Discovery and Regeneration of Hidden Emails

Giuseppe Carenini, Raymond Ng, Xiaodong Zhou, Ed Zwart
Department of Computer Science
University of British Columbia
Vancouver, Canada
{carenini, rng, xdzhou, ez}@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 reconstruction of hidden emails. A hidden email is an original email that has been quoted in subsequent emails but is not itself present in the folder; it may have been deleted or may never have been received. This paper proposes a method for reconstructing hidden emails using the embedded quotations found in messages further down the thread hierarchy. To do so, we model all the quoted fragments in a precedence graph, from which hidden emails are regenerated as bulletized documents. The bulletized model is our solution to the situation when a total ordering of fragments is not possible. We give a necessary and sufficient condition for each component of the precedence graph to be captured in a single bulletized email, and we develop heuristics that minimize the number of regenerated emails when the condition is not met. Finally, we present empirical results showing the scalability of our approach.

Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communications Applications—*Electronic mail*

Keywords

Text Mining, Hidden Email Discovery, Document Forensics

1. INTRODUCTION AND MOTIVATION

Since email has became one of the major communication tools over the past decade, more and more information is stored in one's inbox [1]. In addition, with the increasing use of handhelds, there is a trend to read email in mobile devices [2]. The popularity of email has triggered researchers to look for ways to help users better organize and use their mail folders, e.g., classification [3], task management [4] and user interface [5].

The challenges of automatically processing emails differ from those of other types of documents in various ways. One well-understood difference is the threaded nature of emails. According to one study, over 60% of emails are threaded [6]. Most sets of emails are hierarchically inter-related. Because managing an email folder is still largely a random activity prone to many errors, there is a pressing need to effectively support every aspect of this process. In this paper, we propose a solution to the hidden email problem. Although we concentrate on emails, the solution can also be applied to the more general task of reconstructing any document that exists only in fragments, where a total ordered recovery is not feasible.

A hidden email is an original email document that has been quoted in subsequent emails but is not present itself in the same folder. 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 or intentional deletion may 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. The problem this paper attempts to solve is how hidden emails can be reconstructed using the embedded quotations found in messages further down the thread hierarchy.

Ideally, the hidden email can be reconstructed in full and represented exactly as it was first written, i.e., a total order representation. However, doing so will often not be possible. Parts of the original may not have been quoted in subsequent emails. Even if the entire text does exist scattered among the various quotations, some uncertainty about the correct ordering may remain. In these cases, it is still desirable to settle for a partial order representation, while maintaining utility. As a preview, this paper makes the following contributions.

- We introduce 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 email fragments. The key technical result is a necessary and sufficient condition for each component of the precedence graph to be represented as a single bulletized email. These concepts are discussed in Section 3.
- In Section 4, we develop an algorithm to generate the bulletized email, if the necessary and sufficient condi-

SAC'05 March 13-17, 2005, Santa Fe, New Mexico, USA

tion is met. We also develop heuristics for generating the emails if the graph fails the condition. The biclique problem of graph theory is applicable to our heuristics. We show one way to minimize the number of edges removed from the graph to reconstruct a useful bulletized email. The various heuristics are evaluated empirically in Section 5.

There are many applications for which we may need to reconstruct a hidden email. A related text mining research project of ours deals with email summarization. Reconstruction of hidden emails is critical to this endeavour's success. Document reconstruction is also important for the growing field of document forensics. It is indispensable to many applications in archaeology, crash recovery and security.

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. Agrawal et al. [7] studied the social networks extracted from newsgroups. They found that replies to a posting often entails disagreement. Google's recently introduced *gmail*, a new system that applies the company's search engine technology to email management, also includes conversation threading.

In [8], Lam et al. propose to summarize a set of emails based on their threading hierarchy. They mention the existence of deleted messages in the hierarchy which they point out distinguishes them from messages in newsgroups. However, they do not study how to regenerate them.

In the closely related area of newsgroup summarization, Newman [9] describes MailContent, an operational system to explore email-based discussion lists. This system integrates many facilities such as email content analysis, thread structure analysis and a new visualization tool to represent thread structures. The author 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. [10] 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. De Vel et al. [11] investigate how to use data mining and machine learning techniques to learn both structural and linguistical characters, and hence identify the email authorship.

Our research on the reconstruction of missing emails can be generalized to the area of document forensics, where document reconstruction from fragments is crucial. Shanmugasundaram et al. propose the reconstruction of a total ordered document in [12]. They take the maximum weight Hamiltonian path in a complete graph as the optimal result. 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 constitute a reasonable compromise between accuracy and completeness concerns.

3. BULLETIZED 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. In this section, we first describe how to use a precedence graph to represent hidden fragments. Then we describe how to reconstruct hidden emails based on a bulletized email model. Finally, we give precise conditions under which (parts of) a precedence graph can be transformed into a bulletized email.

3.1 Identifying Quoted Fragments

Given a folder of emails $\{M_1, \ldots, M_n\}$, we first conduct the following two preprocessing steps to identify fragments that are quoted in some email M_i but do not originate from emails in the folder.

1. **Extracting quoted fragments**: Given email M_i, tokenize it into the quoted and the new (non-quoted) *fragments*. A quoted fragment is a maximally contiguous string preceded by a quotation symbol, e.g. >, or by proper indentation[1]. A new fragment is a maximally contiguous string delimited by quoted fragments (or by either end of the email). For convenience, we use $M_i.quote$ and $M_i.new$ to denote the set of quoted and new fragments in M_i.

2. **Eliminating quoted fragments originating from the folder**: For each quoted fragment F in $M_i.quote$, we match it against $M_j.new$ for all $1 \leq j \leq n$. Let τ be the longest match, i.e., contiguous string, in F and in some fragments in $M_j.new$. There are 3 cases:

   (a) $\tau = F$:
   F originates from M_j and is not a fragment from a hidden email. Remove F from $M_i.quote$.

   (b) $length(\tau) < minLen$:
   τ has fewer characters than the threshold $minLen$ (e.g., 40 characters long). Some common words or phrases may match in isolated parts of the text, but there is no reason to believe that F, or parts of F, originates from M_j. Thus, F remains in $M_i.quote$.

   (c) Otherwise, split F into 3 (contiguous) fragments F_1, τ and F_2. Replace F in $M_i.quote$ with F_1 and F_2 and continue.

3.2 Creating The Precedence Graph

After the preprocessing described above, every item in $M_i.quote$ is a hidden fragment. The next step is to look for overlap among the fragments. The same fragment, or parts of it, could be quoted in multiple emails. Such duplication must be removed.

[1]We omit details regarding nested quotations, and we assume quotations at the same level to all be from the same original email.

Let Fdr be the union of $M_i.quote$ for all $1 \leq i \leq n$. For each fragment F in Fdr, match it against every other fragment F' in Fdr. Apply a similar process as in step 2 above.

1. If $F'$is a duplicate of F, then F' is removed from Fdr.
2. If F and F' do not (sufficiently) overlap, then both remain in Fdr.
3. If F and F' overlap sufficiently but not fully, they are split to avoid duplication. F is split into F_1, τ, F_2, and F' is split into F_3, τ, F_4. Then F and F' are replaced by $F_1, \ldots, F_4, \tau$ in Fdr.

After applying the duplication removal process described above, fragments remaining in Fdr are used to create a precedence graph $G = (V, E)$. The set of nodes V is exactly the set of fragments in Fdr. An edge is created from node F to F' if (i) there exists an email M_i such that both fragments are quoted in M_i, and (ii) fragment F precedes fragment F' textually. Thus, the precedence graph G represents the relative textual ordering of all fragments from hidden emails in M.

An edge $(F, F') \in E$ is redundant if the relative ordering of F and F' can be inferred from the rest of the graph. Thus, the *Minimum Equivalent Graph* (MEG) of G is sufficient to represent the relative ordering. The MEG is a subgraph of G such that MEG maintains all reachability relations in G and the number of edges is minimized. This is the *transitive reduction* problem described in [13]. In the rest of this paper, when we talk about a precedence graph, we refer to its minimum equivalent graph unless otherwise stated.

3.3 The Bulletized Email Model

The precedence graph G describes the relationship between hidden fragments, i.e., their textual ordering. In the ideal case, the edges link all the fragments into a single chain of nodes, which amounts to a total order natural reconstruction of the hidden email. In practice, however, selective quoting may lead to the the following complications:

- There may be multiple hidden emails in the folder. This will be evidenced by the existence of disconnected components in G, which may be the result of there having been more than one original email in the folder in the first place, or only one original email but with some missing connecting edges. This happens when responders selectively quote the original email with some intermediate section never being quoted. To identify the relationship between two components in the precedence graph, we need to use methods other than the preprocessing steps stated before.
- There may be cycles in G. This corresponds to a situation when in one email, fragment F is quoted before fragment F', and in another email, the opposite is encountered. There are various heuristics to handle cycles in G; however, in this paper, for simplicity, we do not consider this situation and assume that G is acyclic.
- A node may have more than one outgoing edge to nodes which are not connected. That is, in one email, fragment F was quoted before fragment F_1; in another email F was quoted before F_2; and there is no path connecting F_1 and F_2 in G. We refer to these two nodes as *incompatible*, i.e., nodes with a common ancestor but not otherwise connected to each other.

Given the precedence graph G from which hidden emails are to be regenerated, there are three overall objectives for the reconstruction process:

1. **(node coverage)** Every node must appear in at least one regenerated email. This is natural since this hidden fragment was indeed quoted. If the fragment is not included in any regenerated emails, a real loss in information results.
2. **(edge soundness)** The textual ordering of the fragments in a regenerated email must not correspond to a spurious edge not existing in G. That is, two incompatible nodes in G must remain incompatible in a regenerated email. It is undesirable to impose an arbitrary ordering on incompatible nodes.
3. **(minimization)** The number of regenerated emails should be minimized. This guarantees that the edges in G are reflected as much as possible in the regenerated emails.

People usually read a document sequentially and are not accustomed to reading graphical representations of document fragments. The possible presence of incompatible nodes and the objective of not introducing arbitrary ordering implies that our email model for representation has to account for partial ordering among fragments. Bullets represent a standard way to show unordered paragraphs in documents. So, we adopt a bulletized email model using *bullets* and *offsets* text devices described in [14]. Bullets show no ordering among fragments at the same level, which implies that bulletized fragments come in sets of at least two. They are suitable to represent incompatible nodes. Offsets can only apply to bulletized fragments, and show a nested relationship from the set of bulletized fragments to the fragment from which they are offset. We give an inductive definition below.

Definition 1. A bulletized email $d = [item_n]$ is a sequence of items, each of which is either a fragment (base case), or a set of bulletized emails (inductive case).

Figure 1 shows a precedence graph of fragments and the corresponding bulletized email. There are two groups of incompatible fragments: B, D and C, E, F. In the bulletized email, this incompatibility is reflected in the two bullets beginning with B and C. Within the bullet of C, there are two sub-bullets of E and F, reflecting their incompatibility. Hereafter, we adopt the notation of using [ ] to denote a sequence of items and using { } to denote a set of incompatible bullets. For example, the bulletized email shown in the figure is represented as $[A; \{[B; D], [C; \{[E], [F]\}]\}; G; H]$.

3.4 Completeness and Strictness

So far, we have described how to construct a precedence graph of hidden fragments from an email folder, and what is the corresponding bulletized email. Since each component of the graph, a maximum weakly connected subgraph, corresponds to one hidden email, the key scientific questions

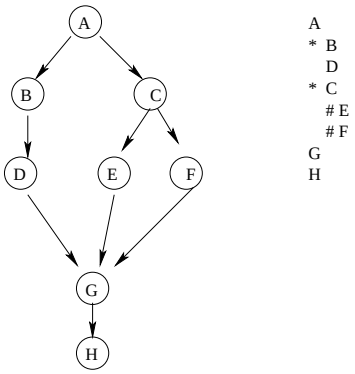


Figure 1: Example of a Bulletized Email

to ask are whether each component can be represented as a single bulletized email, and if not, under what conditions it can be done. The rest of this section, and a key result of this paper, deals with the development of a necessary and sufficient condition. In the next section, we look at graphs that fail this condition.

Given a weakly connected precedence graph $G = (V, E)$ and a node $v \in G$, we use $child(v)$ and $parent(v)$ to denote the set of all child nodes and parent nodes of v respectively. We generalize this notation to a set of nodes by taking the union of the individual sets.

Definition 2. **(Completeness)** A *parent-child subgraph* $PC = (P \cup C, E')$ is a subgraph of $G = (V, E)$, such that:

- for every node $p \in P$, $child(p) \subseteq C$;
- for every node $c \in C$, $parent(c) \subseteq P$;
- for every edge $(u, v) \in E'$, $u \in P$ and $v \in C$;
- for every node $u \in P, v \in C$, edge $(u, v) \notin E - E'$;
- PC is weakly connected, i.e., the underlying undirected graph is connected.

PC is *complete* if PC is a biclique, i.e., a complete bipartite graph. G is a *complete parent-child* graph iff every parent-child subgraph in G is complete.

Definition 3. **(Strictness)** A parent-child subgraph $PC = (P \cup C, E)$ *precedes* node u if there exists a node v in P such that P is an ancestor of u. A precedence graph G is *strict* if for any pair of vertices x, y such that x, y share the same child u, then for any parent-child subgraph PC preceding x or y, but not both, all the nodes in PC must be ancestors of u.

Figure 2 illustrates the notion and importance of strictness. Let us first consider the situation with A, x, y, u as shown but without node B. (This corresponds to the situation when in one email, A, x, u were quoted in this order, and in another email y, u were quoted.) This graph is strict because node A is an ancestor of u. The corresponding bulletized email is $[\{[A; x], y\}; u]$. Now if in another email A, B were quoted in this order, the edge connecting A, B makes the graph non-strict. Consider the parent-child subgraph with $PC = (P \cup C, E)$ with $P = \{A\}$. PC precedes x but

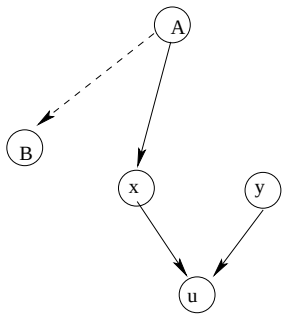


Figure 2: Example of strictness

not y. But node B in PC is not an ancestor of u, which violates the strictness condition. To see the importance of strictness, consider where to put B into $[\{[A; x], y\}; u]$. If we add it into $[\{[A; x], y\}$, we add a spurious edge between B and u. Similarly, grouping B with u adds spurious edges between x, y and B.

In the following theorem, we give a necessary and sufficient condition that any component of a precedence graph must meet to be captured in a single bulletized email.

THEOREM 1. *A weakly connected precedence graph G can be represented by a single bulletized email with every edge captured and no inclusion of spurious edges, iff G is a strict and complete parent-child graph.*

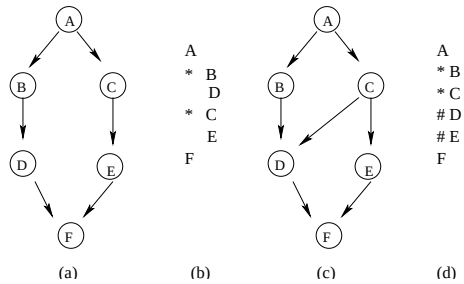


Figure 3: An incomplete parent-child graph

Figure 3 illustrates the situation. The graph in Figure 3(a) is complete and strict, and it can be represented as in (b). However, if we add an edge between C, D, as in (c), the graph is no longer complete parent-child graph as there is no edge between B, E. The bulletized emails in Figure 3(b) and (d) represent two failed attempts to correctly capture the graph. This shows the importance of completeness. Figure 4(a) shows a graph that is complete but non-strict. The problem is that F is not an ancestor of G, making it impossible to find a location for G for the graph to be captured by a single bulletized email.

4. REGENERATION ALGORITHMS

4.1 Algorithm graph2email

Theorem 1 gives a necessary and sufficient condition for a precedence graph to be captured in a single bulletized email. Figure 5 shows a skeleton of an algorithm that (i)

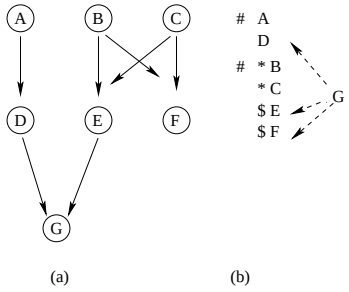


Figure 4: A non-strict graph

checks whether the graph satisfies the condition, and if so (ii) generates the bulletized email. The algorithm starts from the set of 0-indegree vertices in the precedence graph, traverses the whole graph and generates a bulletized email. It takes as input graph G and a set of nodes S, traverses S's descendents T, whose ancestors are connected to at least one node in S, and returns an email for the subgraph induced by $S \cup T$ and a frontier S'. The *frontier* is the set of nodes at which graph2email stops traversing, i.e., each node in the frontier either has no outgoing edge or has at least one parent that is not a descendent of S.

A description of Figure 5 follows. In step 1, we partition S into $\{S_0, ..., S_n\}$ where S_i denotes the overlapped nodes of parent set P_i and S. In step 2, for each S_i found in step 1, we check for completeness, and exit on failure. If the test passes, since each complete parent-child subgraph $PC_i = (P_i \cup C_i, E_i)$ can be represented as $[\{P_i\}; \{C_i\}]$, we set $d_i = [\{S_i\}]$ and we recursively call $[d'_i, S'_i] = graph2email(G, C_i)$, if $S_i = P_i$. The returned email d'_i is appended to d_i. Now we have a frontier $S' = \{S'_0, S'_1, ..., S'_n\}$, each S'_i corresponds to an email d_i. In step 3, we check for strictness, and exit on failure. When two frontiers overlap but are not equal, G is not strict. For example, in Figure 2, the frontier of $\{A\}$ is $\{B, u\}$, the frontier of $\{y\}$ is $\{u\}$. The overlap of $\{B, u\}$ and $\{u\}$ implies that B is in a parent-child subgraph $PC = (\{A\} \cup \{B, x\}, E)$ which precedes x. So, A, B and x have to be ancestors of u. However, B is not connected with x, y and u. Thus, we conclude that the graph is not strict. At this point in the algorithm, we have a set of emails each of which is associated with one frontier. In step 3(b), we union all emails which have the same frontier into a new email d_t, and collapse those frontiers into one S_t. If all parents of the collapsed frontier are included in email d_t, we recursively call $[d'_t, S'_t] = graph2email(G, S_t)$, and append d'_t to d_t. This recursive call compensates for some missed ones in step 2(c), where $S_i \neq P_i$, but all nodes in P_i have already been included in all emails generated in step 2. We repeat step 3 until there are no identical frontiers. In step 4, we simply generate a new email d, which is the set of emails generated above, and return it along with frontier S'.

4.2 Incomplete or Non-strict Graphs

Algorithm graph2email regenerates the hidden email corresponding to a complete and strict precedence graph. In the remainder of this section, we develop regeneration algorithms for graphs that are incomplete or non-strict.

Recall from our three design objectives that textual ordering of the fragments in a regenerated email must not correspond to a spurious edge not existing in G. If G is non-strict or incomplete, our approach is to remove edges from G so that G' will become strict and complete. A bulletized email is then generated corresponding to G' using Algorithm graph2email. There are many ways to show the removed edges to the user. One solution is that we consider the removed edges as a precedence graph G'' and apply Algorithm graph2email on G''. Thus, the removed edges are covered in subsequent regenerated hidden emails. Another solution is to display the missing edges to the user in the output of G', e.g., a few arrows showing the missing precedence. This is a question of user interface design, and is not covered in this paper.

Given the definition of strictness from the previous section, edge removal is an effective solution for fixing non-strict graphs. Specifically, there are two kinds of edges that can be removed: (1) the edge that makes two nodes, say x, y, share the same child u; and (2) the edge between an ancestor and one of its children nodes that is not itself an

Algorithm: graph2email
Input: a weakly connected precedence graph G and a set of nodes $S \in G$.
Output: an email d and a frontier S'.

1. Find all parent-child subgraph $PC_i = (P_i \cup C_i, E_i), i \in [1, n]$, where P_i overlaps with S and $S_i = P_i \cap S$. Union all 0-outdegree nodes as a set S_0.

2. For each $S_i, i \in [0, n]$, generate an email starting from S_i as follows:
   (a) if PC_i is not a biclique, exit the program.
   (b) set email d_i as a set of nodes in S_i, $d_i = [\{S_i\}]$, and set frontier $S'_i = C_i$
   (c) if $S_i = P_i$, generate the email starting from C_i, $(d'_i, S'_i) = graph2email(G, C_i)$, set email $d_i = [d_i; d']$

3. (a) Check for strictness as follows: If there exist $S'_i \cap S'_j \neq \emptyset$ and $S'_i \neq S'_j$, $\Rightarrow G$ is not strict and exit.
   (b) Union emails that correspond to the identical frontier together into one email d_t, i.e.,$d_t = [\{d_{i_1}, ..., d_{i_{t_i}}\}]$ and collapse those identical frontiers together into one S_t. We replace those frontiers in S' with S_t
   (c) For each collapsed frontier in the last step, $S'_t, t \in [1, k]$, if $parent(S_t)$ are all included in d_t, we recursively call $(d'_t, S'_t) = graph2email(G, S_t)$.
   Repeat step 3 until there are no identical frontiers.

4. Union all emails generated by step 3 into one email $d = [\{d_{t_1}, ..., d_{t_k}\}]$, and union all frontiers S'_{t_i} into one set of nodes S'. Return d and S'

Figure 5: Algorithm graph2email

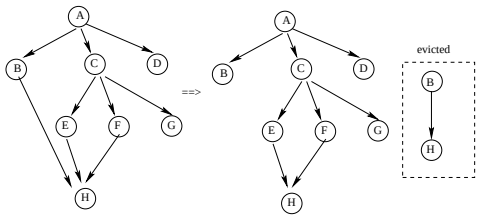


Figure 6: Example of edge removal for strictness

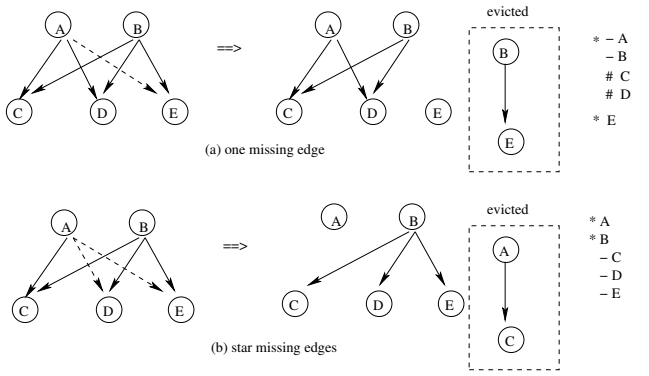


Figure 7: Edge Removal for Completeness

ancestor of u. Figure 6 shows that the removal of the edge between B and H makes the graph strict. Removing edge (C, G) also makes the graph strict. Identifying the appropriate edge to remove to overcome non-strictness is easy to incorporate into Algorithm *graph2email*. In step 3(a), instead of exiting the program we can apply a heuristic to remove edges. However, overcoming incompleteness is more complicated.

Given an incomplete parent-child precedence graph G, there must exist an incomplete parent-child subgraph of G, which is detected in step 2 of Algorithm graph2email. In stead of exiting the program we can apply heuristics described in this section and proceed to the next step.

Figure 7 (a) shows an example of how removing an edge can lead to a biclique. The edge between A and E is missing and the graph is incomplete. Our strategy is to remove the edge between B and E, so that the subgraph involving A, B, C, D now becomes a complete bipartite graph. In the example above, there is only one biclique left. The bulletized email model also accepts multiple bicliques. We generalize this as *multiple biclique decomposition* problem:

Given a bipartite graph $G = (V_1 \cup V_2, E)$ and a weight function $w : E \rightarrow N$, find a subset of edges $E_c \subseteq E$ with minimum total weight, s.t., $G' = (V_1 \cup V_2, E - E_c)$ can be partitioned into multiple bicliques, i.e., $V_1 = V_{11} \cup ... \cup V_{1k}, V_2 = V_{21} \cup ... \cup V_{2k}, E - E_c = E_1 \cup ... \cup E_k$ and $G_i = (V_{1i} \cup V_{2i}, E_i), i \in [1, k]$ is a biclique.

This problem is similar to the *maximum edge biclique* problem [15], which is NP-hard. We hypothesize that the multiple biclique decomposition problem is NP-hard as well. However, in the following discussion we point out that in some cases we can obtain the optimal result by the well-known min-cut algorithm.

For an incomplete parent-child subgraph, which is a weakly connected bipartite graph, a missing edge between two nodes is considered necessary for completeness if the two nodes are connected some other way in the underlying undirected graph. Thus, our aim is to find a way to break the existing path between the two nodes so the remaining subgraph can be represented by a bulletized email.

Let a *cut* $c(u, v)$ denote a set of edges whose removal disconnects vertices u, v. The *capacity* of a cut is the total weights of all edges in the cut. $c(u, v)$ is called *min-cut*, iff the capacity of $c(u, v)$ is the minimum among all cuts between u, v. For each missing edge (u, v), we can disconnect its two ends by deleting any cut $c(u, v)$.

Algorithm star-cut
Input: a connected bipartite graph $G = (L, R, E)$
Output: a set of edges E_c

1. find every missing edge $e_i = (u_i, v_i), u_i \in L, v_i \in R, i \in [1, k]$. Let $U = union(u_i), V = union(v_i)$, sort nodes in U descendently according to the number of missing edges incident on it.
2. for each $u \in U$ in order, do the following:
   (a) get all missing edges $e_j = (u, v_j)$
   (b) add vertex t and edges (v_j, t), set the capacity of edge (v_j, t) to infinity.
   (c) get the min-cut $c_{u,t}$, add $c_{u,t}$ into E_c and remove all edges in $c_{u,t}$ and vertex t

Figure 8: Algorithm star-cut

When all the missing edges share the same node u, i.e., $e_i = (u, v_i)$, we can add an additional node t and edges (v_i, t), and set the capacity of edges (v_i, t) to a big enough integer(bigger than the degree of v_i), the *min-cut* $c_{u,t}$ will only contain edges in the original graph, and will therefore give the optimal solution. The algorithm in Figure 8 uses a greedy approach to first process the node that has the largest number of missing edges incident on it and apply the min-cut in step 2. To find the min-cut, we use the Edmonds-Karps maxflow algorithm [16]. For a flow network with V nodes and E edges, when all edges are integers and bounded by U, the time complexity is $O(VEU)$.

In order to represent an incomplete parent-child subgraph, the remaining graph can be multiple bicliques. A single maximum edge biclique, i.e., with a minimum number of deleted edges, is acceptable as well. The maximum edge biclique problem has recently been proved to be NP-complete [17]. To the best of our knowledge, Hochbaum's 2-approximate algorithm [18] is the only c-approximate algorithm for the edge-deletion problem and is used here as a comparison. Hochbaum's algorithm, which is called *max-biclique* here, also applies the well-know min-cut algorithm. The details can be found at [18]. Both *max-biclique* and *star-cut* can be applied in step 2 of Algorithm graph2email.

5. EMPIRICAL EVALUATION

5.1 A Real Example

Figure 9 shows an original email stored in the teaching-assistant folder of one of the authors. This email is deliberately deleted from the folder to show how our regeneration algorithms work. (For the ease of representation, we use a, b, ..., h to represent the corresponding paragraphs.) Figure 10 shows 5 emails in the folder, all of which quoted the original email.

If we only have email 1 and 4, the precedence graph is disconnected and two hidden emails will be generated. With all the five emails included, the precedence graph is shown in Figure 11. This is an incomplete and non-strict graph. Our heuristics then delete the edges between e and f, and between b and h. Thus, the regenerated email is
$[a; \{[b], [e], [c; \{[\{[d], [f]\}; h], [g]\}]\}]$.

5.2 Scalability Evaluation

In the following, we study how well our algorithms per-

Subject: Midterm Details

Here are the midterm details:

a) I need to meet with a faculty recruit at lunch tomorrow. We'll be at Sage restaurant, so I'm going to walk to class from Sage (at the north end of campus)

b) Don, can you go directly to SOWK 124 ... it's just behind the LSK building, and the easiest entrance for you coming from CICSR is to go to the entrance of the corner at University Boulevard and West Mall. SOWK stands for Social Work. I will meet you at SOWK 124 about 15 minutes before the exam. In other words, I'll meet you around 13:30 on Friday afternoon (today).

c) Warren and Qiang, can you go directly to LSK 201. I'll meet you there about 10 minutes before the exam ... I'll come right over from SOWK 124. We need to get students double−spaced, so they're sitting behind one another in columns.

d) I will bring the exams with me to Sage, so you don't have to bring anything.

e) Students whose last names begin with the letters M−Q will write in SOWK 124; the rest (A−L and R−Z) will write in their normal classroom: LSK 201.

f) The exam is 48 minutes long, and it has 48 possible marks.

g) I will bring classlists with me. We need to check off the names and IDs of all students present. If there's a red serial number in the upper right hand corner of the exam, you should write that beside the student's name on the check−off list.

h) This is a closed book exam, with no help sheet, no calculators, no other aids.

Thanks.

−Ed

Figure 9: Case study - the missing/hidden email

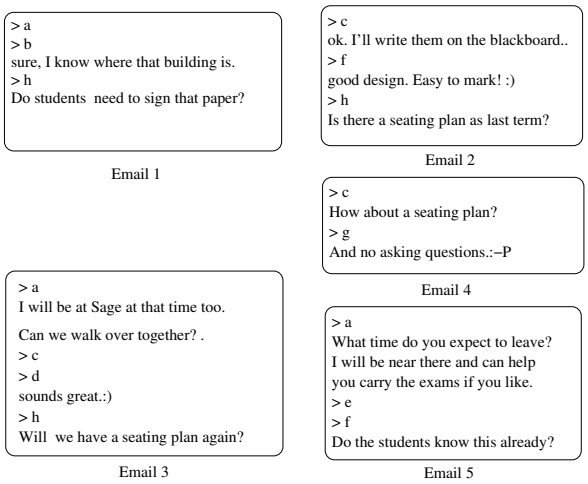


Figure 10: Case study - emails in the folder

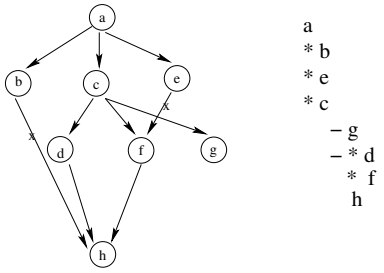


Figure 11: The precedence graph and regenerated hidden email

form with respect to the key parameters that characterize the hidden email problem, e.g., *folder size*. In our experiments, we use a synthetic dataset to evaluate these parameters. We first build a default setting as follows. We take as input the size of an email folder, say M, and set M=1000. Then we generate 1%*M hidden emails. Each hidden email contains 30 fragments, which is called *hidden email length*.

After generating a set of hidden emails, we generate the set of emails quoting them. 30% of emails in the folder quote hidden emails. Each email quotes about 10% of the fragments in that hidden email, which is called *quotation length*. Some hidden emails and fragments are more likely to be quoted since in real life, hot topics are more frequently discussed. In our experiments, we begin with the default setting, and scale up one parameter each time to see the effect on performance. For each setting, we generate 10 datasets and use the average in the following figures.

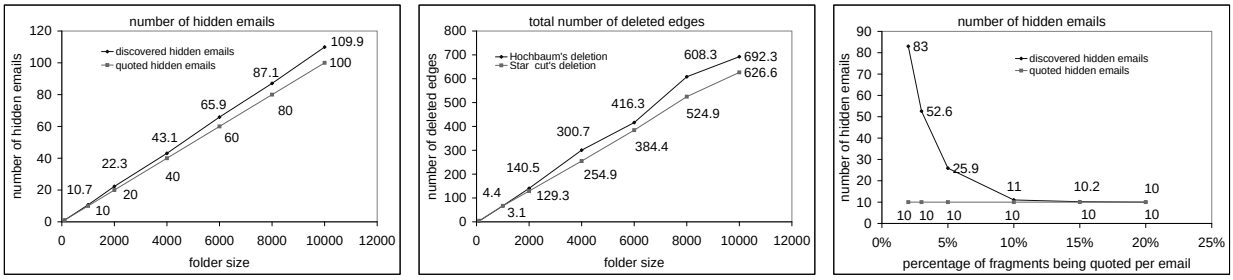
One of the key findings of our experiments deal with scalability with respect to the folder size and the quotation length. Figure 12(a) shows the number of hidden emails discovered and the number of hidden emails quoted as the folder size increases. The number of discovered emails increases linearly to the folder size and is about 10% more than the number of emails quoted. This difference indicates that one original email may be discovered as several independent hidden emails. Figure 12(b) shows that the number of edges deleted by the two heuristics also increases linearly as the folder size increases, and *star-cut* has a better performance over *hochbaum* in the number of deleted edges. As for runtime, both heuristics show increases proportional to the folder size. For the folder size of 1000 and 10000, both algorithms take about 2 seconds and 20 seconds respectively. However, *star-cut* takes more time than *hochbaum*. The difference is about 3 seconds at M=10000.

Notice that in the default setting, if one email quotes a hidden email, it only quotes 10% of the fragments in that hidden email, i.e., each email only quotes 3 hidden fragments. Figure 12(c) shows the number of discovered and quoted hidden emails with respect to various quotation length. This figure shows that with the increase of quotation length, the number of discovered hidden emails is greatly reduced. When the quotation length is 20%, no original hidden emails are discovered as separate ones. This shows that longer quotations greatly increase connectivity.

The discussion so far has only dealt with change in folder size and quotation length. Our experimentation covered many other parameters, and we provide a brief summary of how the situation changes when those parameters are altered:

- In the default setting, the number of emails quoting hidden emails is set to be 30%. When we change this percentage from 2% to 90%, we find that with more emails quoting hidden emails, the number of discovered emails first increases and then decreases. The increase is natural because more hidden emails are quoted at the same time. When more and more emails quote hidden emails, the precedence graph becomes more connected, which account for the decrease in the number of discovered emails.

  Similarly, the number of deleted edges of both heuristics has an up-hill phase followed by a down-hill phase. The reason is that when more emails quote hidden emails, the precedence graph becomes more connected,



(a) number of hidden emails (b) number of deleted edges (c) number of hidden emails

Figure 12: Scalability with respect to folder size and quotation length

and hence one parent-child subgraph contains more nodes and edges. Such increase results in more incomplete parent-child subgraphs, and then more edges are deleted. While more emails may be quoting hidden emails, both the size of a parent-child subgraph and the number of incomplete parent-child subgraphs decrease. Hence, the down-hill phase is generated. This reveals the fact that more quotations do not necessarily imply better connectivity and less edge deletion.

- Comparison of the two heuristics for incompleteness was inconclusive, In the default setting, *star-cut* takes more time, but has a better performance in the number of deleted edges. However, we found that in some settings, there is little difference between in both runtime and the number of deleted edges, while in some settings *star-cut* may delete more edges than that of *hochbaum*. Their relative performance depends on the kind of precedence graph on which they process and their manner of constructing a flow network. Further exploration into this problem may result in an improvement of both heuristics in the future.

6. CONCLUSION

The work in this paper represents an important first step toward the reconstruction of hidden emails. Its major contribution is the introduction of the bulletized email model and a necessary and sufficient condition for fitting a precedence graph into the model. The heuristics we present apply an edge deletion solution to graphs that do not meet the condition. 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. As such, we plan to extend it to related problems such as email summarization.

7. REFERENCES

[1] Jacek Gwizdka. Reinventing the inbox: supporting the management of pending tasks in email *CHI '02 extended abstracts on Human factors in computing systems*, Minneapolis, Minnesota, USA, 2002, pp. 550–551.

[2] Li-te Cheng and Daniel Gruen. A Mobile User Interface for threading, Marking, and Previewing Email *IBM Technical Report*, 2003

[3] 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.

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

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

[7] Rakesh Agrawal and Sridhar Rajagopalan and Ramakrishnan Srikant and Yirong Xu. Mining newsgroups using networks arising from social behavior *Proceedings of the twelfth international conference on World Wide Web*, Budapest, Hungary, 2003, pp. 529–535.

[8] 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,

[9] 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.

[10] 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.

[11] Olivier Y. de Vel and A. Anderson and M. Corney and George M. Mohay. Mining Email Content for Author Identification Forensics *SIGMOD Record v.30(4), 55-64, 2001*

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

[13] Samir Khuller and Balaji Raghavachari and Neal E. Young. Approximating the minimum equivalent digraph *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, Arlington, Virginia, United States, 1994, pp. 177–186.

[14] Eduard H. Hovy and Yigal Arens. Automatic generation of formatted text *Proceedings of Ninth National Conference on Artificial Intel ligence (AAAI-1991)*, 1991, pp. 92-97.

[15] Milind Dawande and Pinar Keskinocak and Jayashankar M. Swaminathan and Sridhar Tayur. On bipartite and multipartite clique problems *J. Algorithms*, v.41(2), 2001 , pp. 388–403.

[16] Jack Edmonds and Richard M. Karp. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems Journal of ACM, vol.19(2),1972,pp. 248–264

[17] Rene Peeters. The maximum edge biclique problem is NP-complete *Discrete Appl. Math.*, v.131(3), 2003 , pp. 651–654.

[18] Dorit S. Hochbaum. Approximating clique and biclique problems *J. Algorithms*, v.29(1) ,1998, pp. 174–200.