Scalable Discovery of Hidden Emails from Large Folders

Giuseppe Carenini, Raymond Ng, Xiaodong Zhou
Department of Computer Science
University of British Columbia, Canada
{carenini, rng, xdzhou}@cs.ubc.ca

ABSTRACT

The popularity of email has triggered researchers to look for ways to help users better organize the enormous amount of information stored in their email folders. One challenge that has not been studied extensively in text mining is the identification and reconstruction of hidden emails. A hidden email is an original email that has been quoted in at least one email in a folder, but does not present itself in the same folder. It may have been (un)intentionally deleted or may never have been received. The discovery and reconstruction of hidden emails is critical for many applications including email classification, summarization and forensic. This paper proposes a framework for reconstructing hidden emails using the embedded quotations found in messages further down the thread hierarchy. We evaluate the robustness and scalability of our framework by using the Enron public email corpus. It contains about half a million emails, distributed in folders belonging to 150 users. Our experiments show that hidden emails exist widely in the Enron corpus and also that our optimization techniques are effective in processing large email folders.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms, Performance, Experimentation

Keywords

text mining, hidden email, forensics, word index

1. INTRODUCTION AND MOTIVATION

The popularity of email has triggered researchers to look for ways to help users better organize and use their mail folders, e.g., classification [7], task management [4] and user interface [9]. One key difference between emails and other types of documents is the threaded nature of emails. According to one study, over 60% of emails are threaded [5].

In this paper, we study a problem largely overlooked in text mining - the problem of discovering and reconstructing hidden emails. A hidden email is an original email that has been quoted in subsequent emails kept in a user's folder, but is not itself present in the folder(s). Anyone who has ever managed a folder is accustomed to the tedium of manually shunting messages between folders, as well as deciding which messages to keep and which to delete. Accidental deletion often occur. Hidden emails also occur when new recipients are included in an existing thread. Whether the original email was deleted or never existed, it still may be found in the quotation of subsequent emails.

Beyond helping users to better manage their ever growing folders, hidden email reconstruction also find many forensic applications. For instance, emails may be deliberately deleted by a malicious user to avoid certain information to be revealed. In this case, investigators will need to reconstruct parts of the deleted emails that may be quoted in other emails stored in other people's folders.

The problem this paper attempts to solve is: *how hidden emails can be discovered and reconstructed using the embedded quotations found in messages further down the thread hierarchy.*

As a concrete case study, we report in this paper the results of applying our framework to the Enron email corpus. The Enron email dataset was made public by the US Federal Energy Regulatory Commission during the ex-Enron investigation. To the best of our knowledge, it is the biggest public accessible email dataset, and has shown to be of great value for research on many areas, such as email classification. The two key purposes of using this dataset are to verify the robustness of our method using real data, and to ascertain its scalability.

While we discuss related work in Section 2, we present in Section 3 a bulletized model of a reconstructed email that accounts for partial ordering. The email is built from a precedence graph which represents the relative orders among hidden fragments. We present a necessary and sufficient condition for each component of the precedence graph to be represented as a single bulletized email. A basic algorithm is then given to reconstruct hidden emails from the graph.

While this basic framework can deliver the functionality, it cannot deliver the efficiency, nor the robustness, to deal with large real folders. In Section 4, we develop two ways to optimize the performance of the algorithms. Both are based on the use of word-indexing. They are designed to scale up to large folders and long emails.

In Section 5, we report a comprehensive case study based on the Enron dataset. We examine the prevalence of hidden emails and fragments to assess the importance of the hidden email reconstruction problem. We evaluate the robustness of our algorithm when applied to real folders. Last but not least, we evaluate the success of our optimizations in scaling up to the needs of large folders and long emails.

2. RELATED WORK

From a research perspective, email and newsgroups differ from traditional documents in many aspects. For example, there is a high level of hierarchical and referential relationship among emails in any folder, i.e., document threading. This relationship has caught the attention of many researchers. In [6], Lam et al. propose to summarize a set of emails based on their threading hierarchy. They mention the existence of deleted emails in the hierarchy, which create a complication not present in newsgroups. However, they do not study how to regenerate those deleted emails.

In the closely related area of newsgroup summarization, Newman [8] indicates the possibility of orphaned quotations and warns that applications such as classification and summarization would be adversely affected as a result, but does not explore the issue further.

Carvalho et al. [2] studied the problem of signature and quotation detection within an email. Their work can help us find the right content in the process of quotation identification.

In [1], we present a preliminary report on the hidden email discovery and regeneration problem using a small synthetic dataset. Section 3 gives an overview of the underlying model and summarizes the key results from that report. However, the basic algorithm in Section 3 does not scale up to large real datasets and is not robust enough to deal with the Enron dataset. Thus, the key contributions of the work presented here is to devise robust and scalable algorithms, as well as to evaluate them rigorously with real email folders.

Our research on the reconstruction of missing emails is also relevant to the area of document forensics, where document reconstruction from fragments is crucial. For instance, Shanmugasundaram et al. in [10] propose the reconstruction of a totally ordered digital evidence from randomly scattered fragments. With respect to our goal of reconstructing the hidden email, as well as in document forensics, a total order is not always possible. Forcing one where none exists may be incorrect and even misleading. We believe that a partial order representation, i.e., the bulletized model, constitutes a reasonable solution that adequately compromises between accuracy and completeness concerns.

3. A BASIC FRAMEWORK FOR RECONSTRUCTING HIDDEN EMAILS

For any given email folder, some emails may contain quotations from original messages that do not exist in the same folder; the originals may have been deleted or were never received at all. Each of those quotations is considered a *hidden fragment*, as part of a *hidden email*. Several hidden fragments may all originate from the same hidden email, and a hidden fragment may be quoted in multiple emails. Our goal is to reconstruct hidden emails from the hidden fragments by finding the relative ordering of the fragments and, where possible, piecing them together.

Figure 1 shows the overall algorithm called HiddenEmailFinder, which integrates the framework proposed in our previous paper [1]. In HiddenEmailFinder, we first describe how we identify hidden fragments(Step 1&2). Then we discuss how we create and use a *precedence graph* to represent hidden fragments(Step 3). After that, we describe how to reconstruct hidden emails based on a *bulletized email model*(Step 4&5). Moreover, in [1], we also give precise conditions under which (parts of) a precedence graph can be transformed into a bulletized email.

```
Algorithm HiddenEmailFinder
Input: email folder $MF$, and reference email folders $RF_1, \dots, RF_k$
Output: a set of bulletized hidden emails in $MF$

  1. For each mail $M \in MF$, extract all the quoted fragments.
  2. For each quoted fragment $F$, match $F$ against all emails in $MF$
     as well as those in $RF_1, \dots, RF_k$. In particular, identify the
     LCS between $F$ and $M$ for every email $M \in MF$ or $M \in RF_i$
     for some $i$. See Section ??. Depending on the length of the
     LCS, $F$ may be declared hidden or not.
  3. Find possible overlaps between hidden fragments as described
     in Section 3.2, split them if necessary, and create a precedence
     graph $G$.
  4. Decompose $G$ into its weakly connected components.
  5. For each weakly connected component, do:
       (a) Process $G$ with Algorithm graph2email as described in
           Figure 4. If the graph is complete and strict, output the
           reconstructed hidden emails. Otherwise, use the heuris-
           tics described in Section 3.5 to deal with non-strictness
           and/or incompleteness. Output the reconstructed hidden
           emails.
```

Figure 1: A Skeleton of Algorithm HiddenEmailFinder

4. OPTIMIZATIONS FOR LARGE FOLDERS

Even though HiddenEmailFinder delivers all the required functionalities, a preliminary experimental evaluation revealed two bottlenecks in the hidden fragment identification steps (1, 2 in Figure 1) when dealing with large folders and long emails. The first bottleneck is due to the large number of matches that may need to be performed between quoted fragments and other emails in the folders, while the second bottleneck is due to how well LCS matching is performed (as discussed in Section 3.1 in [1]). Below we describe two optimizations to overcome these bottlenecks.

4.1 Email Filtering by Indexing

Step 2 of HiddenEmailFinder requires that a quoted fragment F be matched against every single email M in the primary folder MF, as well as with every email in the reference folders $RF_1, \dots, RF_k$. Note that this is a slight generalization of the hidden fragments defined in [1]. While HiddenEmailFinder finds only hidden fragments and emails in the folder MF, a fragment F is hidden only if it cannot be found in MF, as well as not found in any of the reference folders $RF_1, \dots, RF_k$. The reference folders are useful because a fragment may be hidden from MF simply because the user filed the original email into another folder. Thus, to take out the complication introduced by email classification, HiddenEmailFinder can take as input multiple reference folders for additional checking.

The matching in step 2 stops when either a match is found

```
Algorithm EmailFiltering
Input: the word index, the frequent word list $FW$, a quoted fragment $F$
Output: a list of email ids possibly matching $F$

  1. Tokenize $F$ to a set of words $w$, with all the stop-words removed.
  2. For each $w$ not in the list $FW$, use the word index to identify $L_w$.
  3. Return the unioned list, i.e., $\cup_{w \in F \wedge w \notin FW} L_w$.
```

Figure 2: A Skeleton of Algorithm EmailFiltering

(in which case the quoted fragment F is not a hidden fragment), or a match is not found anywhere (in which case F is considered hidden). When the folders are small, a straightforward string comparison is acceptable. But this does not scale up to large folders.

The first optimization is to use a word index. Each index entry is of the form: $\langle w, L_w \rangle$, where w is a word in the email corpus, and L_w is a list of ids of emails containing at least one occurrence of w. For example, the word "available" may have the following entry in the word index: $\langle available, \langle id = 17, id = 278 \rangle\rangle$. Like information retrieval systems, the word index does not contain high frequency closed-class terms (i.e., stop-words) such as the definite article "the". In general, the index is created by one complete pass over all the emails in the corpus.

Given the word index, and a quoted fragment F to be matched, the quoted fragment is first tokenized to words with all the stop-words removed. Then for each word w, the word index is used to return the list L_w. To support LCS matching between a quoted fragment and an email, a match is allowed even if not all the words are found in the email. Thus, we take the union of the lists, i.e., $\cup_{w \in F} L_w$. This filtering process guarantees no false dismissals in the sense that only emails in the unioned list can ever match F (unless F is made up of stop-words only!).

Figure 2 shows a skeleton of this process. It incorporates an additional optimization to reduce the size of the unioned list $\cup_{w \in F} L_w$. Specifically, it further excludes the most frequent open-class words (i.e., non-stop-words) in the corpus. Hereafter, we denote this list of words as FW. In other words, we only obtain the unioned list $\cup_{w \in F \wedge w \notin FW} L_w$. We define the length of FW as *frequent word threshold* ft, i.e., the top-ft most frequent words are kept in FW. In Section 5.4, we show that the choice of frequent word threshold has a great impact on the runtime.

4.2 LCS-Anchoring by Indexing

While the email filtering algorithm reduces the number of emails to be matched against F, we still need to optimize how well each match can be performed. Recall from Section 3.1 that we extract the longest common substring (LCS) between F and the email currently being checked against. To scale up this part, it is important to review why we choose LCS to begin with.

Let us say that the original email is a sequence of fragments $OM = \langle F_1, F_2 \ldots, F_5 \rangle$. When the user quotes this email, the user might perform various actions to this sequence, as she can edit the fragments as free text. She can quote the exact sequence verbatim; or she can delete the beginning and/or the end parts (e.g., $QF_1 = \langle F_2, F_3, F_4 \rangle$). In a more sophisticated setting, she may quote $QF_2 = \langle F_2, F_4 \rangle$ to reduce the length. Furthermore, she may copy another fragment F_6 from another email to form $QF_3 = \langle F_2, F_6, F_4 \rangle$.

```
Algorithm LCS-Anchoring
Input: the word index, the frequent word list $FW$, a quoted fragment $F$, and an email $M$
Output: the LCS between $F$ and $M$

  1. Tokenize $F$ to a set of words $w$, removing the stop-words and keeping only those not in $FW$.
  2. If $M$ does not appear in any of the lists $L_w$ for all the remaining $w$'s, return the empty string.
  3. Otherwise, for each such $w$,
     (a) for each anchor position $pos_i$
         i.  Align $w$ at $pos_i$ in $M$ and $F$.
         ii. Expand the matched substring forward and backward as much as possible.
  4. Return the longest matched substring in the nested loop.
```

Figure 3: A Skeleton of Algorithm LCS-Anchoring

So given a quoted fragment, the task is to match it against other emails. In the case of QF_1, a simple substring searching is sufficient to determine that QF_1 originates from OM. However, substring searching is not able to handle QF_2 and QF_3. In contrast, LCS matching can correctly handle QF_1, QF_2 and QF_3. Here the question is whether a simple substring matching is sufficient for real data. In our experimentation with the Enron dataset, we find that many quoted fragments contain quotations from more than one email. Thus, to maximize the robustness of HiddenEmailFinder, it is necessary to use LCS.

The problem with LCS is that its complexity is quadratic in the length of the fragment and the email. For long emails and/or quotations, implementing LCS naively is not scalable. We propose to extend the word index from the email filtering step to reduce the number of comparisons. In particular, for each email in the list L_w, we also record the positions at which the word w occurs in the corresponding email, i.e., each entry in L_w is of the form $\langle id, \{pos_1, \ldots, pos_k\}\rangle$. For example, the word "available" may have the following index entry: $\langle available, \langle\langle id = 17, pos = \{89, 3475\}\rangle$, $\langle id = 278, pos = \{190, 345, 3805\}\rangle\rangle\rangle$.

Then given a quoted fragment F, as before, F is tokenized to words. For each word w, and each email M in L_w, we can use the list $\{pos_1, \ldots, pos_k\}$ as "anchors" to facilitate the matching between F and M. For example, let us say that F contains the word "available." Then position 89 in email 17 is used as an anchor to match up the word "available" in F and email 17. By expanding forward and backward from the anchor position as much as possible, the longest common substring with respect to the anchor position is formed. Similar anchoring and expansion occurs at position 3475 in email 17, and the three specified positions in email 278. If the quoted fragment is tokenized to multiple words, the above process is conducted for each word w, and the longest common substring is picked. Figure 3 gives a skeleton of this optimization step called LCS-anchoring. This optimization is intended to be used in step 2 of HiddenEmailFinder. It can also be used in step 3 to optimize the identification of possible overlaps between hidden fragments.

5. THE ENRON CASE STUDY

5.1 The Data and the Setup

The Enron email dataset was made public by the US Federal Energy Regulatory Commission during the ex-Enron investigation. It was then purchased by Leslie Kaelbling at MIT, cleansed by Melinda Gervasio et al. at SRI and put on the web by William W. Cohen from CMU. To the best of our knowledge, this is the largest public accessible email dataset. This dataset contains about half a million messages belonging to 150 users and 3500 folders with all attachments deleted. A detailed description can be found at William W. Cohen's website at www-2.cs.cmu.edu/˜enron. The Enron corpus is of great value for many research areas, including social network analysis, spam detection and email classification. Many analyses and preprocessing studies have been done on the Enron dataset. For example, the SIAM'05 Workshop on Link Analysis, Spam Detection and Anti-terrorism published several indexes of the Enron dataset. In our experiments, we use their word indexes instead of building our own. The word index contains 160,203 unique words. Recall that whenever we refer to a frequent word threshold (ft), we mean that the top-ft words are considered too frequent to be used in EmailFiltering or LCS-Anchoring. Below we vary ft from 1000 to 80,000, corresponding to about 0.6% and 50% respectively.

For most of the results reported below, we focus on the inbox folders of the users. Of the 150 users, 137 have an inbox folder. The number of emails in those folders ranges from 3 to 1466. The average and median number of emails are 327 and 223 respectively.

In the experimental results reported below, we examine the prevalence of hidden fragments in the inbox folders. We also examine the percentage of hidden fragments that can be recollected from other folders of the same user. Then we consider robustness issues related to our algorithms. The first issue is the choice of the parameter *minLen*, and the second issue is the impact of signature files. Last but not least, we evaluate the effectiveness of the optimization algorithms EmailFiltering and LCS-Anchoring. We compare the savings in runtime against the possible reduction in the quality of the output (i.e., reconstructed emails). Runtime figures are obtained based on experiments done on a Sun Fire 880, 900MHz UltraSPARC-III machine.

5.2 Prevalence of Emails Containing Hidden Fragments

For each user in the Enron dataset, we identify all the hidden fragments in the inbox folder. Figure 4(a) shows the number of emails that contains at least one hidden fragment. Due to lack of space, we only show the largest 50 inbox folders sorted by ascending folder size (ranging from 338 to 1466 emails). As can be seen from the figure, there are 5 inbox folders with more than 300 emails containing at least one hidden fragment.

While Figure 4(a) shows the absolute values, Figure 4(b) displays the percentage of emails containing at least one hidden fragment (i.e., relative to the folder size). Because percentage may not make sense for small folders, we exclude folders with less than 50 emails. The x-axis of the graph shows the percentage, ranging from 0% to 60%. The y-axis shows the number of users with the given percentage. It is interesting to see that about half of the users are within the range of 15% to 30%.

Note that so far the analysis is based on emails in the inbox only. That is to say, a quoted fragment is designated to be hidden, as long as it cannot be found in the inbox folder. The reader may wonder whether hidden fragments just represent a phenomenon of the user being diligent in filing her emails into an appropriate folder. To examine this effect, we check other folders of the same user. Hereafter, we refer to a hidden fragment as "global" if it is a fragment that cannot be found in *all* the folders of the user. We refer to a hidden fragment as "local" if it is a hidden fragment within the (inbox) folder, but is otherwise found in some other folder of the user. Let us denote the numbers of global and local hidden fragments by n_g and n_l respectively. We define the *recollection rate* as the ratio of $n_l/(n_l + n_g)$. That is to say, the closer the ratio is to 1, the smaller is the number of global hidden fragments.

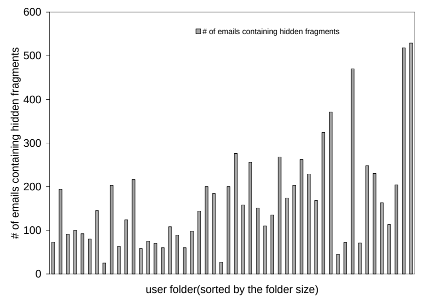
Figure 4(c) shows a histogram of the recollection rates for all the users. It is interesting to see that most users have a recollection rate of less than 15%. That is to say, there is less than 15% of hidden fragments that can be found in the other folders of the user. For the Enron dataset, the average and median $(n_l + n_g)$ values are 211 and 102 fragments respectively. Using the median figure of 102 hidden fragments as an estimation, a recollection rate of less than 15% corresponds to at most 15 local hidden fragments and at least 87 global hidden fragments. Thus, hidden fragments do not seem to be simply a phenomenon of the user filing the emails to other folders; they are truly missing from the user's folders.

While it is clear that hidden fragments are prevalent in the Enron corpus, the immediate question here is how general this phenomenon is for a "typical" real user. Let us review how the Enron dataset was prepared. As reported in [3], emails were deliberately deleted from the first published version of the Enron dataset on the users' request for privacy and redaction purposes. It is estimated that about 8% of the emails were deleted to form the current version. Consider the following two aspects:

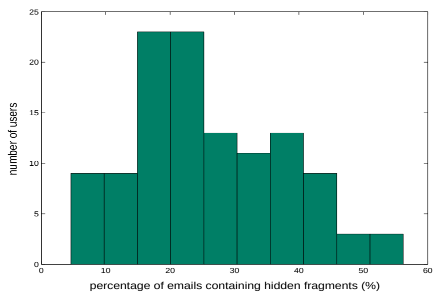
- First, the deleted emails are believed to be more personal in nature. It is reasonable to assume that they were less likely to be quoted in many Enron business emails.
- Second, as will be discussed later in Figure 5, the average number of reconstructed hidden emails per user is about 60. Given that the average inbox folder size is 327 emails, if the 8% of the deleted emails were evenly distributed in the inbox folder of each user, this would correspond to 26 emails in the folder. The gap between 26 and 60 is significant.

The question of whether hidden fragments are prevalent in a typical user's folder is hard to be answered definitively. But for the two reasons discussed above, the abundance of (global) hidden fragments we found in the Enron corpus may well generalize to other real datasets.

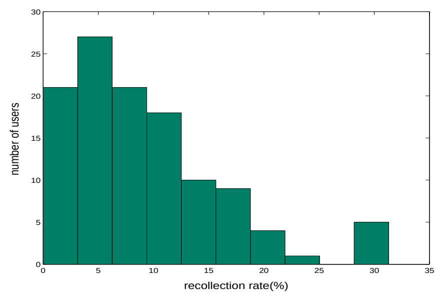
There is actually another interesting point to make here. Emails were deleted partly to protect privacy. However, some of these deleted emails may be recoverable from emails in other folders. Thus, if there is an initial set of emails to be protected, the framework that we develop here can help to strengthen the protection by identifying other emails quoting the initial set.



(a) Emails with hidden fragments



(b) Histogram on the percentage of emails containing hidden fragments



(c) Histogram of Recolletion Rates

Figure 4: Prevalence of Hidden Emails

5.3 Robustness: Choosing *minLen*

In the identification of hidden fragments and in the construction of the precedence graph (steps 1 and 2 of HiddenEmailFinder), a key parameter is the choice of *minLen*, the minimum length for the algorithm to declare a match. This choice affects whether one quoted fragment is considered a hidden fragment or not, and whether a fragment need to be split(Section 3.1 & 3.2 in [1]). For instance, a larger *minLen* can avoid incorrect matching of common sentences and quotation, and hence the resulting precedence graph is less connected. On the other hand, it may also miss short quotations, and create more hidden fragments.

Figure 5 shows the average and median number of reconstructed hidden emails with respect to *minLen* across all the inbox folders. As *minLen* increases from 40 to 160, there is only a very slight increase in the number of reconstructed hidden emails. This shows that our algorithmic framework is stable with respect to the choice of *minLen*.

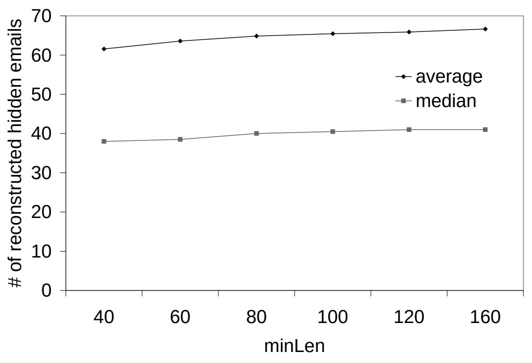


Figure 5: Robustness: Stability of *minLen*

5.4 Effectiveness of Optimizations

In this section, we study the effectiveness of the optimization heuristics of EmailFiltering and LCS-Anchoring. We measure the runtime of step 2 of HiddenEmailFilter, which is the dominant step. We also record the number of reconstructed hidden emails as a way to measure the output quality of HiddenEmailFinder. The intent is to observe the tradeoff between runtime efficiency and output quality.

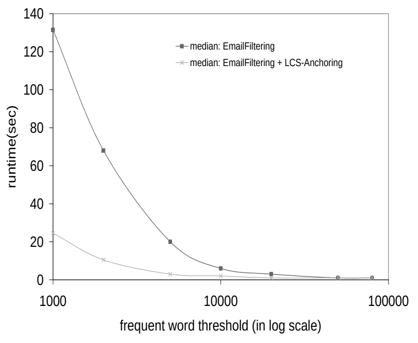
We design our experiments in the following way. In the first round of experiments, we only apply the heuristic of EmailFiltering and change the frequent word threshold (ft). We vary ft from 500 to 80,000. In the second round of experiments, we apply both EmailFiltering and LCS-Anchoring. For both rounds, we record the runtime and the number of reconstructed hidden emails for all the inbox folders.

Figure 6(a) shows the median runtime performance. The x-axis is drawn in log scale of the frequent word threshold ft. Let us first focus on the curve applying only the EmailFiltering algorithm. The basic, unoptimized version of HiddenEmailFinder corresponds to the case when $ft = 0$. The median runtime for this case is about 10 minutes, which is not shown in the figure. As the value of ft increases, the runtime improves by as much as 2 orders of magnitude, down to less than 10 seconds for $ft = 10,000$.

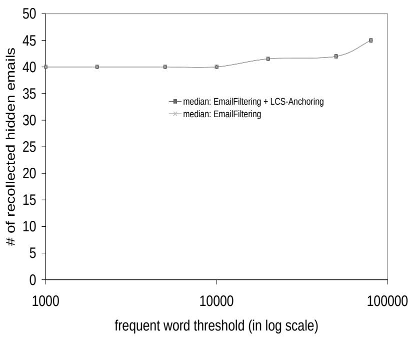
The second curve in Figure 6(a) shows the additional gain in efficiency when LCS-Anchoring is applied on top of EmailFiltering. The gap between the two curves shows that there is a definite bonus in applying LCS-Anchoring. The gain becomes smaller as ft increases because EmailFiltering alone has already eliminated a lot of emails required for matching, thereby reducing the number of times that LCS-Anchoring is performed.

The question is whether the significant gain in efficiency is achieved through reduced quality. Figure 6(b) shows that the number of reconstructed hidden emails when ft changes from 1000 to 80,000. The case when $ft = 0$ is not shown in the figure, but is identical to the value for $ft = 1000$. As ft increases from 1000 to 80,000, the number of reconstructed hidden emails increases very slightly, reflecting the reduced connectivity of the precedence graph. Given that the two curves in Figure 6(b) almost completely coincide, it is clear that both EmailFiltering and LCS-Anchoring can bring about a gain in efficiency without causing a degradation in the output quality.

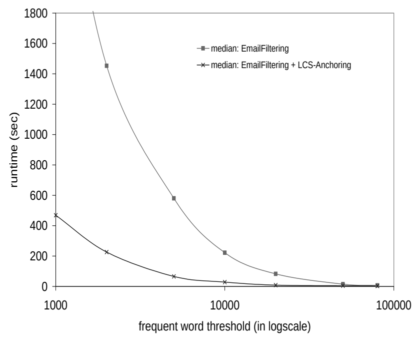
Figure 6(a) does not include the average runtime because there is a large discrepancy between folders on how long it takes to process them. Figure 6(c) shows the extreme case of the top-10 largest folders. Among these top-10 folders, the median folder contains 1,152 emails, with 37 emails each longer than 1,000 words. Large folders and long emails take significantly more time than the smaller ones. The two curves in the figure show the median runtime across the 10 folders when EmailFiltering alone and when EmailFiltering and LCS-Anchoring are applied. Like in Figure 6(a), it is clear that both techniques are effective. For example, when $ft = 10,000$, corresponding to 6% of the total number of unique words, the median runtime even for the top-10 largest folders is now down to 28 seconds. But un-



(a) runtime



(b) number of reconstructed hidden emails



(c) top-10 largest folders

Figure 6: Effectiveness of Optimizations

like in Figure 6(a), this time the gap is far more significant whether LCS-Anchoring is applied. This convincingly shows the importance of LCS-Anchoring for long emails and large folders.

6. CONCLUSION

This paper studies the problem of reconstructing hidden emails using the embedded quotations found in messages further down the thread hierarchy. We propose a bulletized email model and optimize the basic HiddenEmailFinder algorithm to deal with large folders and long emails. The two optimizations are based on word indexing to reduce the number of emails that need to be matched and to reduce the amount of effort required to find the LCS between the fragment and the email under consideration. LCS is vital in making HiddenEmailFinder robust enough to deal with real folders. As a side benefit, given an initial set of emails to be protected, HiddenEmailFinder may be used to strengthen the protection by identifying other emails quoting the initial set.

Another key contribution of this paper is the Enron case study. From our experimentation, many valuable lessons are learned. First, we observe that global hidden fragments are prevalent in the Enron corpus, and that HiddenEmailFinder serves to reconstrucut the hidden emails. There is good reason to believe that the prevalence may well generalize to other real datasets, thereby justifying the importance of the hidden email reconstruction problem. Second, we show that our framework is robust in dealing with real folders. Last but not least, we show that both the EmailFiltering and the LCS-Anchoring techniques are effective in providing scalability to large folders and long emails. They can bring about orders of magnitude improvement in runtime using a frequent word threshold of $ft = 10,000$, while not compromising the output quality.

Our future plans include applying natural language understanding techniques to make even more intelligent decisions about piecing fragments together and representing them to the user. This work is an integral part of a larger project on text mining. The goal is to help users better manage their emails. We plan to develop email summarization and classification tools. These tools will be based on various kinds of graphs extracted from the emails, in a style similar to the precedence graph that HiddenEmailFinder generates.

7. REFERENCES

[1] Giuseppe Carenini, Raymond Ng, Xiaodong Zhou and Ed Zwart. Discovery and Regeneration of Hidden Emails. *ACM Symposium of Applied Computing(SAC'2005)*, Santa Fe, New Mexico, USA, March, 2005.

[2] Vitor R. Carvalho and William W. Cohen. Learning to Extract Signature and Reply Lines from Email. *First Conferences on Emails and Anti-spam*, Mountain View, CA, USA.

[3] US Federal Energy Regulatory Commission. http://www.ferc.gov/industries/electric/indus-act/wem/pa02-2/info-release.asp

[4] Gwizdka, J, Chignell. M.H. Individual Differences and Task-based User Interface Evaluation: A Case Study of Pending Tasks in Email. *Interacting with Computers, Elsevier Science*, Minneapolis, Minnesota, USA , 2004, v.16(4) pp. 550–551.

[5] Bryan Klimt and Yiming Yang. The Enron corpus: a new dataset for email classification research. *European Conference on Machine Learning (ECML 2004)*, Italy, 2004.

[6] Derek Lam, Steven L. Rohall, Chris Schmandt and Mia K. Stern. Exploiting E-mail structure to improve summarization. *CSCW'02 Poster Session*, New Orleans, Louisiana, United States, 2002,

[7] Ani Nenkova and Amit Bagga. Email classification for contact centers *Proceedings of the 2003 ACM symposium on Applied computing*, Melbourne, Florida, 2003, pp. 789–792.

[8] Paula S. Newman. Exploring discussion lists: steps and directions. *Proceedings of the second ACM/IEEE-CS joint conference on Digital libraries*, Portland, Oregon, USA, 2002, pp. 126–134.

[9] Steven L. Rohall. Reinventing email. *CSCW'02 Workshop: Redesigning email for the 21st centry*, Portland, Oregon, USA, November, 2002.

[10] Kulesh Shanmugasundaram and Nasir D. Memon. Automatic Reassembly of Document Fragments via Context Based Statistical Models *ACSAC*, 2003 , pp. 152-159.