

HONOUR YEAR PROJECT REPORT

2006 – 2007

ELIMINATION OF REDUNDANT EMAILS

By

NGUYEN THE HUY

Department of Computer Science

School of Computing

National University of Singapore

Project ID : H114040

Advisor : Professor Wong Lim Soon

Deliverables : Report – 1 Volume

Source code and Test data – 1 CD

Abstract

Email is using by hundred of millions of people worldwide. Unfortunately, the widespread use of email has given rise to several problems. Redundant email is one of them. In this project we are exploring a solution to that problem by examining the applicability of duplicate detection method in eliminating redundant emails. Following fingerprint technique, we have developed a Duplicate Detection component used in our email program. We also invent a new collision-free hash function that encodes more information in a single hash value. Finally we present and discuss the result of our evaluation on various types of redundant emails. Several experiments conducted during the development of project are also included.

Subject Descriptors:

- E.5 Organization / Structure
- H.3.1 Content Analysis and Indexing
- H.3.2 Information Storage
- H.4.3 Communications Application

Keywords:

Duplicate detection, k-gram, fingerprint, shingle, email, hash

Implementation Hardware and Software:

- Platform: POSIX-compliant operating system.
- Programming Language: C++
- Software Library: VMime

Table of Content

LIST OF FIGURE.....	IV
INTRODUCTION.....	1
1.1. CONTEXT	1
1.2. PROJECT OBJECTIVE AND SCOPE.....	2
1.3. REPORT ORGANIZATION.....	3
DIFFERENT TYPES OF REDUNDANCY & POSSIBLE SOLUTIONS.....	4
2.1. DIFFERENT TYPES OF REDUNDANCY.....	4
2.2. POSSIBLE SOLUTIONS	7
DUPLICATE DETECTION.....	10
3.1. A QUICK SURVEY.....	10
3.2. DUPLICATION, RESEMBLANCE AND CONTAINMENT	11
3.3. DESIRABLE PROPERTIES	12
3.4. WORD	13
3.5. HASH FUNCTION	15
3.6. SELECT FINGERPRINTS	16
3.7. COMPARE FINGERPRINTS	19
EMAIL HANDLING.....	20
4.1. CLIENT-SIDE OR SERVER-SIDE SOLUTION	20
4.2. EMAIL LIBRARY	20
IMPLEMENTATION & EVALUATION.....	22
5.1. OVERALL DESIGN	22
5.2. DUPLICATE CHECK	22
5.3. EVALUATION SETTING	24
5.4. WINDOW SIZE VERSUS PRECISION AND RECALL.....	25
5.5. THRESHOLD VALUE VERSUS PRECISION AND RECALL	26
5.6. FALSE POSITIVE	27
5.7. FALSE NEGATIVE	29
POSSIBLE IMPROVEMENT	31
CONCLUSION.....	33
REFERENCE	34
APPENDIX A: DERIVATION OF a AND b IN SECTION 3.5	36
APPENDIX B: EXAMPLES OF EACH DUPLICATE CATEGORY	38
CATEGORY 1: IDENTICAL CONTENT.....	38
CATEGORY 2: ALMOST IDENTICAL CONTENT WITH SUBTLE DIFFERENCE	39
CATEGORY 3: ARBITRARILY COMPONENT-REARRANGED MESSAGES.....	40
CATEGORY 4: MESSAGE QUOTED INSIDE ANOTHER	41
CATEGORY 5: COMBINATION OF THE FIRST THREE WITH THE FORTH CATEGORY	42

List of Figure

Figure 1: Duplicate caused by emails having identical content.....	4
Figure 2: Duplicate by subtle difference in textual presentation.	5
Figure 3: Duplicate caused by quoting another email inside.....	6
Figure 4: containment and resemblance formulas	11
Figure 5: Delimiters	13
Figure 6: Several stop words.....	14
Figure 7: A possible values for a and b.....	16
Figure 8: Result from experiment with RFC documents	16
Figure 9: Code for select fingerprints	18
Figure 10: Average size of documents's fingerprints.....	19
Figure 11: Overall Design.....	22
Figure 12: Duplicate Check	24
Figure 13: Sample test result.....	25
Figure 14: Window size vs. Precision & Recall, threshold = 95%	25
Figure 15: Windows Size vs. Precision & Recall	26
Figure 16: Threshold vs. Precision & Recall, Window size = 3.....	26
Figure 17: Threshold vs Precision & Recall	27

Chapter 1

INTRODUCTION

1.1. Context

Along with the increasing use of computer has always been the development of storage and communication technologies. Electronic Mail (or Email for short) has taken advantage of those and evolved to an essential communication channel used by hundreds of millions of people worldwide nowadays. According to an study conducted by School of Information Management and System (University of California at Berkeley) [6], email ranks second behind telephone as the largest information flow medium; about 31 billions emails were sent daily in 2003 and that figure is expected to double in 2006. Another result from the 2005 Email Usage Survey by ClearContext Corporation [7] shows that most people have 2-6 email accounts; more than 50% of survey respondents spend over 2 hours and 14% spend over 4 hours in email every day.

Widespread use of email, however, has lead to several problems. In addition to spam, email organization and overload are the other major challenges to many people (40%, Email Usage Survey [7]). According to our observation, duplication of email content makes it contribution to the latter two problems. Not only does duplication waste system's storage and channel's bandwidth to store and transport redundant (duplicated) emails but it also wastes people's time and energy to read and classify them.

Such duplication of email content forms the so-called *redundant message* problem: a message is considered redundant when the information it carries can be found inside other message(s). Several problems may arise from having redundant messages: wasting computer storage, channel bandwidth, increasing overhead in organizing and retrieving emails... Besides email system, redundant message problem are also common in bulletin board, newsgroup, discussion forum...

1.2. Project Objective and Scope

In our experience, there has been no complete solution to email problems yet. Most current email solutions aim to resolve spam issue and / or automate the classification of email into different folders. Quite a few email users have too many folders that organizing them becomes yet another burden. Besides, our experience with Enron dataset shows that many non-spam emails in a same folder are indeed redundant. Those redundant emails create obstruction on email classification and retrieval process.

In order to help resolve email's issues, we have put our effort in filling the gap left by existing solutions: eliminating redundant emails. Our solution centers on the idea of detecting duplication of email content. Given a list of email messages, our program will analyze each email's content, detect duplication between them and finally output a list of emails being made redundant by others.

Our main objective is to examine the applicability of duplication detection approach to redundant message problem. During this process, we discovered several interesting facts about possible alternative approaches and word frequency in email communication. We also invented a new collision-free hash function that encodes more information on a single hash value. This collision-free hash function is a core component in our duplicate detection method which satisfies all three important properties of a general duplicate detection technique.

In order to concentrate on our core duplicate detection component, we have limited our system to plain text messages. Nevertheless, by employing open strategy in system design, features not considered in current implementation such as attachments, HTML mails... can be added with minimal cost.

Currently, our solution is implemented to work on email message. However, because our duplicate detection component is implemented to work on arbitrary document type, it can be integrated to other messaging systems where redundant message due to duplication occurs (e.g. newsgroup, bulleting, discussion forum).

1.3. Report Organization

This report is organized into seven Chapters numbered from 1 to 7 and two Appendixes.

Below is a brief description about each of them:

- **Chapter 1 Introduction:** An overview on current email usage and its problem. A brief introduction on the project, its objective and scope.
- **Chapter 2 Different Type of Redundancy and Possible Solutions:** A closer look to various situations where redundant messages occurs. A discussion on possible solutions and their common property are presented in this chapter.
- **Chapter 3 Duplicate Detection:** A detailed explanation of our duplicate detection method. Each subsection includes a discussion on related works and description of our own. **Section 3.5** talks about hash function.
- **Chapter 4 Email handling:** A short note on how email is handled in our project.
- **Chapter 5 Implementation and Evaluation:** A brief description on implementation and discussion on evaluation results.
- **Chapter 6 Possible Improvement:** Discussion on possible future enhancement.
- **Chapter 7 Conclusion:** a short summary on what have been achieved.
- **Appendix A:** A detailed mathematical proof on how our hash function satisfies collision-free property.
- **Appendix B:** Examples of each duplicate category.

Chapter 2

DIFFERENT TYPES OF REDUNDANCY & POSSIBLE SOLUTIONS

2.1. Different types of redundancy

An email message consists of two major sections: header fields - Meta information about email (e.g. Message-ID, sender and receiver's addresses) - and body – the information it carries in text form. In this report, we will refer to content of an email as the information contained in its body.

If *duplicate* is defined as “a copy that corresponds to an original exactly” [3] then only emails with identical content are considered *duplicates*. In the following example, owner of email address sally.beck@enron.com will receive two emails with the same content as the one shown in **Figure 1**. Each copy of this email is made redundant by the other; obviously she only needs to keep one copy in her mailbox.

```
Message-ID: <15928422.1075855993904.JavaMail.evans@thyme>
Date: Fri, 17 Nov 2000 00:50:00 -0800 (PST)
From: heidi.mason@enron.com
Subject: Reporting Line for Sydney Risk Officer
Cc: sally.beck@enron.com
Bcc: sally.beck@enron.com
Scott
```

```
With all our reporting changing to London Office and now it
appears some restructure of risk and the broadening of
Sally's role, what we are going to do for the second report
for our new risk manager, Justin Den Hertog - he has taken
over from Alan.
```

```
With thanks
```

Figure 1: Duplicate caused by emails having identical content.

Different senders (e.g. people, mailing lists) sending a same text to a person will also results in a similar situation as in our previous example. Occasionally, we have received multiple copies of school announcement sent by different administrative officers. They have exactly the same content except for the sender address and Message-ID fields.

However, detecting identical copy can be achieved easily and reliably by comparing message¹'s checksums (with a special treat to Message-ID header field and the like); therefore that is not our focus. We turn to a more general definition of duplicate: “*the same semantic content whether or not it is a precise syntactic match*” [5]. This new definition covers several cases where two emails essentially carry same information yet differ in their body's character strings. Consider a modified version of the previous message sent by heidi.mason@enron.com haft an hour later.

```
Message-ID: <15928422.1075855993904.JavaMail.evans@thyme>  
Date: Fri, 17 Nov 2000 01:20:00 00:50:00 -0800 (PST)  
From: heidi.mason@enron.com  
Subject: Reporting Line for Sydney Risk Officer  
Cc: sally.beck@enron.com  
Bcc: sally.beck@enron.com  
Dear Scott,
```

```
With all our reporting changing to London Office and now it  
appears some restructure of risk and the broadening of  
Sally's role, so what we are going to do for the second  
report for our new risk manager, Justin Den Hertog - he has  
taken over from Alan.
```

```
With thanks
```

```
Thanks
```

Figure 2: Duplicate by subtle difference in textual presentation.

In spite of some addition and deletion of characters from the former message, the two carry same information and are considered *duplicates* according to our definition. Note that we have deliberately discarded their formatting codes and hence will not consider

¹ We will use *email* and *message* interchangeably.

any difference in formatting. In our experience, having formatting code (e.g. HTML, CSS) create more falsity while it is still debatable whether visual appearance can affect original information or not.

Going further, we want to detect reproduction of email content inside another. The quoted content may remain as a whole or be torn into smaller portions placed in arbitrary order inside the containing message. Next example demonstrates a common situation where the quoted email is made redundant by the reply.

```
Message-ID: <4627792.1075855692236.JavaMail.evans@thyme>
Date: Wed, 9 Feb 2000 02:27:00 -0800 (PST)
From: phillip.allen@enron.com
To: keith.holst@enron.com
Subject: RE: W basis quotes
I'll get back to them on this. I know we have sent
financials to Clinton Energy...I'll check to see if this is
enough. In the meantime, is it possible to show me
indications on the quotes I asked for? Please advice.
George
```

```
> -----Original Message-----
> Sent: Monday, February 07, 2000 5:54 PM
> To: george.rahall@acnpower.com
> Subject: Re: W basis quotes
>
> George,
> Can you please call my credit desk at 713-853-1803.
> They have not received any financials for ACN Power.
> (The bolded reply above may appear here, as some people would prefer to do so)
>
> Thanks,
>
> Phillip Allen
```

Figure 3: Duplicate caused by quoting another email inside.

Originally our definition applies to entire body of an email messages. But in order to cover cases of this type, we need to extend our definition of duplicate to *sub-document* level. Where each message is a document, its paragraphs, sentences... are sub-documents, if all sub-documents of a message are *duplicated* in another, that message is considered duplicated and is mark redundant. Situations like this are seen commonly in email threads when people conduct their discussion via emails.

Having defined what *duplicate* mean in this project, we classifies duplicate into 5 main categories. Namely they are:

- ***Identical content***: Messages with identical content, each message is made redundant by the others.
- ***Almost identical content with subtle difference***: Messages with subtle difference in their content.
- ***Arbitrarily Component-rearranged messages***: Messages which are difference only in the order of their sentence, paragraph...
- ***Message quoted inside another***: Message whose content is quoted inside other messages.
- ***Combination of the first three with the forth category***: Content of the quoted message remain unchanged, or is subtly changed, or is disorderly re-arranged.

Appendix B gives examples of each category.

2.2. Possible Solutions

There have been several attempts to tackle *redundant message* problem. Grouping discussion messages into thread is one. A well-known email system utilizing this idea is Gmail¹. Gmail groups all replies with their original message, creating thread or *conversation*. However, this idea does not work for other types of redundancy mentioned before. Furthermore, messages in a same thread need not be redundant, e.g. the reply does not quote the original message. Thus organizing messages into threads does not

¹ Google Mail <http://gmail.google.com>

guarantee detection of redundant emails but rather provides a way to group possibly redundant emails together.

Automating classification of messages is also a possible approach. This idea is implemented by many email filter programs such as procmail, slocal and ifile. Unfortunately, duplication of messages in a same folder is not their concern. Another system described in US paten 5404488 routes messages into pre-defined categories and retains only the newest message in each category. For instance, in a financial data feed system, only newest share prices are kept in Share Prices category. Kwok and Wong [17] note that “while the result may assist in eliminating redundant message, it also eliminates all information about previous messages in the same category”. Besides, this type of solution only seems to work with messages with special characteristics (e.g. characteristics that satisfy classification rules) but does not fit in general email communication.

Another approach described in European paten 1327192 [17] detects redundancy by considering overlap between messages. Specifically, the paten makes use of string matching algorithms to compute the overlap. While the technique does promise good results, its performance may not be as good as expected when operating on large messages; current online string algorithms do not provide adequate performance [11] while indexed searching algorithms which are faster come at high cost of space. Nevertheless, the strategy – computing overlap between messages – is a promising hint to redundant message problem.

After carefully studying these possible solutions, we recognize that the crucial point in solving our problem is to measure the similarity between messages. Similarity appears in different forms: communication context (original – reply, grouping messages into thread), special characteristics of message (automating classification) and repeat of content (computing overlap). The last form of similarity seems to fit well in our definition of *duplication*. It is also the most flexible approach compared to others.

Based on that discovery, our next step is to develop an efficient and cost-effective duplicate detection solution. **Chapter 3** will describe in detail our invention.

Furthermore, it would be of great benefit if our technique can combine with other email solutions. We find that our preprocessing mechanism is similar to many email classification methods. Xiao-lin Wang and Ian Cloete [18] note that “most systems treat a message as a bag of words”. Coincidentally, that is exactly how we preprocess email messages. This suggests the ability of integrating email classification techniques into our program.

Chapter 3

DUPLICATE DETECTION

3.1. A Quick Survey

In recent decades, the increase in volume of data is tremendous. A study in 2003 [6] reveals that new stored information grew 30% a year between 1999 and 2002 to approximately 5 exabytes (one exabyte equals 10^{18} bytes) in 2002. Email generates 400,000 terabytes of new information each year worldwide. That growth in volume of stored information emphasizes the importance of detecting duplicates.

Both industry and research communities have placed considerable interest on developing efficient and cost-effective method to solve *duplicate* problem. Many systems such as MOSS, COPS, and SCAM are results from their effort. Distinguished by their approaches, their techniques can be classified into two major categories [2]:

- *Shingling approach*: A document is divided into a set of contiguous terms or *shingles*. By comparing the number of matching shingles we can determine whether two document are duplicates or not. Some examples are COPS (Brin, Davis et al. 1995), KOALA (Heintze 1996), and DSC (Broder 1998).
- *Similarity Measures Calculation approach*: Each document is represented by a set of *features*. Similarity between two documents can be measured by calculating distance between two sets. Usually the feature is extracted using Information Retrieval techniques and distance is represented by cosine. An famous example is SCAM system.

Another approach to solve *duplicate* problem is to compare two documents' character string directly. However, as argue by Broder [5], this approach is not feasible for very large collection of documents. Furthermore, none of the existing standard distance defined on string seems to well capture the notion of *resemblance* and *containment*.

In this project, we follow the *Shingling* approach and develop a mechanism to detect duplicate in our system. Specifically, we divide a document into *shingles*, hash them and select a subset of these hashes to be the document's *fingerprints*. By comparing fingerprints, we can detect duplication between two documents.

We will discuss the background of our method in subsequent sections.

3.2. Duplication, Resemblance and Containment

Recall from our discussion in section 2.2 **Possible Solutions**, we need a metric to measure the similarity between messages. Broder [5] has already defines two similarity metrics: *resemblance* and *containment*. The *resemblance* $r(A, B)$ of two documents, A and B , is a number between 0 and 1, such that when the resemblance is close to 1 it is likely that the documents are roughly the same. Similarly, the *containment* $c(A, B)$ of A in B is a number between 0 and 1 that, when close to 1, indicates that A is roughly contained within B . They fit well within our definition of *duplicate*.

However, we notice that if A and B roughly resemble each other, resemblance can be written as: A is roughly contained within B and B is roughly contained within A . That suggests resemblance can be written in term of containment. If $content(A)$ returns a quantity representing amount of information in A , $overlap(A, B)$ returns a quantity representing amount of overlapped information between A and B , we can compute $containment(A, B)$ and $resemblance(A, B)$ by the following formulas:

$$\begin{aligned} containment(A, B) &= overlap(A, B) / content(A) \\ resemblance(A, B) &= (containment(A, B) + containment(B, A)) / 2 \end{aligned}$$

Figure 4: containment and resemblance formulas

It is followed from these formulas that when $containment(A, B)$ get close *enough* to 1, the portion of A inside B – $overlap(A, B)$ – is large enough for us to say that

content of A is duplicated inside B ; that means A is made redundant by B . We use a threshold value t such that when $\text{containment}(A, B)$ is larger than t , $\text{containment}(A, B)$ is considered close enough to 1. In **Chapter 5**, we use different value of t to measure our system's precision and recall ratios.

Applying the above to resemblance, if $\text{resemblance}(A, B)$ get close enough to 1 then we can assert that A is made redundant by B and B is made redundant by A .

In our method, $\text{content}(A)$ is the number of fingerprint of document A and $\text{overlap}(A, B)$ is the number of matched fingerprints resulted from comparing A and B 's fingerprints. **Section 3.5** and **3.6** go into details how to select and compare fingerprints.

3.3. Desirable Properties

The inventors of Winnowing [4] have defined several properties that a duplicate detection method should possess. Namely, they are:

- *White spaces insensitivity*: In matching text files, matches should not be affected by such things as extra white spaces, capitalization, punctuation, etc. In other domains the notion of what string should be equals is different – for example, in matching software text it is desirable to make matching insensitive to variable names.
- *Noise suppression*: Any match must be large enough to imply that the material is copied and is not simply a common word or idiom of the language in which document is written.
- *Position independence*: Coarse-grained permutation of the contents of a document (e.g. scrambling the order of paragraphs) should not affect the set of discovered matches. Adding to a document should not affect the set of matches in the original portion of the new documents. Removing part of a document should not affect the set of matches in the portion that remains.

In subsequent sections we will show that our method satisfies all of the above properties.

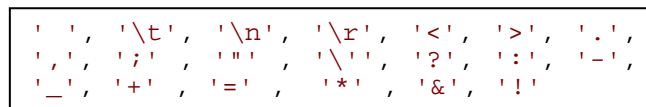
3.4. word

In many *shingling* techniques, a *shingle* is a contiguous substring of length k , called *k-gram*. The value of k is usually fixed. Schleimer et al [4] note that “*the larger k is, the more confident we are that matches between documents are not coincidental; however larger value of k will limit the sensitivity to detect relocation of substring of length less than k* ”. A sufficiently long k is expected to satisfy the *noise suppression* property.

However, in several other techniques [1, 8, 9], a *shingle* is not a fixed sized *k-grams*; instead they choose strings to be hashed to be sentences or paragraphs or strings with “anchor” words. However the implementation of such methods will be limited to a particular type of data (e.g. English text). Schleimer et al claim that using *k-gram* is more robust. [4]

Our approach is a mix between the above two. We introduce the notion of *word* as a sequence of characters delimited by a list of *delimiters*. For instance, in a document containing “*Hello world!*” with space and exclamation mark (!) as our *delimiters*, we have the following words: *Hello, world*. Our notion of *word* works well on various data type such as English and software code. To some extends, we can also consider each *word* as a *k-gram* with variable k .

Figure 5 presents a list of delimiters used in our tests.



```
' ', '\t', '\n', '\r', '<', '>', '.', ',',  
';', '"', '\\', '?', ':', '-',  
'_', '+', '=', '*', '&', '!'
```

Figure 5: Delimiters

To satisfy *white space insensitivity* property, we include all punctuation marks in our list of *delimiter* and convert all *words* to lower case.

We also want to remove *stop words* from our messages. A stop word is one that has low information content yet frequently occurs. Our frequency analysis of 112,554 email messages extracted from Enron data set¹ allows us to construct a list of *stop words*. **Figure 6** presents some *stop words* with corresponding frequencies.

3,146,830	the
2,522,569	to
1,407,508	and
1,293,236	of
1,217,486	a
985,325	in
812,264	from
795,889	for
647,838	on
620,620	is
612,054	that
200,202	http
198,926	an
188,304	was
131,296	www
115,992	about
10,788	hi
5,666	hey

Figure 6: Several stop words

(As a side note, the present of `http` and `www` in our list may suggest a close link between the World Wide Web and email.)

In our method, each word corresponds to an entry in a dictionary. Each entry is associated with a unique odd number which will be used in our hash function later. The dictionary is constructed during preprocessing stage which essentially tokenizes document (message) into words.

¹ <http://www.cs.cmu.edu/~enron>

Although our choice of *word* itself is not sufficient to satisfy *Noise suppression* property, contribution from our hash function and hash selection process will allow us to do so.

3.5. Hash Function

Hash function usually carries the burden of accuracy and performance. Hash collision will possibly lead to false positives. Many common hash functions such as MD5, SHA... can be used because they are easy to compute, have low probability of collision and can be calculated on arbitrary data/document length. Campbell et al [1] define their *near-perfect* sentence-based hash function that claims to produce at most one collision in their method. MOSS, an implementation of Winoing [4] use a tuned rolling hash function originally developed by Karp-Rabin.

We observe that the resulted hash values from the above hash functions do not contain any information about the before and after *shingles*. We took a new approach: the hash value returned by our function still maintains information about the preceding or succeeding word. We demonstrate a version of our hash function that maintains information about the succeeding word.

This hash function takes in a sequence of two contiguous words and returns the hash value for the former one. For example, *Current_Word* and *Next_Word* are the function's input and the value returned is the hash of *Current_Word*.

Previous_Word	Current_Word	Next_Word
---------------	--------------	-----------

Each word associates with a unique odd number. Let x be the *Current_Word*'s odd number and y be the *Next_Word*'s odd number; each pair of x and y will uniquely identify the sequence *Current_Word Next_Word*. We would like to find a value E such that E can only be computed from a unique pair of x and y .

Consider the following equation:

$ax + by = E \quad (1)$ <p>with a and b are integers</p>
--

We found that with appropriate values of a and b , we can find such E . Our function will return E as the hash value of *Current_Word*. **Appendix A** explains in details how to find a and b . **Figure 7** shows a possible pair of values for a and b , largest (y) can be obtained after the preprocessing stage.

$$\begin{array}{l}
 b = -1 \\
 a = \frac{\text{largest}(y)}{2} + 1
 \end{array}$$

Figure 7: A possible values for a and b

Because the returned hash value contains information about the succeeding word, each value will differentiates a sequence of two from others. In other words, our hash function is collision-free. In fact, result from our experiment with the entire collection of RFC (Request For Comment) documents¹ verifies that conclusion. **Figure 8** shows the result of that experiment.

```

Total number of documents      : 4728
Total size of collection       : 227 MB
Total different words          : 212,913
Total different sequences      : 3,965,797
Total different hashes         : 3,965,797
No collision is detected

```

Figure 8: Result from experiment with RFC documents

3.6. Select Fingerprints

A subset of *word*'s hashes is selected to be the document's fingerprints. A naïve approach is to select all hashes. In Winnowing method, the size of the fingerprints would be much larger than the original document itself [4]. For our technique, considering an English document with average word length of 5, applying naïve scheme on 4-byte hashes will not result in a larger index as in Winnowing; however there are other cases (e.g. when we

¹ Downloadable from <http://www.rfc-editor.org/download.html>

switch to 8-byte or 64-bit hash) which will generate fingerprints larger than the document.

Another approach suggested by Heintze [10] is to choose n smallest hashes. However, having a fixed number of hashes limit the effect of detection, only near copies of entire documents could be detected. Documents vastly different in size also does not fit well in this approach. [4]

Another strategy from Manber [9] is to choose all hashes that are $0 \bmod p$, for some p . This method generates variable size fingerprint for document and work well according to Manber's observation. However, as pointed out in [4], this approach does not guarantee that matches are detected. Since the distance between consecutive k -grams whose hashes are selected can be quite large in some cases, matches inside the gap are not detected.

It is our best interest not to restrict our solution to limited application by following these schemes. Instead, we choose the approach described in Winking [4]. Let a *window* of size w be w consecutive hashes. In each window select the minimum hash value. If there are more than one hash with the same minimum hash value, select the rightmost occurrence. Each hash is selected only once.

```
void DupDetection::fingerprint(vector<string> &words,
                               vector<unsigned int> &fprt){
    //document is tokenized into a vector of words
    //its fingerprints will be stored inside fprt
    int pInd = -1, mInd = -1; //previously chosen hash's index
                               //and minimum hash's index
    unsigned int min = UINT_MAX; //min hash value in a window

    for (int i=0; i<=words.size() - wsize-1; i++){
        //fill in window and find minimum hash
        int p1 = dict[words[i]].odd;
        for (int j=1; j<=wsize; j++) {
            Word w2 = dict[words[i+j]];
            unsigned int h = hash(p1, w2.odd);
```

```

        if (h <= min) {           //select the right most min
            min = h;
            mInd = i + j-1;
        }
        p1 = w2.odd;
    }
    if (mInd != pInd) {         //each hash is chosen once
        fprt.push_back(min);
        pInd = mInd;
    }
    //prepare for new window
    min = UINT_MAX;
    mInd = -1;
}
}

```

Figure 9: Code for select fingerprints

Scheimer et al [4] in their paper claim this approach more efficient than others. Central to the approach in WInnowing is the notion of *local* algorithm. “An algorithm is local if, for every window of w consecutive hashes h_i, \dots, h_{i+w-1} , the algorithm decides to select one of these hashes as a fingerprint and this choice depends only on the window’s content” [4]. It is supported by a mathematical proof that any local algorithm is able to detect match between substrings of length at least $w + k - 1$ with k is number of characters in a k -gram. Hash value of a k -gram differentiates that k -gram from others. So far no collision on hash value is reported [4].

There is a direct mapping from Scheimer et al’s solution to ours: Our hash function operates on sequence of $L (= 2)$ consecutive words while theirs operates on k consecutive characters; we both apply WInnowing algorithm to select document’s fingerprints. These direct correspondences allow us to draw similar conclusion: our solution is able to detect match between sequences of length at least $w + L - 1$ with $L = 2$. With appropriate value of w , we are confident that our method satisfies *noise suppression* property.

The choice of w , in our experience, depends on document types. Consider a sequence of 4 words in an email sent to a friend listing possible tourist destinations.

New York, Tokyo, London, Paris

In the reply, the sequence has become:

```
> New York
NO
> Tokyo
NO
> London
YES
> , Paris
NO
```

If a large window size is used (e.g. 100 in Winowing [4]), this duplication is not guaranteed to be detected. Smaller window size, however, results in larger set of fingerprints. Thus, there is a trade of between space and precision that we have to decide. In our experiment, window size of 3 gives best precision.

Window Size	1	2	3	4	5	6	7	8	9	10
Average size of Fingerprints	250	166	123	97	81	68	60	53	48	43

Figure 10: Average size of documents's fingerprints

3.7. Compare Fingerprints

Comparison of two documents' fingerprints should be independent with the order these fingerprints were selected since re-ordering of paragraphs or sentences can result in new selection order.

Firstly, we sort fingerprints of each document in ascending order. This is to ensure that re-ordering of document's substrings does not exit our detection. Then we perform one-by-one comparison between two fingerprints of two documents. Our comparison runs from the smallest fingerprint (hash) to the largest until one of two documents run out of fingerprint. The total number of matches is then returned to the calling function.

Chapter 4

EMAIL HANDLING

This chapter discusses various email-related issues encountered in this project.

4.1. Client-side or Server-side solution

Electronic Mail originally referred to “technologies that allowed people to send documents to one another through electronic means” [13]. Nowadays, “email” generally equals “network email” which means “the asynchronous transmission of messages by using computers and data-communication networks” [13]. That brings up a question: Client-side or server-side solution?

The ultimate goal of this project is to detect redundant emails and it can be performed either on client side or server side or both. However, we feel that it is the client-side solution that offers the most flexibility to user. For instance, many email service providers impose storage limit on their user’s accounts. A server-side solution, if ever built, is only able to operate on a limited number of messages while a client-side one can provide the same service on a much larger number of messages.

4.2. Email Library

There are quite a few protocols defining how emails are transferred across network. Among them, POP3 (Post Office Protocol, RFC 1939) and IMAP4 (Internet Message Access Protocol, RFC 3501) are the most prevalent protocols for email retrieval. In addition to basic services, these protocols also support authentication and security. More than a handful of RFC documents were written for them. A full implementation of IMAP at <http://www.washington.edu/imap/> requires 36 RFCs!

Parsing email message is also a tedious job. Format of a message is defined in RFC 2822 and extended through various other documents including those describing MIME (Multimedia Internet Mail Extension). The introduction of MIME allows an email message to contain “text in character sets other than ASCII, non-text attachments,

multipart message bodies and header information in non-ASCII character sets” [14]. However, it complicates the parsing process.

Since implementing these protocols and parsing email message can be a very tedious job and we need to focus on our duplicate detection component, we decided to use ready built email library instead. We found VMime¹, a powerful email library written in C++. VMime has support for POP3, IMAP4, SASL authentication and various email extensions including MIME. VMime is used in our project to implement email parsing and retrieving functions.

¹ <http://www.vmime.org/index.shtml>

Chapter 5

IMPLEMENTATION & EVALUATION

5.1. Overall Design

Figure 11 captures an overview of our system. Our system composes of three main components: Mail Client responsible for retrieval emails from mail server, Mail Store responsible for storing and managing email messages on local file system and Duplicate Detection responsible for duplicate check. The first two components both have access to Profile objects in order to differentiate different email accounts. Duplicate Detection can be invoked through Mail Store.

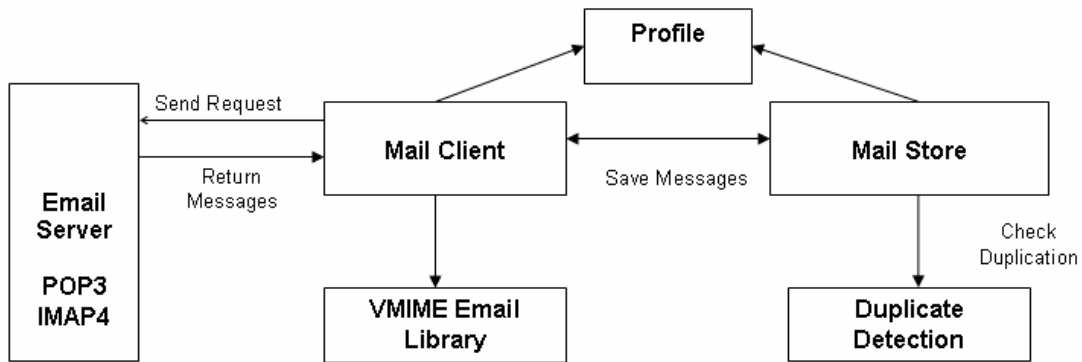


Figure 11: Overall Design

A sample run of our system would be:

- Connect to email server using Mail Client component.
- Download email messages using Mail Client and save them to local file system using Mail Store
- Run duplicate check on downloaded mail messages by invoking Duplicate Detection component via Mail Store.

5.2. Duplicate Check

Figure 12 presents three main stages in duplicate detection process. Namely they are:

- *Preprocessing*: A vector of email messages (documents) is passed in as input. Each message is then tokenized into words; all *stop words* are cleaned out. Each word in a document maps to an entry in a dictionary which is build concurrently. Each entry associates with a unique odd number. The largest odd value in the dictionary is used to compute a used in hash function (see **Section 3.5**)
- *Fingerprinting*: A vector of words obtained from each document after preprocessing stage is passed to fingerprint function which performs two operations: hashing and select fingerprints. **Figure 9** shows code for fingerprint function.
- *Comparing fingerprints*: Each document fingerprints are compared against others. The number of matches between message A and B is chosen to be *overlap (A, B)*, while the total number of fingerprints of message A is *content (A)*. By dividing *content (A)* by *overlap (A, B)* we get *containment (A, B)*. If *containment (A, B)* is greater than a threshold τ , A is said to be made redundant by B. List of messages being made redundant by others are output to file output.txt by default.

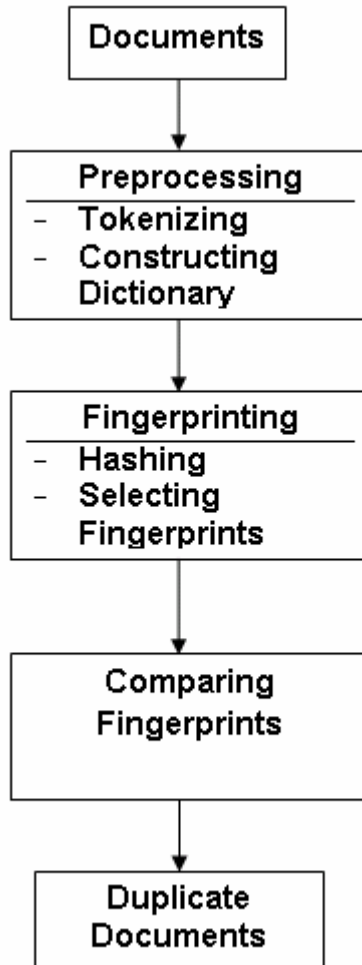


Figure 12: Duplicate Check

5.3. Evaluation Setting

In order to test our system, we compare result from our system (output.txt) versus manual classification (ManualClassification.txt). **Figure 13** shows an example of how a test result looks like.

```

1   Finish loading ManualClassification.txt.
2   Total = 128
3   Finish loading output.txt
4   Total = 127
5   Fail to find email with ID =
6     <18046211.1075861682008.JavaMail.evans@thyme>
7   Size does not match for email with ID =
8     <8089597.1075858304759.JavaMail.evans@thyme>,
9     output.txt: 2, ManualClassification.txt: 3
10  Fail to find email with ID =
11   <8969154.1075861682403.JavaMail.evans@thyme>
12  Correct   = 124
13  Precision = 0.976378
  
```

14 Recall = 0.96875

Figure 13: Sample test result

Line 1 and 2 tell us that the checking program has finished loading ManualClassification.txt and find 128 redundant emails. Similarly, in line 3 and 4 checking program find 127 redundant email in our system output (output.txt). Line 5 and 6 say that email with ID = 18046211.1075861682008.JavaMail.evans@thyme is not found in the ManualClassification.txt, which indicate a false positive. Line 7, 8 and 9 say that our system only found 2 other messages that made email with ID = 8089597.1075858304759.JavaMail.evans@thyme redundant, while it should have found 3 as required by ManualClassification.txt. Finally line 12, 13 and 14 output total number of correct classifications, precision and recall.

Since window size and threshold value can vary, we would like to measure our system's precision and recall against these two variables. Our test data consists of 143 email messages belong to 4 categories: Identical Content (20 messages), Almost identical with subtle differences (16 messages), Arbitrarily component-rearranged messages (20 messages) and Message quoted inside another (Category 4 and 5, **Appendix A**) (87 messages). Average running time of our program is 5 seconds.

5.4. Window size versus Precision and Recall

Figure 14 and **Figure 15** show the changes of precision and recall with variable window size and fixed threshold 95%.

Windows Size	1	2	3	4	5	6	7	8	9	10
Precision (%)	97.14	97.14	97.16	97.14	97.08	94.78	94.74	96.24	96.24	95.42
Recall (%)	96.45	96.45	97.16	96.45	94.33	90.07	89.36	90.78	90.78	88.65

Figure 14: Window size vs. Precision & Recall, threshold = 95%

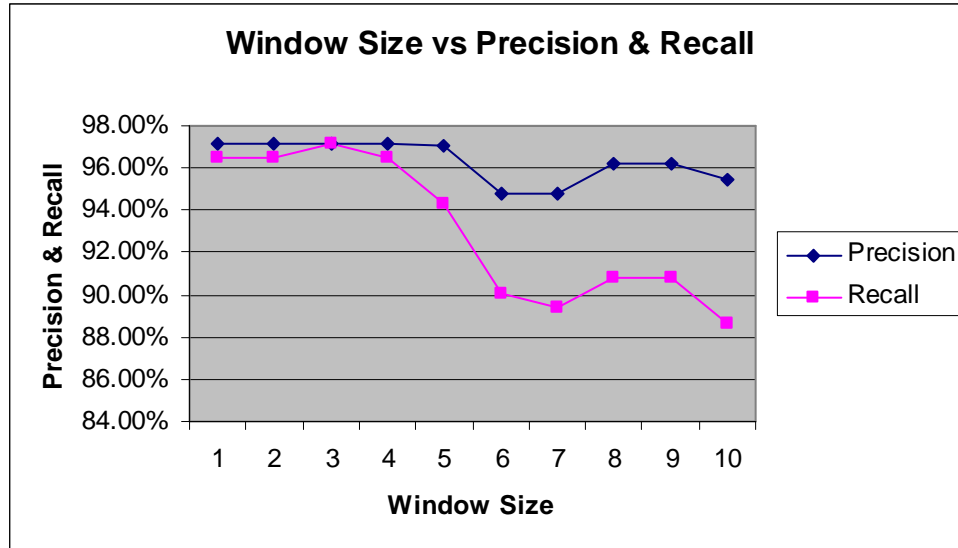


Figure 15: Windows Size vs. Precision & Recall

As we have seen in **Figure 15**, a window with size 3 results in best precision and recall. While precision is relatively high and stable in window size of range 1 – 5, it drops significantly when size get larger than 5. Change in recall, even though seems to follow precision’s pattern, reacts stronger to change in window size. Perhaps loss of sensitivity due to larger window size (our method can detect matches between sequence of length at least $w + 1$, **Section 3.6**) accounts for this strong reaction.

5.5. Threshold Value versus Precision and Recall

Figure 16 and **Figure 17** show the changes of precision and recall with variable threshold and fixed window of size 3.

Threshold	85	88	90	92	95	96	97	100
Precision (%)	89.74	91.50	92.11	94.48	97.16	97.04	99.20	100
Recall (%)	99.29	99.29	99.29	97.16	97.16	92.91	88.65	68.09

Figure 16: Threshold vs. Precision & Recall, Window size = 3

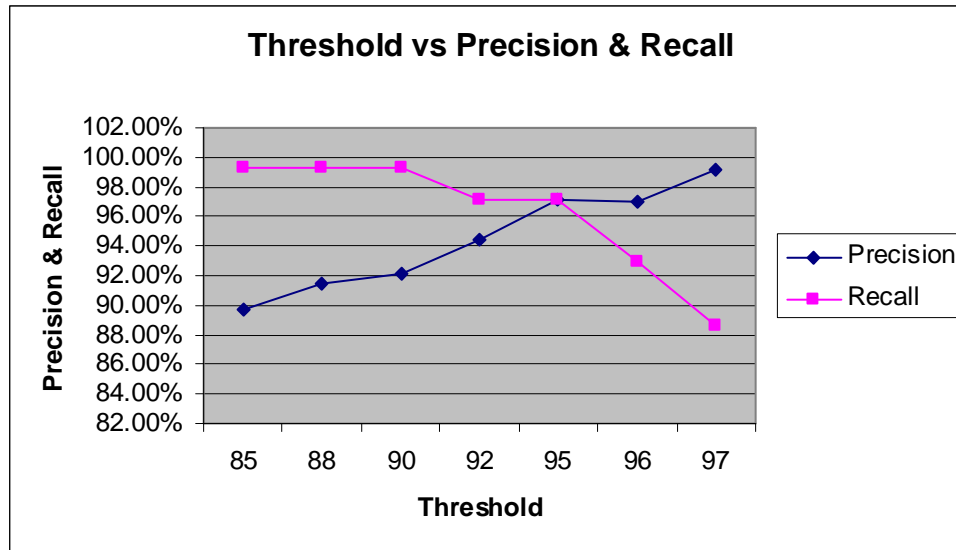


Figure 17: Threshold vs Precision & Recall

The graph in **Figure 17** shows two opposite trends of precision and recall. Threshold value 95% seem to give us most balanced and satisfied performance. Lower value of threshold results in more false positive but higher recall; in contrast, a higher value of threshold gives lower false positive and lower recall.

It is noticeable that when threshold value is set to 100%, precision reaches 100% and recall drops significantly to 68.09% (not shown in graph). In this setting, our system will not allow any difference between messages. However, in spite of high precision, a 100% threshold is not favorable since it offers minimal help to user.

5.6. False Positive

False positive is seen in long emails. The duplicated content between two long emails is so large that it containment measure surpasses threshold value. For example, we encounter a false report of redundant email of two messages:

Message 1:

those bad Dynegy seeds are infecting already.

-----Original Message-----

From: =09Forney, John M. =20

Sent:=09Thursday, November 15, 2001 10:26 AM

To:=09Twiggs, Thane
Subject:=09RE: ASAP please: ERCOT Questions OOMC & OOME

I kind of lost my temper a bit with Mark. Probably wont sound good on tap but I told him that a monkey-man wouldnt interpret the protocols that way. Its the Dynegy thang talking.

JMF

(220 duplicated lines)

Message 2:

I wanted to get a response in writing rather than verbal so I sent an e-mail to Kent Saathoff. I will call and prod a response again today.

-----Original Message-----

From: =09Portz, David =20
Sent:=09Friday, November 16, 2001 10:56 AM
To:=09Twiggs, Thane
Cc:=09Ryall, Jean; Forney, John M.; Nicolay, Christi L.; Gilbert-smith, Dou=
g
Subject:=09RE: ASAP please: ERCOT Questions OOMC & OOME

Thane -- I think word back from you is the next step needed in this process. If ERCOT's position is that within the text of the Protocols, OOMC means "running", then unacceptable risks are presented to our Customer and to EPMI as their QSE. If such is ERCOT's position, then they should cite us chapter and verse for that position within the Protocols, offsetting the clear implications for the contrary position in the protocols' other ancillary services products, in the Operating Guides, the Market Guide, ERCOT's prior

operating and marketing guides (predating this marketplace), and based on customary industry understanding of the term "capacity". -David
(220 duplicated lines)

Their difference as shown here is not enough to lower their containment measure which is 96.6%, higher than our threshold 95%. We expected that more false positive will occurs on long email with a large duplicated body.

5.7. False Negative

False negative occurs on small messages with subtle different. An example of false negative encountered in our evaluation is shown below.

Message 1:

Dear all,

Skilling will be speaking at the National Press Club next week. He'll give overview of Enron's business, talk about what's going on in power markets, and what should be done to fix the current problems.

California will get covered as well. Other than what we have already discussed, is there anything Jeff should know or address in his remarks?

Message 2:

Skilling will be speaking at the National Press Club next week. He'll give overview of Enron's business, talk about what's going on in power markets, and what should be done to fix the current problems.

California will get covered as well. Other than what we have already discussed, is there anything Jeff should know or address in his remarks?

Because the number of fingerprint of message 2 is so small such that even a mismatch will cause its containment to fall below threshold value: In this case, the number of matches is 7 over a total of 8 fingerprints. That leads to $7 / 8 = 87.50\%$ smaller than threshold 95%.

A good list of stop words can help reduce this type of false negative. By adding “dear”, “I” and “will” into the list, we have significantly reduce the number of false negative. However, it is not always possible to do so because some words contain rich meaning that an insertion of such words to stop word list can cause serious information loss.

Chapter 6

POSSIBLE IMPROVEMENT

Even though evaluation result is very good, lots of things can still be improved. The existence of false positive and false negative are indeed challenges to us. We list here a few possible improvements.

Current implementation of our duplicate detection component aims to work for arbitrary type of document. We expect that an optimization based on understanding email characteristic would reduce the number of false positive. For example, if we can group conversation emails together into thread, like what Gmail does, we can isolate them from other messages and impose higher threshold. This promises better elimination of false positive comparing to current implementation. Another improvement can be achieved is about including attachment in duplicate check. Different checksum values of message's attachments suggest messages are not redundant.

In addition, further preprocessing of email message can be of great benefit. Stemming can be incorporated into our current preprocessing stage to reduce the number of entries in the dictionary and further enhance the sensitivity of our program. Consider two false negative messages with almost identical content except for one uses "introduce" and the other uses "introduces", probably due to typing mistake. Applying stemming would allow our program to detect this duplication and avoid false negative. Going further, if our program can correct simple typo error like "introdcue" to its correct form "introduce", our system can produce much better result. It is feasible, as demonstrated in Microsoft Word program.

A further improvement is to enable detection of one message made redundant by two or more other messages. Currently, we did not implement that feature in our system, but the approach is simple. We build a database consist of all fingerprints of our messages. Each fingerprint associates with a set of documents containing that fingerprint. For a message

to be checked, we search each of its fingerprint in our database and record the set of documents associated with that fingerprint. A union of all recorded set of documents is the desired result.

Chapter 7

CONCLUSION

In brief, we have presented our duplicate detection technique as a good candidate solution to *redundant messages* problem. We introduce the notion of word as a basic unit in a message and invent a novel hash function that encodes information about succeeding word in the hash value. We have also proved that our hash function is collision-free and verify that conclusion by experimenting on the entire collection of RFC documents.

We also discuss the evaluation of our system against various type of redundancy. The obtained result is promising in spite of the existence of falsity. We have analyzed and suggest several possible improvements that can further enhance performance and reduce error.

Furthermore, our duplicate detection satisfies all three important properties of a duplicate detection method. This promises opportunities to deploy our technique on many different scenarios, including eliminating redundant messages in Bulletin board, newsgroup...

REFERENCE

- [1] Douglas M. Campbell, Wendy R. Chen, Randy D. Smith. *Copy Detection Systems for Digital Documents*. Advances in Digital Libraries, 2000.
- [2] Abdur Chowdhury. *Duplicate Data Detection*. <http://ir.iit.edu/~abdur/Research/Duplicate.html>
- [3] “duplicate”. WordNet 1.7.1, Princeton University. Published by Princeton University. <http://www.answers.com/topic/duplicate>
- [4] Saul Schleimer, Daniel S. Wilkerson, Alex Aiken. *Winnowing: Local Algorithms for Document Fingerprinting*. ACM SIGMOD 2003.
- [5] Andrei Z. Broder. *On the resemblance and containment of documents*. SEQS: Sequences '91, 1998
- [6] School of Information Management and System, University of California at Berkeley, *How much Information in 2003*. <http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/>
- [7] ClearContext Corporation. *2005 Email Usage Survey*. http://blog.clearcontext.com/2005/04/making_email_wo.html
- [8] Sergey Brin, James Davis, Hector Garcia-Molina. *Copy detection mechanisms for digital documents*. Proceedings of ACM SIGMOD Conference, pages 398-409, 1995.
- [9] Udi Manber. *Finding similar files in a large file system*. Proceedings of the USENIX Winter 1994 Technical Conference, page 1-10, 1994.
- [10] Nevin Heintze. *Scalable document fingerprinting*. 1996 USENIX Workshop on Electronic Commerce, November 1996.
- [11] Gonzalo Navarro. *A guided tour to approximate string matching*. ACM Computing Survey, March 2001.
- [12] *Enron Corpus*. <http://www.cs.cmu.edu/~enron/>
- [13] "e-mail." McGraw-Hill Encyclopedia of Science and Technology. The McGraw-Hill Companies, Inc., 2005. *Answers.com* 01 Apr. 2007. <http://www.answers.com/topic/e-mail>

- [14] "MIME". *Wikipedia*. Retrieved April 01, 2007, from Answers.com Web site:
<http://www.answers.com/topic/mime-1>"Maildir."
- [15] "Maildir". *Wikipedia*. *Wikipedia*, 2007. *Answers.com* 02 Apr. 2007.
<http://www.answers.com/topic/maildir>
- [16] "mbox." *Wikipedia*. *Wikipedia*, 2007. *Answers.com* 02 Apr. 2007.
<http://www.answers.com/topic/mbox-1>
- [17] Kwok Chong See, Wong Lim Soon. *Method for eliminating and identifying redundant message information*. European Patent No 1327192.
- [18] Xiao-lin Wang, Ian Cloete. *Learning to classify email: a survey*. Proceeding of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou 2005.

Appendix A: Derivation of a and b in Section 3.5

Consider the following two equations:

$$ax + by = E \quad (1)$$

$$xy = F \quad (2)$$

With a, b, x, y, E, F are integers; $a, E > 0$; x, y are positive odd number.

Since a value of F in equation (2) can be traced back to two different pair of x and y : (X, Y) and (Y, X) , equation (2) cannot be used to compute hash function. We are interested in using equation (1). If we can find a value of a and b such that E can only have one possible pair of x and y , equation (1) will be used to compute our hash function, E will be the resulted hash value.

From equation (2):

$$x = F/y \quad (3)$$

From equation (1) and (3):

$$aF/y + by = E$$

$$\Leftrightarrow by^2 - Ey + aF = 0 \quad (4)$$

Since x can be computed from y , we will find a and b such that equation (4) have exactly one positive odd number solution.

Substitute $y = 2z + 1$ into equation (4), we get:

$$(4) \Leftrightarrow b(2z + 1)^2 - E(2z + 1) + aF = 0$$

$$\Leftrightarrow 4bz^2 + (4b - 2E)z + (b - E + aF) = 0 \quad (5)$$

Consider: $X = (4b)(b - E + aF)$. As we all know, if $X < 0$ equation (5) have exactly one positive solution z , which map to a unique positive odd y .

By choosing $b = -1$, we have:

$$X = (-4)(-1 - E + aF)$$

$$E = ax - y$$

$$F = xy$$

Substitute E and F into X, we get:

$$\begin{aligned} X &= (-4)(-1 - ax + y + axy) \\ &= (-4)(y - 1)(1 + ax) \end{aligned}$$

Thus we have $X < 0$ for all valid x , y and a .

Since E must be positive:

$$E = ax - y > 0$$

Or equivalently

$$a > y / x$$

Consider case where a hash collision occurs:

$$\begin{aligned} ax_1 - y_1 &= ax_2 - y_2 \\ \Leftrightarrow a(x_1 - x_2) &= y_1 - y_2 \\ \Leftrightarrow a &= (y_1 - y_2) / (x_1 - x_2) \end{aligned}$$

Since $y_1 - y_2 < \text{largest}(y)$ and $x_1 - x_2 \geq 2$
 $(y_1 - y_2) / (x_1 - x_2) \leq \text{largest}(y) / 2$

Thus if we choose $a = \text{Ceil}(\text{largest}(y) / 2)$, hash collision will never occurs.

Conclusion:

$$\begin{aligned} b &= -1 \\ a &= \frac{\text{largest}(y)}{2} + 1 \end{aligned}$$

Appendix B: Examples of Each Duplicate Category

Category 1: Identical content

Message 1:

Message-ID: <15928422.1075855993904.JavaMail.evans@thyme>
Date: Fri, 17 Nov 2000 00:50:00 -0800 (PST)
From: heidi.mason@enron.com
Subject: Reporting Line for Sydney Risk Officer
Cc: sally.beck@enron.com
Bcc: sally.beck@enron.com
Scott

With all our reporting changing to London Office and now it appears some restructure of risk and the broadening of Sally's role, what we are going to do for the second report for our new risk manager, Justin Den Hertog - he has taken over from Alan.

With thanks

Message 2:

Message-ID: <43928422.107585599387.JavaMail.evans@thyme>
Date: Fri, 17 Nov 2000 00:50:00 -0800 (PST)
From: heidi.mason@enron.com
Subject: Reporting Line for Sydney Risk Officer
Cc: sally.beck@enron.com
Bcc: sally.beck@enron.com
Scott

With all our reporting changing to London Office and now it appears some restructure of risk and the broadening of Sally's role, what we are going to do for the second report for our new risk manager, Justin Den Hertog - he has taken over from Alan.

With thanks

Category 2: Almost Identical content with subtle difference

Message 1:

Message-ID: <43928422.107585599387.JavaMail.evans@thyme>
Date: Fri, 17 Nov 2000 00:50:00 -0800 (PST)
From: heidi.mason@enron.com
Subject: Reporting Line for Sydney Risk Officer
Cc: sally.beck@enron.com
Bcc: sally.beck@enron.com
Scott

With all our reporting changing to London Office and now it appears some restructure of risk and the broadening of Sally's role, what we are going to do for the second report for our new risk manager, Justin Den Hertog - he has taken over from Alan.

With thanks

Message 2:

Message-ID: <199284238.1075855993904.JavaMail.evans@thyme>
Date: Fri, 17 Nov 2000 **01:20:00** ~~00:50:00~~ -0800 (PST)
From: heidi.mason@enron.com
Subject: Reporting Line for Sydney Risk Officer
Cc: sally.beck@enron.com
~~Bcc: sally.beck@enron.com~~

Dear Scott,

With all our reporting changing to London Office ~~and~~ now it appears some restructure of risk and ~~the~~ broadening of Sally's role, **so** what we are going to do for the second report for our new risk manager, Justin Den Hertog - he has taken over from Alan.

~~With thanks~~

Thanks

Category 3: Arbitrarily component-rearranged messages

Message 1:

Message-ID: <4209920.1075861681494.JavaMail.evans@thyme>
Date: Thu, 15 Nov 2001 05:33:13 -0800 (PST)
From: allan_taylor@anadarko.com
To: m..forney@enron.com
Subject: Course offer

- 1) Fundamentals of Energy Futures, Options & Derivatives
- 2) Fundamentals of the Electric Power Industry
- 3) Gas-to-Electricity Arbitrage & How to Maximize the Profitability of Electric Generation Assets
- 4) Developing Effective Risk Management Policies & Procedures (John Wengler)
- 5) Fundamentals of Statistical Analysis (Dr. Ken Skinner)
- 6) How to Value Electric Generation Assets as Real Options
- 7) Fundamentals of Value-at-Risk (Soli Forouzan)

Message 2:

Message-ID: <1329920.1075861681494.JavaMail.evans@thyme>
Date: Thu, 15 Nov 2001 06:33:13 -0800 (PST)
From: allan_taylor@anadarko.com
To: m..forney@enron.com
Subject: Course offer

- Fundamentals of Energy Futures, Options & Derivatives
- Fundamentals of the Electric Power Industry
- **Developing Effective Risk Management Policies & Procedures (John Wengler) (item 3 and 4 switch their position)**
- **Gas-to-Electricity Arbitrage & How to Maximize the Profitability of Electric Generation Assets**
- Fundamentals of Statistical Analysis (Dr. Ken Skinner)
- **Fundamentals of Value-at-Risk (Soli Forouzan) (item 6 and 7 switch their position)**
- **How to Value Electric Generation Assets as Real Options**

Category 4: Message quoted inside another

Message 1:

Message-ID: <30675736.1075840054052.JavaMail.evans@thyme>
Date: Thu, 13 Dec 2001 15:15:07 -0800 (PST)
From: alan.comnes@enron.com
To: alan.comnes@enron.com, tim.belden@enron.com
Subject: RE: ISO Disbursed \$404 million this morning

I am told that ISO disbursed to participants the CERS monies it received last week for trade month Februrary '01 for approx. \$404 million. Every week another month is supposed to be processed but I beleive Feb was the biggest month.

Alan

Message 2:

Message-ID: <23733199.1075840053967.JavaMail.evans@thyme>
Date: Fri, 14 Dec 2001 20:05:14 -0800 (PST)
From: legal <.hall@enron.com>
To: alan.comnes@enron.com, tim.belden@enron.com
Subject: RE: ISO Disbursed \$404 million this morning

Of this amount, we received approximately \$536,000.

-----Original Message-----

From: Comnes, Alan
Sent: Thursday, December 13, 2001 11:15 AM
Subject: ISO Disbursed \$404 million this morning

I am told that ISO disbursed to participants the CERS monies it received last week for trade month Februrary '01 for approx. \$404 million. Every week another month is supposed to be processed but I beleive Feb was the biggest month.

Alan

Category 5: Combination of the first three with the forth category

Message 1: (As in category 4)

Message 2:

Message-ID: <23733199.1075840053967.JavaMail.evans@thyme>
Date: Fri, 14 Dec 2001 20:05:14 -0800 (PST)
From: legal <.hall@enron.com>
To: alan.comnes@enron.com, tim.belden@enron.com
Subject: RE: ISO Disbursed \$404 million this morning

Of this amount, we received approximately \$536,000.

-----Original Message-----

From: Comnes, Alan
Sent: Thursday, December 13, 2001 11:15 AM
Subject: ISO Disbursed \$404 million this morning

I am told that ISO disbursed to participants the CERS monies it received last week for trade month Februrary '01 for approx. \$404 million. Every week another month is supposed to be processed but I beleive Feb was the biggest month.

Alan

Message 3:

Message-ID: <23733199.1075840053967.JavaMail.evans@thyme>
Date: Fri, 14 Dec 2001 20:05:14 -0800 (PST)
From: legal <.hall@enron.com>
To: alan.comnes@enron.com, tim.belden@enron.com
Subject: RE: ISO Disbursed \$404 million this morning

> I am told that ISO disbursed to participants the CERS
> monies it received last week for trade month Februrary
> '01 for approx. \$404 million.
Of this amount, we received approximately \$536,000.

-----Original Message-----

From: Comnes, Alan

Sent: Thursday, December 13, 2001 11:15 AM

Subject: ISO Disbursed \$404 million this morning

> Every week another month is supposed to be processed but

> I beleive Feb was the biggest month.

>

> Alan