_i(A)$, $F_i(B)$ are the respective number of occurrences of the i_{th} word in A and B. It is obvious that $S_{VSM}(A,B) = S_{VSM}(B,A)$. Because the similarity of A to B and that of B to A is the same, i.e. $S(A,B) = S(B,A)$, we call it symmetric similarity. For symmetric similarity, the copies' (same documents) value is 1 and the more overlapped words between documents the higher score. But they cannot distinguish the subset copies from partly overlapped documents. We know that A is included in B is different from B is included in A, i.e. $A \subset B \neq B \subset A$. So the inclusion measure of $A \subset B$ should be different from that of $B \subset A$. However the symmetric similarity does not satisfy that.

In RFM the subset measure of document A to be a subset of document B to be:

$$Subset(A,B)=\frac{\sum_{w_i \in c(A,B)}\alpha_i^2 F_i(A)F_i(B)}{\sum_{i=1}^{N}\alpha_i^2 F_i^2(A)}$$

It is obvious that $Subset(A,B) \neq Subset(B,A)$ and $Subset(A,A^c) = 1$ if A^c is a copy of A. Hence we call this type as asymmetric similarity measure. The RFM similarity measure between two documents A and B is:

$$S_{RFM}(A,B) = \max\{Subse(A,B), Subse(B,A)\}$$

The $Subset(A,B)$ may be greater than 1. In order to regularize the similarity value in [0,1], the final RFM similarity of documents A and B is:

$$S_{RFM}(A,B) = \min\{1, \max\{Subset(A,B), Subset(B,A)\}\}$$

Figure 1: the Vector Space Model (VSM) and Relative Frequency Model (RFM)

"RFM detects subset copy well because it can distinguish A⊂B from B⊂A by asymmetric metric. But it cannot find out ***n*** to 1 partial copy because lack of local detail information" [1].

The document copy detection (DCD) is to decide whether some part of the whole document is a copy of another [1] which we can call it plagiarism. DCD plays an important role in Intellectual Property Protection [9].

Plagiarism detection cares about the text identity more than the similarity. The very similar documents may not be identical, but plagiarizing documents must be very similar [1]. Even tough the DCD can detect plagiarism using the String Matching Scheme, it is still cannot resist the noise or modification. The action of rewording the sentences makes the plagiarism detection precision become weak or unused.

Due to the issue, Jun-Peng Rao, Jun-Yi Shen, Xiao-Dong Liu and Qin-Bao Song present a new text feature extraction model Semantic Sequence Model (SSM) [1].the SSM is based on the concepts of word distance, word density, and semantic sequence. Compare to RFM, SSM contains both global and local features so as to detect ***n*** to **1** partial copy well while RFM lack of local detail information.

Definition 1 Let S be a sequence of words, i.e. $S=w_1w_2...w_n$. We denote the position i in S by i_S, the word at i denoted by $w(i_S)$. The word distance of position i_S ($1\le i\le n$), denoted by $d(i_S)$, is the number of words between $w(h_S)$ and $w(i_S)$, i.e. $d(i_S)= i_S - h_S$, where $w(h_S)=w(i_S)$ and $w(k_S)\neq w(i_S)$ ($1\le h<k<i\le n$). If no $w(h_S)$ exists, i.e. $w(i_S)$ first occurs, then $d(i_S)=\infty$.

Definition 2 Let S be a sequence of words, i.e. $S=w_1w_2...w_n$. The word density of position i_S ($1\le i\le n$), denoted by $\rho(i_S)$, is the reciprocal of $d(i_S)$: $\rho(i_S) = 1/d(i_S)$.

In fact, $d(i_S)$ is the distance of $w(i_S)$ to its last appearance in the sequence S, and $\rho(i_S)$ denotes its local frequency. A document is a long sequence of words so that in a given range the small distance means the high density of word in a local section. That is to say the more small distance the higher density of some words in the local section. We believe that the high-density words in some section indicate the semantic of it.

Definition 3 Let S be a sequence of words, i.e. $S=w_1w_2...w_n$. A semantic sequence of S is a part of continued words $L(S)=w_iw_j...w_k$ ($1<i<j<k\le n$) in S and satisfies the following conditions:

(1) $|L(S)|>1$

(2) $$\forall w(l_{L(S)}) \rightarrow \exists w(x_S)(w(l_{L(S)}) = w(x_S))$$

(3) $$(w(l_{L(S)}) = w(x_S)) \wedge (w(m_{L(S)}) = w(y_S)) \rightarrow (l_{L(S)} < m_{L(S)} \leftrightarrow x_S < y_S)$$

(4) $$(w(l_{L(S)}) = w(x_S)) \wedge (w(m_{L(S)}) = w(y_S)) \wedge (m_{L(S)} = l_{L(S)} + 1) \rightarrow (0 < y_S - x_S \le \varepsilon)$$

(5) $\rho(l_S) \ge \delta, i_S \le l_S \le k_S$

(6) $(h_S = i_S - 1) \rightarrow (\rho(h_S) < \delta)$

(7) $(v_S = k_S + 1) \rightarrow (\rho(v_S) < \delta)$

where δ and ε are user defined parameters (e.g. $\delta=1/20$, $\varepsilon=4$).

Figure 2: Semantic Sequence Model (SSM)

In fact, a semantic sequence in S is a continuous word sequence after omitting the low dense words in S. It contains local semantic and structural information of a document. A long S may have several semantics. We denote all of the semantic sequences in the document S by $\Omega(S)$, which then includes the global and local semantic features as well as local structural information. However, a single semantic sequence may not represent the document global feature [1].

A semantic is a sequence of words, but the authors of SSM [1] do not use string matching to compare two semantics. Their purpose is to tell whether plagiarism happened, so they just need to know whether the common semantics length is longer than a threshold. If the max common semantic is long enough then we believe they are plagiarisms otherwise they are normal.

Hence, they can probe some words in the semantic to find the max common length $(P\#)$, which may not be very precise but it is enough to compare with the threshold. Without comparing the whole word string, they gain a great performance promotion. They define the plagiarism probability of document A and B as $\Gamma(A,B) = min(P\#/\Pi,1)$ where Π is the maximum probe number. The following is the semantic probe algorithm.

```
Function SemanticProbes()
Input parameters: Ω(A), Ω(B)
Output: Γ(A,B)
Begin
For each L(A) in Ω(A)
{
probes = get π words randomly from L(A);
For each L(B) in Ω(B)
{
p = number of probes contained in L(B);
p#=max(p);
}
}
Γ(A,B) = min(p#/π,1)
End
```

Figure 3: Semantic probe algorithm

The value of max probes number (π) need not be very large, 6 or 7 is enough. The threshold of the word density (δ) cannot be too large; otherwise the semantics length is too small to probe. The threshold of the number of contained probes in candidate semantics has crucial effect on the final detection. It seems that 4 or 5 are the best because the false positive and the false negative are both very low.

Bao Jun-Peng, Shen Jun-Yi, Liu Xiao-Dong, Liu Hai-Yan, and Zhang Xiao-Di proposed a Semantic Sequence Kernel (SSK) for DCD application which is a Kernel Method based on the semantic density, not the common global word frequency which is derived from string Kernel (SK) [6] and Word Sequence Kernel (WSK) [5].

The SSK is first finds out the semantic sequences in the documents and then it uses a kernel function to calculate their similarities. SK and WSK only calculate the gap between the first word and the last one [9]. SSK is compared with Relative Frequency Model (RFM) and Semantic Sequence Model (SSM), which is word frequency based model. It shows that SSK is excellent on non-rewording corpus.

Lodhi et al [6] proposed the string kernel method that devides the text category according to the common sequences between documents. The string kernel exploits the structural information instead of word frequency and can outperform the bag of words approach [9].

Cancedda et al. [5] introduced the word sequence kernel that extends the idea of a string kernel. They greatly increase the number of symbols to be considered as symbols are words rather than characters. It reduces the average number of symbols per document and yields a significant improvement in computing efficiency so that the training on large corpus becomes feasible without approximation [9].

Brin et al [3] proposed the first DCD Prototype such as COPS that can detect overlap based on sentence and string matching but it has some difficulties in detecting sentences and finding partial sentence copy [9]. As stated before, SCAM is

developed to improve COPS. The SCAM measures overlap based on word frequency.

Si et al. [7] built a copy detection mechanism called CHECK. CHECK parses each document to build and internal indexing structure called structural characteristic (SC), which is used in document registration and comparison modules.

Figure 4 is the CHECK System architecture [7] which is one of plagiarism detection system architecture that is available in the market. As shown in the architecture diagram, CHECK is composed of three main modules: document registration, document comparison and document parsing. These three modules have been mentioned in the previous paragraph.

According to Si A., Leong H.V. and Lau R. W. H., the three modules of the system provide three basic functions: original document registration, document verification, and normal document registration. From the CHECK system architecture, the author came out with Text-Based Plagiarism Detection System Design by narrowing the scope.

As the models can guide the plagiarism detection system development, there is also some other techniques and method can be used. The common thread between information theory and computer science is the study of the amount of information contained in an ensemble [11], [12] or a sequence [13]. According to Xin Chen, there are question that; given two sequences how do we measure their similarity in the sense that the measure captures all of the intuitive concepts "computable similarities"?

There are many plagiarism detection systems that have been developed and based on their characteristic that they employ to detect plagiarism, they can be grouped into two categories [10] which are; attributes-counting system and structure-metric system.

A simple attribute-counting system [14] only counts the number of distinct operators, distinct operands, total number of operands of all types, and then

construct a profile using the statistic for each program. While structure-metric system extracts and compares representation of program structure.

YAP Family [15] is one of the system that available using the structure-metric system. Such system usually has two phases. The first phase involves the tokenization procedure to convert source codes into token sequences by a lexical analyzer. Even though it is for programs codes, it is believed that the method can also be used for text or documents. The second phase involves the method to compare those tokens sequences.

A token for the case above is referring to the basic unit of programming languages such as "if," "then" and many more.

YAP. A similarity score used in YAP system [15] is a value from 0 to 100, called percent match, representing the range from "no-match' to "complete-match". It is obtained by the formulas

Match = (same – diff) / minfile – (maxfile – minfile)/ maxfile

PercentMatch = max (0, Match) * 100

The formulas that have been used in YAP system can be implemented in Text-Based Plagiarism Detection System to measure the percentage of plagiarism of the students work to the original author.

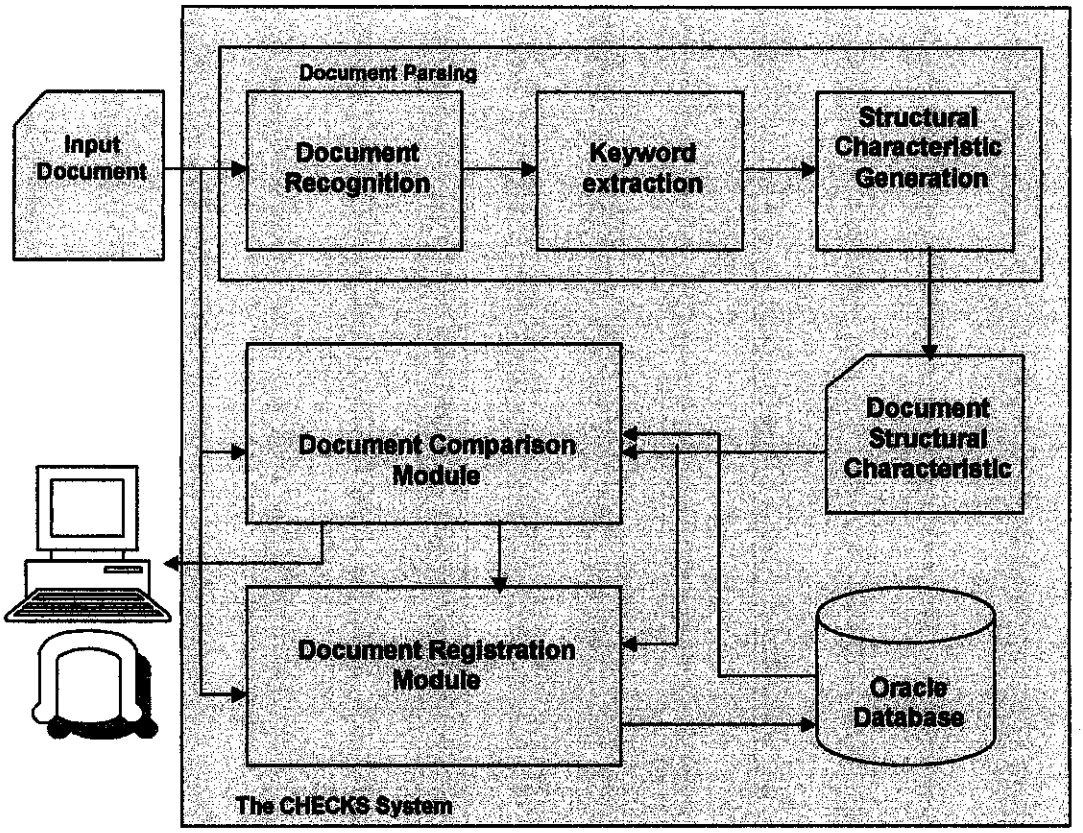


Figure 4: the architecture of CHECK

From the previous studies and research of related work, the author chooses to use the Word Frequency Model (WFM) as the reference for Text-Based Plagiarism Detection System or PlagTest 1.0 development. Besides, PlagTest 1.0 is developed by using simple method which it can enhance system performance in term of response time.

PlagTest 1.0 has two functions which are "Word by Word Comparison" and "Line by Line Comparison". Word by word comparison is suitable for text files that are in paragraph format such as quote and simple text files. While line by line comparison is suitable for text files that contain texts that are in lines format such as coding and poem.

Using the WFM, the Text-Based Plagiarism Detection System is able to count the word occurrence (Word Clustering) in a body of text file. After word clustering, the system will compare the strings to identify the number of differences.

The total difference will determine the plagiarism status. Tokenizer is needed to extract the words into the list and after that the words are sorted ascending order (A-Z). With the words listed, Word clustering will take place.

For the Text-Based Plagiarism Detection System, the author also considers the characters appearance including space and also the characters cluster (the occurrence of same character).

CHAPTER 3

METHODOLOGY/PROJECT WORK

3.1 PROCEDURE IDENTIFICATION

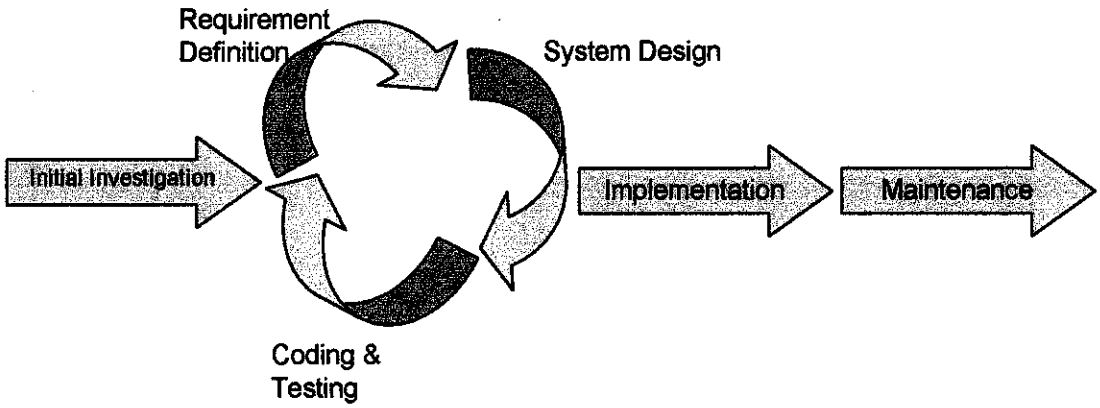


Figure 5: Methodology phases / project procedure

The initial investigation is important to start a system development. During the initial investigation, everything have to be done including to come out with the problem statement, study the previous research paper for information gathering, analyze the significance of the project whether it is worth to be done, the cost involved and also the time frame. The author first gather all related research paper that can be referred in order to come out with the problem statement and as general ideas to develop text-based plagiarism detection system. Besides, the author also did find some plagiarism detection system that available in the market as a sample application.

After the initial investigation is the requirement definition. At this phase, the requirement would be the system functions determination itself and also the method

that will be used. Referring to the model above, requirement definition can be modified as there are changes.

After the requirement definition, the next phase is system design. In the system design the system flow processes is specified and the functions such as system interface design and others are also determined.

The system design phase is followed by 'coding and testing' phase. Based on the system design, the system will be developed by generating the codes and do testing to make sure that it is working. The three phases; requirement definition, system design and coding & testing can be redone due to changes until it satisfy the requirement and ready for implementation.

The next phase is implementation, where the system is ready to be used by the user. The last phase is maintenance. At this phase, the system needs to be keeping updated due to users' preferences and requirements.

3.2 TOOL REQUIREMENT

The tools that are needed for the project are a computer with a good performance, and Microsoft Visual Basic 6.0 software. The system is fully developed using VB 6.0 and as for the database, Microsoft Access version 7.0 is used.

CHAPTER 4

RESULT AND DISCUSSION

4.1 SYSTEM DESIGN

The system design for the system is shown in figure 6 below. The system will have one database which is use to store documents and record. The system must be able to detect similarity among students' assignment after text extraction.

Referring to the system design, the user will first need to input the document or text into the system as the master and target text file. Then the system will extract the text input by the user. After the text extraction, the system will start comparing the text and lastly presents the status.

The system should be able to compare between two text files or more. If it is between two text files then it can be done using the master input file and target file. If it is between many files which are more than two, one text file will be taken as master file and the rest as the target files. The master file will be compared with the documents that are stored in the database. If plagiarism is detected, the system will produce the output or report for the percentage of plagiarism from the original author. The status can be stores as record for future reference in the database.

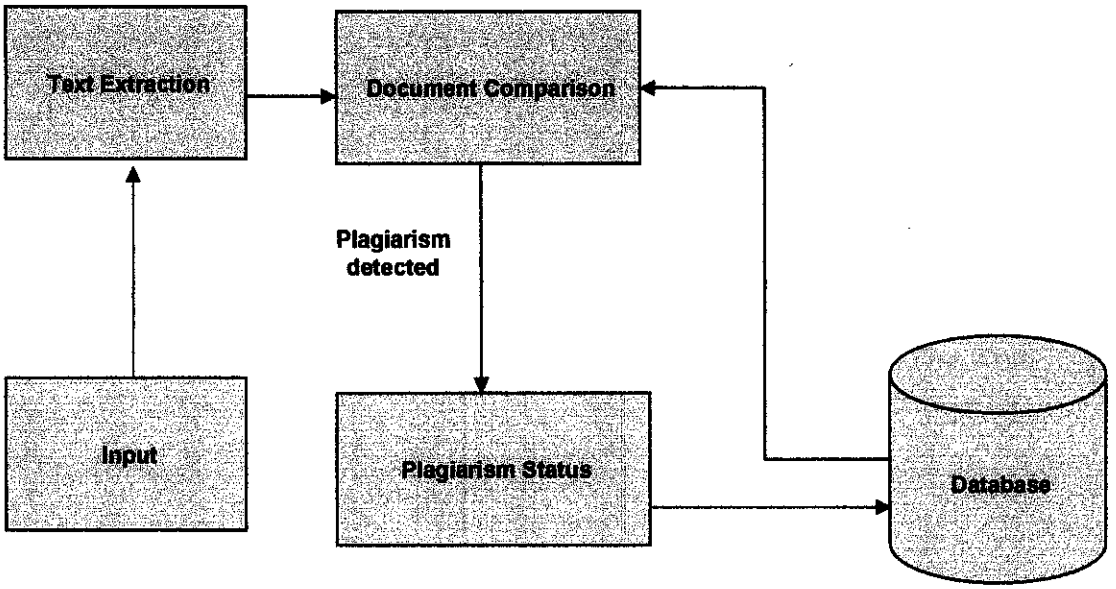


Figure 6: Text-Based Plagiarism Detection System's Design

4.2 SYSTEM FLOW PROCESS

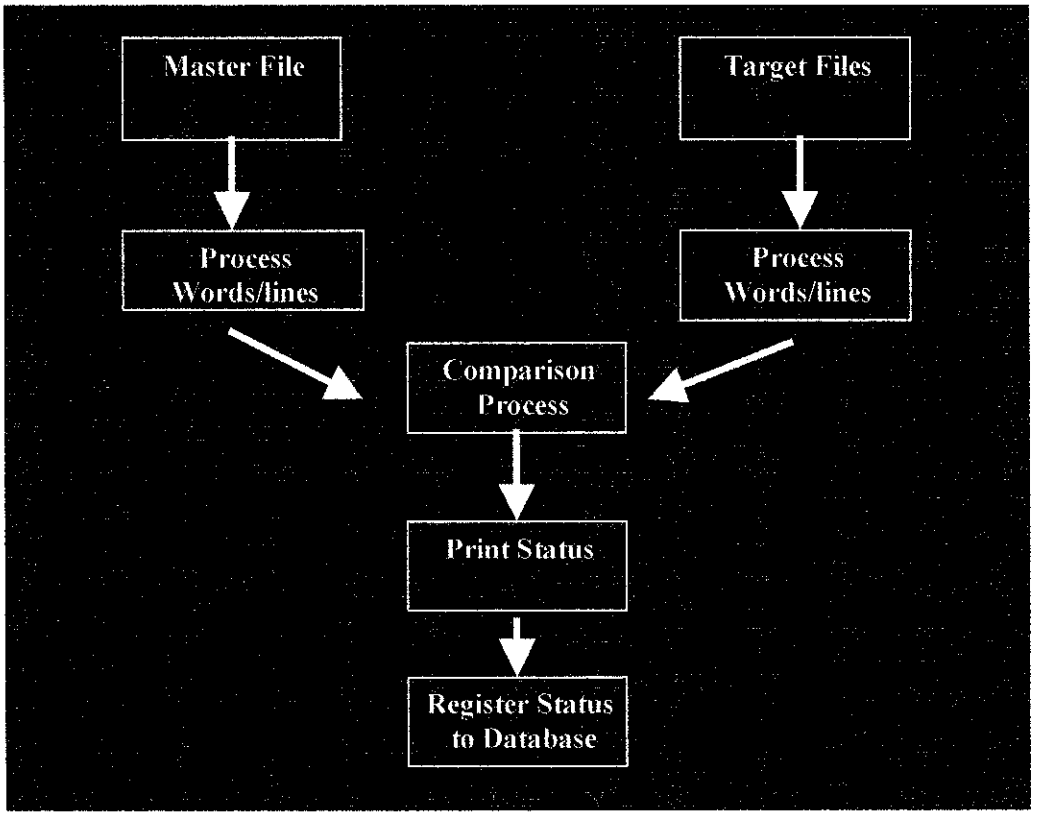


Figure 7: Text-Based Plagiarism Detection System's Flow Process

For word by word comparison, the system involves 5 processes that classified under two stages that are process word and compare files. The process is listed below:

Process Words

1. Text extraction (tokenizing)
2. Sorting (Ascending)
3. Word Clustering (the number of same word occurrence in a text)

Comparison Process

1. Unification for string matching to find the total differences
2. Present the plagiarism status of the similarities based on the total differences using a mathematical calculation

For line by line comparison, the system involves 3 processes that classified under two stages that are process lines and compare files. The process is listed below:

Process Lines

1. Count the number of lines in text files and find the lines that contain differences

Comparison Process

1. Compare lines and find the number of differences
2. Present the plagiarism status of the similarities based on the total differences using a mathematical calculation

Referring to the flow process shown, the system need input files as master file and target files. Each of the files will be first being processed accordingly. After the files are processed and done with comparison, the status is printed and it can be stored as record for reference.

4.3 WORD BY WORD COMPARISON PROCESS (1 – 1 TEXT FILES COMPARISON)

Example:

TEXT

I want to be an engineer. When I was in primary school, I want to be a doctor.

4.3.1 TEXT EXTRACTION USING TOKENIZER

Unsorted	Sorted
I	a
want	an
to	be
be	be
an	doctor
engineer	engineer
when	I
I	I
was	I
in	in
the	primary
primary	school
school	the
I	to
want	to
to	want
be	want
a	was
doctor	when

Table 1: String Extraction

The text is first extracted using a tokenizer and after that they are sorted in ascending oder (A-Z). The purpose of sorting the strings is to ease the word clustering process.

4.3.2 WORD CLUSTERING PROCESS

Word clustering is done to calculate the same words occurrence in a text. By having clustered word, the system can count the amounts of same words and after that place them for string comparison.

Cluster [0] = "a"
Counter [0] = 1
Cluster [1] = "an"
Counter [1] = 1
Cluster [2] = "be"
Counter [2] = 1+1
Cluster [3] = "doctor"
Counter [3] = 1
Cluster [4] = "engineer"
Counter [4] = 1
Cluster [5] = "I"
Counter [5] = 1+1+1
Cluster [6] = "in"
Counter [6] = 1
Cluster [7] = "primary"
Counter [7] = 1
Cluster [8] = "school"
Counter [8] = 1
Cluster [9] = "the"
Counter [9] = 1
Cluster [10] = "to"
Counter [10] = 1+1
Cluster [11] = "want"
Counter [11] = 1+1
Cluster [12] = "was"
Counter [12] = 1
Cluster [13] = "when"
Counter [13] = 1

The list above shows how word clustering is done. It start with cluster 0 with a string name (example: a) and the counter will start counts until there is no more "a". The new cluster will take place and the process will continue until the last cluster is found.

4.3.3 WORD CLUSTERING RESULT

a	-->	1
be	-->	2
doctor	-->	1
engineer	-->	1
I	-->	3
in	-->	1
primary	-->	1
school	-->	1
the	-->	1
to	-->	2
want	-->	2
was	-->	1
when	-->	1

Table 2: Word Clustering Result

The table above shows the clustered word after Word Clustering Process. The total clustered word is 13 and at the right side is the total number each of the clustered words.

4.3.4 UNIFICATION FOR STRING COMPARISON

Text 1 = A

Text 2 = B

Unified Words Cluster = C

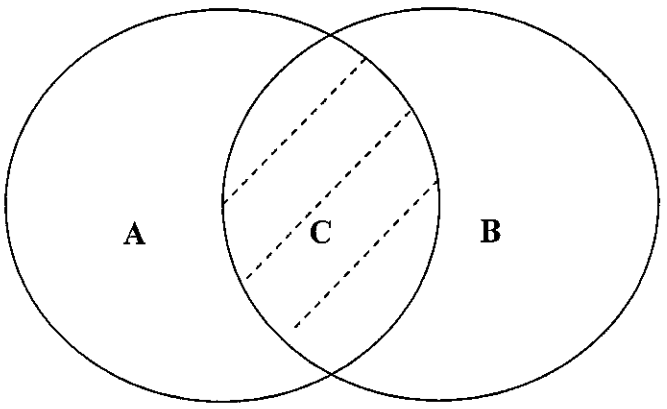


Figure 8: Unification

$$C = A \cup B$$

The purpose of the unification is to trace the overlapping words or strings between the texts. By having the unification subtraction can be done in order to identify difference or score.

Text A as Master

I want to be an engineer. When I was in primary school, I want to be a doctor.

Text B as Target

I want to be a pilot. When I was in primary school, my ambition is to be an astronaut.

A	B	C
a	a	a
an	ambition	ambition
be	an	an
doctor	astronaut	astronaut
engineer	be	an
I	I	be
in	in	doctor
primary	is	engineer
school	my	I
the	pilot	in
to	primary	is
want	school	my
was	to	pilot
when	want	primary
	was	school
	When	the
		to
		want
		was
		when

Table 3: Unification for string comparison

During the unification, all words from text files that have been clustered will be put into the unification field / unified words cluster (C) like the example in table 3. In the unification field there are only different words from the texts and no redundancy occurs. From the column C, the differences calculation can be done.

4.3.5 CALCULATE DIFFERENCES FROM STRING COMPARISON

Total Differences = A – B

C	A	B	A - B
a	1	1	0
ambition	0	1	-1
an	1	1	0
astronaut	2	2	0
an	1	1	0
be	2	2	0
doctor	1	0	1
engineer	1	0	1
I	3	2	1
in	1	1	0
is	0	1	-1
my	0	1	-1
pilot	0	1	-1
primary	1	1	0
school	1	1	0
the	1	0	1
to	2	2	0
want	2	1	1
was	1	1	0
when	1	1	0

A - B
0
1
0
0
0
0
1
1
1
0
1
1
1
0
0
1
0
1
0
0
9

Total Difference

Table 4: Difference Calculation

Total words cluster after unification = 20

- *The negative value resulting from the subtraction (A-B) must be converted to positive value for the accuracy of string matching*

Table 4 shows how the difference is calculated. If text A doesn't have a word of "ambition" the value will be put as 0 and if text A has the word, then the value must not be 0.

4.3.6 PLAGIARISM STATUS

Total word cluster after unification, UWC (Unified Word Cluster) = 20
Total Difference = 9

Status or % of match = 100 – (((A – B) / UWC) * 100)

If Status is ≥ 50% then Plagiarism Status is "Suspected as plagiarized work"
If Status is < 50% the Plagiarism Status is "not plagiarized work"

From the example above:

Status =100 – ((9/20) * 100)
= 55%

Thus, the text file is suspected as plagiarized work.

The concept of this system ***is the lower the different value between the text files, the higher the possibility of plagiarism*** and ***the higher the different value, the lower the possibility of plagiarism***. Condition is added in order to identify plagiarism.

4.4 LINE BY LINE COMPARISON PROCESS (1 – 1 TEXT FILES COMPARISON)

Example:

TEXT A

1 Don't write yourself off yet
2 It's only in your head you feel left out or looked down on
3 Just try your best
4 Try everything you can
5 And don't you worry what they tell themselves when you're away
6 It just takes some time
7 Everything will be okay

TEXT B

1 Don't write yourself off yet
2 It's only in your head you feel left out or looked down on
3 Just try your best
4 Try everything you can
5 And don't you worry what they tell themselves when you're away
6 It just takes some time
7 Everything will be just fine
8 Everything will be all right

The system will count the number of lines of both the text files. As the example above text A has 7 lines and text B has 8 lines. As seen above in text A above the seventh line, there is some modification made and in the text B there is one line that is not in text A which is line seven. Line eight in text B is same like line seven in text A but it is different because there is word modification.

The system will first count the number off different lines. Thus there is only one different line in both text files. The system will compare the lines and after that it will highlight the similar lines and the lines that are different in different color so the lecturer can see the difference.

The percentage of similarity can be calculated based on the different that the system detect. It calculates the percentage of similarity by using the formula shown.

Number of lines in Text A = A
Umber of line in Text B = B
Difference = A – B

Status, % of match = 100 - (difference / A) * 100)

% match ≥ condition = suspected as plagiarized work
Else not plagiarized work

If A – B resulted a negatives value then the system will change it to positive value, so the system can produce accurate status. The system enables the user to set the condition whether the file is plagiarized or not.

4.5 ONE TO MANY TEXT FILES COMPARISON

If there are many files to be compared, then there will be more comparisons occur at once and it will require time to do the comparison. For one to many files comparison, the word by word comparison and line by line comparison process are the same. The different is only the time taken to do comparisons. It is only the matter how comparisons occur with many files. The process is discussed below.

Assume that there are six files to be compared.
Number of files, n = 6

To start comparison, 1 file needs to take as master file or start point. For example, file 1 is taken as master file and file 2 until 6 as target files. Basically, file 1 will be compared to the rest of target files. If the file one is completed with comparisons with the target files, file 2 will start compare with file 2,3,4,5 and 6. The process continues until there is no more files to be compared. The process is shown in figure 9.

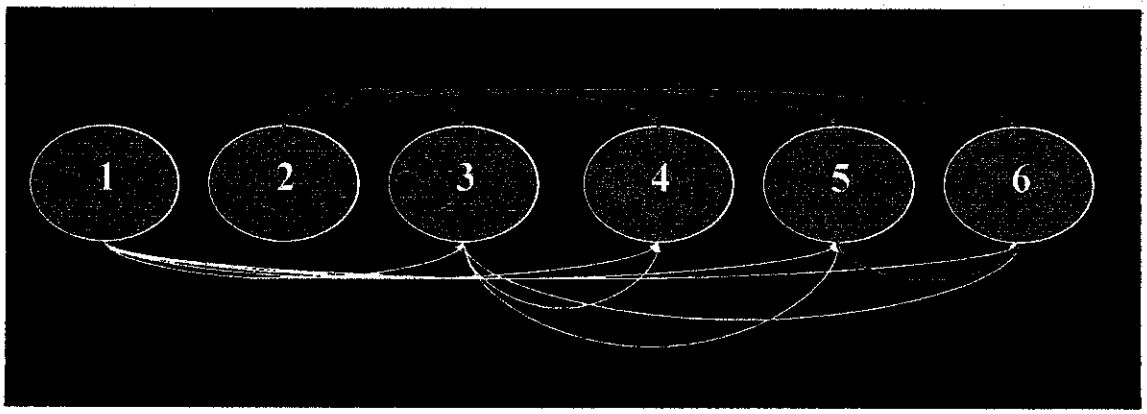


Figure 9: Comparison Process

1 – 2	2 – 3	3 – 4	4 – 5	5 – 6
1 – 3	2 – 4	3 – 5	4 – 6	
1 – 4	2 – 5	3 – 6		
1 – 5	2 – 6			
1 - 6				
5	4	3	2	1

Table 5: Comparison Process And The Number of Comparisons Occur

In the table above it shows that how "one to many" files occur same like in figure 9. As seen in the table, there are numbers that colored in red. The number represent the numbers of comparisons occur for file 1, file 2 and the rest.

To calculate the total comparisons occur for 6 files, the system will total up the value that is in red color or by using a formula shown.

Formula: N = n (n – 1) / 2

N = 5 + 4 + 3 + 2 + 1 = 15

Or

Number of Comparisons occur, N = 6(6 – 6) / 2 = 15

Thus, there are 15 comparisons occur for 6 files.

As stated before, if there are many files to be compared, it means that there are many comparisons will occur and this will require time to process and response to user. From the system testing, the author find out that 1 to 1 text files comparison require 1 comparison and the time required is 2 second. The author provides a table shows the number of files, numbers of comparisons occur and total time required.

n	N	Time Required (Second)	Time Required (Min)	Time Required (Hour)
1	0	0	0.00	0.000
2	1	2	0.03	0.001
3	3	6	0.10	0.002
4	6	12	0.20	0.003
5	10	20	0.33	0.006
6	15	30	0.50	0.008
7	21	42	0.70	0.012
8	28	56	0.93	0.016
9	36	72	1.20	0.020
10	45	90	1.50	0.025
20	190	380	6.33	0.106
30	435	870	14.50	0.242
40	780	1560	26.00	0.433
60	1770	3540	59.00	0.983
100	4950	9900	165.00	2.750
250	31125	62250	1037.50	17.292
300	44850	89700	1495.00	24.917
500	124750	249500	4158.33	69.306
1000	499500	999000	16650.00	277.500
5000	12497500	24995000	416583.33	6943.056

Table 6: Statistic of Comparisons Occur

4.6 THE CHARACTERS CLUSTERING PROCESS

The characters clustering process is same like words clustering process. The characters clustering purpose is to show the statistic of characters occurrence in the texts. For the time being, clustered characters are shown only for statistic and not yet for plagiarism status determination.

TEXT A

I want to be an engineer. When I was in primary school, I want to be a doctor.

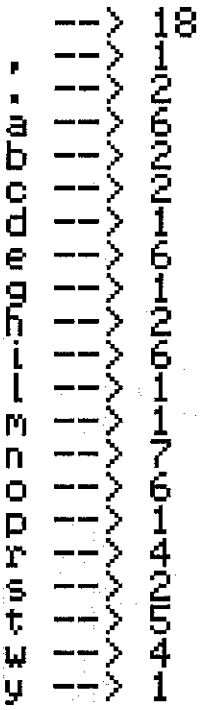


Figure 10: Character clusters including spaces

Figure 11: Total Character clusters including spaces

Figure 12: Total Characters including spaces

TEXT B

I want to be a pilot. When I was in primary school, my ambition is to be an astronaut.

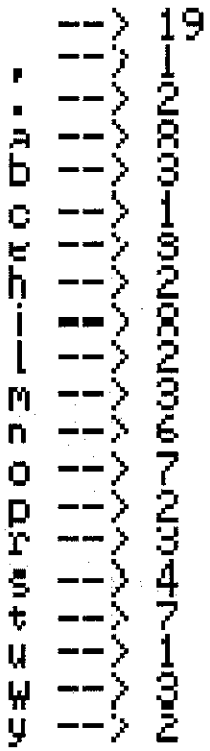


Figure 13: Character clusters including spaces

Figure 14: Total Character clusters including spaces

Figure 15: Total Character including spaces

4.7 TEXT-BASED PLAGIARISM DETECTION SYSTEM INTERFACE AND FUNCTIONS SCREEN SHOTS

Figure 16: Splash Screen

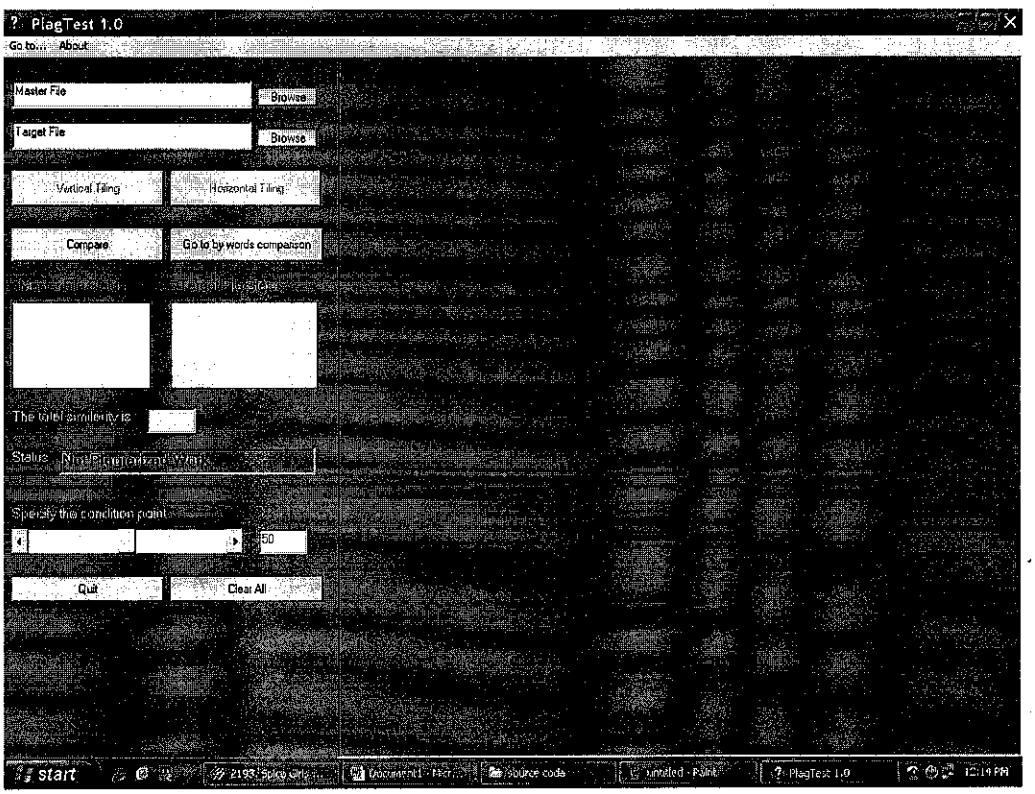


Figure 17: Line by Line Comparison Screen

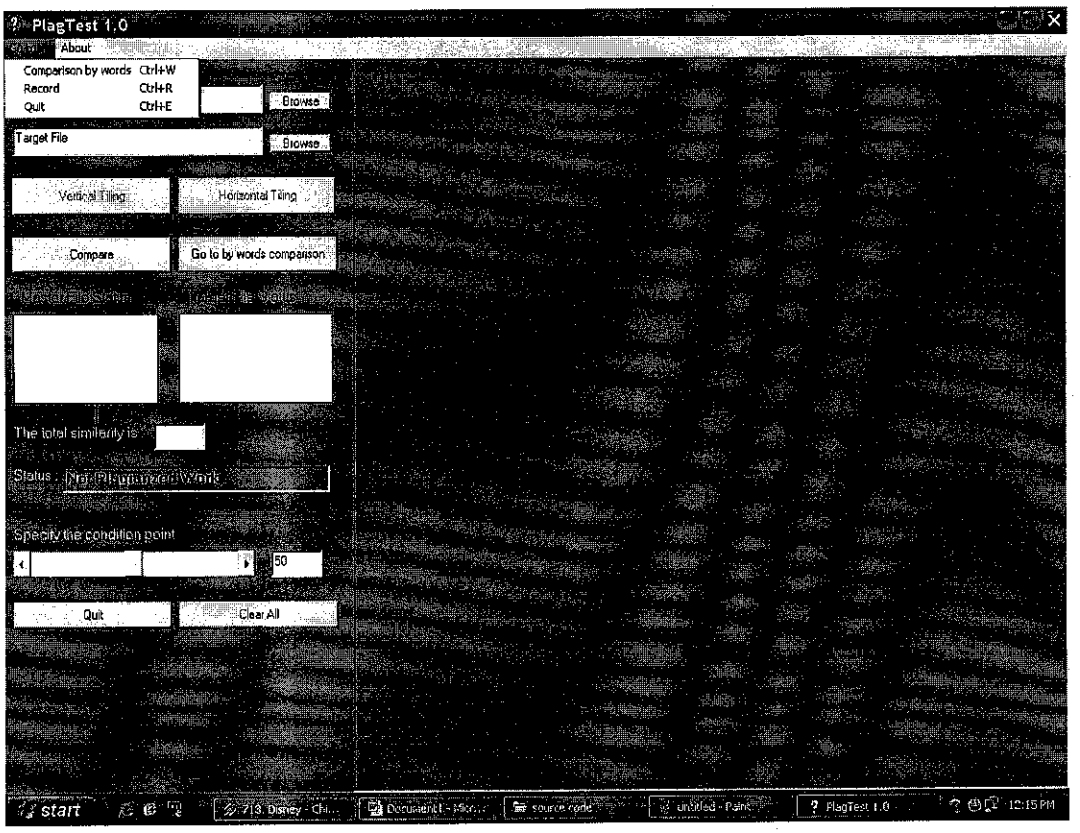


Figure 18: Line by Line Comparison Menus 1

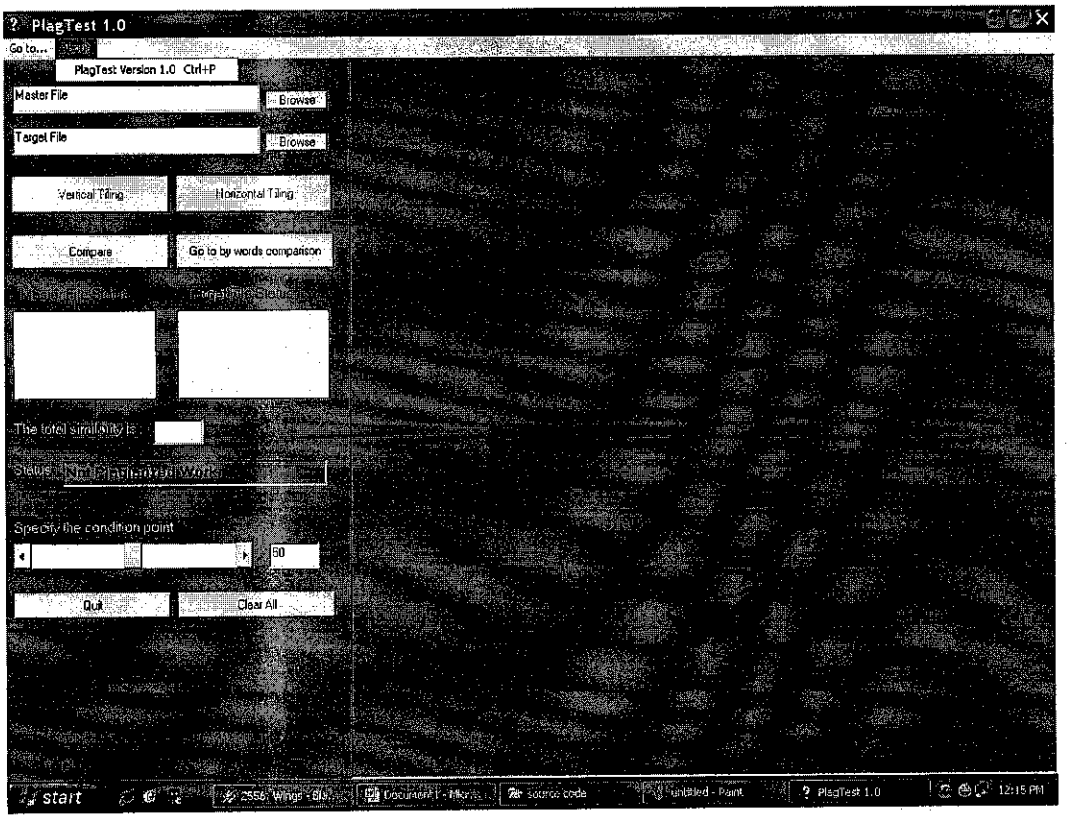


Figure 19: Line by Line Comparison Menus 2

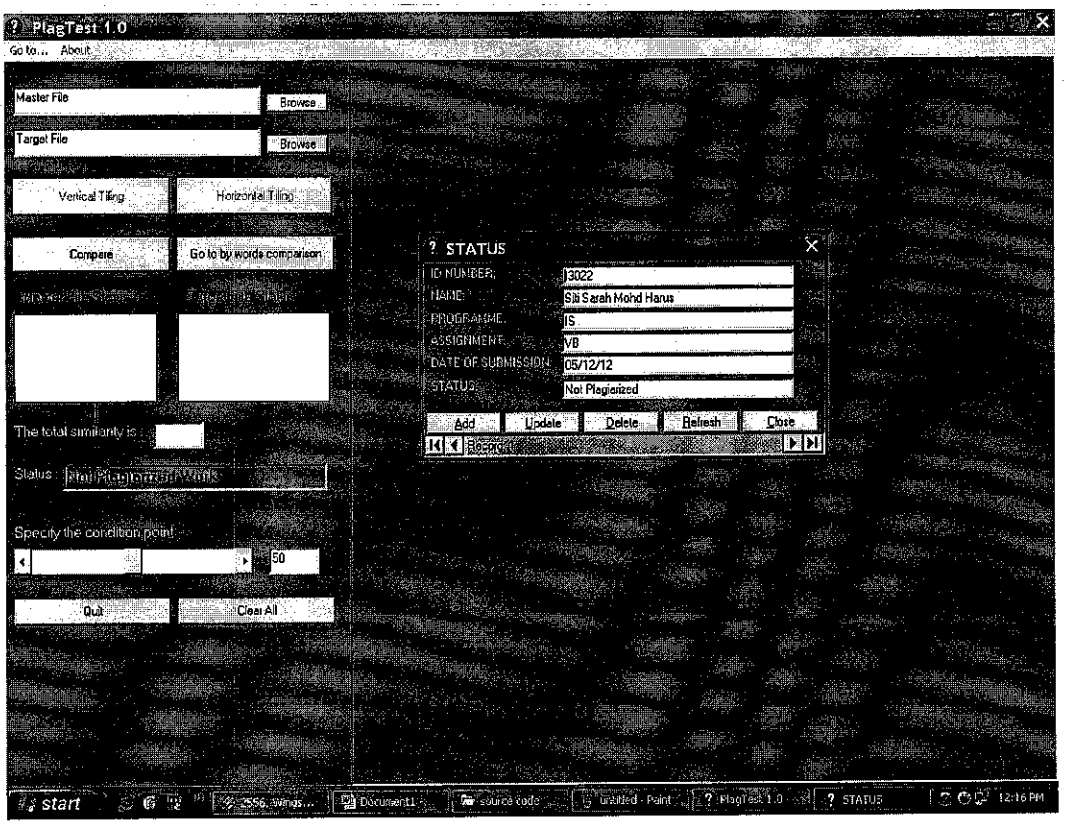


Figure 20: Status Record

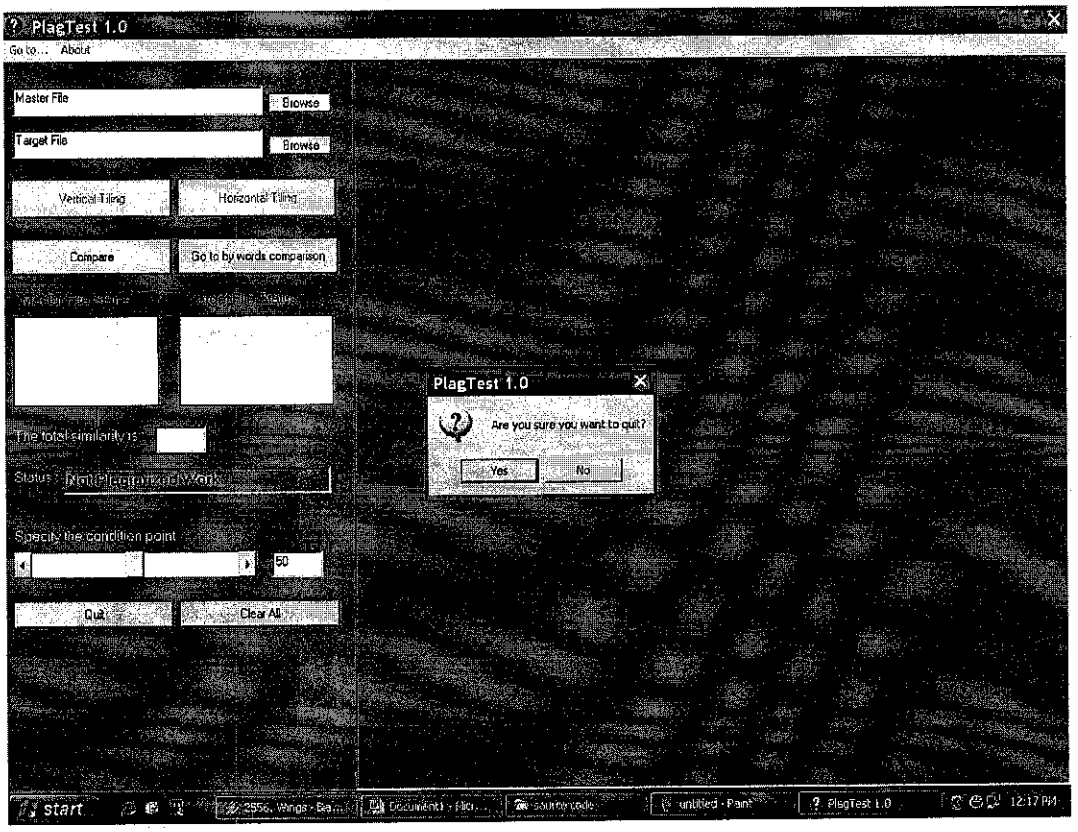


Figure 21: Quit Dialog Box

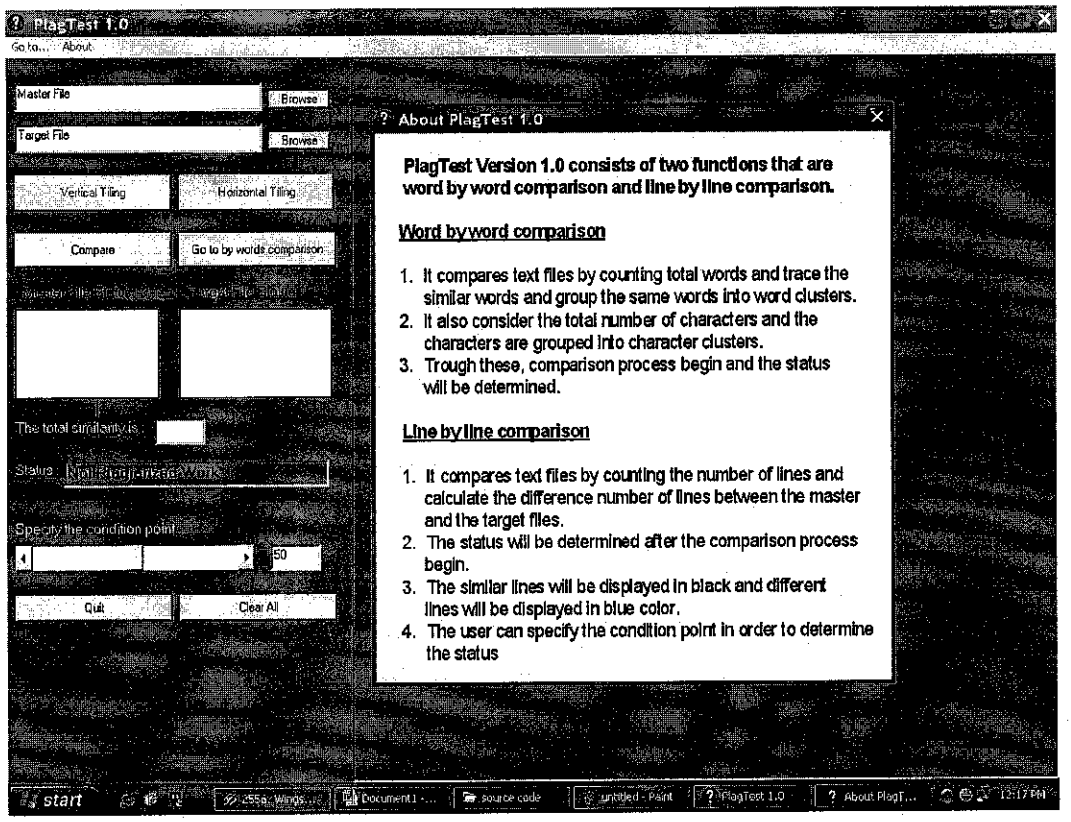


Figure 22: About PlagTest 1.0 Screen Shot

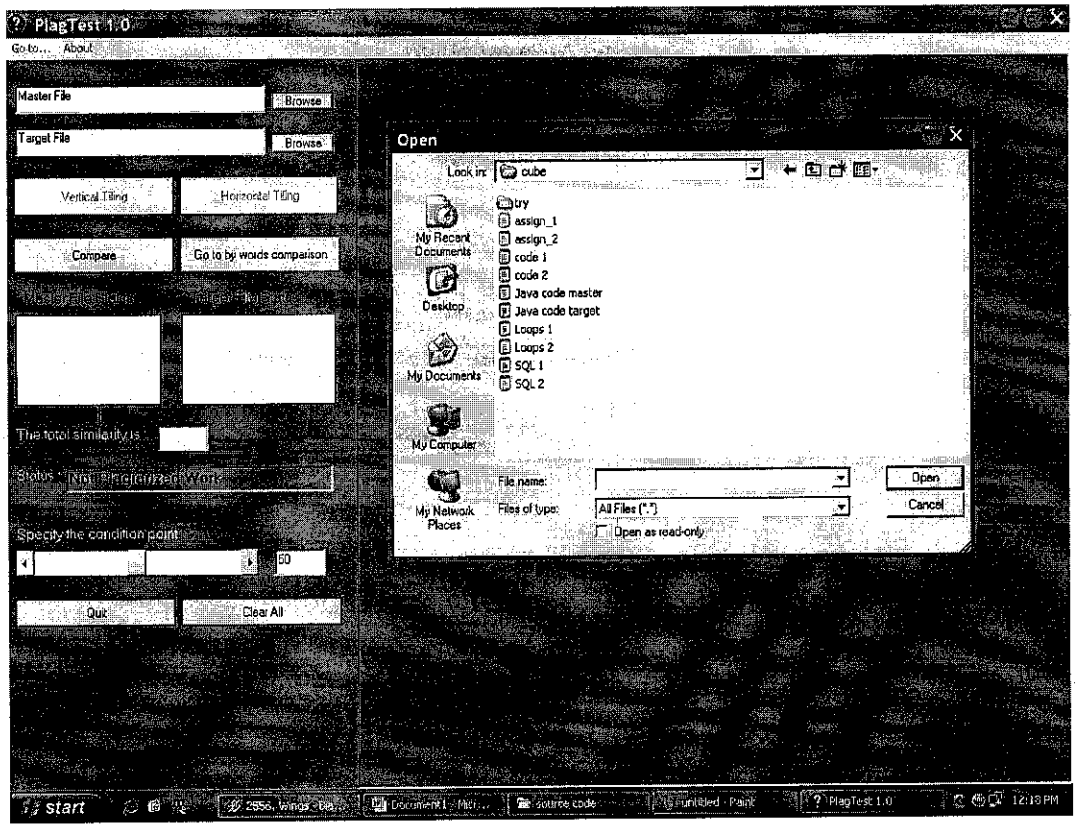


Figure 23: Browse Dialog Box

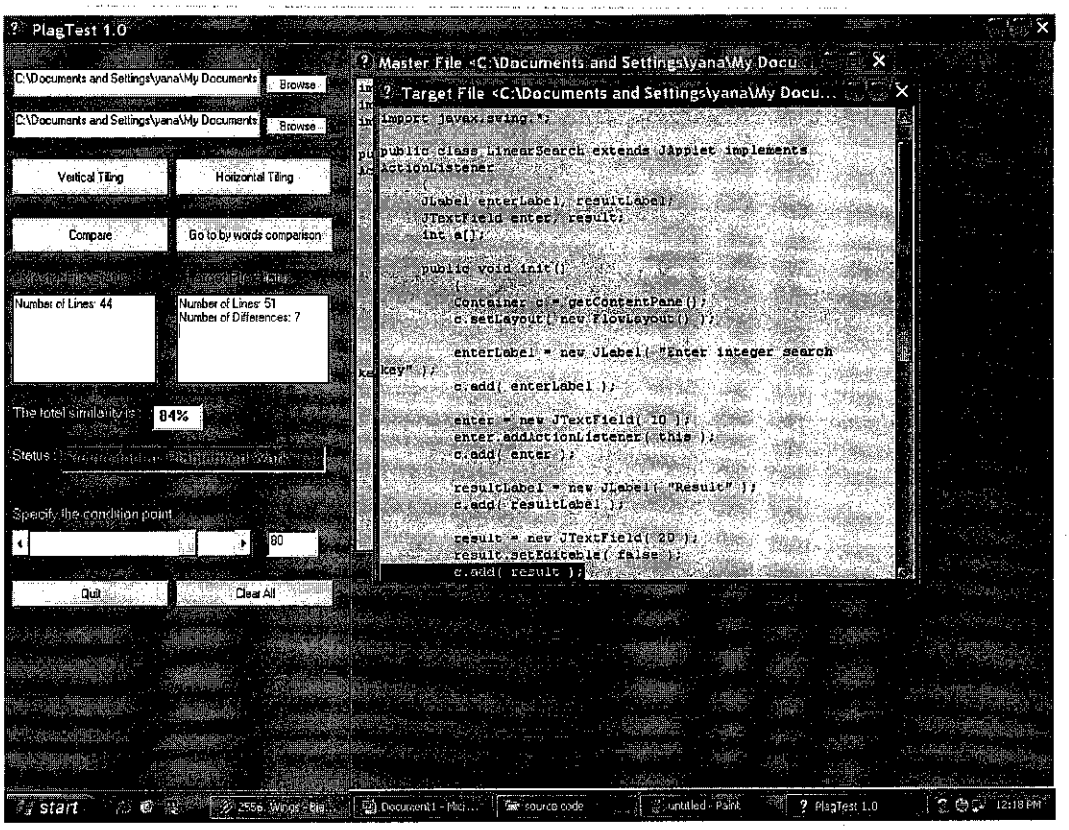


Figure 24: Compare File

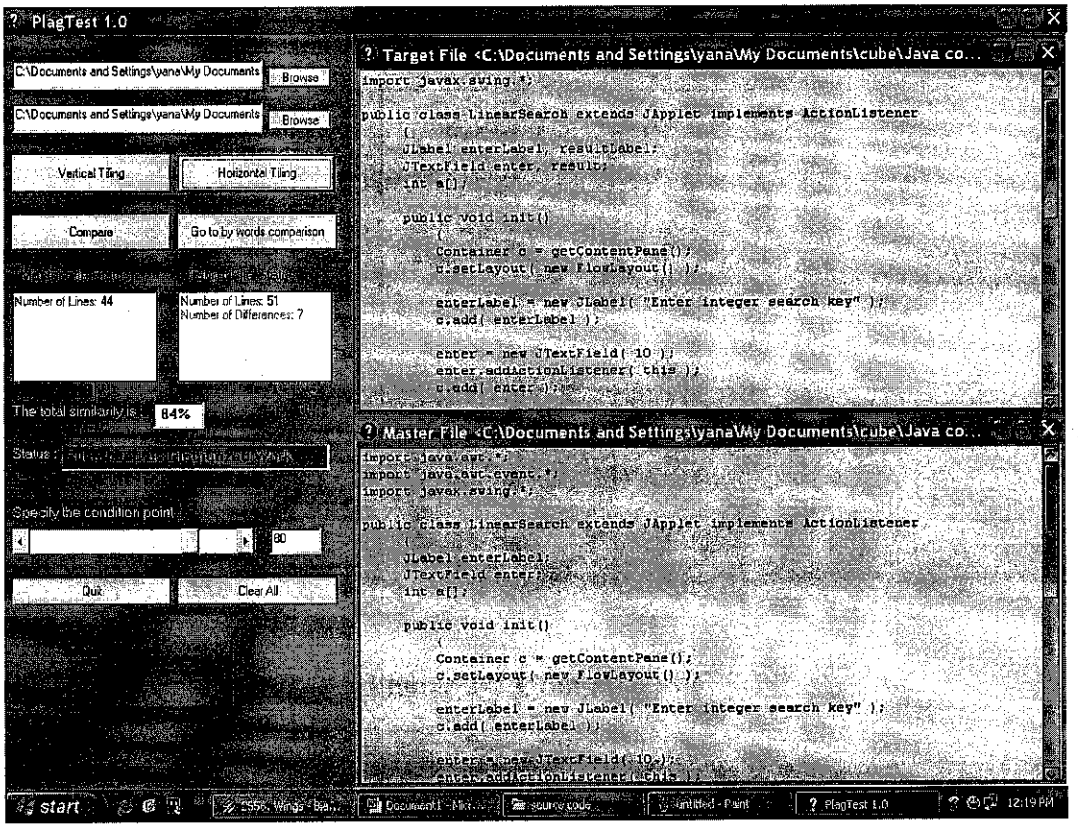


Figure 25: Arrange Text Window in Horizontal Tiling

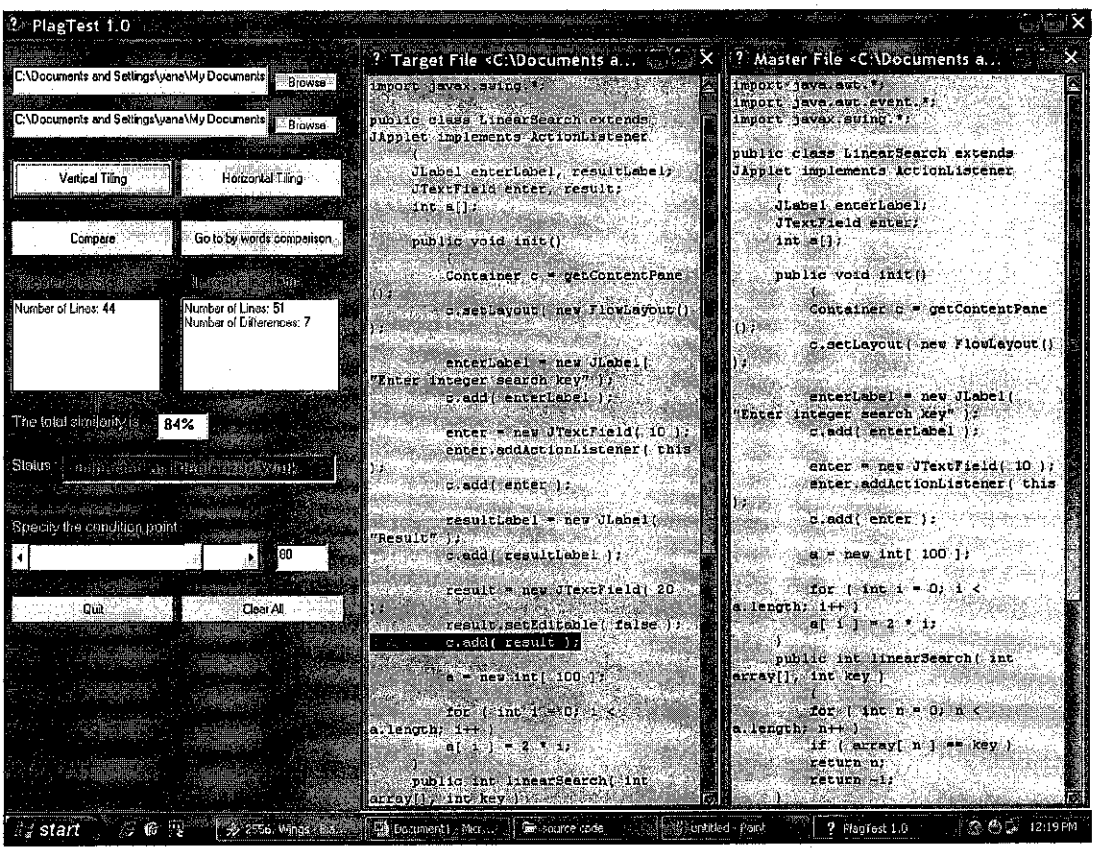


Figure 26: Arrange Text Window in Vertical Tiling

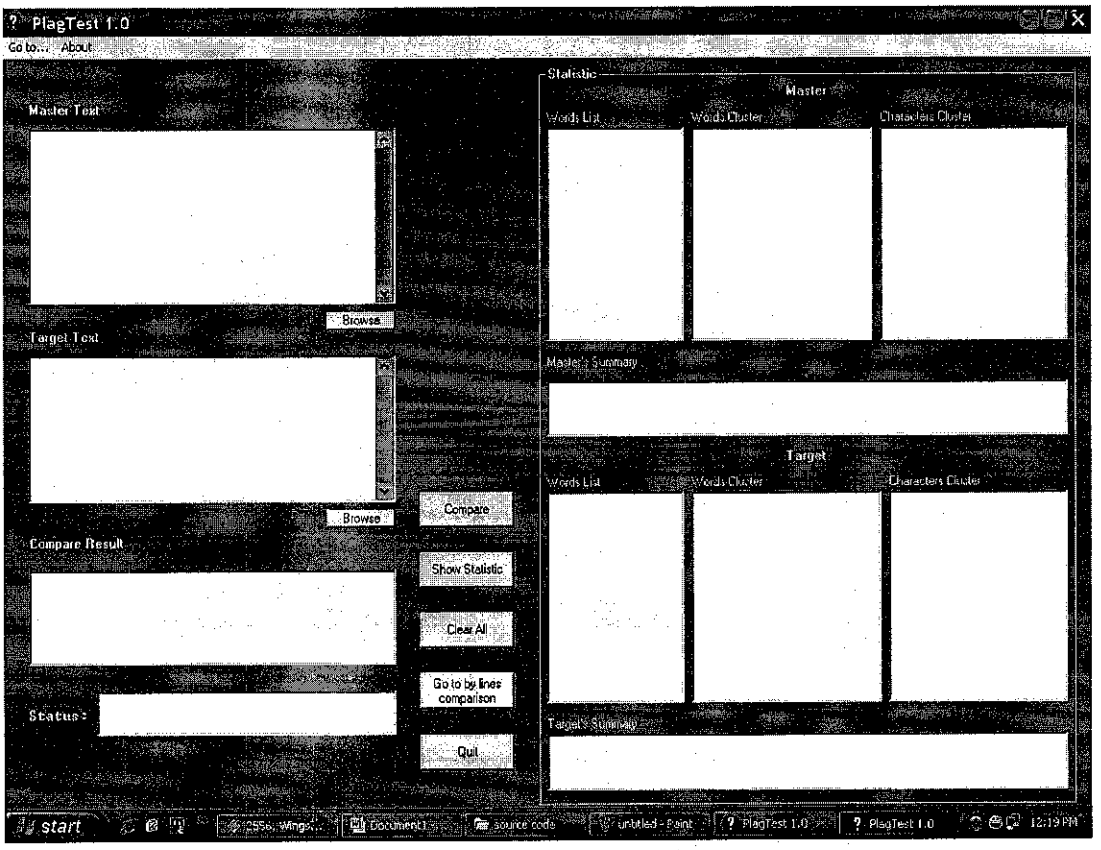


Figure 27: Word by Word Comparison Screen

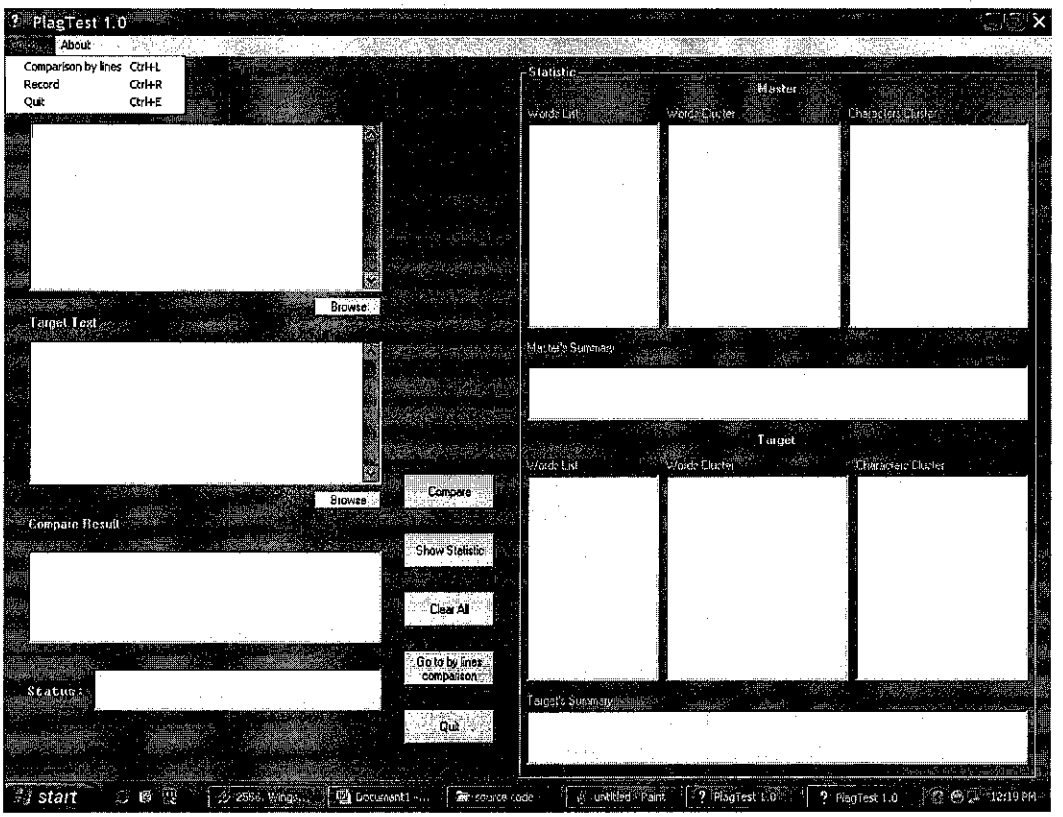


Figure 28: Word by Word Comparison's Menus 1

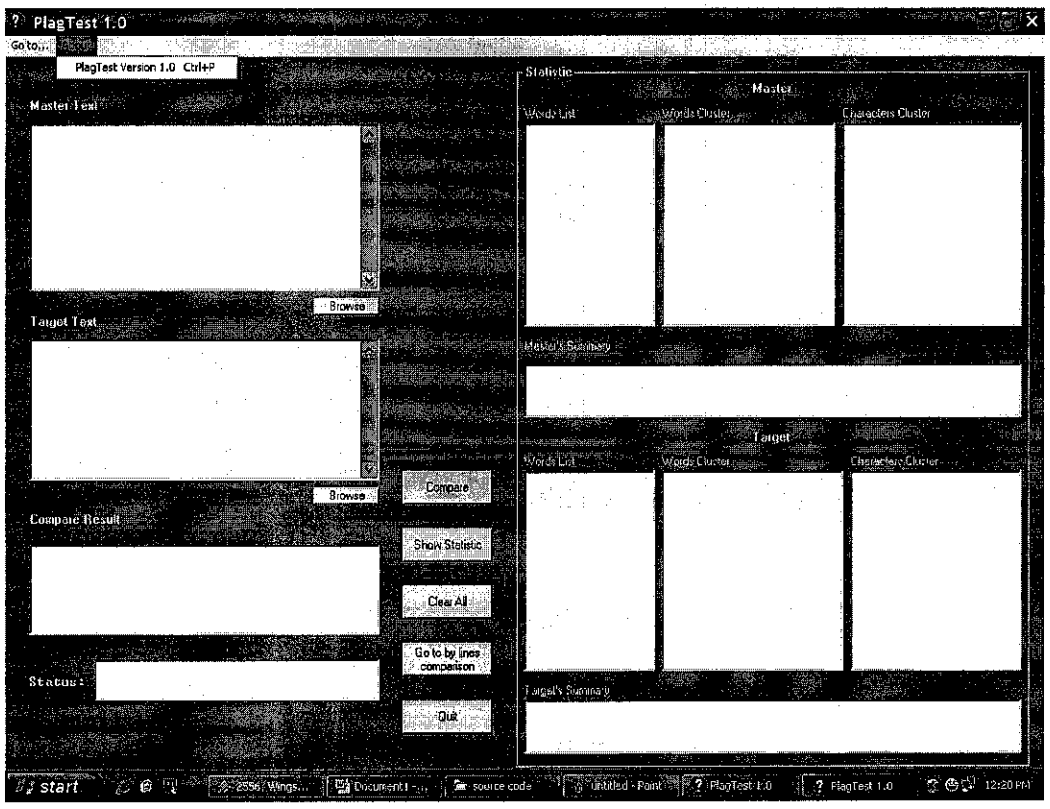


Figure 29: Word by Word Comparison's Menus 2

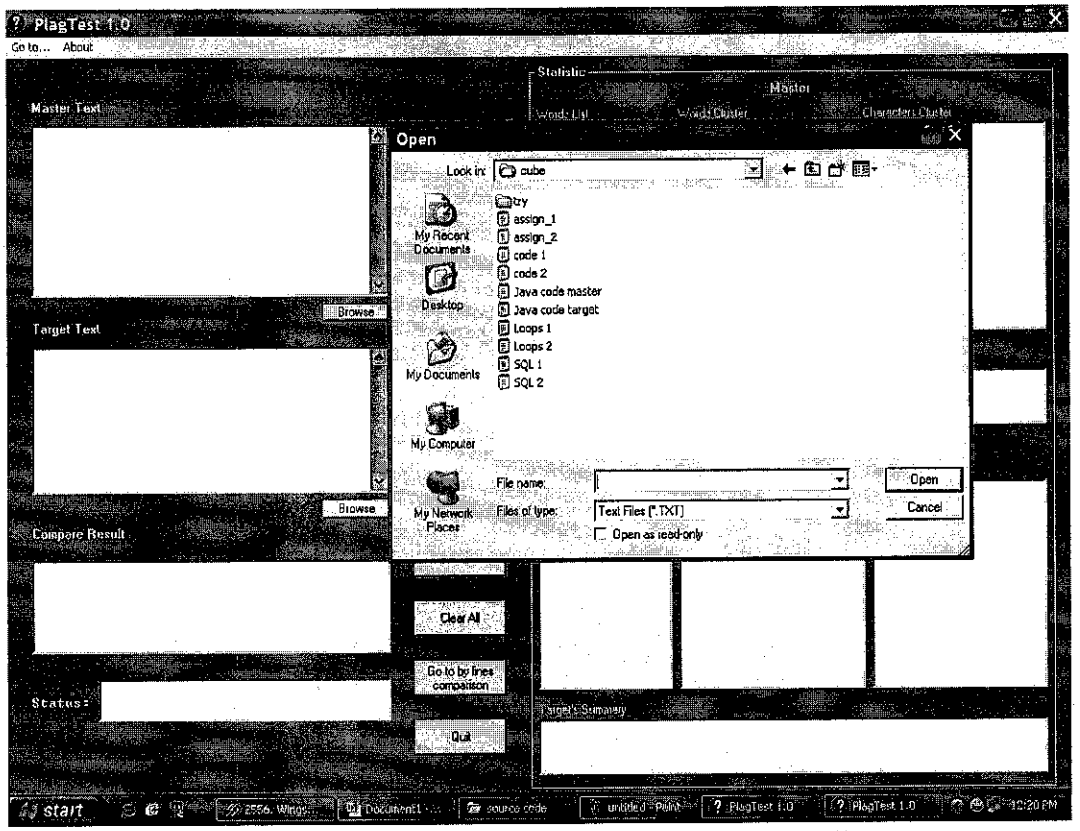


Figure 30: Browse Dialog Box

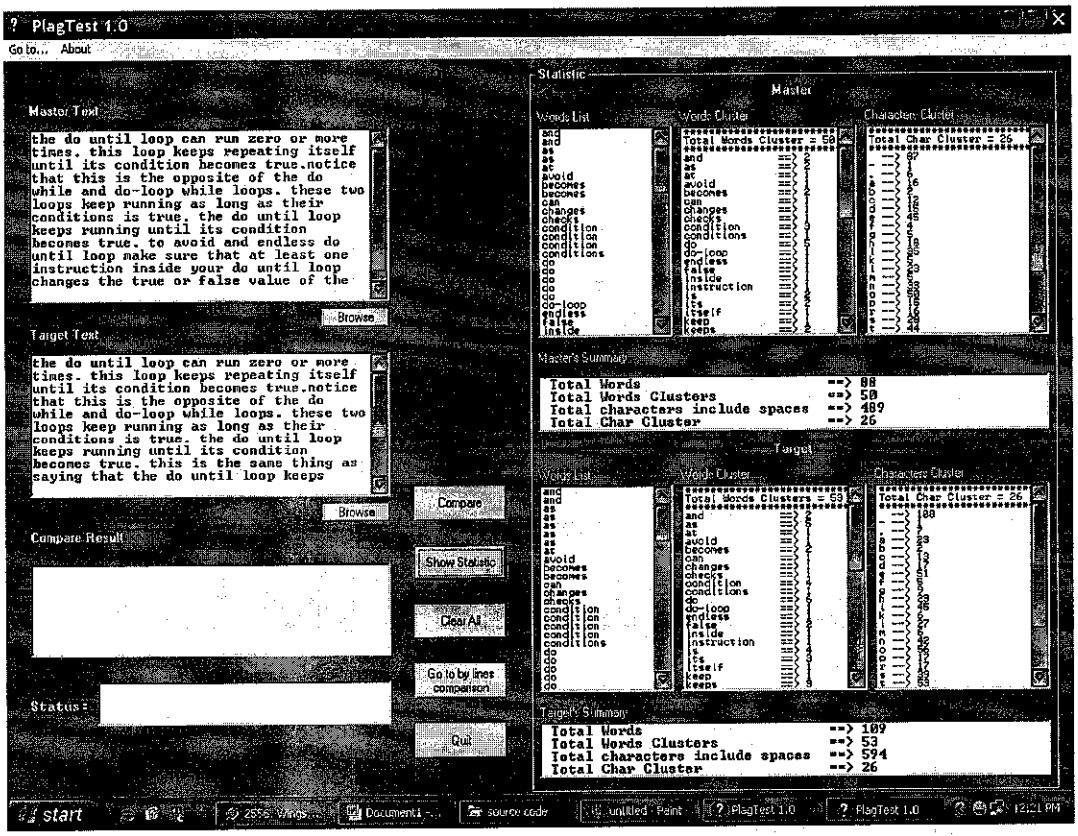


Figure 31: Show Statistic

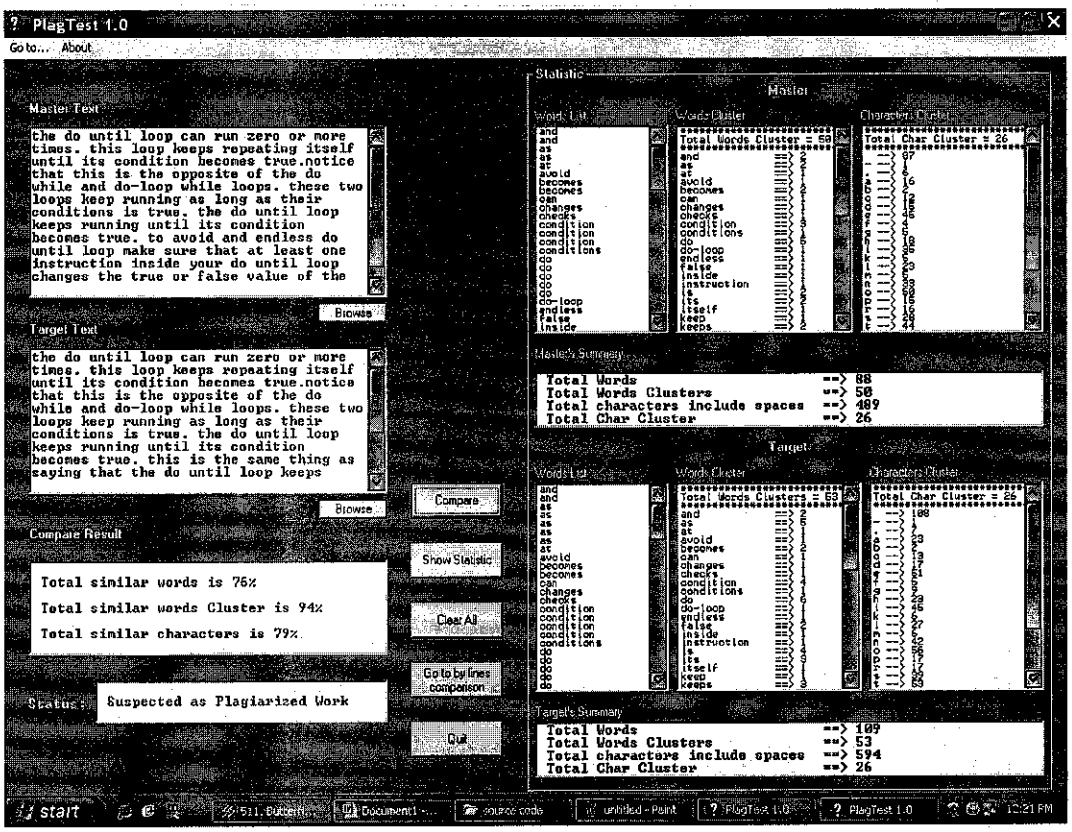


Figure 32: Compare Files

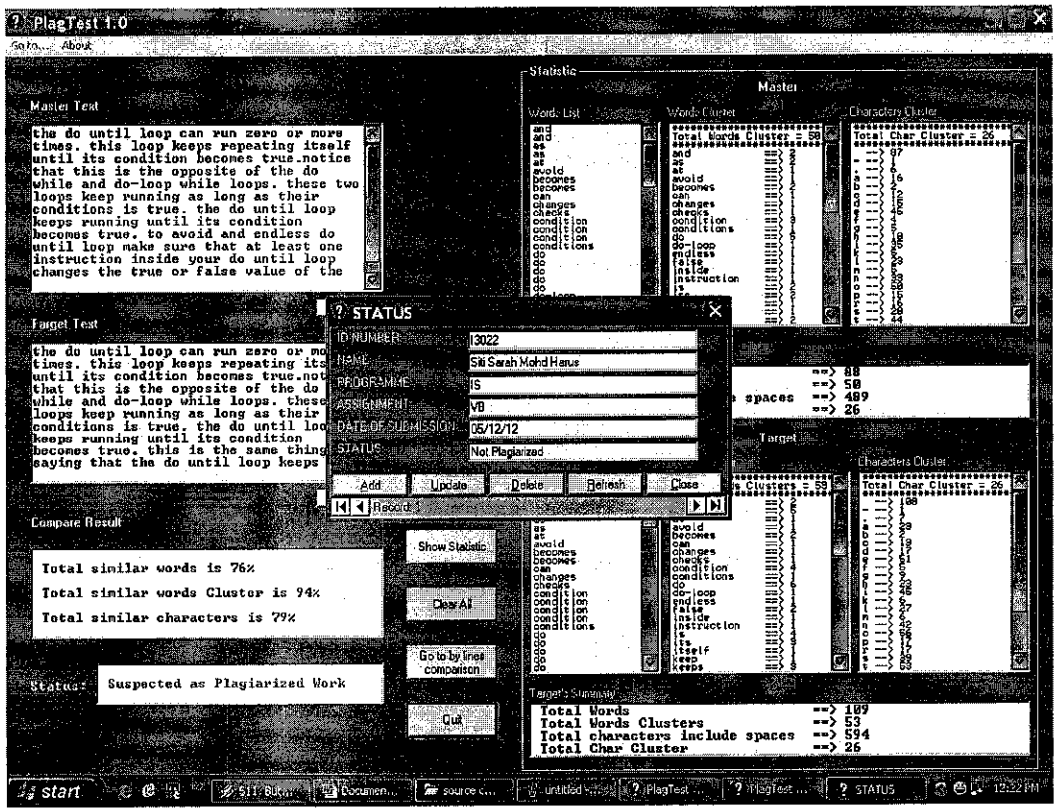


Figure 33: Status Record

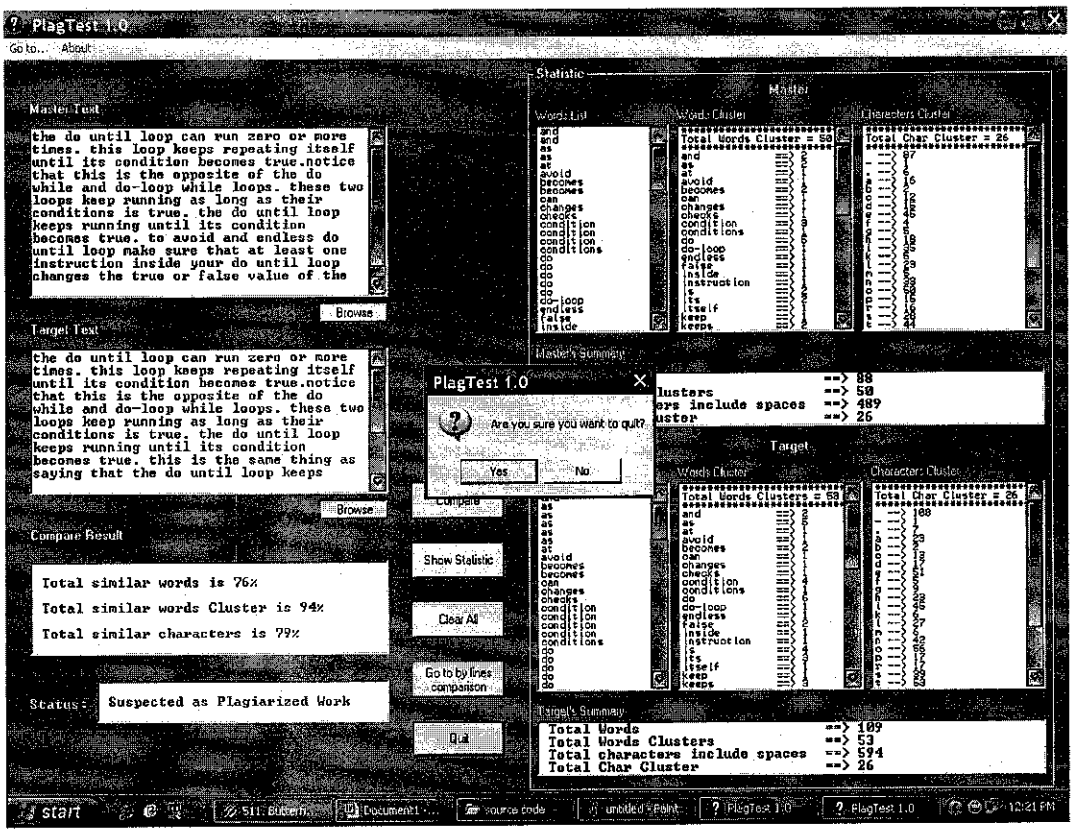


Figure 34: Quit Dialog Box

4.7.1 FUNCTIONS

1. Clear Button
   - To clear the filled field
2. Browse Button
   - To open the text file into the input fields
3. Quit Button
   - To exit the system
4. Show Statistic Button
   - To list the statistic of words, characters, words clusters, characters clusters for both master and target text files. Besides, there is also summary indicate the total of words, total of characters, total of words clusters and total of characters cluster.
5. Compare Button
   - Compare the text files for the status result.
6. Horizontal and Vertical Tiling
   - Arrange the text window horizontally or vertically
7. Go to By Words Comparison
   - Navigate from Line by Line Comparison Page to Word by Word Comparison Page
8. Go to By Lines Comparison
   - Navigate from Word by Word Comparison Page to Line by Line Comparison Page
9. About Menu
   - Prompt a window describing about Plagtest 1.0

CHAPTER 5

RECOMMENDATION AND CONCLUSION

The ultimate goal of plagiarism detection system is the reduction of plagiarism. Many cases of plagiarism can be detected by using the system which would be easily missed by a lecturer. It is recommended that the system can be implemented online whether intranet or internet. This can give easy access to the authenticated user.

The Text-Based Plagiarism Detection System that the author developed is not fully completed and has limitations. Currently, it can only compare one to one text files only. Some of the functions still have small errors. It need enhancement to improve the functions in order to meet the requirements. The main limitation of the system is it cannot identify the original text files. It can only choose one text file as the master and others as the target and decision to penalize the students who do plagiarism is depend to the lecturers.

For a group of text files comparison, it is suggested to use Self-Organizing Maps (SOM). SOM is part of the Neural Network. The SOM can populate the same files into group within the database. From the same files population, the system can detect text files similarity.

The testing result shows that PlagTest 1.0 is applicable to be used in UTP since the number of student per subject offered is less than 300. If there is only 100 students take a subject, it represent 100 files to be compared and the total time required is about two hours.

As the conclusion, the project is feasible and practicable to be developed as the method, equipments and the budget is possible and reasonable. Besides, the

project is beneficial to lecturers and the organization in order to prevent and detect plagiarism.

REFERENCES

[1] Jun-Peng Rao, Jun-Yi Shen, Xiao-Dong Liu, Qin-Bao Song. A New Text Feature Extraction Model and Its Application in Document Copy Detection, 82-87, 2003

[2] Raymond Kosala, Hendrik Blockeel. Web Mining Research: A Survey. ACM SIGKDD,2(1):1-15, 2000

[3] S Brin, J Davis, and H Garcia-Molina. Copy detection mechanisms for digital documents. In Proceedings of the ACM SIGMOD Annual Conference, s San Francisco, CA, May 1995

[4] N Shivakumar, H Garcia-Molina. SCAM: A copy detection mechanism for digital documents. In Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries (DL'95), Austin, Texas, June 1995.

[5] N.Caicedda, E. Gaussier, C. Goutte, J. M. Renders. Word-Sequence Keniels. Jounial of Machine Learning Research, 3:1059-1082, 2003

[6] H. Lodhi, C. Sannders, J. Shawe-Taylor, N. Cristianini, C. Watkins. Text Classification using String Keniels. Joumal of Machine Learning Rescarcb, 2(Fcb):419-444, 2002

[7] Si A., Leong H.V., Lmu R. W. H. CHECK A Document Plagiarism Detection System. In Proceedings of ACM Symposium for Applied Computing, pp.70-77, Feb. 1997.

[8] G Salton. The state of retrieval system evaluation. Information Processing & Management, 28(4):441-453.1992

[9] Bao Jun-Peng, Shen Jun-Yi, Liu Xiao-Dong, Liu Hai-Yan, Zhang Xiao-Di. Document copy detection based on kernel method.

[10] Xin Chen, Brent Francia, Ming Li, Member, IEEE, Brian McKinon and Amit Seker. Shared Information and Program Plagiarism Detection, 1545-1551, July 2004

[11] C.E. Shannon, "A mathematical theory of communications," Bell Syst. Tech. J., Vol.27, pp.379-423, July and October.1948

[12] W.Weaver and C.E. Shannon, The mathematical theory of communication. Chicago,IL:Univ.Illinois Press, 1949

[13] M.Li and P.Vitanyi, An introduction to Kolmogorov Complexity and Its Applications, 2nd ed. New York:Springer-Verlag, 1997

[14] K.Ottenstein, "An algorithmic approach to the detection and prevention of plagiarism."SIGCSE Bull, vol. 8, no. 4, pp. 30-41, 1997

[15] ____, "YAP3: Improved detection of similarities in computer program and other texts In Proc.27th SCGCSE Tech. symp., Philadelphia, PA, 1996, pp. 130-134

APPENDIXES

frmdoc

```
Option Explicit

Private Sub mnFile1_Click()

End Sub


Private Sub CoolBar1_HeightChanged(ByVal NewHeight As Single)
With rtext
   .Top = 0 + NewHeight
   .Left = 0
   .Width = Me.Width - 125
   .Height = Me.Height - 400 - NewHeight
End With
End Sub

Private Sub Form_Load()

With rtext
   .Top = 0
   .Left = 0
   .Width = Me.Width - 125
   .Height = Me.Height - 400
End With

End Sub


Private Sub Form_Resize()
If Me.WindowState <> 1 Then
   With rtext
      .Top = 0
      .Left = 0
      .Width = Me.Width - 125
      .Height = Me.Height - 400
   End With

End If
End Sub


Private Sub mnJump_Click()
Dim compareline As String
Dim I As Integer
compareline = rtext.SelText

If InStr(1, mastertext, compareline) = 0 Then
   I = 0
Else
   I = InStr(1, mastertext, compareline)
End If

With newdoc(0).rtext
   .SetFocus
   .SelStart = I - 1
   .SelLength = Len(compareline)

End With
End Sub

Private Sub mnJumpCompare_Click()
Dim masterline As String
Dim I As Integer

masterline = rtext.SelText
```

```
If InStr(1, comparetext, masterline) = 0 Then
   I = 0
Else
   I = InStr(1, comparetext, masterline)
End If

With newdoc(1).rtext
   .SetFocus
   .SelStart = I - 1
   .SelLength = Len(masterline)
End With
End Sub


Private Sub rtext_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
If Button = 1 Then
   SendKeys "{HOME}"
   SendKeys "+{END}"
End If
If Button = 2 Then
   PopupMenu mnedit
End If
End Sub
```

frmmain

```
Option Explicit
'Private Sub Form_Load()
'Call HScroll_Scroll
'End Sub

Private Sub begin_Click()

If txtmaster = "Master File" Or txtcompare = "Compare File" Then
   MsgBox "You must specify a Master and Target File"
   Exit Sub
End If

Dim masterline As String
Dim compareline As String
Dim mlinecount As Integer
Dim clinecount As Integer
Dim diffcount As Integer
Dim Difstats As Integer
Dim I As Integer
Dim K As Integer

newdoc(0).Show
newdoc(0).Caption = "Master File <" & txtmaster.Text & ">"
newdoc(1).Show
newdoc(1).Caption = "Target File <" & txtcompare.Text & ">"
vtile.Enabled = True
htile.Enabled = True


Open txtmaster.Text For Input As #1
Do Until EOF(1)
   Line Input #1, masterline
   mlinecount = mlinecount + 1
Loop
Close #1

ReDim masterarray(mlinecount)
mlinecount = 0

Open txtmaster.Text For Input As #1
Do Until EOF(1)
   Line Input #1, masterline
   masterarray(mlinecount) = masterline
   mtxtadd (masterarray(mlinecount))
   mlinecount = mlinecount + 1
Loop
Close #1

Open txtcompare.Text For Input As #1
Do Until EOF(1)
   Line Input #1, compareline
   clinecount = clinecount + 1
Loop
Close #1

ReDim comparearray(clinecount)
ReDim diffarray(clinecount)
clinecount = 0

Open txtcompare.Text For Input As #1
Do Until EOF(1)
   Line Input #1, compareline
   comparearray(clinecount) = compareline
   ctxtadd (comparearray(clinecount))
```

```
    clinecount = clinecount + 1
Loop
Close #1
comparecount = clinecount


mastertext = newdoc(0).rtext.Text
comparetext = newdoc(1).rtext.Text

DoEvents

Call mstatsadd("Number of Lines", mlinecount)
Call cstatsadd("Number of Lines", clinecount)

For I = 0 To clinecount
   If InStr(1, mastertext, comparearray(I)) = 0 Then
      With newdoc(1).rtext
         .SelStart = InStr(1, comparetext, comparearray(I)) - 1
         .SelLength = Len(comparearray(I))
         .SelColor = vbBlue
      End With
      diffcount = diffcount + 1
      diffarray(diffcount) = comparearray(I)
   End If
Next I
DoEvents
Call cstatsadd("Number of Differences", diffcount)

Difstats = 100 - (diffcount / (mlinecount) * 100)
Diff.Text = " " & Difstats & "%"

'If Difstats < 50 Then
'Diff2.Text = "Not Plagiarized Work"
'Else: Diff2.Text = "Plagiarized Work"
'End If

End Sub


Private Sub browse1_Click()
'cd1.Filter = "Text Files (*.TXT)|*.TXT|All_Files (*.*)|*.*"
cd1.Filter = "All Files (*.*)|*.*|Text Files (*.TXT)|*.TXT|Batch Files(*.BAT)|*.BAT|Executable Files(*.EXE)|*.EXE"
cd1.ShowOpen
txtmaster.Text = cd1.FileName
End Sub

Private Sub browse2_Click()
'cd1.Filter = "Text Files (*.TXT)|*.TXT|All_Files (*.*)|*.*"
cd1.Filter = "All Files (*.*)|*.*|Text Files (*.TXT)|*.TXT|Batch Files(*.BAT)|*.BAT|Executable Files(*.EXE)|*.EXE"
cd1.ShowOpen
txtcompare.Text = cd1.FileName
End Sub

Private Sub Clear_Click()
txtmaster.Text = ""
txtcompare.Text = ""
mstats.Text = ""
cstats.Text = ""
Diff.Text = ""
Diff2.Text = ""

End Sub

Private Sub Command1_Click()
PlagTest.Show
End Sub



Private Sub Diff_Change()
```

```
Call HScroll_Scroll
End Sub

Private Sub Exit_Click()
On Error Resume Next
If MsgBox("Are you sure you want to quit?", vbQuestion + vbYesNo) = vbYes Then
Unload Me
End If
End Sub

Private Sub HScroll_Change()
Call HScroll_Scroll
End Sub

Private Sub HScroll_Scroll()
Dim Difstats As Integer
'************************************************************************

'************************************************************************
Text1 = HScroll.Value
If Val(Diff) > Val(Text1) Then
  Diff2 = "Suspected as Plagiarized Work"
  Diff2.BackColor = vbRed
Else
  Diff2 = "Not Plagiarized Work"
  Diff2.BackColor = vbGreen
End If

'If Difstats < Val(Text1) Then
'Diff2.Text = "Not Plagiarized Work"
'Diff2.BackColor = vbGreen
'Else: Diff2.Text = "Plagiarized Work"
'Diff2.BackColor = vbRed
'End If
End Sub

Private Sub htile_Click()
Me.Arrange vbTileHorizontal
End Sub

Private Sub MDIForm_Load()
Call HScroll_Scroll
End Sub

Private Sub PT_Click()
about.Show
End Sub

Private Sub Quit_Click()
On Error Resume Next
If MsgBox("Are you sure you want to quit?", vbQuestion + vbYesNo) = vbYes Then
Unload Me
End If
End Sub

Private Sub Rec_Click()
STATUS_FORM.Show
End Sub


Private Sub vtile_Click()
Me.Arrange vbTileVertical
End Sub


Private Sub Words_Click()
PlagTest.Show
End Sub
```

frmSplash

```
Option Explicit

Private Sub Form_KeyPress(KeyAscii As Integer)
   Unload Me
   frmmain.Show
End Sub

Private Sub Form_Load()
   lblVersion.Caption = "Version " & App.Major & "." & App.Minor '& "." & App.Revision
   lblProductName.Caption = App.Title

End Sub

Private Sub Frame1_Click()
   Unload Me
   frmmain.Show
End Sub
```

PlagTest

```
Dim a(120) As String
Dim countClusterWord(120) As Integer
Dim ClusterWord(120) As String
Dim ch(2000) As Byte
Dim countch(2000) As Integer


'try for compare purpose
Dim b(120) As String
Dim countClusterWord1(120) As Integer
Dim ClusterWord1(120) As String
Dim ch1(2000) As Byte
Dim countch1(2000) As Integer

Private Sub ClearText1_Click()
InputText1.Text = ""
ListWords1.Clear
ListChars1.Clear
ListWords11.Clear
TotalChars1.Clear
InputText2.Text = ""
ListWords2.Clear
ListWords22.Clear
ListChars2.Clear
TotalChars2.Clear
Result.Clear
Status.Clear

End Sub

'Private Sub cmdCompare_Click()
'   If Not bIsCompared Then
'      MsgBox "Error is Building Arrays"
'   End If
'
'   'end try
'End Sub

Private Sub Compare_Click()
If InputText1 = "" Or InputText2 = "" Then
   MsgBox "You must specify a Master and Target File"
   Exit Sub
End If

'*************************************************************************

Sentence1 = InputText1
InputText1 = LCase(Sentence1)
Sentence1 = LCase(InputText1)

lok = -1
   ctrWord1 = 1
   'Tokenizer
   While (Not Len(Sentence1) = 0) And (Not lok = 0)
      lok = InStr(1, Sentence1, " ", vbTextCompare)
      If lok = 0 Then lok = Len(Sentence1)
      b(ctrWord1) = Mid(Sentence1, 1, lok - 1)

      'to remove Question mark (?) at the end of the word
      lokasiQmark = InStr(1, b(ctrWord1), "?", vbTextCompare)
      If lokasiQmark = Len(b(ctrWord1)) Then b(ctrWord1) = Left(b(ctrWord1), lokasiQmark - 1)

      'to remove full-stop (.) at the end of the word
      lokasiFSmark = InStr(1, b(ctrWord1), ".", vbTextCompare)
      If lokasiFSmark = Len(b(ctrWord1)) Then b(ctrWord1) = Left(b(ctrWord1), lokasiFSmark - 1)

      lokasiFSmark = InStr(1, b(ctrWord1), ",", vbTextCompare)
```

```
    If lokasiFSmark = Len(b(ctrWord1)) Then b(ctrWord1) = Left(b(ctrWord1), lokasiFSmark - 1)

    lokasiFSmark = InStr(1, b(ctrWord1), "!", vbTextCompare)
    If lokasiFSmark = Len(b(ctrWord1)) Then b(ctrWord1) = Left(b(ctrWord1), lokasiFSmark - 1)

    ctrWord1 = ctrWord1 + 1
    Sentence1 = Mid(Sentence1, lok + 1)

  Wend
  'adjust the value of ctrWord1
  ctrWord1 = ctrWord1 - 1


  'Selection sort --> required for words clustering, see bellow
  For I = 1 To ctrWord1 - 1
    For j = I + 1 To ctrWord1
      If (StrComp(b(j), b(I), vbBinaryCompare) < 0) Then
        temp = b(j)
        b(j) = b(I)
        b(I) = temp
      End If
    Next j
  Next I


Sentence = InputText2
  'convert case for character
  InputText2 = LCase(Sentence)
  'convert case for word
  Sentence = LCase(InputText2)
  lok = -1
  ctrWord = 1
  'Tokenizer
  While (Not Len(Sentence) = 0) And (Not lok = 0)
    lok = InStr(1, Sentence, " ", vbTextCompare)
    If lok = 0 Then lok = Len(Sentence)
    a(ctrWord) = Mid(Sentence, 1, lok - 1)
    'to remove Question mark (?) at the end of the word
    lokasiQmark = InStr(1, a(ctrWord), "?", vbTextCompare)
    If lokasiQmark = Len(a(ctrWord)) Then a(ctrWord) = Left(a(ctrWord), lokasiQmark - 1)

    'to remove full-stop (.) at the ned of the word
    lokasiFSmark = InStr(1, a(ctrWord), ".", vbTextCompare)
    If lokasiFSmark = Len(a(ctrWord)) Then a(ctrWord) = Left(a(ctrWord), lokasiFSmark - 1)

    lokasiFSmark = InStr(1, a(ctrWord), ",", vbTextCompare)
    If lokasiFSmark = Len(a(ctrWord)) Then a(ctrWord) = Left(a(ctrWord), lokasiFSmark - 1)

    lokasiFSmark = InStr(1, a(ctrWord), "!", vbTextCompare)
    If lokasiFSmark = Len(a(ctrWord)) Then a(ctrWord) = Left(a(ctrWord), lokasiFSmark - 1)

    ctrWord = ctrWord + 1
    Sentence = Mid(Sentence, lok + 1)
  Wend
  'adjust the value of ctrWord
  ctrWord = ctrWord - 1
'Selection sort --> required for words clustering, see bellow
  For I = 1 To ctrWord - 1
    For j = I + 1 To ctrWord
      If (StrComp(a(j), a(I), vbBinaryCompare) < 0) Then
        temp = a(j)
        a(j) = a(I)
        a(I) = temp
      End If
    Next j
  Next I

'*************************************************************************
If ctrWord1 < ctrWord Then
diffcount1 = (ctrWord - ctrWord1)
```

```
Else: diffcount1 = (ctrWord1 - ctrWord)
End If
Dim diff1 As Integer
diff1 = 100 - ((diffcount1 / (ctrWord1) * 100))
Result.AddItem " "
Result.AddItem " Total similar words is " & diff1 & "%"

'*************************************************************************
'Word clustering, similar words are group together
   TmpString = b(1)
   ctrCluster1 = 1
   countClusterWord1(ctrCluster1) = 1

   ClusterWord1(ctrCluster1) = TmpString
   For I = 2 To ctrWord1
      If StrComp(TmpString, b(I), vbBinaryCompare) = 0 Then
         countClusterWord1(ctrCluster1) = countClusterWord1(ctrCluster1) + 1
      Else
         ctrCluster1 = ctrCluster1 + 1
         TmpString = b(I)
         ClusterWord1(ctrCluster1) = TmpString
         countClusterWord1(ctrCluster1) = 1
      End If
   Next I

jumlahPktn1 = 0 ' to check the sum of clustered words statistic
   'Populate the clustered words in listbox
   For I = 1 To ctrCluster1
      TmpString = ClusterWord1(I) & Space(20)
      jumlahPktn1 = jumlahPktn1 + countClusterWord1(I)
   Next I

'Word clustering, similar words are group together
   TmpString = a(1)
   ctrCluster = 1
   countClusterWord(ctrCluster) = 1

   ClusterWord(ctrCluster) = TmpString
   For I = 2 To ctrWord
      If StrComp(TmpString, a(I), vbBinaryCompare) = 0 Then
         countClusterWord(ctrCluster) = countClusterWord(ctrCluster) + 1

      Else
         ctrCluster = ctrCluster + 1
         TmpString = a(I)
         ClusterWord(ctrCluster) = TmpString
         countClusterWord(ctrCluster) = 1

      End If
   Next I

jumlahPktn = 0
   'to check the sum of clustered words statistic
   'Populate the clustered words in listbox
   For I = 1 To ctrCluster

      TmpString = ClusterWord(I) & Space(20)
      jumlahPktn = jumlahPktn + countClusterWord(I)
   Next I
'*************************************************************************
'If countClusterWord1(I) < countClusterWord(I) Then
'diffcount2 = (countClusterWord(I) - countClusterWord1(I))
'Else: diffcount2 = (countClusterWord1(I) - countClusterWord(I))
'End If
'Dim Diff2 As Integer
'Diff2 = 100 - ((diffcount2 / ((countClusterWord1(I) + countClusterWord(I)) / 2) * 100))
'Result.AddItem " "
'Result.AddItem " Total similar words Cluster is " & Diff2 & "%"

If ctrCluster1 < ctrCluster Then
```

```
diffcount2 = (ctrCluster - ctrCluster1)
Else: diffcount2 = (ctrCluster1 - ctrCluster)
End If
Dim Diff2 As Integer
Diff2 = 100 - ((diffcount2 / (ctrCluster1) * 100))
Result.AddItem " "
Result.AddItem " Total similar words Cluster is " & Diff2 & "%"

'*****************************************************************************
Sentence1 = InputText1
   ctrCh1 = Len(Sentence1)

For I = 1 To ctrCh1
      ch(I) = Asc(Mid(Sentence1, I, 1))
   Next I
   'sorting characters
   For I = 1 To ctrCh1 - 1
      For j = I + 1 To ctrCh1
         If ch(j) < ch(I) Then
            temp = ch(j)
            ch(j) = ch(I)
            ch(I) = temp
         End If
      Next j

Next I

Sentence = InputText2
   ctrCh = Len(Sentence)

For I = 1 To ctrCh
      ch(I) = Asc(Mid(Sentence, I, 1))

   Next I
   'sorting characters
   For I = 1 To ctrCh - 1
      For j = I + 1 To ctrCh
         If ch(j) < ch(I) Then
            temp = ch(j)
            ch(j) = ch(I)
            ch(I) = temp
         End If
      Next j
   Next I
'*****************************************************************************
If ctrCh1 < ctrCh Then
diffcount3 = (ctrCh - ctrCh1)
Else: diffcount3 = (ctrCh1 - ctrCh)
End If
Dim diff3 As Integer
diff3 = 100 - ((diffcount3 / (ctrCh1) * 100))
Result.AddItem " "
Result.AddItem " Total similar characters is " & diff3 & "%"

'*****************************************************************************

tmpCh = ch(1)
   ctrClusterCh1 = 1
   ch(ctrClusterCh1) = tmpCh
   countch1(ctrClusterCh1) = 1
   For I = 2 To ctrCh1
      If ch(I) = tmpCh Then
         countch1(ctrClusterCh1) = countch1(ctrClusterCh1) + 1
      Else
         tmpCh = ch(I)
         ctrClusterCh1 = ctrClusterCh1 + 1
         countch1(ctrClusterCh1) = 1
         ch(ctrClusterCh1) = tmpCh
      End If
   Next I
```

```
tmpCh = ch(1)
   ctrClusterCh = 1
   ch(ctrClusterCh) = tmpCh
   countch(ctrClusterCh) = 1
   For I = 2 To ctrCh
      If ch(I) = tmpCh Then
         countch(ctrClusterCh) = countch(ctrClusterCh) + 1
      Else
         tmpCh = ch(I)
         ctrClusterCh = ctrClusterCh + 1
         countch(ctrClusterCh) = 1
         ch(ctrClusterCh) = tmpCh
      End If
   Next I

'*****************************************************************************
'If ctrClusterCh1 < ctrClusterCh Then
'diffcount4 = (ctrClusterCh - ctrClusterCh1)
'Else: diffcount4 = (ctrClusterCh1 - ctrClusterCh)
'End If
'Dim diff4 As Integer
'diff4 = 100 - ((diffcount4 / ctrClusterCh1) * 100)
'Result.AddItem "Total similar characters Clusters is " & diff4 & "%"

' characters only have 26. no need to count the diff
'*****************************************************************************
If diff1 >= 50 And Diff2 >= 50 And diff3 >= 50 Then
Status.AddItem "                    "
Status.AddItem " Suspected as Plagiarized Work"
Else: Status.AddItem "                    "
Status.AddItem " Not Plagiarized Work"
End If
End Sub

Private Sub Exit_Click()
'Unload Me
On Error Resume Next
If MsgBox("Are you sure you want to quit?", vbQuestion + vbYesNo) = vbYes Then
Unload Me
End If

End Sub

Private Sub GoCode_Click()
frmmain.Show
End Sub

Private Sub lines_Click()
frmmain.Show
End Sub

Private Sub mnuFileOpen_Click()
Dim WhatFile As String
CommonDialog1.Filter = "Text Files (*.TXT)|*.TXT|All_Files (*.*)|*.*"
'CommonDialog1.Filter = "All Files (*.*)|*.*|Text Files (*.TXT)|*.TXT|Batch Files(*.BAT)|*.BAT|Executable Files(*.EXE)|*.EXE"
CommonDialog1.FilterIndex = 1

CommonDialog1.ShowOpen
WhatFile = CommonDialog1.FileName

If MsgBox("<" & WhatFile & ">", vbQuestion + vbYesNo) = vbYes Then
Load Me
End If

Open WhatFile For Input As #1
Input #1, loadfile1
InputText1.Text = loadfile1

End Sub
```

```
Private Sub mnuFileOpen2_Click()

Dim WhatFile1 As String
CommonDialog2.Filter = "Text Files (*.TXT)|*.TXT|All_Files (*.*)|*.*"
CommonDialog2.FilterIndex = 1

CommonDialog2.ShowOpen
WhatFile1 = CommonDialog2.FileName

If MsgBox("<" & WhatFile1 & ">", vbQuestion + vbYesNo) = vbYes Then
Load Me
End If

Open WhatFile1 For Input As #2
Input #2, MyString2
InputText2 = MyString2

End Sub

Private Sub PT_Click()
about.Show
End Sub

Private Sub Quit_Click()
On Error Resume Next
If MsgBox("Are you sure you want to quit?", vbQuestion + vbYesNo) = vbYes Then
Unload Me
End If
End Sub

Private Sub Rec_Click()
STATUS_FORM.Show
End Sub

Private Sub ShowStatistic_Click()

If InputText1 = "" Or InputText2 = "" Then
   MsgBox "You must specify a Master and Target File"
   Exit Sub
End If

'ListChars1.Clear
'ListWords11.Clear
'TotalChars1.Clear
'If InpuText1 = "Master File" Or InputText2 = "Target File" Then
'   MsgBox "You must specify a Master and Compare File"
'Exit Sub
'End If
'input text 1 codes
Sentence1 = InputText1
'change the case for ctrCluster1 character
InputText1 = LCase(Sentence1)
Sentence1 = LCase(InputText1)
If InputText1 = "" Then
   ListChars1.Clear
   ListWords11.Clear
   TotalChars1.Clear
End If

   lok = -1
   ctrWord1 = 1
   'Tokenizer
   While (Not Len(Sentence1) = 0) And (Not lok = 0)
      lok = InStr(1, Sentence1, " ", vbTextCompare)
      If lok = 0 Then lok = Len(Sentence1)
      b(ctrWord1) = Mid(Sentence1, 1, lok - 1)

      'to remove Question mark (?) at the end of the word
      lokasiQmark = InStr(1, b(ctrWord1), "?", vbTextCompare)
      If lokasiQmark = Len(b(ctrWord1)) Then b(ctrWord1) = Left(b(ctrWord1), lokasiQmark - 1)
```

```
    'to remove full-stop (.) at the end of the word
    lokasiFSmark = InStr(1, b(ctrWord1), ".", vbTextCompare)
    If lokasiFSmark = Len(b(ctrWord1)) Then b(ctrWord1) = Left(b(ctrWord1), lokasiFSmark - 1)

    lokasiFSmark = InStr(1, b(ctrWord1), ",", vbTextCompare)
    If lokasiFSmark = Len(b(ctrWord1)) Then b(ctrWord1) = Left(b(ctrWord1), lokasiFSmark - 1)

    lokasiFSmark = InStr(1, b(ctrWord1), "!", vbTextCompare)
    If lokasiFSmark = Len(b(ctrWord1)) Then b(ctrWord1) = Left(b(ctrWord1), lokasiFSmark - 1)

    ctrWord1 = ctrWord1 + 1
    Sentence1 = Mid(Sentence1, lok + 1)

Wend
'adjust the value of ctrWord1
ctrWord1 = ctrWord1 - 1


'Selection sort --> required for words clustering, see bellow
For I = 1 To ctrWord1 - 1
    For j = I + 1 To ctrWord1
        If (StrComp(b(j), b(I), vbBinaryCompare) < 0) Then
            temp = b(j)
            b(j) = b(I)
            b(I) = temp
        End If
    Next j
Next I

'Populate the sorted words in listbox
ListWords1.Clear

For I = 1 To ctrWord1
    ListWords1.AddItem b(I)
Next I



'Word clustering, similar words are group together
TmpString = b(1)
ctrCluster1 = 1
countClusterWord1(ctrCluster1) = 1

ClusterWord1(ctrCluster1) = TmpString
For I = 2 To ctrWord1
    If StrComp(TmpString, b(I), vbBinaryCompare) = 0 Then
        countClusterWord1(ctrCluster1) = countClusterWord1(ctrCluster1) + 1
    Else
        ctrCluster1 = ctrCluster1 + 1
        TmpString = b(I)
        ClusterWord1(ctrCluster1) = TmpString
        countClusterWord1(ctrCluster1) = 1
    End If
Next I

ListWords11.AddItem "**********************************"
ListWords11.AddItem "Total Words Cluster = " & ctrCluster1
ListWords11.AddItem "**********************************"

jumlahPktn1 = 0 ' to check the sum of clustered words statistic
'Populate the clustered words in listbox
For I = 1 To ctrCluster1
    TmpString = ClusterWord1(I) & Space(20)
    ListWords11.AddItem Left(TmpString, 15) & "==> " & countClusterWord1(I)
    jumlahPktn1 = jumlahPktn1 + countClusterWord1(I)
Next I
ListWords11.AddItem "=================================="
'display the sum of clustered words -> only for checking
ListWords11.AddItem "Total Words   ==> " & jumlahPktn1
```

```
ListWords11.AddItem "=================================="

'Counting characters
Sentence1 = InputText1
ctrCh1 = Len(Sentence1)

TotalChars1.AddItem " Total Words                    ==> " & ctrWord1
TotalChars1.AddItem " Total Words Clusters           ==> " & ctrCluster1
TotalChars1.AddItem " Total characters include spaces  ==> " & ctrCh1
'TotalChars1.AddItem " Total characters include spaces ==> " & ctrCh1 & vbCr
'TotalChars1.AddItem " Total characters include spaces ==> " & jumlahCh


For I = 1 To ctrCh1
   ch(I) = Asc(Mid(Sentence1, I, 1))
Next I
'sorting characters
For I = 1 To ctrCh1 - 1
   For j = I + 1 To ctrCh1
      If ch(j) < ch(I) Then
         temp = ch(j)
         ch(j) = ch(I)
         ch(I) = temp
      End If
   Next j
Next I

tmpCh = ch(1)
ctrClusterCh1 = 1
ch(ctrClusterCh1) = tmpCh
countch1(ctrClusterCh1) = 1
For I = 2 To ctrCh1
   If ch(I) = tmpCh Then
      countch1(ctrClusterCh1) = countch1(ctrClusterCh1) + 1
   Else
      tmpCh = ch(I)
      ctrClusterCh1 = ctrClusterCh1 + 1
      countch1(ctrClusterCh1) = 1
      ch(ctrClusterCh1) = tmpCh
   End If
Next I

'Populate the clustered characters
jumlahCh1 = 1
ListChars1.Clear
ListChars1.AddItem "**********************************"
ListChars1.AddItem "Total Char Cluster = " & ctrClusterCh1
ListChars1.AddItem "**********************************"
TotalChars1.AddItem " Total Char Cluster             ==> " & ctrClusterCh1
For I = 1 To ctrClusterCh1
   ListChars1.AddItem Left(Chr(ch(I)) & Space(5), 2) & "--> " & countch1(I)
   jumlahCh1 = jumlahCh1 + countch1(I)
Next I
ListChars1.AddItem "=================================="
'ListChars1.AddItem "Total Chars ==> " & jumlahCh1
ListChars1.AddItem "Total Chars ==> " & ctrCh1
ListChars1.AddItem "=================================="

'##############################################################################text 2
'
'  ListChars2.Clear
'  ListWords22.Clear
'  TotalChars2.Clear

Sentence = InputText2
'convert case for character
InputText2 = LCase(Sentence)
'convert case for word
Sentence = LCase(InputText2)
iok = -1
```

```
ctrWord = 1
'Tokenizer
While (Not Len(Sentence) = 0) And (Not lok = 0)
   lok = InStr(1, Sentence, " ", vbTextCompare)
   If lok = 0 Then lok = Len(Sentence)
   a(ctrWord) = Mid(Sentence, 1, lok - 1)
   'to remove Question mark (?) at the end of the word
   lokasiQmark = InStr(1, a(ctrWord), "?", vbTextCompare)
   If lokasiQmark = Len(a(ctrWord)) Then a(ctrWord) = Left(a(ctrWord), lokasiQmark - 1)

   'to remove full-stop (.) at the ned of the word
   lokasiFSmark = InStr(1, a(ctrWord), ".", vbTextCompare)
   If lokasiFSmark = Len(a(ctrWord)) Then a(ctrWord) = Left(a(ctrWord), lokasiFSmark - 1)

   lokasiFSmark = InStr(1, a(ctrWord), ",", vbTextCompare)
   If lokasiFSmark = Len(a(ctrWord)) Then a(ctrWord) = Left(a(ctrWord), lokasiFSmark - 1)

   lokasiFSmark = InStr(1, a(ctrWord), "!", vbTextCompare)
   If lokasiFSmark = Len(a(ctrWord)) Then a(ctrWord) = Left(a(ctrWord), lokasiFSmark - 1)

   ctrWord = ctrWord + 1
   Sentence = Mid(Sentence, lok + 1)
Wend
'adjust the value of ctrWord
ctrWord = ctrWord - 1


'Selection sort --> required for words clustering, see bellow
For I = 1 To ctrWord - 1
   For j = I + 1 To ctrWord
      If (StrComp(a(j), a(I), vbBinaryCompare) < 0) Then
         temp = a(j)
         a(j) = a(I)
         a(I) = temp
      End If
   Next j
Next I

'Populate the sorted words in listbox
ListWords2.Clear

For I = 1 To ctrWord
   ListWords2.AddItem a(I)
Next I

'Word clustering, similar words are group together
TmpString = a(1)
ctrCluster = 1
countClusterWord(ctrCluster) = 1

ClusterWord(ctrCluster) = TmpString
For I = 2 To ctrWord
   If StrComp(TmpString, a(I), vbBinaryCompare) = 0 Then
      countClusterWord(ctrCluster) = countClusterWord(ctrCluster) + 1

   Else
      ctrCluster = ctrCluster + 1
      TmpString = a(I)
      ClusterWord(ctrCluster) = TmpString
      countClusterWord(ctrCluster) = 1

   End If
Next I

ListWords22.AddItem "*************************************************************"
ListWords22.AddItem "Total Words Clusters = " & ctrCluster
ListWords22.AddItem "*************************************************************"

jumlahPktn = 0
'to check the sum of clustered words statistic
```

```
'Populate the clustered words in listbox
For I = 1 To ctrCluster

   TmpString = ClusterWord(I) & Space(20)
   ListWords22.AddItem Left(TmpString, 15) & "==> " & countClusterWord(I)
   jumlahPktn = jumlahPktn + countClusterWord(I)
Next I
ListWords22.AddItem "=============================================="
'display the sum of clustered words -> only for checking
ListWords22.AddItem "Total Words   ==> " & jumlahPktn
ListWords22.AddItem "=============================================="

'Counting characters
Sentence = InputText2
ctrCh = Len(Sentence)

TotalChars2.AddItem " Total Words                ==> " & ctrWord
TotalChars2.AddItem " Total Words Clusters       ==> " & ctrCluster
TotalChars2.AddItem " Total characters include spaces  ==> " & ctrCh
'TotalChars2.AddItem " Total characters include spaces ==> " & ctrCh & vbCr
'TotalChars2.AddItem " Total characters include spaces ==> " & jumlahCh & vbCr

For I = 1 To ctrCh
   ch(I) = Asc(Mid(Sentence, I, 1))

Next I
'sorting characters
For I = 1 To ctrCh - 1
   For j = I + 1 To ctrCh
      If ch(j) < ch(I) Then
         temp = ch(j)
         ch(j) = ch(I)
         ch(I) = temp
      End If
   Next j
Next I

tmpCh = ch(1)
ctrClusterCh = 1
ch(ctrClusterCh) = tmpCh
countch(ctrClusterCh) = 1
For I = 2 To ctrCh
   If ch(I) = tmpCh Then
      countch(ctrClusterCh) = countch(ctrClusterCh) + 1
   Else
      tmpCh = ch(I)
      ctrClusterCh = ctrClusterCh + 1
      countch(ctrClusterCh) = 1
      ch(ctrClusterCh) = tmpCh
   End If
Next I

'Populate the clustered characters
jumlahCh = 1
ListChars2.Clear
ListChars2.AddItem "**********************************"
ListChars2.AddItem "Total Char Cluster = " & ctrClusterCh
ListChars2.AddItem "**********************************"
TotalChars2.AddItem " Total Char Cluster          ==> " & ctrClusterCh
For I = 1 To ctrClusterCh
   ListChars2.AddItem Left(Chr(ch(I)) & Space(5), 2) & "--> " & countch(I)
   jumlahCh = jumlahCh + countch(I)
Next I
ListChars2.AddItem "=================================="
ListChars2.AddItem "Total Chars ==> " & ctrCh
'ListChars2.AddItem "Total Chars ==> " & jumlahCh
ListChars2.AddItem "=================================="


End Sub
```

STATUS_FORM

```
Private Sub Form_Unload(Cancel As Integer)
 Screen.MousePointer = vbDefault
End Sub

Private Sub datPrimaryRS_Error(ByVal ErrorNumber As Long, Description As String, ByVal Scode As Long, ByVal Source As
String, ByVal HelpFile As String, ByVal HelpContext As Long, fCancelDisplay As Boolean)
 'This is where you would put error handling code
 'If you want to ignore errors, comment out the next line
 'If you want to trap them, add code here to handle them
 MsgBox "Data error event hit err:" & Description
End Sub

Private Sub datPrimaryRS_MoveComplete(ByVal adReason As ADODB.EventReasonEnum, ByVal pError As ADODB.Error,
adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
 'This will display the current record position for this recordset
 datPrimaryRS.Caption = "Record: " & CStr(datPrimaryRS.Recordset.AbsolutePosition)
End Sub

Private Sub datPrimaryRS_WillChangeRecord(ByVal adReason As ADODB.EventReasonEnum, ByVal cRecords As Long, adStatus
As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
 'This is where you put validation code
 'This event gets called when the following actions occur
 Dim bCancel As Boolean

 Select Case adReason
 Case adRsnAddNew
 Case adRsnClose
 Case adRsnDelete
 Case adRsnFirstChange
 Case adRsnMove
 Case adRsnRequery
 Case adRsnResynch
 Case adRsnUndoAddNew
 Case adRsnUndoDelete
 Case adRsnUndoUpdate
 Case adRsnUpdate
 End Select

 If bCancel Then adStatus = adStatusCancel
End Sub

Private Sub cmdAdd_Click()
 On Error GoTo AddErr
 datPrimaryRS.Recordset.AddNew

 Exit Sub
AddErr:
 MsgBox Err.Description
End Sub

Private Sub cmdDelete_Click()
 On Error GoTo DeleteErr
 With datPrimaryRS.Recordset
  .Delete
  .MoveNext
  If .EOF Then .MoveLast
 End With
 Exit Sub
DeleteErr:
 MsgBox Err.Description
End Sub

Private Sub cmdRefresh_Click()
 'This is only needed for multi user apps
 On Error GoTo RefreshErr
 datPrimaryRS.Refresh
```

```
  Exit Sub
RefreshErr:
  MsgBox Err.Description
End Sub

Private Sub cmdUpdate_Click()
  On Error GoTo UpdateErr

  datPrimaryRS.Recordset.UpdateBatch adAffectAll
  Exit Sub
UpdateErr:
  MsgBox Err.Description
End Sub

Private Sub cmdClose_Click()
  Unload Me
End Sub
```