Summarizing Email Conversations with Clue Words

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

ABSTRACT

Accessing an ever increasing number of emails, possibly on small mobile devices, has become a major problem for many users. Email summarization is a promising way to solve this problem. In this paper, we propose a new framework for email summarization. One novelty is to use a fragment quotation graph to try to capture an email conversation. The second novelty is to use clue words to measure the importance of sentences in conversation summarization. Based on clue words and their scores, we propose a method called CWS, which is capable of producing a summary of any length as requested by the user. We provide a comprehensive comparison of CWS with various existing methods on the Enron data set. Preliminary results suggest that CWS provides better summaries than existing methods.

Categories and Subject Descriptors

H.2.8 [**Database applications**]: [Data mining]

General Terms

Algorithms

Keywords

Text mining, email summarization

1. INTRODUCTION

With the ever increasing popularity of emails, *email overload* becomes a major problem for many email users [17]. Users spend a lot of time reading, replying and organizing their emails. To help users organize their email folders, many forms of support have been proposed, including spam filtering[10], email classification[18] and email visualization[14]. In this paper, we discuss a different form of support - *email summarization*. The goal is to provide a concise, informative summary of emails contained in a folder, thus saving the user from browsing through each email one by one. The summary is intended to be *multi-granularity* in that the user can specify the size of the concise summary (e.g., depending on how much time the user wants to spend on the folder). Email summarization can also be valuable for users reading emails with mobile devices. Given the small screen size of handheld devices, efforts have been made to re-design the user interface. However, providing a concise summary may be just as important.

Email summarization is challenging in at least the following aspects. Many emails are asynchronous responses to some previous messages and as such they constitute a conversation, which may be hard to reconstruct in detail. A conversation may involve many users, many of whom may have different writing styles (e.g., short vs long sentences, formal vs informal). Finally, hidden emails may carry important information to be part of the summary. As defined in [3], a hidden email is an email quoted by at least one email in the folder but is not present itself in the user's folders.

Several recent attempts have been made to capture conversations by email threading. Many email programs provide features to group emails into threads using headers[6]. Research studies by Rambow et al.[9], Wan et al.[15] and Lam et al.[4] go further by using features defined on the threads to generate summaries. While a more detailed comparison will be discussed later, we believe that threading at the granularity level of emails is not sufficient and can be significantly refined. Furthermore, none of these approaches handle hidden emails.

In this paper, we claim the following contributions.

In Section 3, we propose using the *fragment quotation graph* to capture conversations. Based on an analysis of the quotations embedded in emails, the graph provides a fine representation of the referential structure of a conversation. The graph is also capable of dealing with hidden emails.

In Section 4, we propose an email summarization method, called ClueWordSummarizer (CWS), based on a novel concept called *clue words*. A clue word from a node is a word (modulo stemming) that appears also in its parent node(s) and/or child node(s) in the quotation graph. It is important to note that a clue word takes into account simultaneously (part of) the content and the structure of the quotation graph. Moreover, CWS can produce summaries of any size as requested by the user.

It is an open question how human would summarize email conversations. Thus, in Section 5, we present results of a user study on summarizing 20 conversations from the Enron data set. Not only does this study provide a gold standard to evaluate CWS and other summarization methods, but it also sheds light on the importance of clue words and hidden emails to human summarizers.

In Section 6, we evaluate the effectiveness of CWS on the Enron data set. We compare CWS with other summarization approaches. Our preliminary results show that both the quotation graph and clue words are valuable for summarizing email conversations.

2. RELATED WORK

Email summarization can be viewed as a special case of multi-document (MD) summarization. Radev et al. develop MEAD which gives a score to each sentence based on its similarity to the TFIDF centroid of the whole document set and other properties such as position in a document, sentence length and inter-sentence similarity [7]. Erkan et al.[5] develop the LexPageRank to rank sentences based on the eigenvector centrality. They compute a sentence linkage matrix as the sentence similarity and use this matrix with the well-known PageRank algorithm. Wan et al.[16] generate an affinity graph from multiple documents and use this graph for summarization. They consider both the information richness and the sentence novelty based on the sentence affinity graph. However, MD summarization methods, when applied to emails, do not take into account the key differences between emails and conventional documents. Key differences include the referential structure of conversations, the existence of hidden emails and the high variability of writing styles. Section 6 will compare CWS with MEAD for email summarization.

Rambow et al. apply a machine learning approach to email summarization [9]. They use RIPPER as a classifier to determine which sentences should be included in a summary. Features used for learning include linguistic features, and features describing the email and the threading structure. Such an approach requires a large number of positive examples and cannot produce summaries with varying length based on the users request. It is also not clear how this approach can handle hidden emails. Section 6 will compare CWS with RIPPER.

Wan et al. study decision-making summarization for email conversations [15]. Email threading is used. Among the various sets of features explored, their experiments show that a centroid based method is effective.

In our earlier studies, we focus on the *re-construction* of hidden emails [3, 2]. The focus here is completely different in generating summaries of conversations, regardless of whether there are hidden emails or not. In [3, 2], we use a precedence graph to re-construct hidden emails. The fragment quotation graph here is different in at least two ways. First, the nodes are different as a fragment quotation graph creates nodes for both new and hidden fragments. More importantly, the edges in a precedence graph capture textual ordering of the nodes within one hidden email, whereas the edges in a fragment quotation graph reflect the referential relationship among multiple emails.

As for extracting conversations, Yeh et al. study how to use quotation matching to construct email threads [6]. Their experiments show a higher recall than the header-based threading method. This supports our use of the quotation graph as a representation of email conversation. Shrestha et al. propose methods to automatically identify the question-answer pairs from an email thread [11]. Their method may be useful in building the conversation structure for the purpose of email summarization. Agrawal et al. extract social networks from newsgroups [8]. Stolfo et al. study the behavior model of email users based on the social network analysis among email correspondences [12]. They develop an email mining toolkit and use it to identify target emails without analyzing the email content.

3. BUILDING THE FRAGMENT QUOTATION GRAPH

For any given email folder, there may be multiple email conversations. To capture these different conversations, we assume that if one email quotes another email, they belong to the same conversation. We use a fragment quotation graph to represent conversations. A fragment quotation graph $G = (V, E)$ is a directed graph, where each node $u \in V$ is a text unit in the email folder, and an edge (u, v) means node u is in reply to node v. We are aware that this framework cannot represent every kind of email conversations. For example, there are cases where the original email is not quoted by the replies, and there are cases where the quotation is irrelevant to the topic discussed. Nonetheless, we believe that fragment quotation graphs can be applied to many practical situations.

3.1 Identifying Quoted and New Fragments

Quotation identification is itself a research problem [13]. Here we assume that there exist one or more quotation markers (e.g., ">") that are used as a prefix of every *quoted line*. We define the *quotation depth* of a line as the number of quotation markers ">" in the prefix. The quotation depth reflects the number of times that this line has been quoted since the original message containing this line was sent. A *quoted fragment* is a maximally contiguous block of quoted lines having the same quotation depth. A *new fragment* is a maximally contiguous block of lines that are not prefixed by the quotation markers. In other words, an email can be viewed as an alternating sequence of quoted and new fragments, or vice versa.

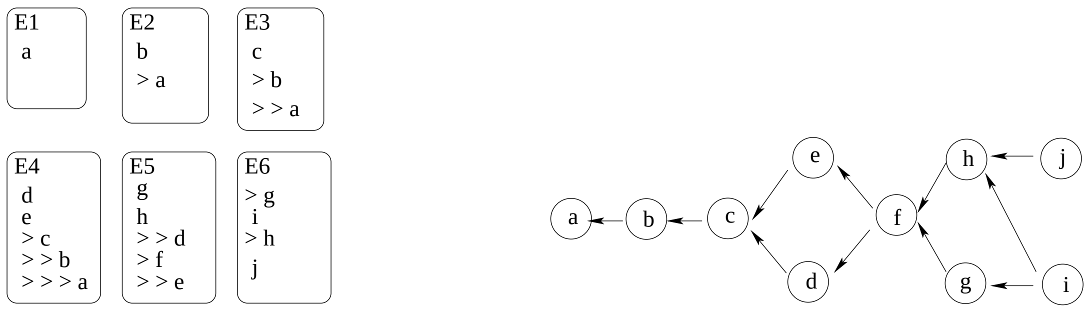
For convenience, we use $M_i.quote$ and $M_i.new$ to denote the set of quoted and new fragments in email M_i respectively. We use $M_i.frag$ to denote the sequence of fragments, both quoted and new ones. The order of the fragments is in accordance to their textual order in M_i. We denote the quotation depth of fragment F as $F.qtDepth$. The quotation depth of a new fragment is defined as 0.

3.2 Creating Nodes

Given an email folder $Fdr = \{M_1, \ldots, M_n\}$, we construct a fragment quotation graph G as follows. After the aforementioned identification of quoted and new fragments in each email M_i in Fdr, the first step is to identify distinct fragments, each of which will be represented as a node in the graph.

Note that when a user quotes an email, the user might perform various actions, as she can edit the fragments as free text. She can quote the exact sequence verbatim; or she can delete some parts of it. For example, a new fragment from an earlier email can be of the form $NF_1 = \langle F_1, F_2, F_3 \rangle$. In a latter email, a quoted fragment may be $QF_2 = \langle F_1, F_3 \rangle$. In another email, it may appear as $QF_3 = \langle F_1, F_4, F_3 \rangle$, where fragment F_4 was quoted from yet another email. The point is that a fragment can be quoted by many emails, with each user possibly performing different edits to it. The goal of the task here is to avoid as much duplication as possible in the quoted and new fragments.

To do so, quoted and new fragments from all emails in Fdr are matched against each other to identify overlaps. We say that there is an overlap between two fragments if there is a common substring that is sufficiently long. In this process, fragments may be split. Using the above examples, when



(a) Conversation involving 6 Emails

(b) Fragment Quotation Graph

Figure 1: A Real Example

QF_1 is matched against QF_2, the fragments F_1, F_2, F_3 are identified (assuming that they are longer than the overlap threshold). QF_1, QF_2 are then replaced by F_1, F_2, F_3. And when QF_3 is processed, F_4 is identified as well. This also shows how hidden fragments (e.g., F_4) can be identified.

At the end of this process, every distinct fragment is represented as a node in the fragment quotation graph G. Note that while the processing is quadratic with respect to the number of emails in Fdr, this is designed to be as accurate as possible to extract the conversations. Indexing techniques similar to those developed in [2] [6] can be used to significantly reduce the processing time.

Figure 1(a) shows a real example of a conversation from a benchmark data set involving 6 emails. For the ease of representation and space limit, we do not show the original content and abbreviate them as a sequence of fragments. In the fragment identification step, all new and quoted fragments are identified. For instance, E_3 is decomposed into 3 fragments: new fragment c and quoted fragments a and b (i.e., different quotation depth). Similarly, 4 fragments are identified from each of E_4, E_5 and E_6. Then in the node creation step, overlaps are identified, fragments are split if necessary (e.g., fragment gh in E_5 split into g and h when matched with E_6), and duplicates are removed. At the end, 10 distinct fragments $a, \ldots, j$ give rise to 10 nodes in the graph shown in Figure 1(b). Note that amongst all the fragments quoted, f never appears as a new fragment, and is hence labeled a hidden fragment.

3.3 Creating Edges

In general, it is difficult to determine whether one fragment is actually replying to another fragment. In this paper, we make the following simplifying assumption. *We assume that any new fragment is a potential reply to neighboring quotations – quoted fragments immediately preceding or following it.* In that case, an edge is added between the two corresponding nodes in the fragment quotation graph. Recall that in the fragment identification step, an email is decomposed into an alternating sequence of new and quoted blocks. For example, E_6 in Figure 1(a) is decomposed into quoted g, new i, quoted h and new j. In general, partly because of possible differences in quotation depth, a block may contain multiple fragments. Thus, for the general situation when QS_p precedes NS, which is then followed by QS_f, we create an edge (v, u) for each fragment $u \in (QS_p \cup QS_f)$ and $v \in NS$.

Let us consider E_3 in Figure 1(a). There are the edges (c, b) and (c, a). As will be pointed out later, Figure 1(b) only shows the minimum equivalent graph with all the redundant edges removed. Thus, because of the edge (b, a), the edge (c, a) is not included in Figure 1(b). Similarly, for E_6, there are two edges from node i to g and h, while there is only a single edge from j to h. Other edges are added for the same reason – except for the edges involving hidden fragment f.

For a hidden fragment, additional edges are created within the quoted block, following the same neighboring quotation assumption. For instance, because of E_5, edges are added from f to d and e.

We use the minimum equivalent graph as the fragment quotation graph, which is transitively equivalent to the original graph. Recall that a folder may contain emails involved in multiple distinct conversations. In the fragment quotation graph, each of these conversations will be reflected as weakly connected components.

Figure 1(b) shows the fragment quotation graph of the conversation shown in Figure 1(a) with all the redundant edges removed. In contrast, if threading is done at the coarse granularity of entire emails, as adopted in many studies, the threading would be a simple chain from E_6 to E_5, E_5 to E_4 and so on. This example clearly shows the advantage of using fragment quotation graphs.

4. EMAIL SUMMARIZATION METHODS

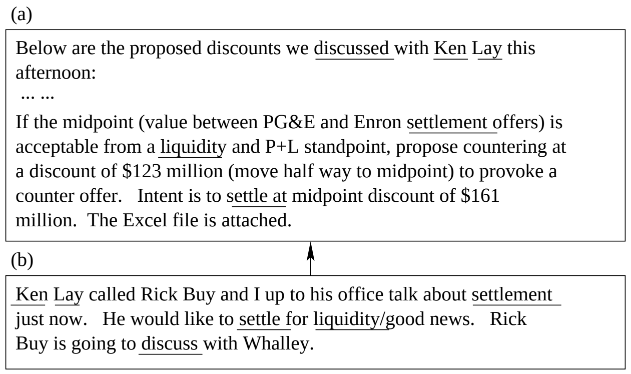
Once the fragment quotation graph corresponding to a conversation is extracted, the remaining task is to identify the most informative sentences to be included in the summary. In this section, we first present a novel method called ClueWordSummarizer. Then we briefly describe how two existing approaches, MEAD and RIPPER, can be applied to a fragment quotation graph.

4.1 Clue Words

It is well known in linguistics that coherent text tends to be lexically cohesive. Words related to the current topic tend to reoccur more frequently in successive utterances. In our preliminary analysis of email folders, we found this to be true also for fragments in the quotation graph. That is to say, some words in a fragment reoccur in the fragments replying to it. We call those words *clue words*.

A clue word in node (fragment) F is a word which also appears in a semantically similar form in a parent or a child node of F in the fragment quotation graph.

Note that the definition of a clue word is established by examining the textual content of a fragment. At the same time, a clue word takes into account of the referential relationship between two nodes connected by an edge. Thus, we believe that clue words are important for summarizing email conversations.



(a)

Below are the proposed discounts we discussed with Ken Lay this afternoon:
... ...
If the midpoint (value between PG&E and Enron settlement offers) is acceptable from a liquidity and P+L standpoint, propose countering at a discount of $123 million (move half way to midpoint) to provoke a counter offer. Intent is to settle at midpoint discount of $161 million. The Excel file is attached.

(b)

Ken Lay called Rick Buy and I up to his office talk about settlement just now. He would like to settle for liquidity/good news. Rick Buy is going to discuss with Whalley.

Figure 2: Example of Clue Words

Figure 2 shows a real example of two nodes from a conversation in the Enron data set. Fragments (a) and (b) are two adjacent nodes with (b) as the parent node of (a). Between the two nodes, there are 5 clue words *Ken, Lay, settle, discuss* and *liquidity*. Note that clue words do not need to reoccur verbatim. The clue words "discussed" and "settle" in (a) reoccur as "discuss" and "settlement" in (b). There are also synonyms, such as "discuss" in (a) and "talk" in (b).

From a preliminary analysis, we observe 3 major kinds of reoccurrence:

- the same root(stem) with different forms, e.g., "settle" vs. "settlement" and "discuss" vs. "discussed" as in the example above.
- synonyms/antonyms or words with similar/contrary meaning, e.g., "talk" vs. "discuss" and "peace" vs. "war".
- words that have a looser semantic link, e.g., "deadline" with "Friday morning".

In our experimentation, we observe that stemming occurs the most frequently among the three types discussed above. Thus, in this paper, we only apply stemming to the identification of clue words. We use the Porter's stemming algorithm to compute the stem of each word, and use the stems to judge the reoccurrence.

Several studies in the NLP literature have explored the reoccurrence of similar words within one document due to the text cohesion. The idea has been formalized in the construct of *lexical chains*, i.e., sequences of semantically related words appearing in sequences of contiguous sentences within a document. Some approaches use lexical chains to generate single-document summaries [1]. While clue words and lexical chains both rely on lexical cohesion, the two concepts are quite different with respect to the kind of linkages considered. For lexical chains, the concept of "chain" is based on similarities between lexical items in contiguous sentences within a single document. In contrast, for clue words, the linkage is based on the existing conversation structure which is represented by the quotation graph. In other words the "chain" in clue word is not only "lexical" but also "conversational", and typically spans over several emails (i.e., documents).

```
Input: fragment quotation graph $G = (V, E)$, summary length $k$
Output: a sequence of sentences $SUM = [s_1; \ldots; s_k]$.

1. For each node $u \in V$, let $u.StemSet$ denote the multiset of all
   words' stems in $u$.
   For every sentence $s \in u$, do the following:
     (a) Tokenize $s$ into a sequence of words $W$.
     (b) Remove all stop-words from $W$.
     (c) For each word $w \in W$ compute its stem $w_{stem}$ with
         Porter's stemming algorithm.
     (d) Insert $w_{stem}$ into $u.StemSet$.
2. For each sentence $s$ in each node $u \in V$, compute the
   $ClueScore(s)$ as follows:
   For each non-stop word $w$, compute $ClueScore(w, u)$ as the
   number of occurrence of $w_{stem}$ in all $x.StemSet$, where $x$ are
   $u$'s parent or child nodes.
   The ClueScore of sentence $s$ is computed as follows:
   $ClueScore(s) = \sum_{w \in s} ClueScore(w, u)$
3. Rank all sentences according to their ClueScore and select the
   top-$k$ ones as $SUM$.
```

Figure 3: Algorithm CWS

4.2 Algorithm CWS

Algorithm ClueWordSummarizer (CWS) uses clue words as the main feature for email summarization. The assumption is that if those words reoccur between parent and child nodes, they are more likely to be relevant and important to the conversation. A skeleton of algorithm CWS is presented in Figure 3.

In order to evaluate the significance of the clue words quantitatively, CWS uses *ClueScore(CW, F)* to represent the importance of a clue word CW in fragment F:

$$\begin{aligned} ClueScore(CW, F) \quad &= \sum_{parent(F)} freq(CW, parent(F)) + \\ &\quad \sum_{child(F)} freq(CW, child(F)) \end{aligned}$$

where $freq$ denotes the frequency the clue word appearing in a fragment. The above formula generalizes to the sentence level:

$$ClueScore(s) \quad = \sum_{CW_i \in s} ClueScore(CW_i, F)$$

where s is a sentence in fragment F.

For the example in Figure 2, consider the ClueScore for the sentence: "If the midpoint ... counter offer." in fragment (a). ClueScore("liquidity") = 1 and ClueScore("settlement") = 2 because "liquidity" appears once and "settlement" appears twice in fragment (b). Thus, the ClueScore of this sentence is 3.

To select sentences to be included in a summary of k sentences, algorithm CWS first tokenizes each sentence in every node into a *multiset* of words. After the stop words are removed and the stem of words are obtained, the above ClueScore formulas are applied first to words then to sentences. CWS selects the sentences with the highest ClueScore to return as the summary.

4.3 MEAD: a centroid-based multi-document summarizer

Algorithm CWS described above uses the ClueScore to evaluate the importance of each sentence based on the quotation graph. Since the ClueScore is computed based on the reoccurrence of the clue words in the parent and child nodes, it may tend to capture more local salience and miss more global salience related to the whole conversation. So it is reasonable to assume that additional information is needed. To satisfy these needs we explore using MEAD, a centroid-based multi-document summarizer to generate email summaries.

As the first step, MEAD computes the centroid of all emails in one conversation. A centroid is a vector of words' average TFIDF values in all documents. MEAD compares each sentence s with the centroid and assigns it a score as the sum of all the centroid values of the common words shared by the sentence s and the centroid. $C(s) = \sum_{w \in s} C_w$

In addition to the centroid, MEAD also uses other features (e.g., the sentence position and the similarity to the selected sentences) to compute the sentence score, MEADScore. Then all sentences are ranked based on the MEADScore, and the top ones are included in the summary. The details can be found in [7].

Compared with MEAD, CWS may appear to use more "local" features. But notice that locality in CWS is defined with respect to the quotation graph. That is, two sentences with clue words are not textually proximal to each other, but are brought together in a parent and a child node in the quotation graph. Because of this, CWS may achieve a good compromise between local and global information, which may explain its promising performance described in Section 6.

4.4 Hybrid ClueScore with MEADScore

As stated above, ClueScore and MEADScore tend to represent different aspects of the importance of a sentence. So it is natural to combine both methods together. In order to avoid either one score overwhelming the other, we standardize both scores as follows. We compute σ_{ctr}, the standard deviation of the centroid of all words, and use $\frac{centroid(w)}{\sigma_{ctr}}$ as the centroid to compute the hybrid score. Similarly, we also standardize the ClueScore for each word $w \in F$ as $\frac{ClueScore(w,F)}{\sigma_{clue}}$, where σ_{clue} is the standard deviation of all positive $ClueScore(w, F)$ for all words in every fragment F. For the ease of representation we still use the previous symbol to represent both standardized scores.

We use the linear combination of ClueScore and MEADScore together as the hybrid score of the sentence s. Let $\alpha \in [0, 1]$ denote the percentage of ClueScore in the final result. When $\alpha = 0$ or 1, it is pure MEADScore or pure ClueScore respectively.

$LinearClueMEAD(s) = \alpha * ClueScore(s) + (1 - \alpha) * MEADScore(s)$.

4.5 Summarization Using RIPPER

In [9], Rambow et al. propose to use machine learning to extract representative sentences from the original emails to form a summary. They use the RIPPER system to induce a classifier for determining if a given sentence should be included in the summary. RIPPER is trained on a corpus in which each sentence is described by a set of 14 features and annotated with the correct classification (i.e., whether it should be included in the summary or not). The features describing each sentence comprise linguistic features (e.g., similarity to the centroid) and features describing the email and the threading structure, (e.g., number of recipients of the email containing the sentence). They use 3 incremental feature subsets out of the 14 features, starting with 8 "basic" linguistic features. Then they add successively 2 ("basic+") and 4 ("basic++") additional structural features. In the end, the results with all features show that including additional features based on the emails and thread structure does improve classification accuracy.

We have adopted the framework proposed in [9] with only the following exception. Since some of the features used in [9] are based on the email thread structure and our algorithm is based on the fragment quotation graph, We needed to slightly change some features to fit the quotation graph. For example, the number of direct response to an email is replaced by the number of direct response to the fragment(node), in which sentence s is contained.

5. RESULT 1: USER STUDY

In order to compare different approaches of email summarization a gold standard is needed. In practice, for comparing extractive summarizers, we need to know what sentences a human summarizer would extract as most important from a target email corpus. Notice that having such a gold standard may also allow us to verify our assumptions on what information and algorithms an effective extractive email summarizer should rely on. In particular, we verify whether some important sentences do come from hidden emails and whether ClueScore correlates with sentence importance. A few gold standards have been developed in previous work [9] [15], but unfortunately they are not public. In addition, their gold standards only involve 2 human summarizers. It is not clear whether personal preferences are avoided in those gold standards. So we build our own in a user study described in the following.

5.1 Dataset Setup

We collected 20 email conversations from the Enron email dataset as the testbed and recruited 25 human summarizers to review them. The 20 email conversations were selected as follows. From the 10 largest *inbox* folders in the Enron email dataset, we discovered 296 email conversations. Since we are studying multiple email summarization in a conversational context, we required that each conversation contained at least 4 emails. Thirty eight conversations satisfied this requirement. Moreover, we wanted to study the effect of conversation structure, i.e., the fragment quotation graph. The selected emails also need to represent different types of conversation structure. According to our preliminary study, we found that the email threads could be divided into two types. One is the single chain type, and the other is the thread hierarchy type. In order to cover both structure in the context of email summarization, we randomly select 4 single chains and 16 trees, which is close to their ratio in the 38 conversations.

Secondly, we recruited 25 human summarizers to review those 20 selected email conversations. All 25 human summarizers were undergraduate or graduate students in University of British Columbia. Their majors covered various disciplines including Arts, Law, Science and Engineering. Since many emails in the Enron dataset relate to business and law issues, the variety of the human summarizers, especially those with business and legal background are of an asset to this user study.

Each summarizer reviewed 4 distinct conversations in one hour. In this way, each email conversation were reviewed by 5 different human summarizers. For each given email conversation, human summarizers were asked to generate a summary by directly selecting sentences from the original emails in that conversation. The generated summary contained about 30% of the original sentences. The selected sentences need to represent the major ideas of the original email conversations. The human summarizers were told that they need to select those sentences in a way that, by reading the generated summary, the readers did not need to refer to the original emails in most cases.

Moreover, human summarizers were asked to classify each selected sentence as either *essential* or *optional*. The essential sentences are crucial to the email conversation and have to be extracted in any case. The optional sentences are not critical to the conversation but are useful to help readers understand the email conversation if the given summary length permits. By classifying essential and optional sentences, we can distinguish the core information from the supporting ones. Moreover, the summarization accuracy changes a lot with different summarization length. With the choice of essential and optional, we ask the human summarizers to include about 30% of the total sentences. Thus, we can analyze the result and find the most convincing sentences that most human summarizers agrees on.

5.2 Result of the User Study

As we all know, summarization is a subjective activity. Different people make different choices. We cannot expect all summarizers agree to each other on all sentences. In addition, according to the design of the user study, essential sentences are more important than the optional ones. Thus, we give more weights to the essential selections. We assign a $GSValue$ for each sentence to evaluate its importance according to human summarizers' selection. The score is designed as follows. For each sentence s, one essential selection has a score of 3, one optional selection has a score of 1. Suppose k_e summarizers classify s as "essential", and k_o summarizers classify s as "optional". The GSValue of sentence s is $3 * k_e + k_o$. The reason that we choose 3 as the coefficient for essential selection is that we assume that two essential selections are more important than five optional selections. In this way, we can reduce the side effect bringing in by the 30% summarization length for each user. Thus, the GSValue of a sentence ranges from 0 to 15. If a sentence has a GSValue no less than 8, we take it as an *overall essential* sentence. The GSValue of 8 corresponds to 2 essential and 2 optional selections. Table 1 shows the possible cases of overall essential sentences. Intuitively, we can see that an overall essential sentence requires that at least 4 human summarizers select it in their summary and at least 2 select it as essential. It is obvious that the overall essential sentence are considered important by most summarizers. Out of the 741 sentences in the 20 conversations, 88 are overall essential sentences which is about 12% of the overall sentences. In addition, this also reveals that in general only about 12% sentences are agreed by most human summarizers.

With the scoring system discussed above, we study the human summarizers' selection in detail with respect to the following two aspects: hidden emails and significance of clue words.

Hidden emails - We sort all sentences by their GSValue and about 17% sentences in the top-30% sentences are from hidden emails. In addition, among the 88 overall essential sentences, about 18% sentences are from hidden emails. This clearly shows that the hidden emails do carry crucial information and have to be considered by the email summarization system.

GSValue	possible selections
8	"2 ess + 2 opt"
9	"2 ess + 3 opt" or "3 ess"
10	"3 ess + 1 opt"
11	"3 ess + 2 opt"
12	"4 ess"
13	"4 ess + 1 opt"
14	impossible score
15	"5 ess"

Table 1: GSValues and Possible Selections for Overall Essential Sentences

Significance of clue words - In the user study, we find that among the overall essential sentences in the gold standard, the clue words are very popular. Recall that the clue words are chosen based on the fragment quotation graph, this validates our hypothesis that the conversation structure(fragment quotation graph) plays an important role in the email summarization.

After we got the ClueScore of all the sentences, we study whether ClueScore is consistent with users' judgment. We do this by comparing the average ClueScore of overall essential sentences in the gold standard with the average of all other sentences. Figure 4 shows how significant the ClueScore is for the overall essential sentences. This figure is the histogram of the distribution of the ratios. The x-axis is the ratio which ranges from 0 to 18, and the y-axis is the number of conversations in that range. There are 3 conversations with a ratio of 1, meaning that there is no difference. At the other extreme, there is one conversation with a ratio of 16. For the remaining conversations, the ratio falls within [2,8]. The average ratio is 3.9. This suggests that ClueScore is significantly different between sentences in the gold standard and those that are not. Though there exist non-gold standard sentences whose ClueScore is very high, and there are gold standard sentences whose ClueScore is very low, in general the higher the ratio, the more likely a sentence with a high ClueScore is included in the gold standard. This suggests that CWS which uses ClueScore to rank sentences can lead to better precision and recall.

6. RESULT 2: EMPIRICAL EVALUATION OF CWS

The User Study provides us with a gold standard (GS), which identifies the overall essential sentences(i.e., GSValue ≥ 8) in the corpus. We apply three summarization methods to the 20 conversations and compare their result with GS. We use precision, recall and F-measure to evaluate the accuracy of those three summarization methods.

6.1 Comparing CWS with MEAD and RIPPER

We start by applying the summarization approach based on RIPPER to our dataset. We try two different ways to train RIPPER and compare it with MEAD and CWS.

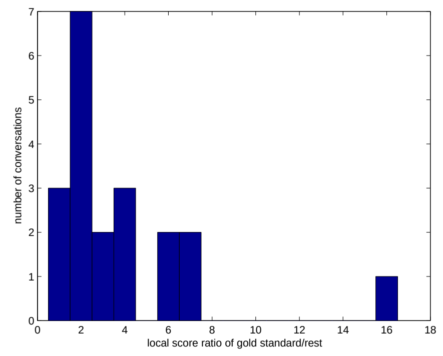


Figure 4: Ratio of ClueScore of Gold Standard Sentences and Non-Gold Standard Sentences

6.1.1 Sentence-level training

In sentence level training, we put all the sentences in the 20 conversations together. We use k-fold cross validation to train and test RIPPER, where k varies from 2 to 20. For each value of k, we repeat the experiment 10 times and take the average.

As described before, there are 3 feature sets: "basic", "basic+" and "basic++". Figure 5 shows that additional features generally improve the accuracy. However, too many features may reduce the accuracy in some cases. This is different from the result in [9], where the full feature set ("basic++") gets the highest accuracy. For the results of RIPPER reported hereafter, it is based on using the feature set "basic++".

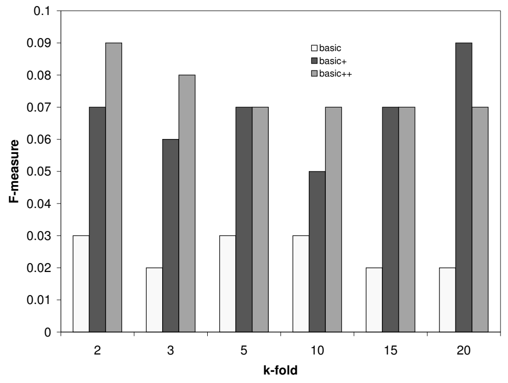


Figure 5: F-measure of RIPPER with 3 Feature Sets

Table 2 shows the precision of the summaries generated by all three methods. The reason why only precision is shown is that RIPPER surprisingly only selects 2% of the sentences to be included in a summary. For such a low percentage, recall is not informative. Remember that CWS and MEAD are capable of generating summaries of any length requested by the user, we force MEAD and CWS to produce summaries of the same length, so as to provide a fair comparison. To make CWS and MEAD function properly when all the sentences from the 20 conversations are pooled together, statistical standardization is applied to the ClueScore of sentences within each conversation before they are pooled. In this way, we avoid selecting too many sentences from one conversation simply because the ClueScore in one conversation is significantly higher than the rest. The first column of the table in Table 2 shows that even when CWS is forced to restrict the summary to a length of 2%, CWS still offers a precision doubling that of RIPPER. MEAD does surprisingly well as compared with RIPPER.

	sentence-level	conversation-level
RIPPER:	0.2	0.225
MEAD:	0.33	0.4
CWS:	0.4	0.35

Table 2: Precision of the Three Methods

6.1.2 Conversation-level training

In the experiment above, we train and test RIPPER by randomly selecting sentences in all conversations. This may not be accurate because the test set itself does not represent a conversation. Thus, to complement the sentence-level training above, we perform a separate experiment using conversation-level training. We apply a 20-fold cross validation as follows. In each round, we select 19 conversations as the training set and the remaining one conversation as the test set. This is repeated 20 times, with every conversation selected once as a test set.

As for CWS and MEAD, we force them to generate exactly the same number of sentences as selected by RIPPER. As is often the case for many conversations, RIPPER does not select any sentence. Whenever that happens, to allow the comparison to be completed, we force CWS and MEAD to return a single sentence with the highest score.

The experiment shows that out of the 20 conversations, RIPPER does not generate any summary for 12 conversations, and picks 9 sentences from the remaining 8. Five out of those 9 sentences belong to the gold standard. In contrast, CWS and MEAD selects 7 and 8 gold standard sentences out of 20 conversations respectively. The second column of the table in Table 2 summarizes the precision.

When compared with the results reported in [9], the relatively poor performance of RIPPER on the Enron data set is surprising in two ways. First, on the ACM data set used in [9], RIPPER could achieve a summary length around 20%. Furthermore, in [9], RIPPER easily outperforms centroid methods, like MEAD. We believe that there are two possible explanations. First, for our testbed, there are 20 conversations with 110 emails. In contrast, the ACM dataset contains 1600 emails. Secondly, the gold standard is obtained differently. In our user study, each conversation is reviewed by 5 human summarizers, and each summarizer directly selects essential and optional sentences. The gold standard sentences need to be selected by at least 4 summarizers and at least 2 of them select it as an essential sentence. Thus, the gold standard sentences are highly consistent among the summarizers. The side effect is that only 12% sentences are selected as overall essential in the gold standard sentences. This is different from the gold standard in [9], where two human summarizers write their own summaries for all the conversations. The gold standard is generated by comparing the similarity of each sentence to the generated one.

To verify the validity of the latter explanation, we relax the threshold for the gold standard and see whether RIPPER selects more sentences. When we reduce the GSValue threshold from 8 to 6, there are 19% gold standard sentences. The summary length of RIPPER increases to 8%. When we further reduce the threshold to 4, there are 30%

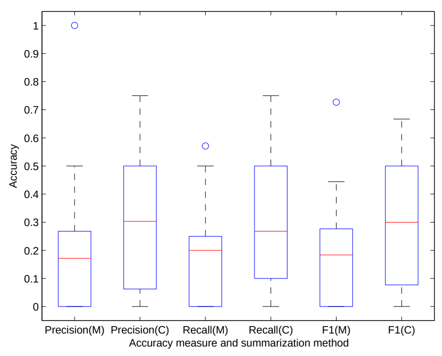


Figure 6: Box-plot for MEAD(M) and CWS(C) with $sumLen = 15\%$

gold standard sentences, and the summary length of RIPPER increases to about 25%. Because the training data is randomly selected, the number of positive examples in the training data generally agrees with the gold standard percentage. This shows that RIPPER is sensitive to the availability of positive examples.

From the above experiments, we can conclude that both CWS and MEAD have the following two advantages over RIPPER. First, they can guarantee to generate a summary for each conversation according to the need of the user. Second, CWS and MEAD show a better accuracy even when they are restricted to match the summary length of RIPPER.

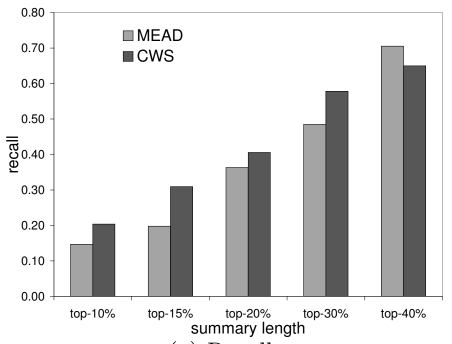
It is a natural idea to include ClueScore as another feature to the existing RIPPER feature set. Our experiment shows that there is little improvement and in some cases, it may be even worse. For lack of space, we do not include the details.

6.2 Comparing CWS with MEAD

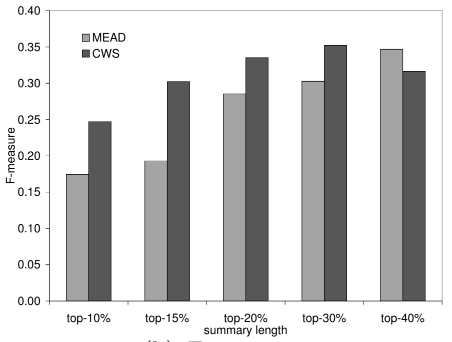
In the previous experiments, in order to match the summary length of RIPPER, we restrict the summary length of CWS and MEAD to a very low 2%. Below we compare CWS and MEAD at various lengths. The default percentage we pick is 15%, as this roughly corresponds to the gold standard percentage.

Figure 6 shows the box-plot of the effectiveness of CWS and MEAD when the summary length is 15%. This figure describes the statistic distribution of the two methods across all 20 conversations. The x-axis represents different evaluation measures in precision, recall and F-measure for MEAD and CWS, e.g., Precision(M) means the precision of MEAD with 15% summary length. The y-axis shows the values in decimals. The line in the middle of the box is the median(med) of all the values. The upper and lower edge of the box show the 75% and 25% of percentile of the data. The range between them is called an interquartile. The dot line outside the box shows the range of the data outside the interquartile. The length of the dot line is no more than the range of the interquartile. The data points beyond two tails are called outliers, which are drawn as circles. This figure also shows that both the median and the box of CWS is higher than that of MEAD respectively. And hence, CWS appear to be more accurate than MEAD with the summary length of 15%.

We also compute the student t-test for the accuracy of MEAD and CWS. The p-value we get for precision, recall



(a) Recall



(b) F-measure

Figure 7: Accuracy of CWS and MEAD with Various $sumLen$

and F-measure is marginally significant at 0.077 for precision, significant at 0.049 for recall and marginally significant at 0.053 for F-measure. Thus, we have at least 90% confidence that the mean of precision, recall and F-measure of MEAD and CWS differ when $sumLen = 15\%$.

In the experiments above, we only compare the accuracy of MEAD and CWS with a fixed summary length. In the following, we compare both methods using different summary lengths. Figure 7 shows the recall and F-measure for both methods under different summary length varying from 8% to 40%. In Figure 7(a) and Figure 7(b), the recall and F-measure of MEAD and CWS both improve with an increase of summary length. Thus, RIPPER may be missing "the boat" for not being able to produce summaries of varying lengths. When the summary length is no more than 30%, CWS seems to have a higher recall and F-measure than MEAD, while MEAD appears to be higher in recall and F-measure when the summary length is greater than 30%. The highest difference, which is greater than 0.1, takes place when summary length is 15% for all accuracy measures including precision, recall and F-measure. We can also see that the highest average F-measure seems to occur when the summary length is 30%.

Note that when the requested summary length is larger, the summarization task is easier. In this case, both methods appear to be fine. However, for shorter summaries, which is a harder summarization problem, CWS dominates, especially when the summary length is close to the size of the gold standard. Recall that CWS focuses on the local importance (wrt. the quotation graph), and MEAD focuses on the global importance. This result may reflect the relative importance of local and global importance with different summary lengths.

6.3 Effectiveness of the Fragment Quotation Graph

As a sanity check, we try two randomized selection algorithms to compare with CWS and MEAD. First, we try to randomly select k% sentences from each conversation and compute the precision, recall and F-measure. This method is labeled as "Random-sent" in Figure 8. Clearly, it shows that CWS dominates such a random approach. For that matter, combining Figure 8 with Figure 7(b) indicates that MEAD also dominates this random approach.

The second random approach is more intelligent. Here we create a random graph G_{random} and replace the fragment quotation graph, in the hope of evaluating the utility of a fragment quotation graph. The random graph contains exactly the same set of nodes as the fragment quotation graph, but the edges are randomly generated. We guaranteed that the minimum equivalent graph of G_{random} contains the same number of edges as the fragment quotation graph. For each fragment quotation graph we generate 3 random graphs and use the average as the accuracy.

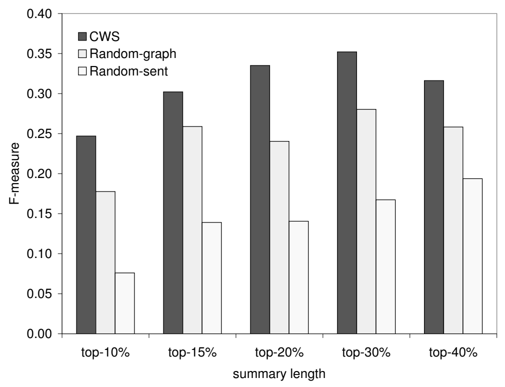


Figure 8: F-measure of CWS and Two Random Alternatives

Figure 8 shows the accuracy of CWS under the fragment quotation graph and the random graph shown with the label "Random-graph". The gap between "Random-graph" and "Random-sent" indicates the contributions of node identification and ClueScore. In all cases, the gap is significant.

The gap between "Random-graph" and CWS indicates the importance of the edges of the fragment quotation graph. In all cases, CWS gives a higher value. Interestingly, combining Figure 8 with Figure 7(b) indicates that "Random-graph" outperforms MEAD in some cases. This set of experiments clearly shows the value of the fragment quotation graph for summarizing conversations.

6.4 Examining Individual Conversations

In addition to measuring the average accuracy, Figure 9 shows the accuracy of CWS in each conversation when the summary length is 15%. The x-axis is the 20 conversations, and the y-axis is the accuracy in terms of precision, recall and F-measure. There are 5 conversations where the summary contains no overall essential sentence. Among those 5 conversations, MEAD cannot select any overall essential sentences in 4 of them either. Table 3 shows the total number of sentences, number of gold standard sentences and their GSValue for the 5 conversations. Among the 5 conversations, we can see that they either have a smaller number of sentences, or have few GS sentences(dasovich-j.3 has only one). Notice that those gold standard sentences have relatively smaller GSValues even for those selected as GS sentences. We can conclude that even human summarizers are not consistent on their summary of those conversations. This shows that email summary is a challenging job when the summary length is short due to the characteristic of emails, e.g., the email length.

folder name	# of sent.	# of GS sent.	GSValues
buy-r.0	25	4	9, 9, 8, 8
buy-r.4	15	1	8
dasovich-j.3	40	1	9
nemec-g.6	23	2	11, 8
shackleton-s.2	50	5	13,10, 9, 9, 9

Table 3: Zero Selection Folders

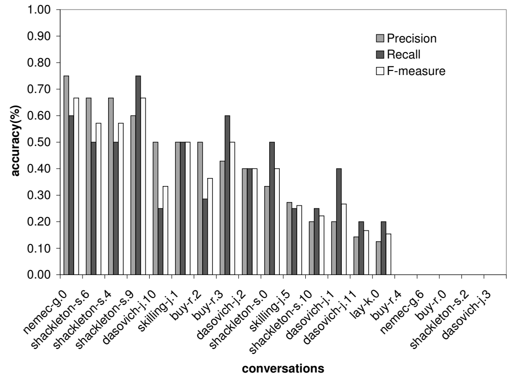


Figure 9: Accuracy of CWS on Individual Folders

6.5 Hybrid Methods

In this section, we study whether the linear combination of ClueScore and MEADScore can improve the accuracy. Figure 10 shows the change of the F-measure with different percentage of ClueScore (from 0% which corresponds to pure MEAD to 100% which corresponds to pure CWS). This figure shows that, when the summary length is less or equal than 20%, the linear combination cannot improve the accuracy of CWS (pure 100% CWS is the max). In contrast, when the summary length is greater than 20%, there is benefit to combine MEAD and CWS together (the max is between 0% and 100%). Notice that the accuracy usually get its maximal value when the percentage of ClueScore is 80%. In other words, CWS can benefit from a little bit of MEAD when the summary is relatively long.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we study how to generate accurate email summaries. We analyze the characters of emails and study the email conversation structure, which we argue have not been sufficiently investigated in previous research on email summarization. We build a novel structure: the fragment quotation graph, to represent the conversation structure. This graph includes hidden emails and can represent the conversation in more details than a simple threading structure. Based on the fragment quotation graph, we also develop a new summarization approach CWS to select important sentences from an email conversation. Our experiments

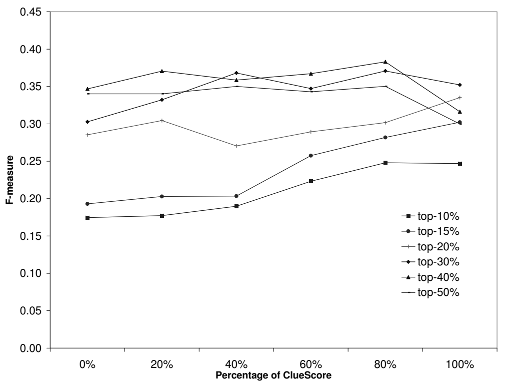


Figure 10: Average F-measure of Linear Combination of CWS and MEAD

with the Enron email dataset not only indicate that hidden emails have to be considered for email summarization, but also shows that CWS can generate more accurate summaries when compared with other methods.

The CWS approach relies on a simple algorithm and is very easy to implement. Yet, it appears to work better than other existing approaches. Since CWS is built on the fragment quotation graph, we believe this can be at least partially attributed to the use of the fragment quotation graph. Our experiments on the random graphs also support this. Others have also argued that the conversation structure is very important for email summarizations, we claim that it should be paid even more attention. Furthermore, we believe that it is promising to combine the CWS methods together with other methods.

Our future plan includes improving the fragment quotation graph generation with more sophisticated linguistic analysis. For example, we can use some linguistic analysis to decide whether we need to add an edge or not.

In order to verify the generality of our findings, we are also working on evaluating our methods with different real-life data sets; creating the gold standard for a large real-life data set requires a lot of effort. Last but not least, we want to explore how to combine CWS with several machine learning algorithms. The low selection rate of RIPPER does not mean that the features are not useful. In a preliminary study, we find that different classifiers have very different selection rates and accuracy. We will look into this issue in the near future.

8. REFERENCES

[1] Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. In *Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97), ACL, Madrid, Spain*, 1997.

[2] Giuseppe Carenini, Raymond T. Ng, and Xiaodong Zhou. Scalable discovery of hidden emails from large folders. In *ACM SIGKDD'05*, pages 544–549, 2005.

[3] Giuseppe Carenini, Raymond T. Ng, Xiaodong Zhou, and Ed Zwart. Discovery and regeneration of hidden emails. In *ACM SAC '05: Proceedings of the 2005 ACM Symposium on Applied Computing*, pages 503–510, 2005.

[4] Chris Schmandt Derek Lam, Steven L. Rohall and Mia K. Stern. Exploiting e-mail structure to improve summarization. In *CSCW'02 Poster Session*, 2002.

[5] Günes Erkan and Dragomir R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research(JAIR)*, 22:457–479, 2004.

[6] Aaron Harnly Jen-Yuan Yeh. Email thread reassembly using similarity matching. In *Third Conference on Email and Anti-Spam (CEAS)*, July 27 - 28 2006.

[7] Dragomir R. Radev, Hongyan Jing, Malgorzata Styś, and Daniel Tam. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40:919–938, December 2004.

[8] Ramakrishnan Srikant Rakesh Agrawal, Sridhar Rajagopalan and Yirong Xu. Mining newsgroups using networks arising from social behavior. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pages 529–535, 2003.

[9] Owen Rambow, Lokesh Shrestha, John Chen, and Chirsty Lauridsen. Summarizing email threads. In *HLT/NAACL*, May 2–7 2004.

[10] Gordon Rios and Hongyuan Zha. Exploring support vector machines and random forests for spam detection. In *First Conference on Email and Anti-Spam (CEAS)*, July 30 - 31, 2004.

[11] Lokesh Shrestha and Kathleen McKeown. Detection of question-answer pairs in email conversations. In *Proceedings of COLING'04*, pages 889–895, August 23–27 2004.

[12] Salvatore J. Stolfo, Shlomo Hershkop, Chia-Wei Hu, Wei-Jen Li, Olivier Nimeskern, and Ke Wang. Behavior-based modeling and its application to email analysis. *ACM Trans. Inter. Tech.*, 6(2):187–221, 2006.

[13] Jie Tang, Hang Li, Yunbo Cao, and ZhaoHui Tang. Email data cleaning. In *ACM SIGKDD'05*, pages 489–498, 2005.

[14] Gina Danielle Venolia and Carman Neustaedter. Understanding sequence and reply relationships within email conversations: a mixed-model visualization. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 361–368, 2003.

[15] Stephen Wan and Kathleen McKeown. Generating overview summaries of ongoing email thread discussions. In *Proceedings of COLING'04, the 20th International Conference on Computational Linguistics*, August 23–27 2004.

[16] Xiaojun Wan and Jianwu Yang. Improved affinity graph based multi-document summarization. In *HLT/NAACL*, June 2006.

[17] Steve Whittaker and Candace Sidner. Email overload: exploring personal information management of email. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 276–283, 1996.

[18] Jihoon Yang and Sung-Yong Park. Email categorization using fast machine learning algorithms. In *Discovery Science*, pages 316–323, 2002.