Abstract

In the project, a redundant email detection system is designed and developed to help busy email users detect emails whose body is repeated almost wholly in other emails. The core detection algorithm of the system is implemented using string matching algorithm which is used in DNA sequence matching. However, these algorithms are not meant for normal text matching and hence they must be adapted and optimized for the use of the system. The system is an email agent which allows users to connect to their mail box, retrieve, read and most importantly, see which of their mails are redundant. Currently there are no such systems around. The system is developed using Java, Eclipse and SWT. The report will focus on implementation and optimizations.

Subject Descriptors:

E.1 DATA STRUCTURES

J.3 LIFE AND MEDICAL SCIENCES

H.3.1 Content Analysis and Indexing

H.3.5 On-line Information Services

H.2.8 Database Applications

I.5.3 Clustering

Keywords:

Data Structure and Algorithms, Computational Biology,

Information retrieval / processing, Data Mining / Pattern Recognition

Implementation Software and Hardware:

Intel Pentium 2.4 G, 256 M memory, Windows XP sp2, Java 1.5, Eclipse 3.2, JavaMail, SWT

Acknowledgement

I would like to thank my supervisor, Professor Wong Limsoon, for giving me invaluable guidelines and suggestions to the project as well as his encouragement.

List of Figures:

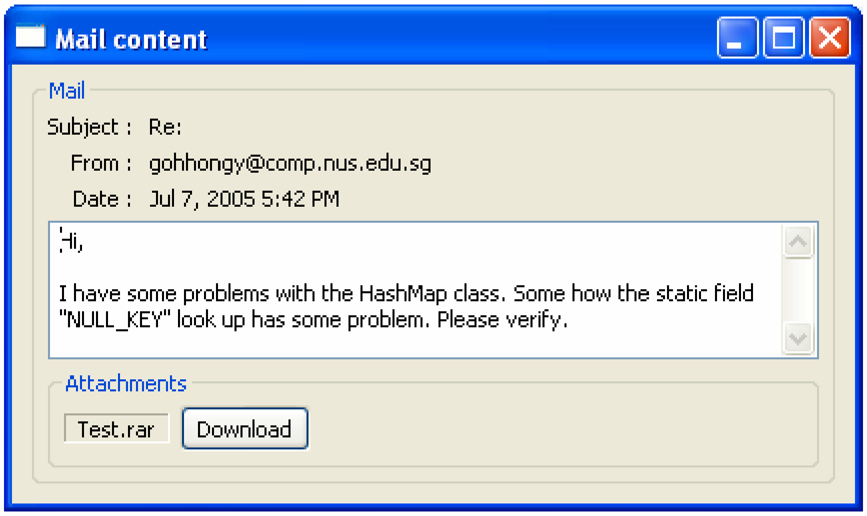
Table of Contents

Chapter 1 Introduction

1.1 Background and objective

In this age of electronic communication, our email boxes are constantly flooded with new emails everyday. However, our busy schedules make it difficult for us to read and reply to all of them. To compound the problem, there are a lot of emails that are redundant. Reading these mails is simply a waste of our precious time. In this project, I have developed an email agent which is able to detect and flag all the redundant emails, to help users save time.

Redundant mails are mails whose essential content are repeated or quoted in later mails, making it unnecessary to read these redundant mails. Here are some examples,

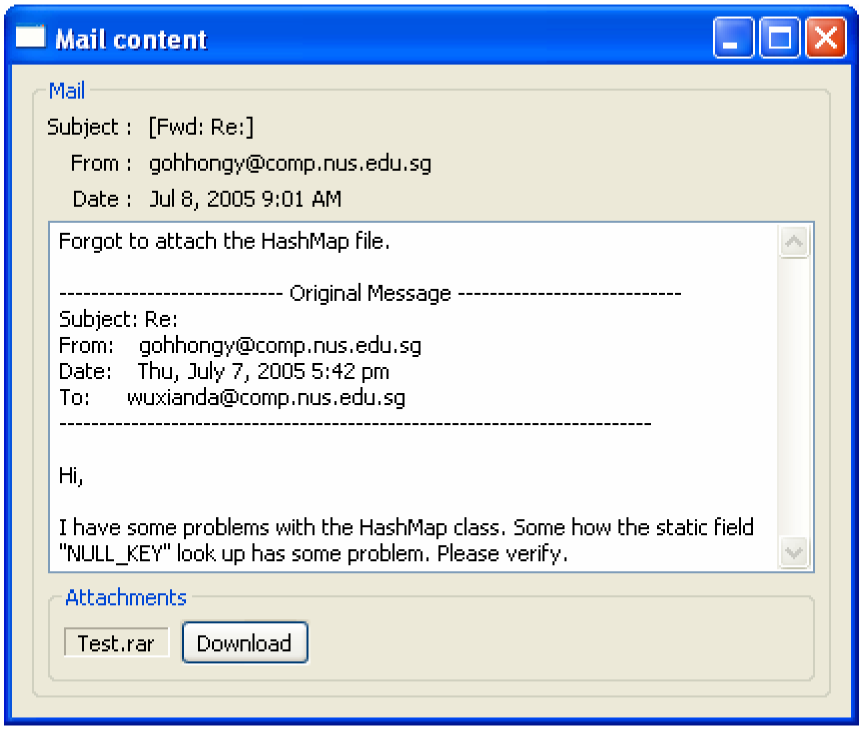


Subject : Re:
From : gohhongy@comp.nus.edu.sg
Date : Jul 7, 2005 5:42 PM

Hi,

I have some problems with the HashMap class. Some how the static field "NULL_KEY" look up has some problem. Please verify.

Fig 1.1.1 Mail 1 from gohhongy



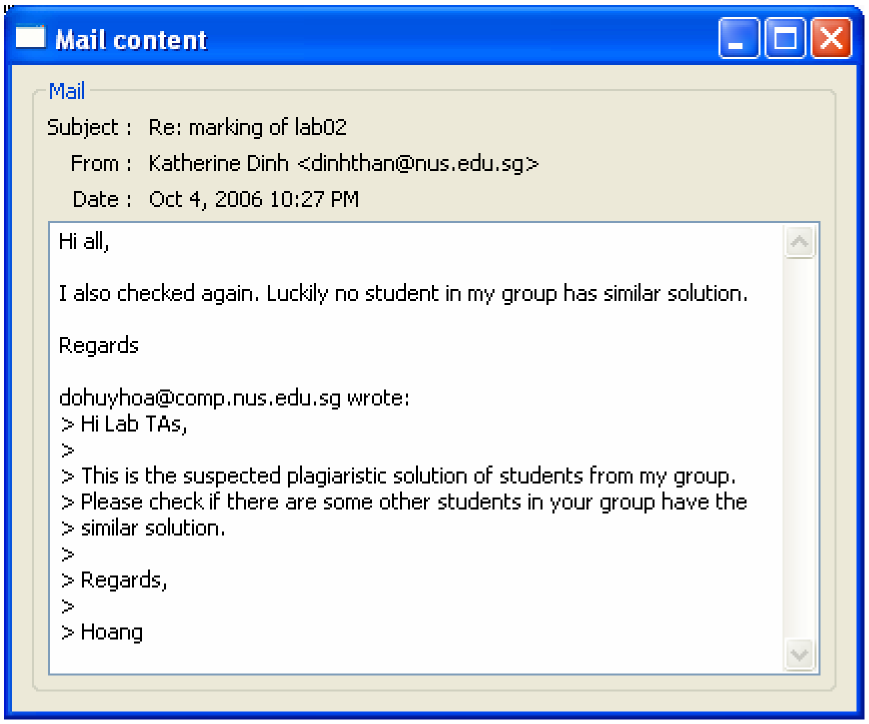
Subject : [Fwd: Re:]
From : gohhongy@comp.nus.edu.sg
Date : Jul 8, 2005 9:01 AM

```
Forgot to attach the HashMap file.

--------------------------- Original Message ---------------------------
Subject: Re:
From:    gohhongy@comp.nus.edu.sg
Date:    Thu, July 7, 2005 5:42 pm
To:      wuxianda@comp.nus.edu.sg
------------------------------------------------------------------------

Hi,

I have some problems with the HashMap class. Some how the static field
"NULL_KEY" look up has some problem. Please verify.
```

Fig 1.1.2 Mail 2 from gohhongy

These are two mails from my friend. The whole body of the first mail is appended at the end of the second mail. The attached file of the first mail is also same as the second mail. So the first mail is actually a redundant mail because of the second mail.

This is another example,



Mail content

Mail

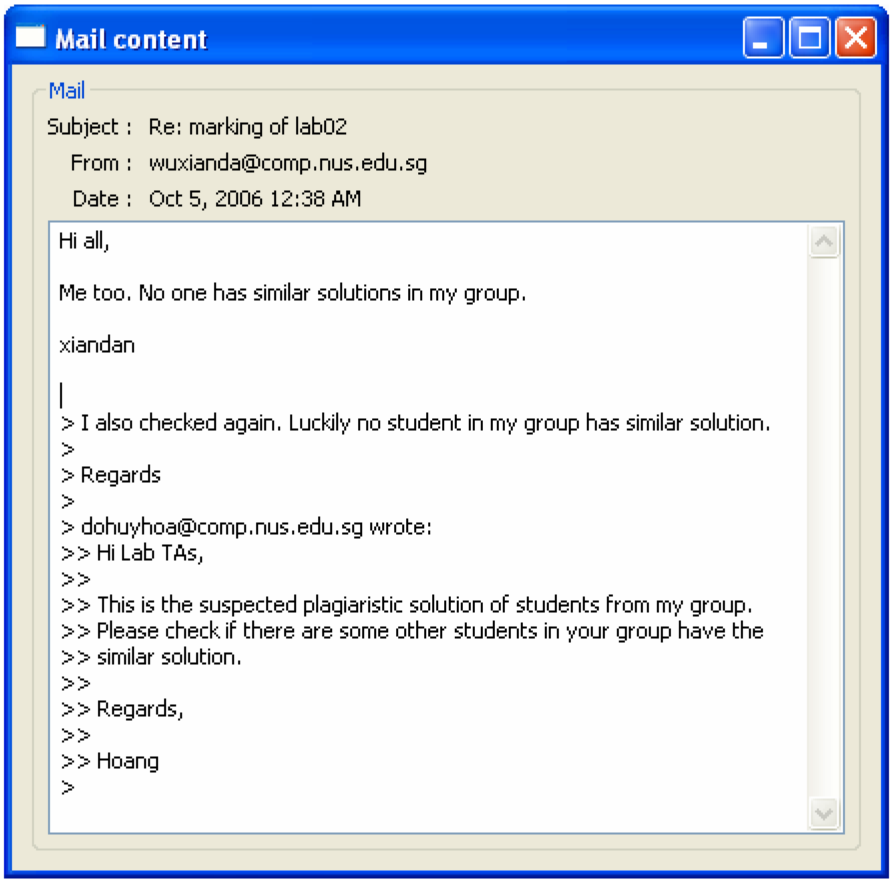
Subject : Re: marking of lab02

From : Katherine Dinh <dinhthan@nus.edu.sg>

Date : Oct 4, 2006 10:27 PM

```
Hi all,

I also checked again. Luckily no student in my group has similar solution.

Regards

dohuyhoa@comp.nus.edu.sg wrote:
> Hi Lab TAs,
>
> This is the suspected plagiaristic solution of students from my group.
> Please check if there are some other students in your group have the
> similar solution.
>
> Regards,
>
> Hoang
```

Fig 1.1.3 Mail 3 from Katherine



Mail content

Mail

Subject : Re: marking of lab02

From : wuxianda@comp.nus.edu.sg

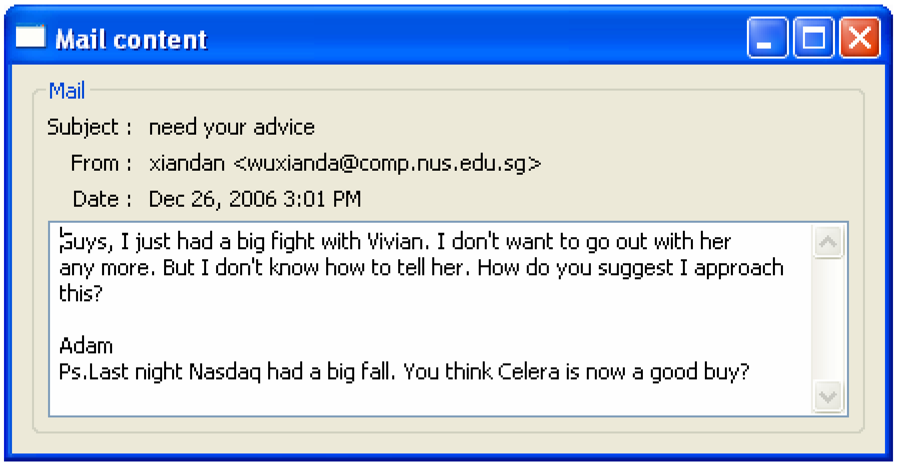
Date : Oct 5, 2006 12:38 AM

```
Hi all,

Me too. No one has similar solutions in my group.

xiandan

|
> I also checked again. Luckily no student in my group has similar solution.
>
> Regards
>
> dohuyhoa@comp.nus.edu.sg wrote:
>> Hi Lab TAs,
>>
>> This is the suspected plagiaristic solution of students from my group.
>> Please check if there are some other students in your group have the
>> similar solution.
>>
>> Regards,
>>
>> Hoang
>
```

Fig 1.1.4 Mail 4 from myself

The content of the first mail is totally repeated in the second mail, except "Hi all" which is not important. Hence the first mail is still considered redundant in respect to the second mail.

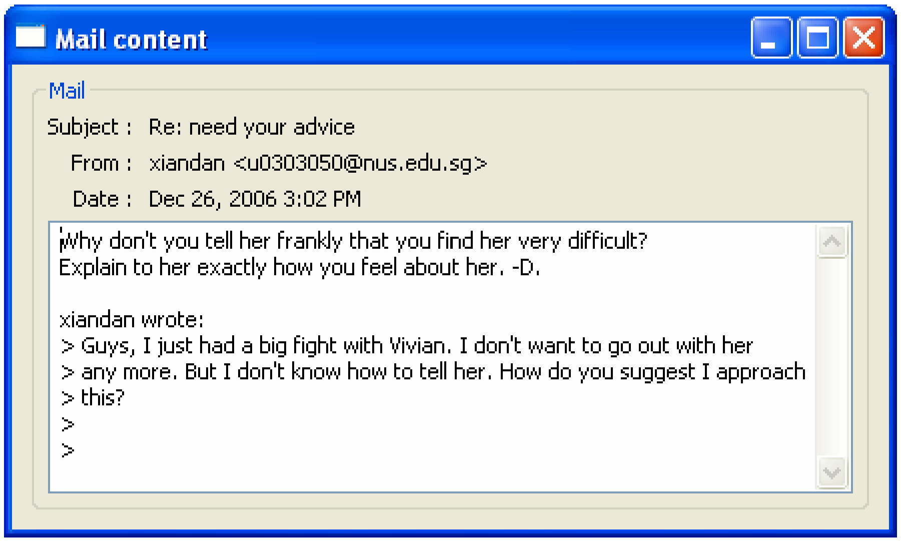
Here is another more complicated example mentioned by Kwok and Wong[1],



Subject : need your advice
From : xiandan <wuxianda@comp.nus.edu.sg>
Date : Dec 26, 2006 3:01 PM

```
Guys, I just had a big fight with Vivian. I don't want to go out with her
any more. But I don't know how to tell her. How do you suggest I approach
this?

Adam
Ps.Last night Nasdaq had a big fall. You think Celera is now a good buy?
```

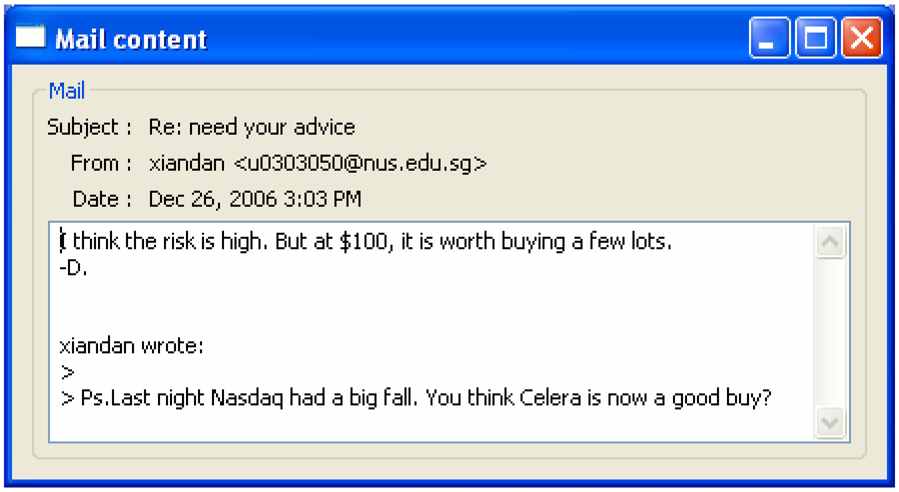
Fig 1.1.5 Mail 5



Subject : Re: need your advice
From : xiandan <u0303050@nus.edu.sg>
Date : Dec 26, 2006 3:02 PM

```
Why don't you tell her frankly that you find her very difficult?
Explain to her exactly how you feel about her. -D.

xiandan wrote:
> Guys, I just had a big fight with Vivian. I don't want to go out with her
> any more. But I don't know how to tell her. How do you suggest I approach
> this?
>
>
```

Fig 1.1.6 Mail 6



Subject : Re: need your advice
From : xiandan <u0303050@nus.edu.sg>
Date : Dec 26, 2006 3:03 PM

```
I think the risk is high. But at $100, it is worth buying a few lots.
-D.


xiandan wrote:
>
> Ps.Last night Nasdaq had a big fall. You think Celera is now a good buy?
```

Fig 1.1.7 Mail 7

Mail 5 is asking two questions. Then mail 6 and mail 7 each quoted one question from mail 5 and answered it. Hence mail 6 and mail 7 together make mail 5 redundant.

From the examples, we can see that there are actually 3 kinds of redundant mails.

- Those that are wholly repeated in another mail
- Those that are mostly repeated in another mail
- Those that are mostly repeated in combination of 2 or more than 2 mails

The objective of this project is to develop an efficient, reliable and user-friendly software program that can help busy professionals and managers, who receive hundreds of new mails everyday, to save time on reading redundant mails.

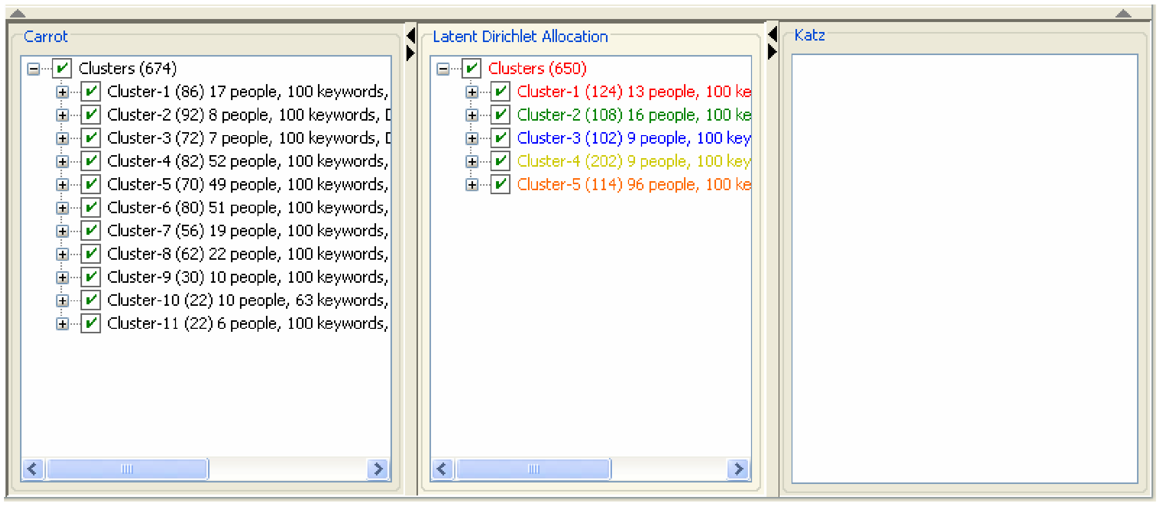
1.2 Report organization

The main part of this report consists of 6 chapters. Chapter 1 is the introduction of this project. It gives a rough description of what this project is about, as well as the background and objectives of this project. Chapter 2 is the evaluation of an existing solution that can be used to roughly detect redundant emails. Chapter describes the architecture of my system. Chapter 4 shows how the system is implemented. Chapter 5 describes benchmarking. The last chapter draws a short conclusion of this project and makes a summary.

Chapter 2 Evaluation of Existing Solution

As redundant emails and the mails that make them redundant are quite similar, one obvious solution is to cluster all the mails into different clusters. If all the mails on top of each cluster are the ones that are worth reading, then our job is done.

Open IRIS 2.3[12] is an open source information organization software. It is able to cluster emails using 3 different algorithms: Carrot, Latent Dirichlet Allocation and Katz. (Clustering function is missing in Open IRIS 3.0) Here are the results clustering my mails.



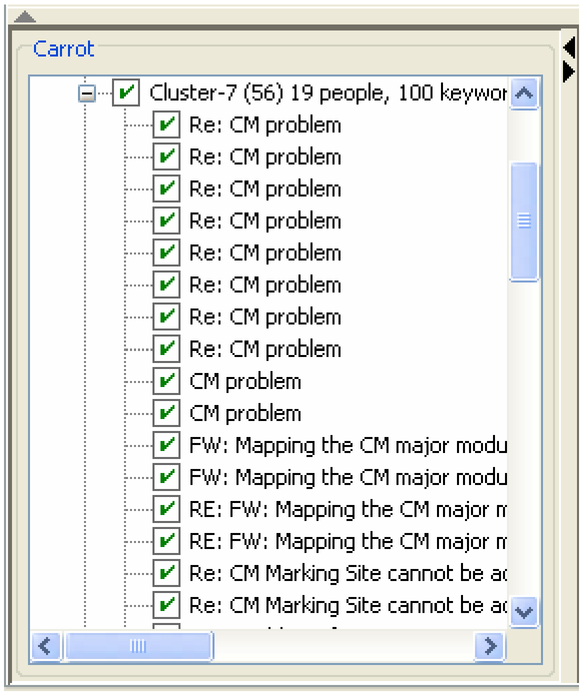
Clusters (674)
Cluster-1 (86) 17 people, 100 keywords,
Cluster-2 (92) 8 people, 100 keywords, D
Cluster-3 (72) 7 people, 100 keywords, D
Cluster-4 (82) 52 people, 100 keywords,
Cluster-5 (70) 49 people, 100 keywords,
Cluster-6 (80) 51 people, 100 keywords,
Cluster-7 (56) 19 people, 100 keywords,
Cluster-8 (62) 22 people, 100 keywords,
Cluster-9 (30) 10 people, 100 keywords,
Cluster-10 (22) 10 people, 63 keywords,
Cluster-11 (22) 6 people, 100 keywords,

Clusters (650)
Cluster-1 (124) 13 people, 100 ke
Cluster-2 (108) 16 people, 100 ke
Cluster-3 (102) 9 people, 100 key
Cluster-4 (202) 9 people, 100 key
Cluster-5 (114) 96 people, 100 ke

Fig 2.1 IRIS clustering results

The first 2 algorithm succeed in clustering a group of over 600 mails. Carrot algorithm cluster them into 11 clusters.

For example, this is one of the clusters,


Carrot

Cluster-7 (56) 19 people, 100 keywor
Re: CM problem
Re: CM problem
Re: CM problem
Re: CM problem
Re: CM problem
Re: CM problem
Re: CM problem
Re: CM problem
CM problem
CM problem
FW: Mapping the CM major modu
FW: Mapping the CM major modu
RE: FW: Mapping the CM major m
RE: FW: Mapping the CM major m
Re: CM Marking Site cannot be ac
Re: CM Marking Site cannot be ac

Fig 2.2 IRIS clustering results

This cluster shows the problems with using clustering method to identify redundant emails.

- All the mails that are related are clustered together. However, we have no clue that which ones are redundant and which ones are not. The ones on the top do not necessarily make the bottom top redundant. So we still need to go through all the mails one by one in order not to miss important messages.
- Mails that are not related but have identical or similar keywords are grouped into same cluster. In the above screen shot, there are mails “CM problem” and mails “FW: Mapping the CM major modu…”. These mails are clustered together because they have the same keyword “CM” in their title. However, “CM” in “CM problem” means “Course Maker”, which is a system to mark students’ lab. “CM” in “FW: Mapping the CM major modu…” means “Communication and Media” which is a major is our School of Computing. They are totally unrelated.

There is currently no existing solution to identify redundant mails. This is the motivation behind this project, to develop a system that is able to do the job.

Chapter 3 Architecture of the system

To achieve the objective of detecting and flagging redundant emails, the system will have to connect to the mail box and retrieve the mails to analysis. Next the system will run the mails through redundancy detection component, which is the core of this system. After that, the result will be communicated to the user through a user friendly interface. The overall architecture is shown below.

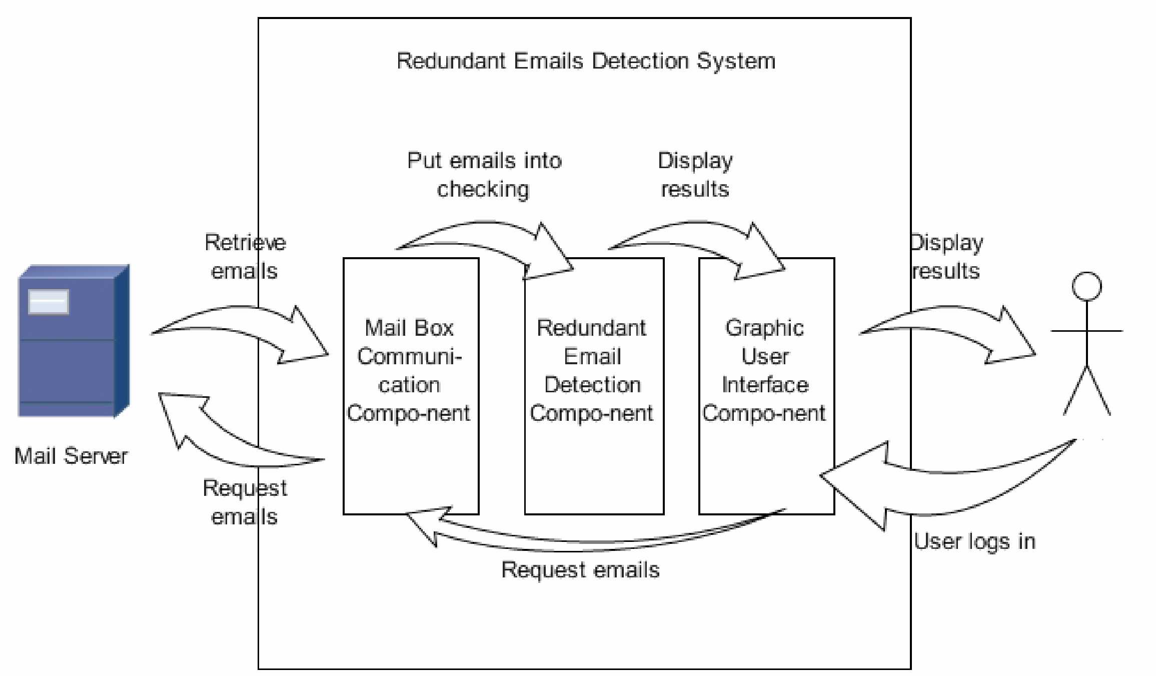

Redundant Emails Detection System
Put emails into checking
Display results
Retrieve emails
Display results
Mail Box Communi-cation Compo-nent
Redundant Email Detection Compo-nent
Graphic User Interface Compo-nent
Mail Server
Request emails
User logs in
Request emails

Fig 3.1 Overall architecture of system

The critical element in the system is the core redundancy detection component. It must be design to be accurate, sensitive and efficient in order to be useful. In next chapter I will devote most of the effort to describe the design and optimization of this component.

Chapter 4 Implementation

The main idea in the redundant mail detection process is to adapt biological sequence alignment algorithm to do email text matching. In this chapter, I will start with how the system connects to mailboxes. Then I will spend most of the chapter describing the main detection mechanism of the system. Lastly I will discuss the graphic user interface, which makes the system user friendly and practical.

4.1 Connecting to mail box

Before the mails can be retrieved and analyzed, we have to connect to a mailbox. JavaMail API[8] is a platform-independent tool to deal with emails and messages using Java. It is an external package that is not part of Java Standard Edition. My system uses JavaMail to connect to mailboxes. By using JavaMail, the communication with mail servers is encapsulated in one single class MailBoxOperation. Here is the interface,

```
public class MailBoxOperation {
    public void connect ( int mailBoxType, String host, String username, String password )
    public String[] getFolderList () {}
    public Folder[] getSubFolders ( String folderName ) {}
    private Folder[] getSubFolders ( Folder f ) {}
    public Message[] getMailEnvelopsFromFolder ( String mailFolderName ) {}
    public Message[] getMailEnvelopsFromFolder ( String mailFolderName, int str, int end ){}
    public ArrayList<Mail> getMailsFromFolder ( String mailFolderName ) {}
    public ArrayList<Mail> getMailsFromFolder ( String mailFolderName, int strMsgNum ){}
    public ArrayList<Mail> getMailObjectsForMsgs ( Message [] msgs, int strMsgNum ){}
    public static Mail readMail ( Message msg ) {}
    private static String getTextFromMultipart ( MimeMultipart mm ) {}
    public void copyMsgs ( ArrayList<Message> msgs, String from, String to ) {}
    public static MailAttachment getAttachment ( Message msg ) {}
    public void closeStore () {}
}
```

In this way, the detection algorithm can be shielded from talking directly to the mail server. The MailBoxOperation can also be easily replaced by another class which communicates with news group server, so the program can detect redundant news group post. New mail protocols can also be easily incorporated.

4.2 Detecting redundant mails

4.2.1 Exact match detection

To detect redundant emails, we have to check for overlapping of emails texts. If texts of email 1 are totally included or almost totally included in email 2, then email 1 is made redundant by email 2. So how are we going to detect these overlapping?

One of the simplest ways is to do brute force string matching. If email 1 is of length *n* and email 2 is of length *m*, to check whether email 1 is made redundant by email 2, we need *n*m* units of time. Suppose we have *q* emails to check, and the average length is of n, then we need *q*q*n*n* units of time to finish checking. This is unacceptable considering the normal inbox size is probably of 200 over mails.

However, there are faster ways of matching 2 strings, for example,

1	2	3	4	5	6	7	8	9	10	11	12
H	E	L	L	O	T	H	I	S	A	G	R
H	E	L	L	W							
					H	E	L	L	W		

When we first compare the last character ‘w’ of “hellw” with the fifth character ‘o’ of “hellothisagr”, they are not a match. We also observe that character ‘o’ does not appear in “hellw”. Hence it is no point to move “hellw” one place right and compare again just like the brutal matching algorithm. We might just shift “hellw” 5 place right and compare the ‘w’ with ‘a’ just as shown above in the 4th row. In this way we accelerate the matching by a factor as large as the length of the second string. This is the idea behind Boyer-Moore exact matching algorithm[4].

The Boyer-Moore algorithm preprocesses the 2 strings and builds 2 arrays which store the shift information. During comparison, when mismatch is encountered, these 2 arrays are consulted so that the program knows how many spaces to shift the second array to the right and start comparing again. The normal running time for Boyer-Moore algorithm is near *n/m* where *n* is the length of first string and *m* is that of the second string. This is a considerable improvement from the brutal force matching algorithm.

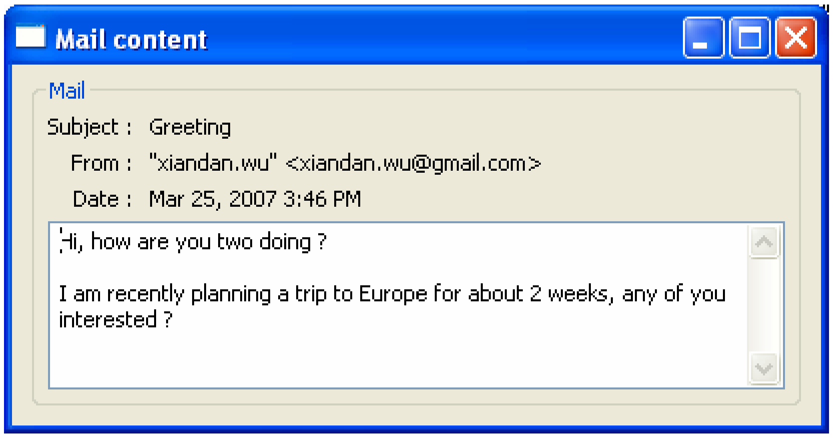
This first redundant email detection algorithm suggested by Kwok and Wong[1] is built using Boyer-Moore algorithm. I use a Boyer-Moore java implementation by Michael Lecuyer[10]. Here is the algorithm for the detection,

```
Loop through every mail i
    For every mail j after mail i
        Strip mail i and mail j off all spaces and punctuations
        Run mail i and mail j through Boyer-Moore algorithm
        If mail i is has an exact match in mail j
            Flag mail i as made redundant by mail j
        End if
    End for
End loop
```

This detection program successfully detects 115 out of 159 redundant mails which I have collected in 59 seconds. This is a 72% detection rate. It is a pretty good result considering the simplicity and speed of this algorithm. However, there is still room for improvement. I will start to describe a better algorithm which is at the heart of my final program in the next section.

4.2.2 Global alignment algorithm

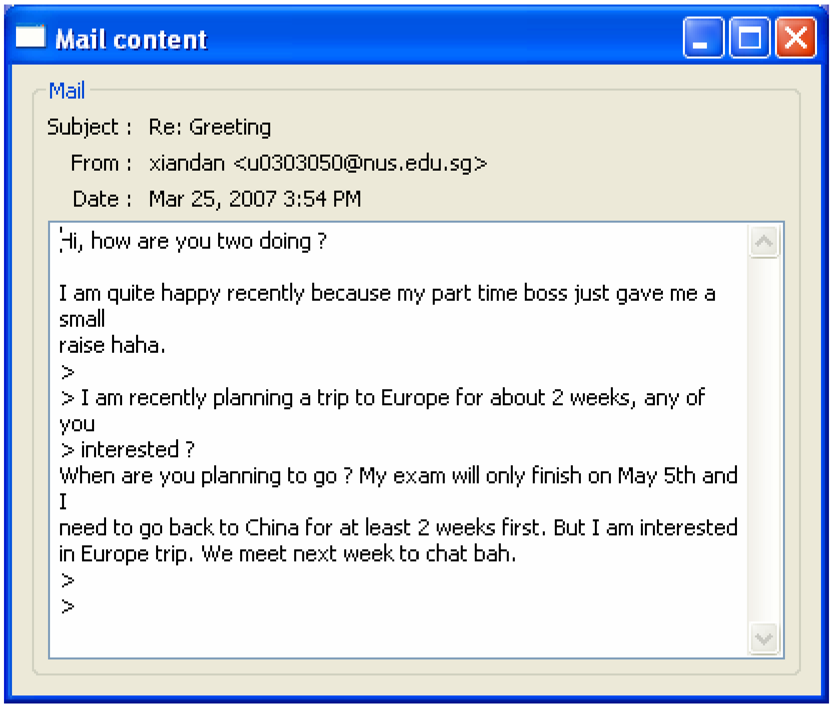
The detection algorithm we just examined is not able to detect redundant emails in the following case.



Subject : Greeting
From : "xiandan.wu" <xiandan.wu@gmail.com>
Date : Mar 25, 2007 3:46 PM

```
Hi, how are you two doing ?

I am recently planning a trip to Europe for about 2 weeks, any of you
interested ?
```

Fig 4.2.2.1 Mail 8



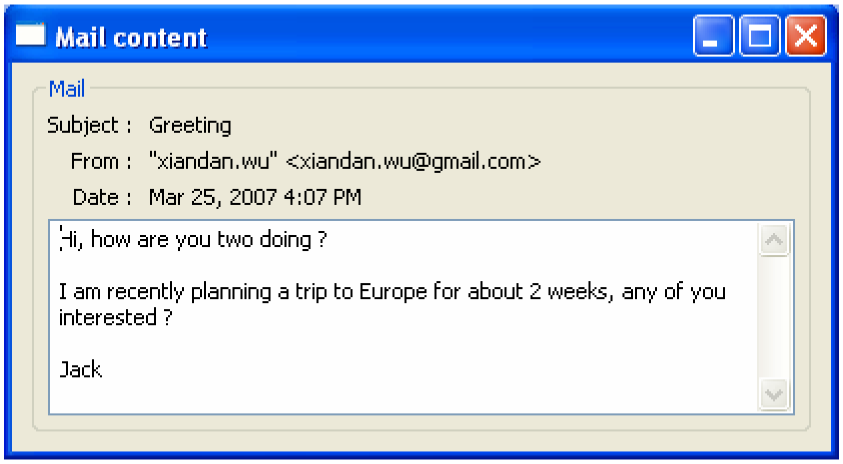
Subject : Re: Greeting
From : xiandan <u0303050@nus.edu.sg>
Date : Mar 25, 2007 3:54 PM

```
Hi, how are you two doing ?

I am quite happy recently because my part time boss just gave me a
small
raise haha.
>
> I am recently planning a trip to Europe for about 2 weeks, any of
you
> interested ?
When are you planning to go ? My exam will only finish on May 5th and
I
need to go back to China for at least 2 weeks first. But I am interested
in Europe trip. We meet next week to chat bah.
>
>
```

Fig 4.2.2.2 Mail 9

Mail 8 is totally included in mail 9 although interrupted by the answer into two parts. But mail 8 is still made redundant by mail 9. If the two questions of mail 8 are included and answered in two separate mails and we combined the 2 answer mails together, we will still have a similar case. However, the exact matching detection is unable to detect redundancy in such case because there is not exact contiguous match for mail 8 in mail 9.

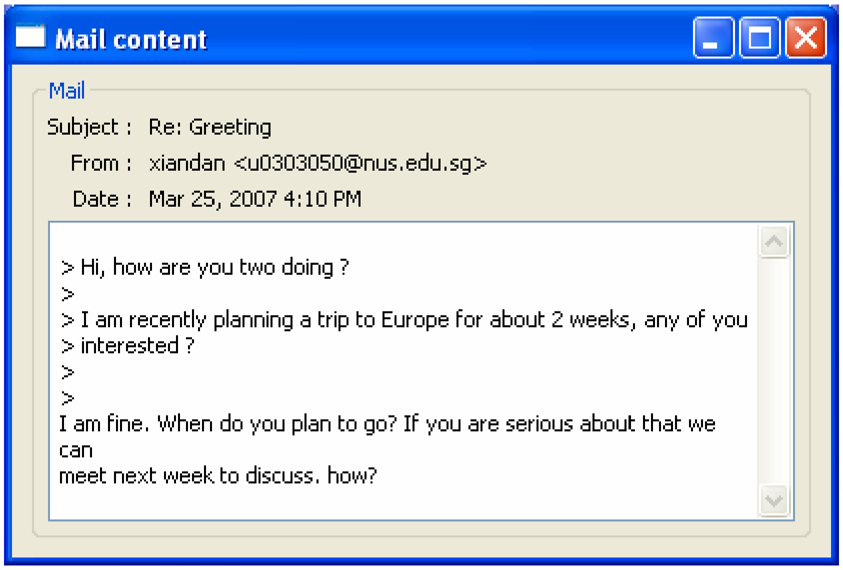
Sometimes, the redundant mail will not be totally included in another mail. Some trivial parts may be missing. For example,



Subject : Greeting
From : "xiandan.wu" <xiandan.wu@gmail.com>
Date : Mar 25, 2007 4:07 PM

```
Hi, how are you two doing ?

I am recently planning a trip to Europe for about 2 weeks, any of you
interested ?

Jack
```

Fig 4.2.2.3 Mail 10



Subject : Re: Greeting
From : xiandan <u0303050@nus.edu.sg>
Date : Mar 25, 2007 4:10 PM

```
> Hi, how are you two doing ?
>
> I am recently planning a trip to Europe for about 2 weeks, any of you
> interested ?
>
>
I am fine. When do you plan to go? If you are serious about that we
can
meet next week to discuss. how?
```

Fig 4.2.2.4 Mail 11

Mail 10 is included in mail 11 except the word “Jack”. The signature “Jack” is unimportant and hence mail 10 is still considered redundant in respect to mail 11. The exact match detection is also unable to detect redundancy in such case.

Hence we have to find a way to allow for some flexibility in the detection algorithm. In bioinformatics, a sequence alignment algorithm is used to identify similar regions in different sequences, in an effort to reveal the possible relationships between these sequences. Kwok and Wong suggest the method of using these algorithm to detect redundancy in emails.[1]

Needleman-Wunsch algorithm[7] is one such algorithm used to identify the best way to align sequences in terms of score. It uses dynamic programming to build up a two dimensional array representing the score during the matching and shifting between the two sequences. It then uses this array to trace back the best alignment found. I will illustrate the principal of this algorithm with an example. Suppose we want to align “ACBQ” with “ABQ”. We first need to decide what score to assign for every possible matching. This is a two dimensional array of the alphabets allowed. In this case, the alphabets are “ABCQ”. I will assume we use the following score matrix,

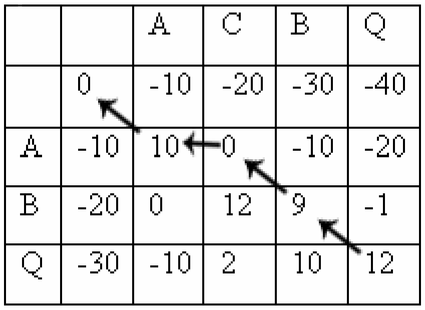
	A	B	C	Q
A	10	2	1	-7
B	2	9	2	-2
C	1	2	6	-3
Q	-7	-2	-3	3

When an ‘A’ is matched with another ‘A’, we increment the total score by 10. When a ‘C’ is matched with a ‘Q’, we decrement the total score by 3. Another parameter we need is the gap cost. This is the cost of matching a letter with a space. Let it be -10 in this case.

Then we can start the dynamic programming to find the maximum alignment score of “ACBQ” and “ABQ”. The follow table will be built.

		A	C	B	Q
	0	-10	-20	-30	-40
A	-10	10	0	-10	-20
B	-20	0	12	9	-1
Q	-30	-10	2	10	12

After the table is built, the algorithm traces back to top left hand corner from the bottom right hand corner. It finds how the score ‘9’ is obtained and an alignment will be traced out.



		A	C	B	Q
	0	-10	-20	-30	-40
A	-10	10	0	-10	-20
B	-20	0	12	9	-1
Q	-30	-10	2	10	12

Moving diagonally is a match while moving horizontally or vertically will represent a insertion of a space. The alignment found is

ACBQ

A_BQ

The allowance of spaces between characters while aligning sequences makes Needleman-Wunsch algorithm a good candidate for redundancy detection algorithm[1]. In the next section, I will describe how Needleman-Wunsch algorithm is adapted to the redundant email detection program.

4.2.3 Adaptation of Needleman-Wunsch algorithm

Several things need to be changed to adapt the standard Needleman-Wunsch algorithm to serve my redundant email detection.

Biology sequences use only limited alphabets. The set of alphabets need to be expanded be accommodate ‘A’-‘Z’, ‘a’-‘z’ and ‘0’-‘9’. New line symbols and punctuation symbols will be removed before the detection is run and hence they need not to be included.

The alphabet score matrix need to be changed. As my example score matrix in last section shows, the score between ‘A’ ‘A’ and that between ‘B’ ‘B’ are different. That serves the purpose of amino acid sequence matching but not emails text matching. The score matrix is changed in this way. Score between any two exact character matches will be 10 and any mismatch will be -10. The gap cost will be 8. These numbers are purposely chosen so that the algorithm would rather leave a gap than to allow a mismatch. In this way, interruption of the email texts will be allowed.

The original Needleman-Wunsch algorithm only returns the best alignment of the two strings. However, the detection program needs to find how much of one string is contained in another string. So the alignment returned by Needleman-Wunsch algorithm need to be processed. The class MatchedStringExtractor is design for this purpose, to extract degree of overlapping and the overlapping regions from the alignment string return by Needleman-Wunsch algorithm. Here is a example. Alignment strings returned from Needleman-Wunsch algorithm is shown below. [the previous sentence seems incomplete. Something is missing.]

HOWAREYOUJOHNIAMMISSINGYOU

HOWAREYOU_____IAMMISSINGYOU

The second string has the ‘_’ characters in it, indicating those corresponding characters in first string is not found in the second. The MatchedStringExtractor class will process and return the number of character that is exactly matched. In this case there are 22. This number can then be used to check what percentage of the mail body is included in another mail.

However, for short mail body, it may also be totally matched. For example,

HOWAREYOUJOINIAMMISSINGYOU

H________________I___________YOU

The second string “HIYOU” is totally matched with the first string. But it should not be considered as redundant. This case is solved by restricting the minimum number of contiguous matched characters to be considered valid. If the minimum number is set at 7, then all three matched region, ‘H’, ‘I’ and “YOU” falls below the minimum number and hence not counted as matched characters. The number of matched characters in this case will be 0 and “HIYOU” is not redundant.

Another example will show how this alignment algorithm solves the problem of some missing text in the redundant emails. Mail 10 in fig 3.2.2.3 is made redundant by mail 11 in fig 3.2.2.4, although mail 11 does not have the word “Jack” which appears in mail 10. Below is the matched string of these two emails using the alignment algorithm.

Mail 11:

hihowareyoutwodoingiamrecentlyplanningatriptoeuropeforabout2weeksanyofyouinterestediamfinewhendoyouplantogoifyo
uareseriousaboutthatwecanmeetnextweektodiscusshow

Mail 10:

hihowareyoutwodoingiamrecentlyplanningatriptoeuropeforabout2weeksanyofyouinterestedja-------------------------------------
------------c-------------k------------

In this alignment, mail 10 is almost totally matched in mail 11 but only with small bits of texts missing, indicated by the scattered ‘c’ and ‘k’. So the percentage of overlapping is high and we are still able to determine that mail 10 is made redundant by mail 11.

Another example will be mail 8 in figure 3.2.2.1 and mail 9 in figure 3.2.2.2. Mail 8 contains two questions and mail 9 inserts answers in between the questions. Using the alignment algorithm, the matched string obtain is as shown below.

Mail 9:

hihowareyoutwodoingiamquitehappyrecentlybecausemyparttimebossjustgavemeasmallraisehahaiamrecentlyplanningatript
oeuropeforabout2weeksanyofyouinterestedwhenareyouplanningtogomyexamwillonlyfinishonmay5thandineedtogobacktoch
inaforatleast2weeksfirstbutiaminterestedineuropetripwemeetnextweektochatbah

Mail 8:

hihowareyoutwodoingiam----------recently---------p---lanningatriptoeuropeforabout2w
eeksanyofyouinteres--t--e---d---------

We can see 3 regions of matched texts in mail 8 separated with mainly spaces. So we would still be able to conclude that mail 8 is made redundant by mail 9.

In order to determine whether a mail is redundant using the percentage of matched texts, a threshold is needed. This threshold is the redundancy threshold. If the percentage is over this threshold, a mail will be flagged as redundant. The suitable value of this threshold will be discussed in section 3.2.6.

Now a practical and effective way of detecting redundant emails is found. In the next section, I will describe the complete process of detecting all redundant emails in a mail box folder.

4.2.4 Complete detection process

After the core algorithm is found, the whole detection process can be drafted. First, all the mails will need to be retrieved from the mail server. The mail body texts need to be processed. All end line symbols and punctuation symbols will be removed. Next, sort the mail in ascending order of send dates. Only mails that are sent later can make mails that are sent earlier redundant, because it is impossible for a mail to repeat the texts of another mail that has not been sent. So we just need to check whether a mail is made redundant by any of the mails that are after it. Usually mails in mail boxes are sorted in received date order. But mails may not always be received in the same order as they are sent. So we need to sort them in the sent date order. Next, the program will go through every mail. For every mail, all mails behind are chained together and detection algorithm is run on the mail and the chained mails. The reason to chain up all the mails is to cater to the case that an email is answer is two or more mails and all these answer mails together make the original mail redundant. After the matched string is obtained, the percentage of repeat characters is calculated. If the percentage is over a preset threshold, the mail will be considered redundant. Then we have to find out which mail or mails actually contribute to the redundancy. This is quite a complicated and bug-prone process. When the mails are chained together, the length of each individual mail must be noted. After examining the matched strings, we will be able to locate the position of the matched regions. Using these location and individual mail length information, with painstaking carefulness, the mail or mails that contribute to the redundancy will be located. So this is a very basic description of the detection process.

This process will work theoretically. However, after I implemented it, it only works for mail boxes with very a small number of mails. The number of mails is around 20. For a mail box with more mails, the program takes very long to run and most of the time, memory is used up. Most normal mail boxes have more than 20 mails. So the program needs serious optimization to be practical and efficient. In next section, I will examine all the optimization techniques I have used to make the program efficient.

4.2.5 Optimizations

4.2.5.1 Sender grouping

Needleman-Wunsch algorithm is of $n*m$ complexity both in terms of time and space, where n and m are the lengths of the two strings respectively. This algorithm is hence much slower than Boyer-Moore algorithm. In the detection process, Needleman-Wunsch algorithm will have to be run once for every mail with a very long chain of mail texts. This takes significant time and memory, and the program described in last section is unable to finish for this reason. If we are able to make the chained mail texts shorter, it will take significantly less memory and prevents the out of memory error.

The reason to check a mail against all other mails chained together is to detect redundancy that is contributed together by multiple mails, such as case like mail 8 and mail 9 in figure 3.2.2.1 and figure 3.2.2.2. However, after some observations, I found that only the same sender will separate a mail into multiple parts and reply to them in different mails separately. This implies that it is not needed to chain all mails together. Instead, only mails from the same sender need to be chained up and check. This significantly shortens the length the chained texts, because a usual mail box will include mails from 10 to several hundred senders. The optimized detection algorithm for a mail is shown below.

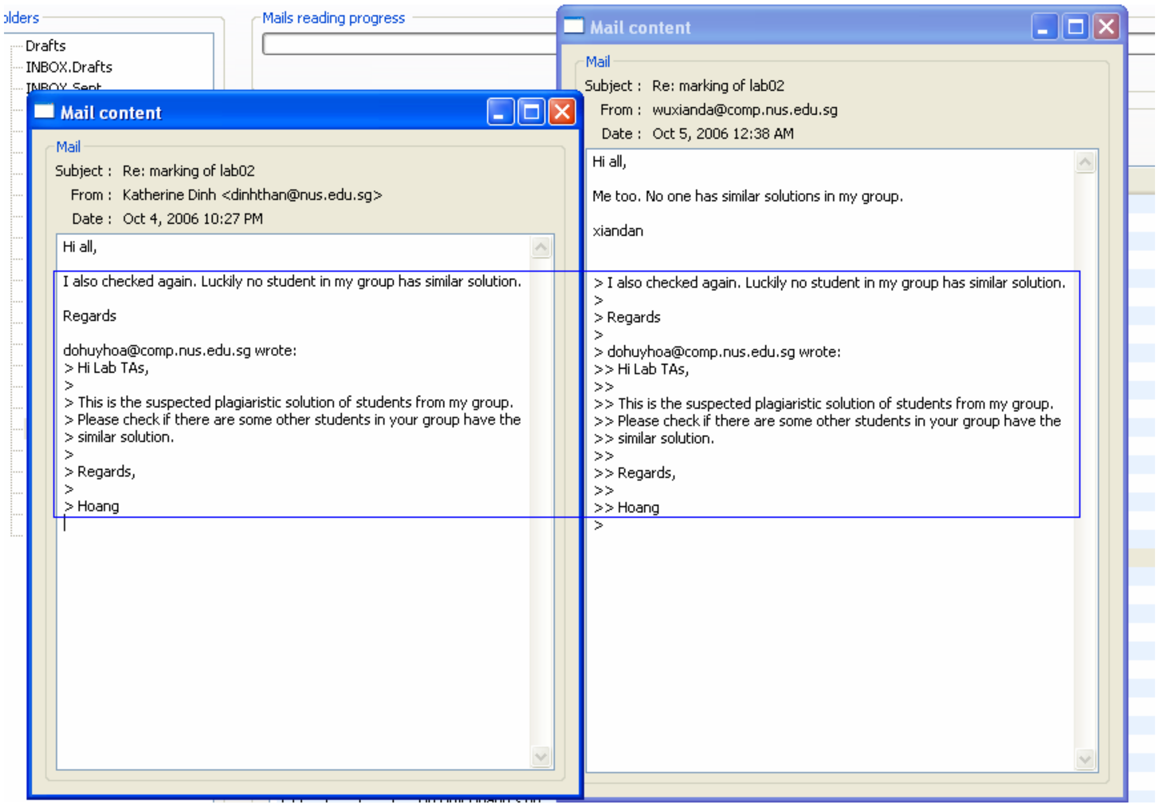
```
For every mail i
    For every sender j
        Chain up all mails behind mail i by sender j
        Run Needleman-Wunsch algorithm on mail i and chained mails
        Analyze result
    End for
End for
```

In this way, Needleman-Wunsch algorithm needs to be run multiple times for one mail. However, the memory requirement for each run will be significantly lower such that it will fit into a normal desktop memory capacity. After this optimization is implemented, my testing mail collection of 325 mails is run through the program. It takes 1 hour 32 minutes to finish. It is an improvement as the program is able to finish now. However, it is not practical to wait for one and a half hour just to check a mail box consisting of about 300 mails, which is not particularly big. More optimizations are needed.

4.2.5.2 Sub string screening

Since Needleman-Wunsch algorithm is very time consuming, one way to optimize the program is to find a way to reduce the number of times this algorithm is run. If there is a way to easily identify those cases that the mail is probably not made redundant by another specific mail or mails, Needleman-Wunsch algorithm would not need to be run. This would save a great deal of time.

In figure 1.1.3 and figure 1.1.4, mail 3 is made redundant by mail 4. These 2 mails are juxtapose here,



Subject : Re: marking of lab02
From : Katherine Dinh <dinhthan@nus.edu.sg>
Date : Oct 4, 2006 10:27 PM

```
Hi all,

I also checked again. Luckily no student in my group has similar solution.

Regards

dohuyhoa@comp.nus.edu.sg wrote:
> Hi Lab TAs,
>
> This is the suspected plagiaristic solution of students from my group.
> Please check if there are some other students in your group have the
> similar solution.
>
> Regards,
>
> Hoang
```

Subject : Re: marking of lab02
From : wuxianda@comp.nus.edu.sg
Date : Oct 5, 2006 12:38 AM

```
Hi all,

Me too. No one has similar solutions in my group.

xiandan

> I also checked again. Luckily no student in my group has similar solution.
>
> Regards
>
> dohuyhoa@comp.nus.edu.sg wrote:
>> Hi Lab TAs,
>>
>> This is the suspected plagiaristic solution of students from my group.
>> Please check if there are some other students in your group have the
>> similar solution.
>>
>> Regards,
>>
>> Hoang
>
```

Fig 4.2.5.2.1 Mail 3 and mail 4

From the lines in the blue box, all the lines in mail 3, except "Hi, all," and the symbol '>', are repeated exactly in mail 4, including end line symbols. Normally when mail texts are repeated in another mail, these mail texts are copied automatically or manually to the new mail. The copying process will often keep all the end line symbols, so the mail texts appear in the new mail in exactly the same format. The author will then delete some lines, but not some words in a line. With this observation, it can be concluded that if most of the lines from a mail does not appear in another mail unchanged, the mail is probably not redundant. Hence a pre screening can be done before invoking the Needleman-Wunsch algorithm. This screening will check if number lines in a mail appear in the chained mail texts is over a threshold. If the result is positive, Needleman-Wunsch algorithm will be run. Else it can be safely concluded that the mail is not redundant. The threshold is called sub string threshold and its value will be discussed in section 3.2.6.

After sub string screening is implemented, the program is able to finish the checking process for the 315 mails collected. The checking time is 6 minutes 16 seconds, which is about a 15 fold improvement.

4.2.5.3 Thread Tracing

When Needleman-Wunsch algorithm is run, it is guessed that a mail may be made redundant by some mail or mails in the chained mails. The chained mails are long and hence slow the Needleman-Wunsch algorithm. During checking of a mail, if another mail that will most probably make the mail redundant can be identified first, then Needleman-Wunsch algorithm can be run with this mail first. Since the mail will probably be identified as redundant in this checking, the amount of time used run Needleman-Wunsch algorithm against a much long mail chain will be saved.

When a mail is received from another person, this mail resides in our mail box and let it be mail A. This mail will be replied with the content append in the replied mail. Let the replied mail be B. Mail B will then be replied again with all contents of mail A and mail B attached. Let this replied mail be C. This process goes on. This is quite common in our daily emailing. When mail A and mail B are both from other person to a group of recipients and we are in the group, both A and B will be in our mail box and mail A is made redundant by mail B. When mail A and C are from our friend, B is our reply to A, both A and C will reside in our mail box and A is made redundant by C. These two cases are the most common source of redundancy. Recognizing this, another optimization can be crafted.

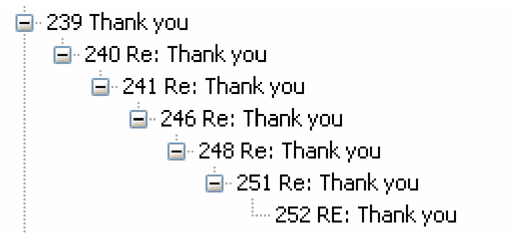
A mail is stored in a mail box with a message id that unique identify the mail in the mail box. A mail has also another field “reply-to”, which stores the message id of the mail it replies to. A mail and the mail it replies to are in the same thread. The mail A, B and C in last paragraph can be visualized as,

```
A
|_ B
   |_ C
```

C replies to B and B replies to A. A is at top of the thread. Before the checking starts, all the mails will be processed and put into their respective threads. Then for every mail, the thread is traced and the 2 mails that are one level and two levels down the thread are checked first. In this example, A will be checked against B and C first before being checked against the chained mails by different senders.

This optimization will also allow the program to line up the redundant mails in a thread fashion as shown below.

Without this optimization, a mail may not be identified to be made redundant by its immediate reply but by some other reply deeper down the thread and it is not possible to line up the redundancy result in this nicely thread fashion.



```
⊟ 239 Thank you
    ⊟ 240 Re: Thank you
        ⊟ 241 Re: Thank you
            ⊟ 246 Re: Thank you
                ⊟ 248 Re: Thank you
                    ⊟ 251 Re: Thank you
                        252 RE: Thank you
```

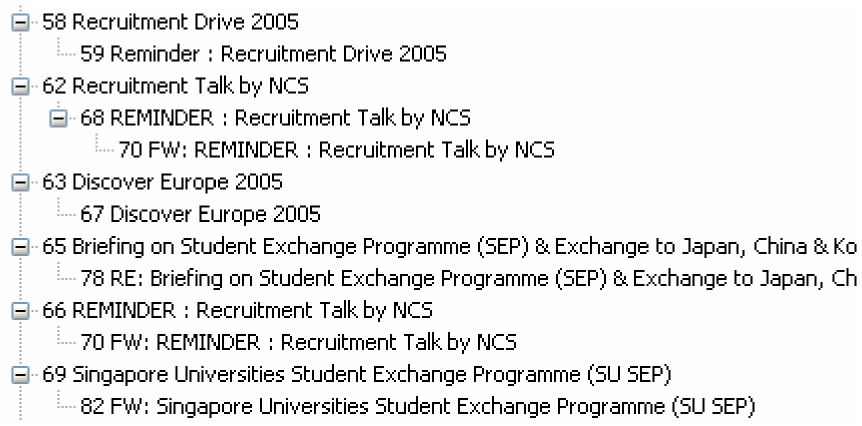
Fig 4.2.5.3.1 Redundancy result shown in thread fashion

In this screen, all mails are in the same thread and mails on the top are made redundant by mails below. It is clear to see the redundancy in this way than to show all mails made redundant by the same last mail at the bottom of the thread, which is mail 252 is this case.

After thread tracing optimization is implemented, the program now finish the checking process for the 315 mails collected in 4 minutes 25 seconds, down from 6 minute 16 seconds at the end of last section.

4.2.5.4 Mail title similarity

A fair amount of redundant mails results from forwarding the same mail by different person or organizations. It may also result from sending a same mail multiple times from the same person or organization. In both cases, the titles of the mails will probably be very similar. Here are some examples.



```
⊟ 58 Recruitment Drive 2005
    └ 59 Reminder : Recruitment Drive 2005
⊟ 62 Recruitment Talk by NCS
    ⊟ 68 REMINDER : Recruitment Talk by NCS
        └ 70 FW: REMINDER : Recruitment Talk by NCS
⊟ 63 Discover Europe 2005
    └ 67 Discover Europe 2005
⊟ 65 Briefing on Student Exchange Programme (SEP) & Exchange to Japan, China & Ko
    └ 78 RE: Briefing on Student Exchange Programme (SEP) & Exchange to Japan, Ch
⊟ 66 REMINDER : Recruitment Talk by NCS
    └ 70 FW: REMINDER : Recruitment Talk by NCS
⊟ 69 Singapore Universities Student Exchange Programme (SU SEP)
    └ 82 FW: Singapore Universities Student Exchange Programme (SU SEP)
```

Fig 4.2.5.4.1 Redundant mails made redundant by mails with similar names

In this result tree, a mail in the parent position is made redundant by a mail in a child position. It is observed that all these mails are made redundant by a mail with very similar name, with totally the same, or with some prefix like “REMINDER : “, “FW:” or “RE:”. Making use of this observation, another optimization can be implemented. When checking a mail, the mails with same name or similar with popular prefixes are checked first. This achieves the same effect as thread tracing optimization. Time is saved for not have to invoke Needleman-Wunsch algorithm against much long mail chains.

After mail title similarity optimization is implemented, the program now finish the checking process for the 315 mails collected in 3 minutes 8 seconds, down again from 4 minute 25 seconds at the end of last section.

4.2.5.5 Attachment check

Attachments have been ignored in previous discussions and implementations. However, they should not be ignored in determining whether a mail is redundant. A mail may have its texts totally repeated in another mail, but has a unique attachment and hence the mail is not redundant.

The checking of attachment is done in the following way. If mail A has no attachment, checking will proceed normally. If mail A has attachments, all the attachments must be included in mail B, in order for mail B to make mail A redundant. If mail B does not included all of mail A's attachment, Needleman-Wunsch algorithm need not to be run as mail A will not be made redundant by mail B. If mail A has attachments and it is detected to be made redundant by a combination of multiple mails, then everyone of mail A's attachment must appear in at least one of the mails in the combination of mails for the redundancy to be true. Two attached files are considered to be the same if their names and file sizes are the same. This is a safe assumption, as a different file may have the same name but seldom the same size.

The accuracy in redundancy detection in the 315 mails collection without attachment check is 0.878 (redundant threshold = 0.9, sub string threshold = 0.9, discussed in section 3.2.6). With attachment, the accuracy is increased to 0.962. Running time is not discernibly different as the slight overhead cancels the slight saving of Needleman-Wunsch algorithm in some cases.

4.2.5.6 Keyword check

In the detection of redundancy, some missing words from the redundant mails are allowed. In figure 3.2.5.2.1, "Hi, all," in mail 3 are allowed to be missing in mail 4. However, sometimes there are important keywords that cannot afford to be missing. For example, Jack sent a long proposal to his boss and his colleagues. His boss replies first by just adding a word "Approved" on top of the long proposal. His colleague then replies to Jack's original mail by adding some suggestions on top of the long proposal. After the checking is run, the program will detect that the email by Jack's colleague makes the email by Jack's boss redundant, because most of the boss's email, the proposal, appears in Jack’s colleague’s email. So Jack’s boss email will be flagged redundant and risk not being read by Jack. In this case, “Approved” is a keyword that cannot be missed.

Keyword check allows the user of the program to specify a list of words that are important. After a mail is determined to be redundant, keyword check will be invoked. The check will look at all the missing words gathered by the class MatchedStringExtractor. If the words include any of the important keywords, the mail will not be flagged redundant. This optimization greatly increases the reliability of the program.

4.2.5.7 Missing line limitation

The sub string threshold dictates the percentage of the lines required to be present in the sub string check. If percentage of lines present is below this threshold, further check will not be run and the mail is flagged as not redundant. Needleman-Wunsch algorithm will be run if percentage of lines present is above this threshold. However, with a 50 lines email, a 0.8 sub string threshold will allow 10 lines missing. This may be too many and if 10 lines of a mail are missing in another mail, it is quite indicative that the mail is not made redundant. Hence in extra to the sub string threshold, a maximum missing limitation in sub string check is imposed. Number of lines allowed to be missing will be the minimum of the number dictated by sub string threshold and the maximum missing line limit.

The effect of this limitation is more obvious with lower redundancy threshold and sub string threshold setting. When both redundancy threshold and sub string threshold are set to 0.8, accuracy is 0.939 with missing line limitation being 10 and accuracy is 0.957 with missing line limitation being 5.

4.2.5.8 Combo re-checks

In the example shown in figure 3.2.5.2.1, if mail 3 is checked against a chain of mails, the program may find most of the body of mail 3 to be in mail 4 and find the “hi, all” to be in another mail, as it is a very common phrase in emails. So the program may declare mail 3 to be made redundant by a combo of mail 4 and another mail which is not true. Such cases can be detected.

Whenever the match string from Needleman-Wunsch algorithm from is analyzed by MatchedStringExtractor and the program finds a mail to be made redundant by a combo of multiple mails, the program will look at the number of matched texts in each of the individual mails in the combo. If the contribution from any of the individual mail exceeds the redundancy threshold, the mail will be flagged to be made redundant by that mail alone. This optimization reduces the number of false flagged combo redundancy case.

Before implementing combo re-checks optimization, the number of combo redundant mail is found to be 10 in the 315 mail collection. After implementation of this optimization, the number falls to 1 which is correct.

4.2.6 Setting of parameters

The following table shows the accuracy and sensitivity of detecting redundancy in the 315 mail collection, varying 3 parameters: redundancy threshold, sub string threshold and missing line limitation.

Missing line limitation	Redundancy Threshold	Sub string Threshold	Sensitivity	Accuracy
	Redundancy threshold and sub string threshold the same			
5	0.8	0.8	0.975	0.957
10	0.8	0.8	0.975	0.939

5	0.9	0.9	0.950	0.962
10	0.9	0.9	0.950	0.962
5	0.95	0.95	0.918	0.973
5	0.98	0.98	0.862	0.992
5	0.99	0.99	0.824	0.992
	Redundancy threshold and sub string threshold are not the same			
5	0.95	0.8	0.943	0.974
5	0.95	0.75	0.943	0.974
5	0.96	0.8	0.943	0.974
5	0.97	0.8	0.918	0.986
5	0.98	0.8	0.881	0.993
5	0.99	0.8	0.836	0.993
	Varying missing line limitation			
3	0.95	0.8	0.943	0.974
7	0.95	0.8	0.943	0.974
10	0.95	0.8	0.943	0.974
3	0.98	0.98	0.862	0.992
7	0.98	0.98	0.862	0.992
10	0.98	0.98	0.862	0.992

Table Parameter setting and performance

Sensitivity and accuracy are the performance indicators of the program, hence these 2 columns are plotted as a scatter chart to find the dominant points and hence the default optimum settings of parameters. Sensitivity is ratio of the number of correct flagged redundant mails over the true number of redundant mails. Accuracy is the ratio of the number of correct flagged redundant mails over the total number of flagged redundant mails. [you should briefly explain what is sensitivity and what is accuracy.] The chart is shown below.

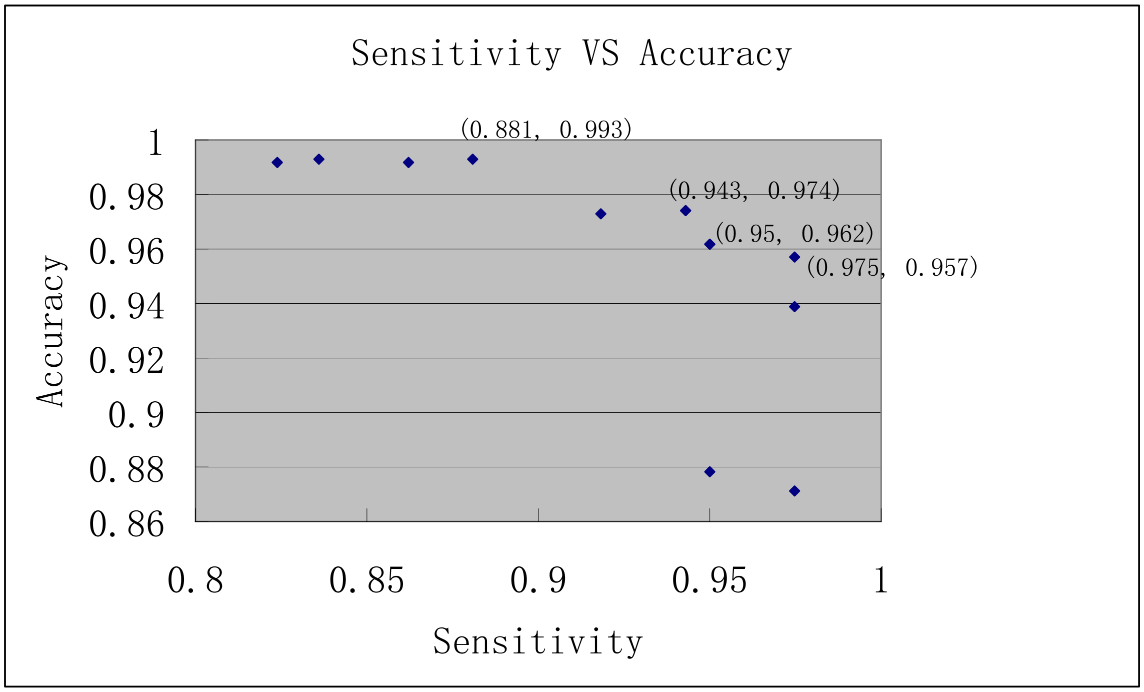


Fig 4.2.6.1 Scatter plot of sensitivity VS accuracy

In the chart, the only four points that are labeled are those that clearly dominate over others. However, the point (0.881, 0.993) offers an extremely high accuracy. So the corresponding setting (redundancy threshold = 0.98, sub string threshold = 0.8) will be used when very high accuracy is needed and lower sensitivity is tolerable. The point (0.975, 0.957) offers the highest sensitivity. Its corresponding setting (redundancy threshold = 0.8, sub string threshold = 0.8) will be used when very high sensitivity is needed and moderate accuracy is tolerable. Over the remaining point, (0.943, 0.974) is chosen as an optimum default point as we favor higher accuracy, so that useful email may not be flagged as redundant. Its setting (redundancy threshold = 0.95, sub string threshold = 0.8) will be used as default setting. The parameters can be set in the user interface to allow for flexibility. User will also be able to select 3 preset combinations described above, namely high accuracy, high sensitivity and default. The parameter setting interface will be described in section 3.3.3.

The last section in the parameter setting and performance table in last page shows the setting of missing line limitation is irrelevant when redundancy threshold and sub string threshold is high. The default setting of redundancy threshold will be 0.95 and that of sub string threshold will be 0.8 corresponding to point (0.943, 0.974) in the chart, while missing line limitation will be set to be 5.

4.3 Graphic user interface

There are two choices for coding a java graphic user interface, swing and SWT. Swing is included in standard java distribution. SWT stands for stand widget toolkit and is under the eclipse project and it is an alternative to Swing. SWT is light weight compared to Swing and takes up less memory. SWT is faster than Swing. Also SWT gives the interface a native look of the operating system instead of a java look of Swing. SWT is chosen to implement the system, as the redundancy detection process already takes up a fair amount of memory and we do not want the GUI to take up more memory, which will make the system difficult to run in slower machines.

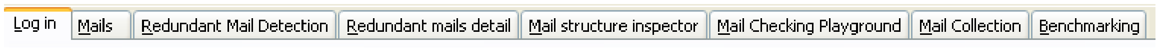


Fig 4.3.1 Tabs of the GUI

The interface is broken into 8 tabs as shown above catering to different functions. These 8 tabs and their functions will be described respectively below in each section.

4.3.1 Log in tab

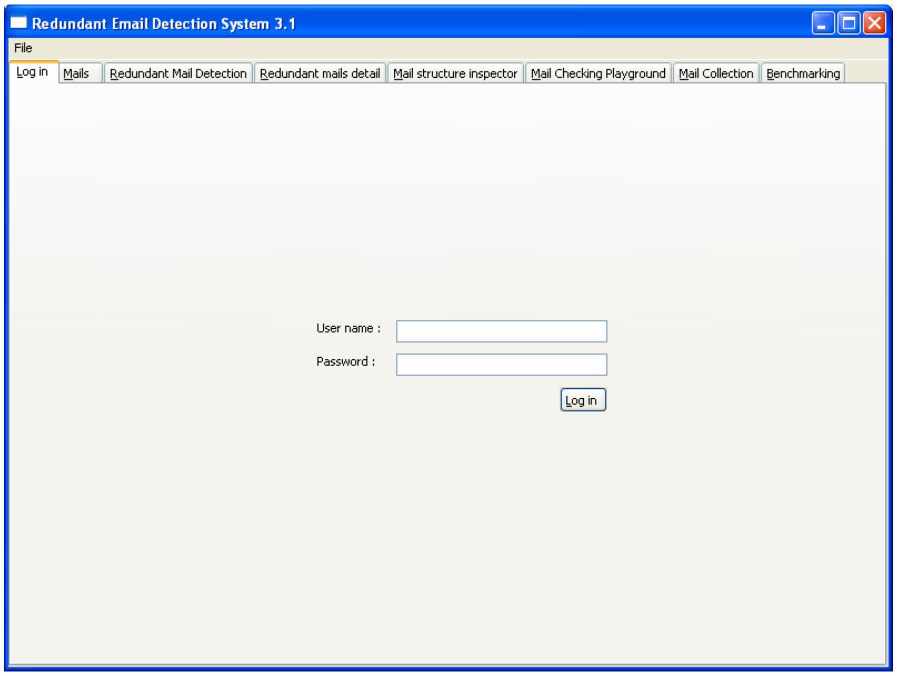
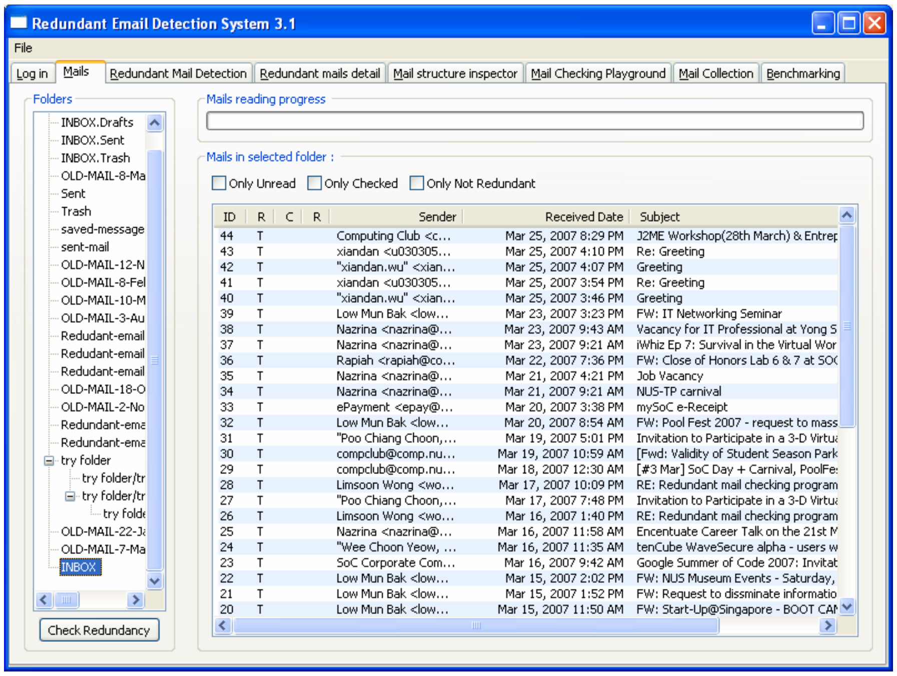


Fig 4.3.1.1 *Log in tab*

This tab is used for logging in and logging off the user.

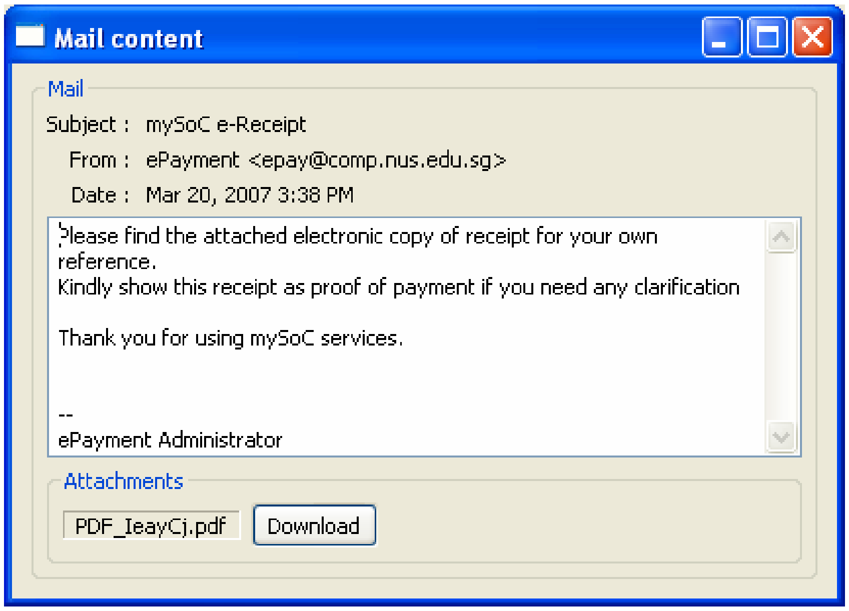
4.3.2 Mails tab



ID	R	C	R	Sender	Received Date	Subject
44	T			Computing Club <c…	Mar 25, 2007 8:29 PM	J2ME Workshop(28th March) & Entrep
43	T			xiandan <u030305…	Mar 25, 2007 4:10 PM	Re: Greeting
42	T			"xiandan.wu" <xian…	Mar 25, 2007 4:07 PM	Greeting
41	T			xiandan <u030305…	Mar 25, 2007 3:54 PM	Re: Greeting
40	T			"xiandan.wu" <xian…	Mar 25, 2007 3:46 PM	Greeting
39	T			Low Mun Bak <low…	Mar 23, 2007 3:23 PM	FW: IT Networking Seminar
38	T			Nazrina <nazrina@…	Mar 23, 2007 9:43 AM	Vacancy for IT Professional at Yong S
37	T			Nazrina <nazrina@…	Mar 23, 2007 9:21 AM	iWhiz Ep 7: Survival in the Virtual Wor
36	T			Rapiah <rapiah@co…	Mar 22, 2007 7:36 PM	FW: Close of Honors Lab 6 & 7 at SOC
35	T			Nazrina <nazrina@…	Mar 21, 2007 4:21 PM	Job Vacancy
34	T			Nazrina <nazrina@…	Mar 21, 2007 9:21 AM	NUS-TP carnival
33	T			ePayment <epay@…	Mar 20, 2007 3:38 PM	mySoC e-Receipt
32	T			Low Mun Bak <low…	Mar 20, 2007 8:54 AM	FW: Pool Fest 2007 - request to mass
31	T			"Poo Chiang Choon,…	Mar 19, 2007 5:01 PM	Invitation to Participate in a 3-D Virtu
30	T			compclub@comp.nu…	Mar 19, 2007 10:59 AM	[Fwd: Validity of Student Season Park
29	T			compclub@comp.nu…	Mar 18, 2007 12:30 AM	[#3 Mar] SoC Day + Carnival, PoolFe
28	T			Limsoon Wong <wo…	Mar 17, 2007 10:09 PM	RE: Redundant mail checking program
27	T			"Poo Chiang Choon,…	Mar 17, 2007 7:48 PM	Invitation to Participate in a 3-D Virtu
26	T			Limsoon Wong <wo…	Mar 16, 2007 1:40 PM	RE: Redundant mail checking program
25	T			Nazrina <nazrina@…	Mar 16, 2007 11:58 AM	Encentuate Career Talk on the 21st M
24	T			"Wee Choon Yeow, …	Mar 16, 2007 11:35 AM	tenCube WaveSecure alpha - users w
23	T			SoC Corporate Com…	Mar 16, 2007 9:42 AM	Google Summer of Code 2007: Invitat
22	T			Low Mun Bak <low…	Mar 15, 2007 2:02 PM	FW: NUS Museum Events - Saturday,
21	T			Low Mun Bak <low…	Mar 15, 2007 1:52 PM	FW: Request to dissminate informatio
20	T			Low Mun Bak <low…	Mar 15, 2007 11:50 AM	FW: Start-Up@Singapore - BOOT CAN

Fig 4.3.2.1 *Mails tab*

This tab is used to show the mails in a selected folder. Folder can be selected in the left column and its mails will be loaded and displayed on the right. Nested folders are supported. Using the check boxes above the mail list, users can select to see only unread mails, only mails that are checked for redundancy, only mail that are not flagged as redundant or any combination of them. When a mail is double clicked, its content and attachments will be brought up in a new window as shown below.



Subject : mySoC e-Receipt
From : ePayment <epay@comp.nus.edu.sg>
Date : Mar 20, 2007 3:38 PM

Please find the attached electronic copy of receipt for your own reference.
Kindly show this receipt as proof of payment if you need any clarification

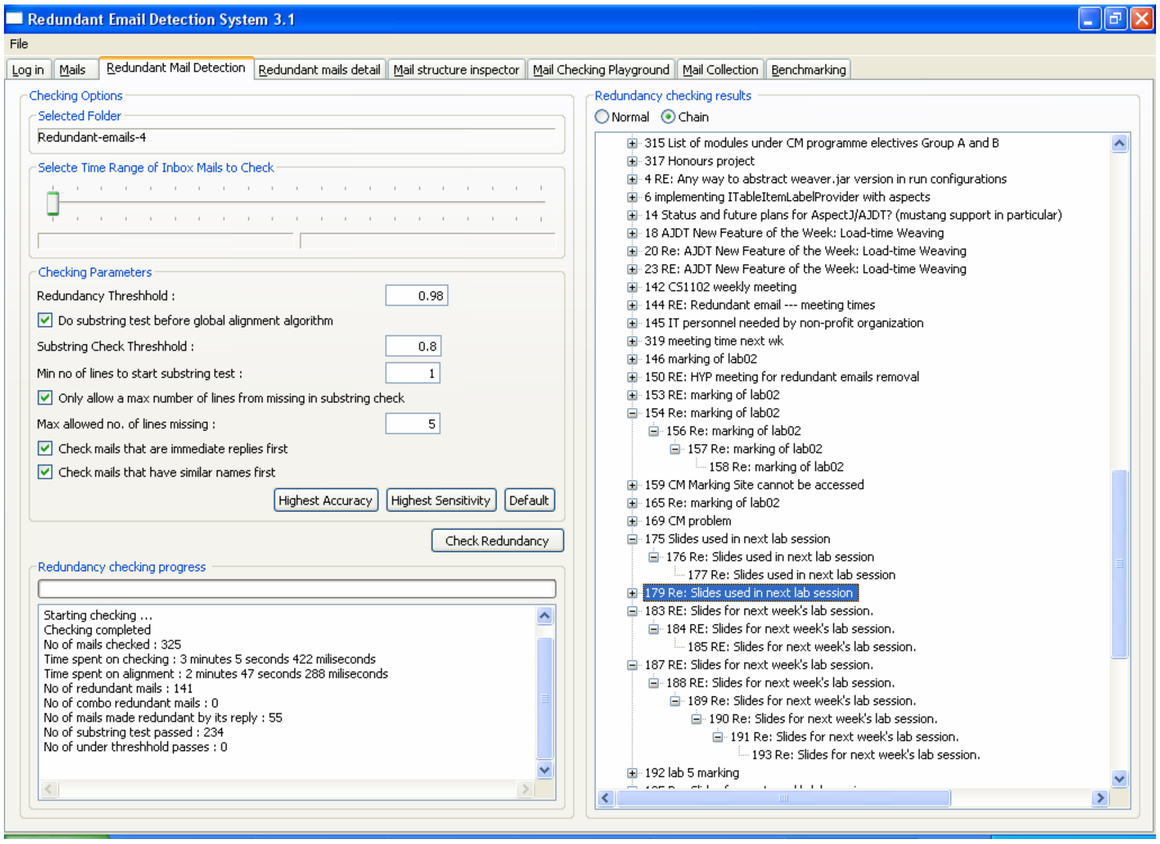
Thank you for using mySoC services.

--
ePayment Administrator

PDF_IeayCj.pdf

Fig 4.3.2.2 Mail content window

4.3.3 Redundant mails detection tab



Checking Options

Selected Folder: Redundant-emails-4

Selecte Time Range of Inbox Mails to Check

Checking Parameters

- Redundancy Threshhold : 0.98
- [✓] Do substring test before global alignment algorithm
- Substring Check Threshhold : 0.8
- Min no of lines to start substring test : 1
- [✓] Only allow a max number of lines from missing in substring check
- Max allowed no. of lines missing : 5
- [✓] Check mails that are immediate replies first
- [✓] Check mails that have similar names first

Redundancy checking progress

```
Starting checking ...
Checking completed
No of mails checked : 325
Time spent on checking : 3 minutes 5 seconds 422 miliseconds
Time spent on alignment : 2 minutes 47 seconds 288 miliseconds
No of redundant mails : 141
No of combo redundant mails : 0
No of mails made redundant by its reply : 55
No of substring test passed : 234
No of under threshhold passes : 0
```

Redundancy checking results

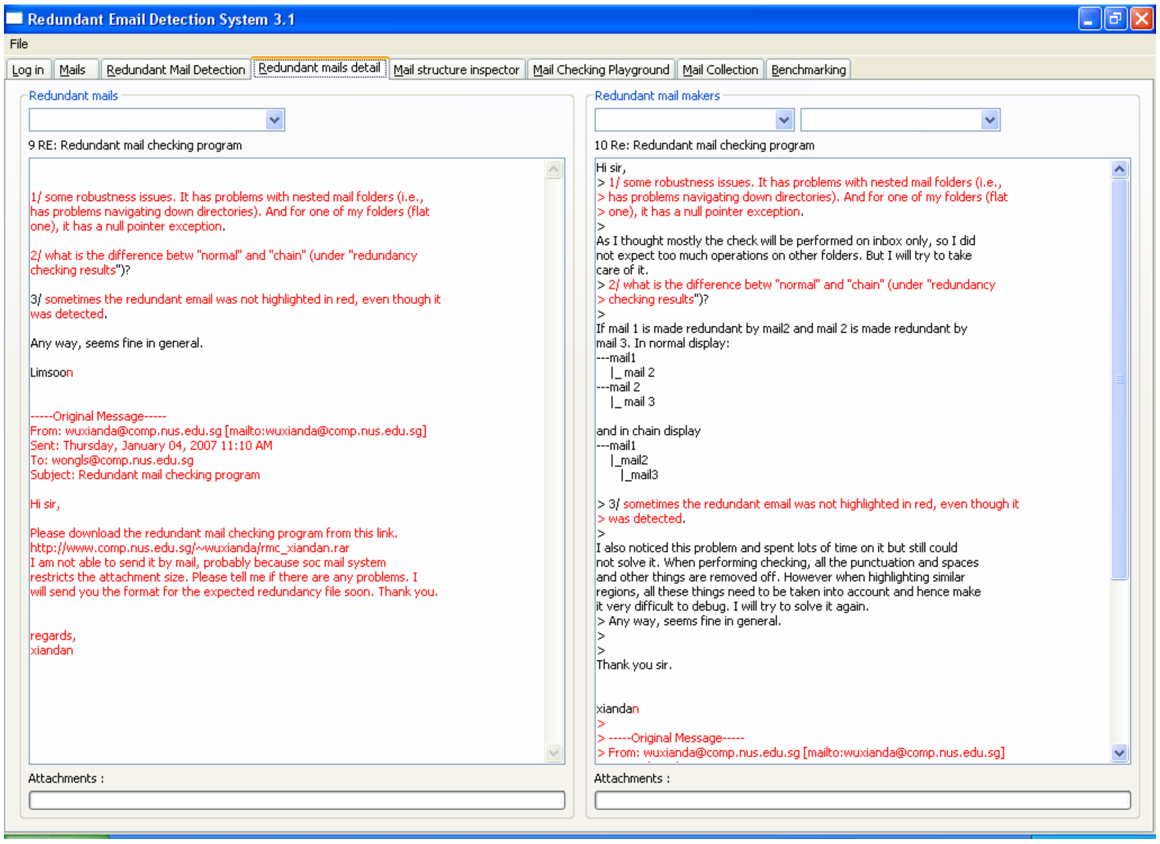
Normal [✓ Chain]

```
+ 315 List of modules under CM programme electives Group A and B
+ 317 Honours project
+ 4 RE: Any way to abstract weaver.jar version in run configurations
+ 6 implementing ITableItemLabelProvider with aspects
+ 14 Status and future plans for AspectJ/AJDT? (mustang support in particular)
+ 18 AJDT New Feature of the Week: Load-time Weaving
+ 20 Re: AJDT New Feature of the Week: Load-time Weaving
+ 23 RE: AJDT New Feature of the Week: Load-time Weaving
+ 142 CS1102 weekly meeting
+ 144 RE: Redundant email --- meeting times
+ 145 IT personnel needed by non-profit organization
+ 319 meeting time next wk
+ 146 marking of lab02
+ 150 RE: HYP meeting for redundant emails removal
+ 153 RE: marking of lab02
- 154 Re: marking of lab02
    - 156 Re: marking of lab02
        - 157 Re: marking of lab02
            158 Re: marking of lab02
+ 159 CM Marking Site cannot be accessed
+ 165 Re: marking of lab02
+ 169 CM problem
- 175 Slides used in next lab session
    - 176 Re: Slides used in next lab session
        177 Re: Slides used in next lab session
+ 179 Re: Slides used in next lab session
- 183 RE: Slides for next week's lab session.
    - 184 RE: Slides for next week's lab session.
        185 RE: Slides for next week's lab session.
- 187 RE: Slides for next week's lab session.
    - 188 RE: Slides for next week's lab session.
        - 189 Re: Slides for next week's lab session.
            - 190 Re: Slides for next week's lab session.
                - 191 Re: Slides for next week's lab session.
                    193 Re: Slides for next week's lab session.
+ 192 lab 5 marking
```

Fig 4.3.3.1 Redundant mails detection tab

This tab is used for the detection process and displaying the redundancy tree. The left hand side includes the parameter setting panel and checking process message window. The optimum preset parameter combination described in section 3.2.2 can be set using three buttons, high accuracy, high sensitivity and default. The right hand side displays result redundant email tree after detection process is finished. On this tree, mail in the parent node is made redundant by the mail in the child node. This tree can be displayed in normal fashion where every redundant mail is displayed as a root node, or in the thread fashion discussed in section 3.2.5.3, where redundant mails of the same thread are trained together under one root node. Double click any redundant mail in the result tree will bring user to the redundant mail detail tag where the details about the redundancy in the mail is displayed and highlighted.

4.3.4 Redundant mails detail tab



```
9 RE: Redundant mail checking program


1/ some robustness issues. It has problems with nested mail folders (i.e.,
has problems navigating down directories). And for one of my folders (flat
one), it has a null pointer exception.

2/ what is the difference betw "normal" and "chain" (under "redundancy
checking results")?

3/ sometimes the redundant email was not highlighted in red, even though it
was detected.

Any way, seems fine in general.

Limsoon


-----Original Message-----
From: wuxianda@comp.nus.edu.sg [mailto:wuxianda@comp.nus.edu.sg]
Sent: Thursday, January 04, 2007 11:10 AM
To: wongls@comp.nus.edu.sg
Subject: Redundant mail checking program

Hi sir,

Please download the redundant mail checking program from this link.
http://www.comp.nus.edu.sg/~wuxianda/rmc_xiandan.rar
I am not able to send it by mail, probably because soc mail system
restricts the attachment size. Please tell me if there are any problems. I
will send you the format for the expected redundancy file soon. Thank you.


regards,
xiandan
```

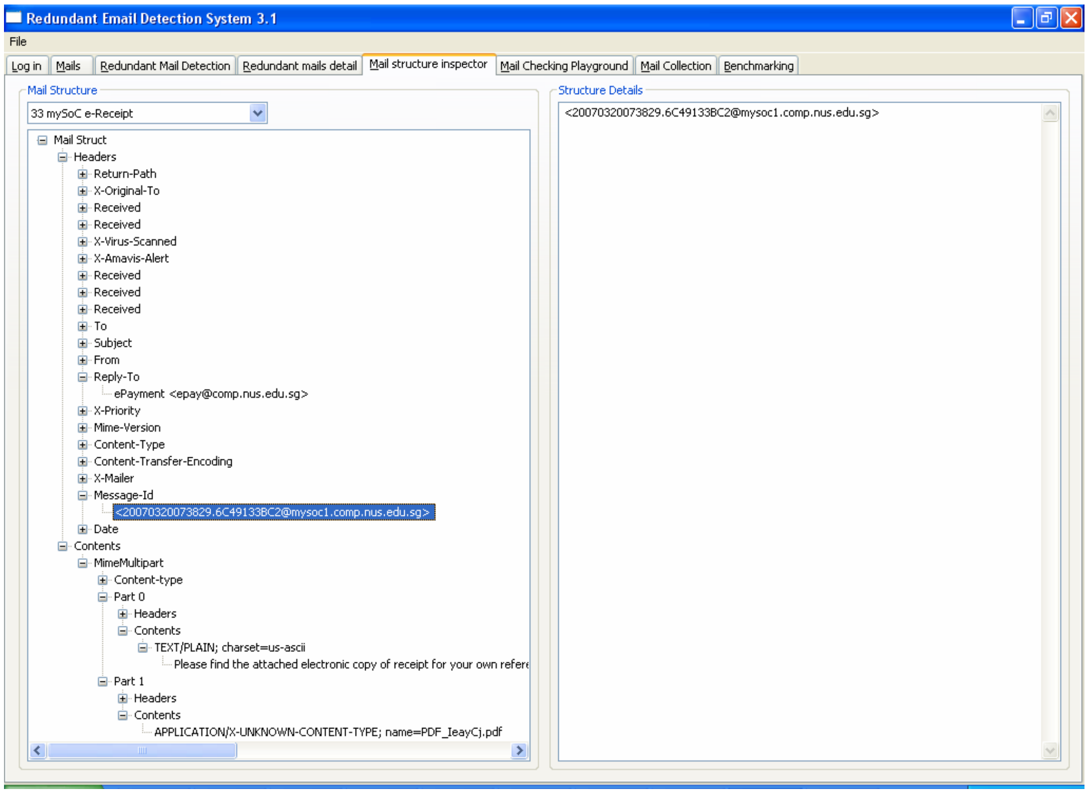
```
10 Re: Redundant mail checking program

Hi sir,
> 1/ some robustness issues. It has problems with nested mail folders (i.e.,
> has problems navigating down directories). And for one of my folders (flat
> one), it has a null pointer exception.
>
As I thought mostly the check will be performed on inbox only, so I did
not expect too much operations on other folders. But I will try to take
care of it.
> 2/ what is the difference betw "normal" and "chain" (under "redundancy
> checking results")?
>
If mail 1 is made redundant by mail2 and mail 2 is made redundant by
mail 3. In normal display:
---mail1
   |_ mail 2
---mail 2
   |_ mail 3

and in chain display
---mail1
   |_mail2
     |_mail3

> 3/ sometimes the redundant email was not highlighted in red, even though it
> was detected.
>
I also noticed this problem and spent lots of time on it but still could
not solve it. When performing checking, all the punctuation and spaces
and other things are removed off. However when highlighting similar
regions, all these things need to be taken into account and hence make
it very difficult to debug. I will try to solve it again.
> Any way, seems fine in general.
>
>
Thank you sir.


xiandan
>
> -----Original Message-----
> From: wuxianda@comp.nus.edu.sg [mailto:wuxianda@comp.nus.edu.sg]
```

Fig 4.3.4.1 Redundant mails detail tab

This tab displays the details of why the redundant email is marked redundant. The left hand panel displays the redundant mail while the right hand panel displays the mail that makes the redundant mail redundant. In both panel, the repeated texts in both mails are highlighted in red.

Highlighting of the common texts is a difficult and bug prone task. During detection of redundancy, all the spaces, new line symbols and punctuations are removed. After redundancy is identified, the repeated texts obtained only contain all the characters. These characters are separated into regions. Now the characters need to be fitted back to the original contexts. The repeated texts and the original mail body are compared one character by one character, skipping all spaces, new line symbols and punctuations. Starting position and ending position of each corresponding regions must be recorded in order to highlight them in the GUI.

4.3.5 Mail structure inspector tab



33 mySoC e-Receipt

```
Mail Struct
  Headers
    Return-Path
    X-Original-To
    Received
    Received
    X-Virus-Scanned
    X-Amavis-Alert
    Received
    Received
    Received
    To
    Subject
    From
    Reply-To
      ePayment <epay@comp.nus.edu.sg>
    X-Priority
    Mime-Version
    Content-Type
    Content-Transfer-Encoding
    X-Mailer
    Message-Id
      <20070320073829.6C49133BC2@mysoc1.comp.nus.edu.sg>
    Date
  Contents
    MimeMultipart
      Content-type
      Part 0
        Headers
        Contents
          TEXT/PLAIN; charset=us-ascii
            Please find the attached electronic copy of receipt for your own refer
      Part 1
        Headers
        Contents
          APPLICATION/X-UNKNOWN-CONTENT-TYPE; name=PDF_IeayCj.pdf
```

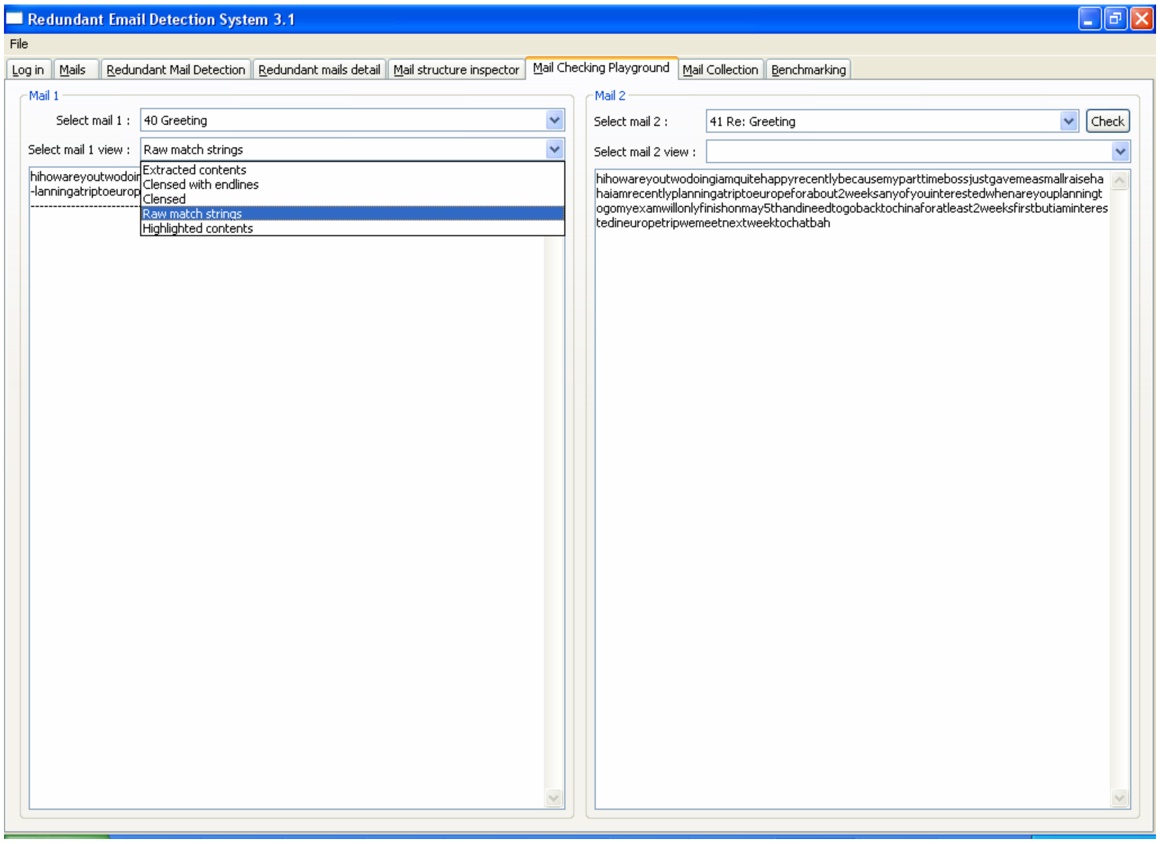
<20070320073829.6C49133BC2@mysoc1.comp.nus.edu.sg>

Fig 4.3.5.1 Mail structure inspector tab

This tab is used to inspect the internal structure of how a mail is stored. It is mainly used to assist in debugging the program. When a particular mail causes problem, this inspector can be used to look into the mail in great detail to see what differences in the mail are cause problem.

A mail is store with a lot of headers field and then contents. Some header fields include message-id which uniquely identify a mail in a server and reply-to which store the message-id of the mail that this mail is replying to. Contents include the mail body and attachments. Mail body may not only be stored once but possibly several times. The reason of this is different presentation of the contents are store, for example, plain text and html. The presentation and the actual body are not separated and hence the body is stored once for every presentation format. This has caused some problem during the development.

4.3.6 Mail checking playground tab



Select mail 1 : 40 Greeting
Select mail 1 view : Raw match strings

```
hihowareyoutwodoin
-lanningatriptoeurop
--------------------
```

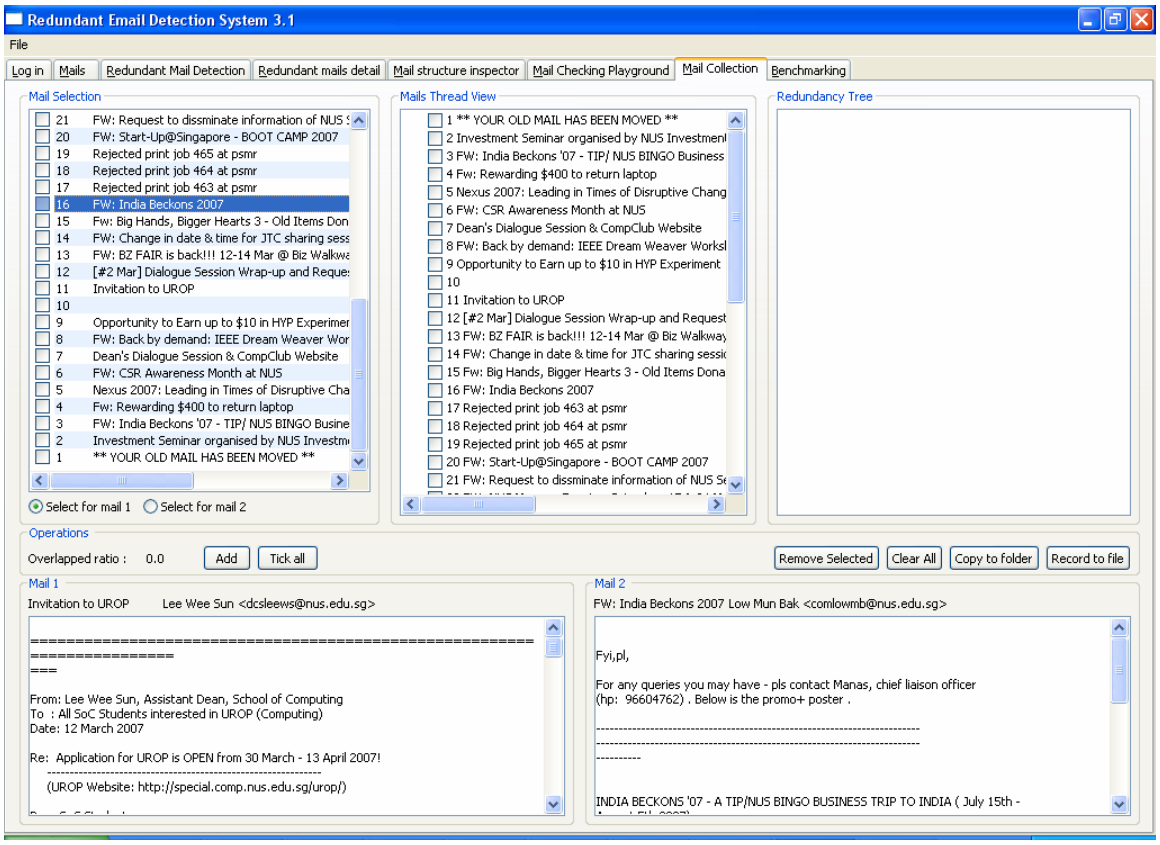
Select mail 2 : 41 Re: Greeting
Select mail 2 view :

```
hihowareyoutwodoingiamquitehappyrecentlybecausemyparttimebossjustgavemeasmallraiseha
haiamrecentlyplanningatriptoeuropeforabout2weeksanyofyouinterestedwhenareyouplanningt
ogomyexamwillonlyfinishonmay5thandineedtogobacktochinaforatleast2weeksfirstbutiaminteres
tedineuropetripwemeetnextweektochatbah
```

Fig 4.3.6.1 Mail checking playground tab

In this tab, any two mails in the current folder can be selected and run through the detection algorithm to see whether the left one is made redundant by the right one. After that, the two mails can be selected displayed in several formats, including “extracted contents”, “cleansed with endlines”, “cleansed”, “raw match strings” and “highlight contents”. “Extracted contents” option will display the body texts that are going to be processed. “Cleansed with endlines” option will display the mail body with all spaces and punctuation symbols removed. “Cleansed” option will display the mail body with all spaces, end line symbols and punctuation symbols removed. “Raw match strings” displays the match string returned by the Needleman-Wunsch algorithm. “Highlight contents” display mails in the same fashion as in redundant mails detail tab.

4.3.7 Mail collection tab



```
21    FW: Request to dissminate information of NUS S
20    FW: Start-Up@Singapore - BOOT CAMP 2007
19    Rejected print job 465 at psmr
18    Rejected print job 464 at psmr
17    Rejected print job 463 at psmr
16    FW: India Beckons 2007
15    Fw: Big Hands, Bigger Hearts 3 - Old Items Don
14    FW: Change in date & time for JTC sharing sess
13    FW: BZ FAIR is back!!! 12-14 Mar @ Biz Walkwa
12    [#2 Mar] Dialogue Session Wrap-up and Reque
11    Invitation to UROP
10
9     Opportunity to Earn up to $10 in HYP Experimer
8     FW: Back by demand: IEEE Dream Weaver Wor
7     Dean's Dialogue Session & CompClub Website
6     FW: CSR Awareness Month at NUS
5     Nexus 2007: Leading in Times of Disruptive Cha
4     Fw: Rewarding $400 to return laptop
3     FW: India Beckons '07 - TIP/ NUS BINGO Busine
2     Investment Seminar organised by NUS Investm
1     ** YOUR OLD MAIL HAS BEEN MOVED **
```

[✓ Select for mail 1] Select for mail 2

```
1 ** YOUR OLD MAIL HAS BEEN MOVED **
2 Investment Seminar organised by NUS Investment
3 FW: India Beckons '07 - TIP/ NUS BINGO Business
4 Fw: Rewarding $400 to return laptop
5 Nexus 2007: Leading in Times of Disruptive Chang
6 FW: CSR Awareness Month at NUS
7 Dean's Dialogue Session & CompClub Website
8 FW: Back by demand: IEEE Dream Weaver Worksl
9 Opportunity to Earn up to $10 in HYP Experiment
10
11 Invitation to UROP
12 [#2 Mar] Dialogue Session Wrap-up and Request
13 FW: BZ FAIR is back!!! 12-14 Mar @ Biz Walkway
14 FW: Change in date & time for JTC sharing sessic
15 Fw: Big Hands, Bigger Hearts 3 - Old Items Dona
16 FW: India Beckons 2007
17 Rejected print job 463 at psmr
18 Rejected print job 464 at psmr
19 Rejected print job 465 at psmr
20 FW: Start-Up@Singapore - BOOT CAMP 2007
21 FW: Request to dissminate information of NUS Se
```

Overlapped ratio : 0.0

Mail 1: Invitation to UROP   Lee Wee Sun <dcsleews@nus.edu.sg>

```
=================================================================
=================
===

From: Lee Wee Sun, Assistant Dean, School of Computing
To  : All SoC Students interested in UROP (Computing)
Date: 12 March 2007

Re:  Application for UROP is OPEN from 30 March - 13 April 2007!
    -------------------------------------------------------------
    (UROP Website: http://special.comp.nus.edu.sg/urop/)
```

Mail 2: FW: India Beckons 2007 Low Mun Bak <comlowmb@nus.edu.sg>

```
Fyi,pl,

For any queries you may have - pls contact Manas, chief liaison officer
(hp:  96604762) . Below is the promo+ poster .

--------------------------------------------------------------------------
--------------------------------------------------------------------------
----------


INDIA BECKONS '07 - A TIP/NUS BINGO BUSINESS TRIP TO INDIA ( July 15th -
```

Fig 4.3.7.1 Mail collection tab

This tab is used to collect the 315 mails used in benchmarking and testing. Mail collection is a complicated process and hence it is worthwhile to build an interface to facilitate collection. Any two mails can be selected and a reference checking can be run on them. I will them look through the two mails to check whether they are suitable mails for benchmarking. If they are suitable, I will mark their redundancy relation and add them to the top right hand panel. The program will check whether the mails are already in the collection and sort out the redundancy relation with the existed mails in the collection. For example, if mail A and mail B are already in the collection and mail A is marked to be made redundant by mail B. Then when I add another relation that mail B is made redundant by mail C into the collection. The program will only copy C into the collection. At the same time, because C makes B redundant and B makes A redundant and C must make A redundant. This new relationship must also be recorded. After collection, these relations must be written into a file which will be used in benchmarking.

The grammar definition of this file is shown below.

*File ::= ReFile**

*ReFile ::= MsgID NUMBER MakerFile MakerFile**

MakerFile ::= MsgID | ComboMakers

*ComboMakers ::= "COMBO" NUMBER MsgID MsgID**

Mails are recorded by their message-ids. A mail is followed by the number of cases that make it redundant and then each of these cases. Each case can be a mail or a combo of multiple mails. A example is shown below.

<000d01c656e3$63468a10$f55a8489@comp.nus.edu.sg>

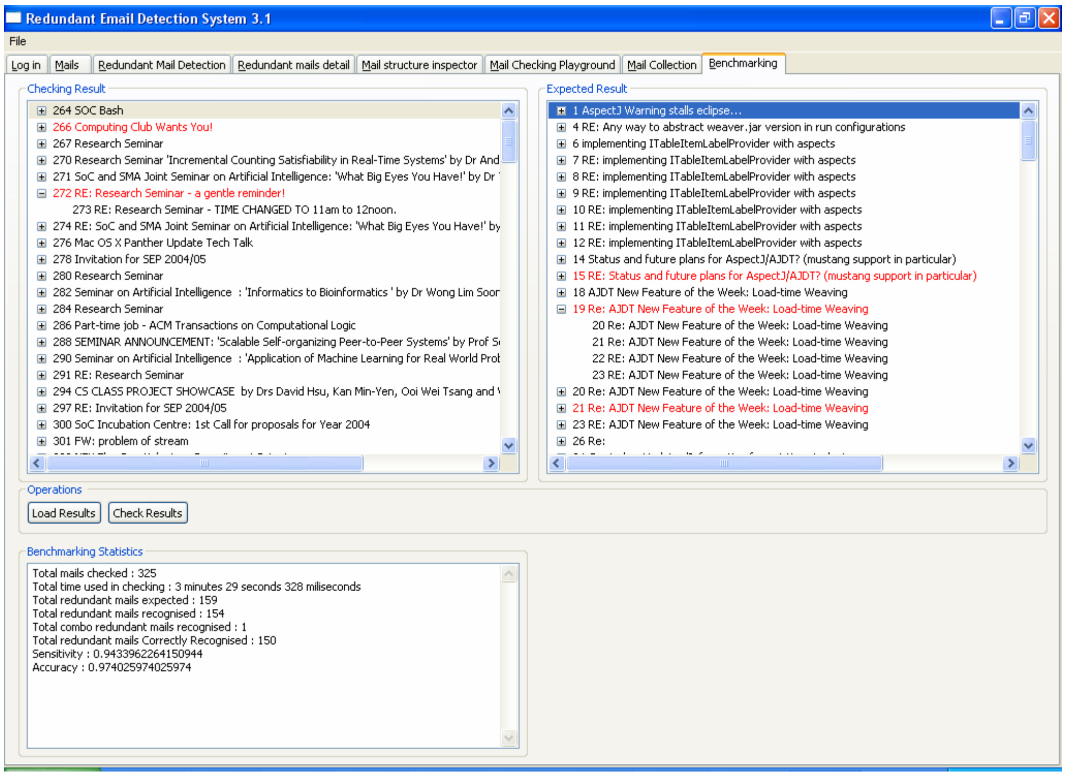
2

<001101c656e7$924b99a0$f55a8489@comp.nus.edu.sg>

<6C24CBA42F567B4380B621C6378C19351735D4@MBX01.stf.nus.edu.sg>

The mail identified by the id in the first line can be made redundant by 2 cases. Each case is a mail.

4.3.8 Benchmarking tab



```
264 SOC Bash
266 Computing Club Wants You!
267 Research Seminar
270 Research Seminar 'Incremental Counting Satisfiability in Real-Time Systems' by Dr And
271 SoC and SMA Joint Seminar on Artificial Intelligence: 'What Big Eyes You Have!' by Dr
272 RE: Research Seminar - a gentle reminder!
    273 RE: Research Seminar - TIME CHANGED TO 11am to 12noon.
274 RE: SoC and SMA Joint Seminar on Artificial Intelligence: 'What Big Eyes You Have!' by
276 Mac OS X Panther Update Tech Talk
278 Invitation for SEP 2004/05
280 Research Seminar
282 Seminar on Artificial Intelligence  : 'Informatics to Bioinformatics ' by Dr Wong Lim Soon
284 Research Seminar
286 Part-time job - ACM Transactions on Computational Logic
288 SEMINAR ANNOUNCEMENT: 'Scalable Self-organizing Peer-to-Peer Systems' by Prof S
290 Seminar on Artificial Intelligence  : 'Application of Machine Learning for Real World Prob
291 RE: Research Seminar
294 CS CLASS PROJECT SHOWCASE  by Drs David Hsu, Kan Min-Yen, Ooi Wei Tsang and
297 RE: Invitation for SEP 2004/05
300 SoC Incubation Centre: 1st Call for proposals for Year 2004
301 FW: problem of stream
```

```
1 AspectJ Warning stalls eclipse...
4 RE: Any way to abstract weaver.jar version in run configurations
6 implementing ITableItemLabelProvider with aspects
7 RE: implementing ITableItemLabelProvider with aspects
8 RE: implementing ITableItemLabelProvider with aspects
9 RE: implementing ITableItemLabelProvider with aspects
10 RE: implementing ITableItemLabelProvider with aspects
11 RE: implementing ITableItemLabelProvider with aspects
12 RE: implementing ITableItemLabelProvider with aspects
14 Status and future plans for AspectJ/AJDT? (mustang support in particular)
15 RE: Status and future plans for AspectJ/AJDT? (mustang support in particular)
18 AJDT New Feature of the Week: Load-time Weaving
19 Re: AJDT New Feature of the Week: Load-time Weaving
    20 Re: AJDT New Feature of the Week: Load-time Weaving
    21 Re: AJDT New Feature of the Week: Load-time Weaving
    22 RE: AJDT New Feature of the Week: Load-time Weaving
    23 RE: AJDT New Feature of the Week: Load-time Weaving
20 Re: AJDT New Feature of the Week: Load-time Weaving
21 Re: AJDT New Feature of the Week: Load-time Weaving
23 RE: AJDT New Feature of the Week: Load-time Weaving
26 Re:
```

```
Total mails checked : 325
Total time used in checking : 3 minutes 29 seconds 328 miliseconds
Total redundant mails expected : 159
Total redundant mails recognised : 154
Total combo redundant mails recognised : 1
Total redundant mails Correctly Recognised : 150
Sensitivity : 0.9433962264150944
Accuracy : 0.974025974025974
```

Fig 3.3.8.1 Benchmarking tab

This tab is used for benchmarking of the mail collection. After checking on the mail collection is done, the tab is used to do the automatic comparisons of the actual result and the expected results. The top left panel displays the actual results, with detected redundant mails that are not really redundant highlighted in red. The top right panel displays the expected results, with expected redundant mails not detected highlighted in red. Statistics about the detection are shown at the bottom panel. These statistics will be described in details in chapter 4 benchmarking.

Chapter 5 Benchmarking

315 mails are collected and using the mail collection tab in the GUI from my friend's and my mail boxes and their redundancy relationships are marked and stored in a expected redundancy file, whose format is described in section 3.3.7. This mail collection is used for benchmarking. Two performance indicators are used, namely sensitivity and accuracy. Sensitivity is defined as the ratio of number of correctly flagged redundant mails over the number of expected redundant mails. Accuracy is defined as the ration of number of correctly flagged redundant mails over the total number of flagged redundant mails. These two indicators have been used to fine tune parameters, as shown in section 3.2.2. Accuracy is favored as the program must be reliable and the users do not want to miss any mail that is not actually redundant.

After all the optimizations are implemented and enabled, the program is able to finish the detection process for the 315 mails in around 3 minute 10 seconds. As these 315 mails have high concentration of redundant mails, it takes longer than normal mail boxes to process, because more sub string tests are passed and hence Needleman-Wunsch algorithm are invoked more frequently. The redundancy detection process of a mail box with 500 over mails can be expected to be finished in about 5 minute. This is an acceptable time.

Chapter 6 Conclusion

6.1 Limitations

There are several limitations in this project.

- Language limitation

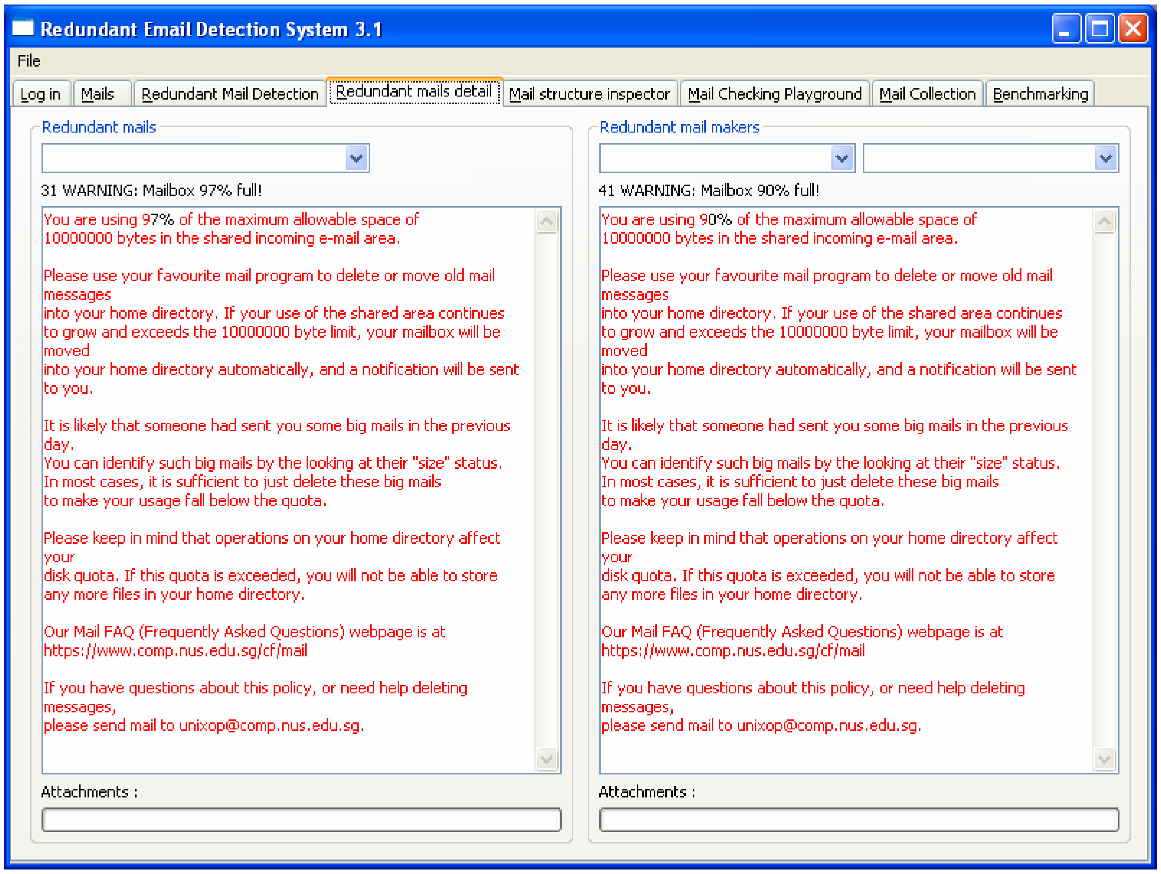
Currently, the program is only able to deal with English mails. Singapore is a multi cultural society and we can expect a lot of mails to be written in other languages. Being limited to only English mails will limit the usefulness of the program in identifying redundant emails.

- Dealing with redundant emails

In the program, redundant emails identified are only flagged and not dealt with. To boost the usefulness of the program, redundant emails should be copied to some other folders just as junk mails are copied to junk folders. The user can then check the redundant mail folder from time to time to see whether any useful mails are thrown there.

- Accuracy problem

The program is still not 100% accurate. The kind of mails that will most likely to be identify as redundant mails are machine generated mails. These mails are highly similar except for a small part of the body. Below is an example.



31 WARNING: Mailbox 97% full!

You are using 97% of the maximum allowable space of
10000000 bytes in the shared incoming e-mail area.

Please use your favourite mail program to delete or move old mail
messages
into your home directory. If your use of the shared area continues
to grow and exceeds the 10000000 byte limit, your mailbox will be
moved
into your home directory automatically, and a notification will be sent
to you.

It is likely that someone had sent you some big mails in the previous
day.
You can identify such big mails by the looking at their "size" status.
In most cases, it is sufficient to just delete these big mails
to make your usage fall below the quota.

Please keep in mind that operations on your home directory affect
your
disk quota. If this quota is exceeded, you will not be able to store
any more files in your home directory.

Our Mail FAQ (Frequently Asked Questions) webpage is at
https://www.comp.nus.edu.sg/cf/mail

If you have questions about this policy, or need help deleting
messages,
please send mail to unixop@comp.nus.edu.sg.

41 WARNING: Mailbox 90% full!

You are using 90% of the maximum allowable space of
10000000 bytes in the shared incoming e-mail area.

Please use your favourite mail program to delete or move old mail
messages
into your home directory. If your use of the shared area continues
to grow and exceeds the 10000000 byte limit, your mailbox will be
moved
into your home directory automatically, and a notification will be sent
to you.

It is likely that someone had sent you some big mails in the previous
day.
You can identify such big mails by the looking at their "size" status.
In most cases, it is sufficient to just delete these big mails
to make your usage fall below the quota.

Please keep in mind that operations on your home directory affect
your
disk quota. If this quota is exceeded, you will not be able to store
any more files in your home directory.

Our Mail FAQ (Frequently Asked Questions) webpage is at
https://www.comp.nus.edu.sg/cf/mail

If you have questions about this policy, or need help deleting
messages,
please send mail to unixop@comp.nus.edu.sg.

Fig 6.1.1 Machine generated mail falsely identified as redundant

These two mails are totally the same except the percentage of the fullness of the mail box.

6.2 Further developments

- Detection of redundancy across different mail accounts

Usually people will have more than one email accounts. All the accounts can be logged in at the same time and all the mails in all the accounts can be checked together. In this way, a user will not need to read similar mails in different email accounts.

- Hiding the redundancy detection process

The redundancy detection process can be programmed to run in the background once the agent is opened. The user can proceed to read new mails while the agent is checking on the background. Result will be shown once checking is finished.

- Redundancy checking plug in

The redundancy checking algorithm can be developed into a plug in of any popular email agents such as Thunderbird. User normally would prefer to use their favorite mail agent and the plug in can enable them to use the same agent while having the ability to identify redundancy in mails.

- Possible application of redundancy detection algorithm in RSS readers

A lot of people use RSS readers nowadays and they subscribe to a large number of feeds. There may be redundant post from different feeds. The redundancy detection algorithm can be possibly added to RSS readers to detect redundant posts.

6.3 Summary

This project developed a redundant email detection program that enables busy professionals and managers to detect redundant emails in their mail boxes and save the time reading them. The critical element of this system is the redundant emails detection algorithm. It has been designed from global alignment algorithm and is able to detect redundant emails accurately within an acceptable time. The objectives of efficiency, reliability and user friendliness have been achieved.

References

[1] Chong-See Kwok, Limsoon Wong. **A method for eliminating redundant email messages**. European Patent No. 1327192, 20 April 2005. Singapore Patent No. 95931 [WO 00/33981], 31 May 2005.

[2] Dan Gusfield. **Algorithms on Strings, Trees, and Sequences.** Cambridge Univ Press, 1997.

[3] Richard Durbin**. Biological sequence analysis : probalistic models of proteins and nucleic acids.** Cambridge University Press, 1998.

[4] RS Boyer, JS Moore. **A fast string searching algorithm.** Comm, ACM, 1977.

[5] DE Knuth, JH Morris, VB Pratt. **Fast pattern matching in strings.** SIAM J. Comput, 1977.

[6] TF Smith, MS Waterman. **Identification of common molecular subsequences.** J. Mol. Biol, 1981.

[7] SB Needleman, CD Wunsch. **A general method applicable to the search for similarities in the amino acid sequence of two proteins.** J. Mol. Biol, 1970.

[8] JavaMail. *http://java.sun.com/products/javamail/*

[9] Standard widget toolkit. *http://www.eclipse.org/swt/*

[10] Michael Lecuyer. Boyer Moore algorithm java implementation. *http://www.theorem.com/java/BM.java*

[11] Peter Sestoft. Needleman-Wunsch algorithm java implementation. *http://www.dina.kvl.dk/~sestoft/bsa.html*

[12] Open IRIS. *http://www.openiris.org/*