# Generating Overview Summaries of Ongoing Email Thread Discussions

**Stephen Wan**
Department of Computing
Macquarie University
Sydney NSW 2109
swan@ics.mq.edu.au

**Kathy McKeown**
Columbia University
Department of Computer Science
1214 Amsterdam Avenue
NY - 10027-7003, USA
kathy@cs.columbia.edu

## Abstract

The tedious task of responding to a backlog of email is one which is familiar to many researchers. As a subset of email management, we address the problem of constructing a summary of email discussions. Specifically, we examine ongoing discussions which will ultimately culminate in a consensus in a decision-making process. Our summary provides a snapshot of the current state-of-affairs of the discussion and facilitates a speedy response from the user, who might be the bottleneck in some matter being resolved. We present a method which uses the structure of the thread dialogue and word vector techniques to determine which sentence in the thread should be extracted as the main issue. Our solution successfully identifies the sentence containing the issue of the thread being discussed, potentially more informative than subject line.

## 1 Introduction

Imagine the chore of sifting through your overflowing email inbox after an extended period away from the office. The discovery that some of these emails form part of a larger decision-making discussion only heightens the sense of urgency and stress. Such a discussion may require an urgent response and a user's lack of contribution may be a bottleneck in some matter being resolved. Such a scenario is seems quite familiar and intuitively, one would expect that better solutions to presenting the contents of the email inbox might be useful in facilitating a timely reply to a missed email discussion.

One such solution might be a summary of that very email discussion. However, it would be much more useful if the summary did not just tell the user what the thread is about. Such information might be easily obtained from the subject line, or if not, a conventional off-the-shelf summarizer might provide the gist of the thread quite easily.

However, in contrast to a conventional sentence extraction summary in Figure 1, the ideal summary ought to provide sufficient information about the current state-of-affairs of the discussion, in order to minimize any further delay in the matter being resolved. Specifically, this might include a description of the matter being discussed and the responses received so far. An example of such a summary is presented in Figure 2. In this example, it is not sufficient to know that a plaque is being designed. Crucially, the wording of the plaque is under discussion and requires feedback from the thread participants. It is not difficult to appreciate the usefulness of such a summary to avoid writing responses to older, and hence irrelevant, emails. Accordingly, we envisage that the resulting summary to be not just indicative of the thread content but informative Borko (1975).

1. Here's the plaque info.
2. http://www.affordableawards.com/plaques/ordecon.htm
3. I like the plaque, and aside for exchanging Dana's name for "Sally Slater" and ACM for "Ladies Auxiliary", the wording is nice.
4. We just need to contact the plaque folks and ask what format they need for the logo.

Figure 1. Example summary from a conventional sentence extraction summarizer

*Issue:* Let me know if you agree or disagree w/choice of plaque and (especially) wording.

*Response 1:* I like the plaque, and aside for exchanging Dana's name for "Sally Slater" and ACM for "Ladies Auxiliary", the wording is nice.
*Response 2:* I prefer Christy's wording to the plaque original.

Figure 2. Example summary from our system

We present a novel approach which identifies the main issue within the email and finds the responses to that issue within subsequent emails. Our approach uses a combination of traditional vector space techniques and Singular Value Decomposition (SVD). We rely on the premise that the participants of the discussion have implicitly determined which sentence from the initiating email of the thread is most important and that we can see evidence of this inherent in the content of their respective replies.

In the remainder of the paper, we provide background on email usage and our observations of discussion thread structure in Section 2 to support our basic premise. Section 3 provides a description of related work in the area. To date, use of dialogue structure has mostly been limited to finding question-answer pairs in order to extract the pairing as a whole for the sake of coherence. We present a more formal description of the problem we are addressing and our algorithms for issue in Section 4. Section 5 outlines our handling of response extraction. In Section 6, we present a preliminary evaluation we have conducted along with the results. Finally we end with concluding remarks in Section 7.

## 2 Background: Email Threads

### 2.1 Email Discussions supporting a Decision-Making Process

The focus of this paper is on email discussions supporting a group decision-making process. In contrast to studies on individual email usage (for an overview see: Ducheneaut and Bellotti, 2001), this research area has been less explored. Occasionally, such discussions end with an online vote. However, Ducheneaut and Belotti do note that voting is relatively infrequent and our own experience with our email corpora tends to support this.

In general, we expect that these threads contain supporting discussions, and the actual decision might occur outside of the email medium, for example in a board meeting. What we hope to observe is that, for some issue discussed, candidate solutions and responses highlighting the pros and cons of a solution are introduced via email.

Decision-making discussion threads occur frequently enough in environments which depend on professional usage of email. In the corpus we examined, 40% of the threads were decision-making discussions.

### 2.2 Constraints on and Choice of a Corpus of Email Discussions

To collect a corpus of these threads, we placed a few constraints on the mailing list archives we found online.

To begin with, we focused on threads from mailing lists that were set up to support organization activities as these often involve decision-making processes. As we are also interested in examining the role of dialogue, we required access to the email thread structure from which we can infer a basic dialogue structure.

We chose to use the archives of the Columbia University ACM Student Chapter Committee as this group has organized several events and used email as their primary mode of communication outside of meetings. For practical reasons, it was relatively straightforward to obtain the necessary permissions to use the data, something that might be more difficult for other archives. Possible alternative corpora might be the mailing lists of organizing committees, for example that of a conference organizing committee or a steering group. Project-based mailing lists might also be potentially used, especially if the group participants have sufficient shared background to engage in discussions.

### 2.3 Observations on Thread Structure

The Columbia University ACM Student Chapter Committee was made up of about 10 people. Upon initial examination of the data, we found that we could classify the threads of email according to its purpose. The set of group tasks facilitated by the email correspondence were: decision-making, information provision, requests for action and social conversation.

However, it is natural for the group to engage in multiple tasks. Thus, we use the term “task shift” to refer to adjacent segments of the thread (comprised of emails) which reflect distinct group goals. In the corpus we use, we observe that these tasks usually occur sequentially. In some cases, a single email proposes more than one issue for discussion, and subsequent responses address each of these in turn.

Intuitively, it makes sense to create a summary for a single task. Accordingly, we have designed our algorithm to accept only dialogue structures addressing a single group task. If discussions invoke short clarification questions, these should not be treated differently if the task remains the same. One supporting reason for this is the syntactic variation with which participants express disagreement. We have observed that disagreement is often expressed as a clarification question, or as a question which offers an alternative suggestion.

## 3 Related Work

To date, email thread summarization has not been explored in any great depth within the Natural Language Processing (NLP) research community.

Research on thread summarization has included some work on using dialogue structure for email summarization. Nenkova et al. (2003) advocate the use of *overview sentences* similar to ours.

They extract sentences based on the presence of subject line key words. However, should the subject line not reflect the content of the thread, our method has the potential to extract the true discussion issue since it based on the responses of other participants.

Lam et al. (2002) use the context of the preceding thread to provide background information for email summaries. However, they note that even after appropriate preprocessing of email text, simply concatenating preceding context can lead to long summaries. In contrast, instead of extracting email texts verbatim, we extract single sentences from particular emails in the thread. As a result, our summaries tend to be much shorter.

Murakoshi et al. (1999) describe an approach which extracts question-answer pairs from an email thread. Extraction is based on the use of pattern-based information extraction methods. The summary thus provides the question-answer pair intact, thereby improving the coherence. Question-answer summaries would presumably be suited to discussions which support an information provision task, a complementary task to the one we examine.

Rambow et al. (2004) apply sentence extraction techniques to the thread to construct a generic summary. Though not specifically using dialogue structure, one feature used marks if a sentence is a question or not.

Work has also been done on more accurately constructing the dialogue structure. Newman and Blitzer (2003) focus on clustering related newsgroup messages into dialogue segments. The segments are then linked using email header information to form a hierarchical structure. Their summary is simply the first sentence from each segment. We envisage dialogue structure summaries showing an overview of topics would be combined with approaches such as ours which provide summaries of segments.

We also note the existing work that explores the summarization of speech transcripts. Speech is a very different mode of communication. An overview of the differences between asynchronous and synchronous modes of communication is provided by Clark (1991) and Simpson-Young et al. (2000). Alexandersson et al. (2000) note that in speech there is a tendency not to repeat shared conversation context. They use the preceding dialogue structure, modeled using Dialogue Representation Theory, to provide additional ellipsed information. It is unclear how such an approach might apply to an email corpus which has the potential to cover a broader set of domains.

More recently, Zechner and Lavie (2001) identify question-answer dialogue segments in order to extract the pair as a whole.

Hillard et al. (2003) have also produced a system which generates summaries of speech discussions supporting a decision-making process. Their work differs from ours in that they focus on categorizing the polarity of responses in order to summarize consensus.

## 4 Issue Detection

To make the problem more manageable we make the following assumptions about the types of threads that our algorithm will handle. To begin with, we assume that the threads have been correctly constructed and classified as discussions supporting decision-making. Needless to say, the first assumption is a little unrealistic given that thread construction is a difficult problem. For example, it is not uncommon to receive emails with recycled subject lines simply because replying to an email is often more convenient than typing in an address.

The other assumptions we make have to do with the dialogue structure of the threads. The first is that the issue being discussed (usually a statement describing the matter to be decided) is to be found in the first email. The second is that the email thread doesn’t shift task, nor does it contain multiple issues.

The first assumption is based on what we have observed to be normal behavior. Exceptions to this rule are broken threads and cases where the participants have responded to a forwarded email. In the first case, this can be seen as an error in thread construction and identification. In such cases however, even in such a thread, the first email usually contains a reference to the issue at hand, although it may be an impoverished paraphrase. Our algorithm extracts these paraphrases in lieu of the original wording. Cases where participants have responded to a forwarded email are not common. For such threads, we attempt to extract the sentence participants respond to. However, again, this may not be the best formulation of the issue.

Secondly, we assume that a text segmentation algorithm (for examples see Hearst’s “Text-Tiling” algorithm 1997, Choi et al. 2000) has already segmented the threads according to shifts in task. Operationally, our detection of shifts in task would then be based on corresponding changes in vocabulary used.

### 4.1 The Algorithm

Our summarization approach is to extract a set of sentences consisting of one issue, and the corresponding responses – one per participant. Our sentence extraction mechanisms borrow from information retrieval methods which represent text as weighted term frequency vectors (for an overview see: Salton and McGill, 1983).

In Figure 3, we present the general framework of the algorithm. In this framework we divide the thread into two parts, the initiating email and the replies. We create a *comparison vecto*r that represents what the replies are about. We can construct variations of this framework by changing the way we build our comparison vector. The aim is to compare each sentence to the comparison vector for the replies. Thus, we build separate vector representations, called candidate vectors, for each sentence in the first email. Using the cosine similarity metric to compare candidate vectors with the comparison vector, we rank the sentences of the first email. Conceptually, the highest ranked sentence will be the one that is closest in content to the replies and this is extracted as the issue of the discussion.

```
1. Separate thread into issue_email and replies
2. Create "comparison vector" V representing replies
3. For each sentence s in issue_email
   3.1 Construct vector representation S for sentence s
   3.2 Compare V and S using cosine similarity
4. Rank sentences according to their cosine similarity scores
5. Extract top ranking sentence
```

Figure 3. Framework for extracting discussion Issues.

We now discuss the four methods for building the comparison vector. These are:

1. The Centroid method
2. The SVD Centroid method.
3. The SVD Key Sentence method
4. Combinations of methods: Oracles

### 4.1.1 The Centroid Method

In the Centroid method, we first build a term by sentence ($t \times s$) matrix, $A$, from the reply emails. In this matrix, rows represent sentences and columns represent unique words found in the thread. Thus, the cells of a row store the term frequencies of words in a particular sentence. From this matrix, we form a centroid to represent the content of the replies. This is a matter of summing each row vector and normalizing by the number of rows. This centroid is then what we use as our comparison vector.

### 4.1.2 The SVD Centroid Method

Our interpretation of the SVD results is based on that of Gong and Liu (1999) and Hoffman (1999). Gong and Liu use SVD for text segmentation and summarization purposes. Hoffman describes the results of SVD within a probabilistic framework. For a more complete summary of our interpretation of the SVD analysis see Wan et al. (2003).

To begin with, we construct the matrix A as in the Centroid Method. The matrix A provides a representation of each sentence in *w* dimensionality, where *w* is the size of the vocabulary of the thread. The SVD analysis[1] is the product of three matrices *U*, *S* and *V* transpose. In the following equation, dimensionality is indicated by the subscripts.

$$\mathrm{SVD}(A_{t \times s}) = U_{t \times r} S_{r \times r} (V_{s \times r})^{tr}$$

Conceptually, the analysis essentially maps the sentences into a smaller dimensionality *r*, which we interpret as the main "concepts" that are discussed in the sentences. These dimensions, or concepts, are automatically identified by the SVD analysis on the basis of similarities of co-occurrences. The rows of *V* matrix represent the sentences of the first email, and each row vector describes how a given sentence relates to the discovered concepts. Importantly, the number of discovered concepts is less than or equal to the vocabulary of the thread in question. If it is less than the vocabulary size, then the SVD analysis has been able to combine several related terms into a single concept. Conceptually, this corresponds to finding word associations between synonyms though in general, this association may not conserve part-of-speech. In contrast to the values of the *A* matrix which are always positive (since they are based on frequencies), the values of each cell in the *V* matrix can be negative. This represents the degree to which the sentence relates to a particular concept. We build a centroid from the V matrix to form our comparison vector.

### 4.1.3 The SVD Key Sentence Method

The SVD Key Sentence Method is similar to the preceding method. We build the matrix *A*, apply the SVD analysis and obtain the matrix *V*. Instead of constructing a vector which represents all of the replies, we choose one sentence from the replies that is most representative of the thread content. This is done by selecting the most important concept and finding the sentence that contains the most words related to it. The SVD analysis by default sorts the concepts according to degree to which sentences are associated with it. By this definition, the most important sentence is

[1] We use the *SVD* function in the JAMA Java Matrix Package (http://math.nist.gov/javanumerics/jama/) to compute the analysis.

represented by the values in the first column of the matrix *V*. We then take the maximum of this column vector and note its row index, *r*, which denotes a sentence. We use the *rth* row vector of the *V* matrix as the comparison vector.

In both the SVD Centroid method and the SVD Key Sentence method, the comparison vector has a different dimensionality than the candidate vectors. To perform the comparison, we must map the candidate vectors into this new dimensionality. This is done by pre-multiplying each candidate vector with the result of the matrix multiplication: $U_{\text{transpose}} \times S$. Both of the matrices involved are obtained from the SVD analysis.

#### 4.1.4 Combinations of methods: Oracles

Since we have three alternatives for constructing the comparison vector we consider the possibility of combining the approaches. In Wan et al. (2003) we showed that using a combination of traditional TF*IDF approaches and SVD approaches was useful given that SVD provided additional information about word associations. Similarly, our two SVD methods provide complementary information. The vector computed by the SVD centroid method provides information about the replies and accounts for word associations such as synonyms. However, like the centroid method, this vector will include all topics discussed in the replies, even small digressions. In contrast, the SVD Key sentence is potentially better at ignoring these digressions by focusing on a single concept.

We present three heuristic oracles which essentially re-rank the candidate issue sentences identified by each of the three methods. Re-ranking is based on a voting mechanism. The rules for three oracles are presented in Figures 4 and 5.

1. If a majority exists return it
2. If tie then:
   retrieve the lowest index number *i*, where $i \neq 1$
3. If all methods return different answers, then choose Centroid Method's answer

Figure 4. Oracle 1 heuristic rules

The oracle in Figure 4 attempts to choose the best sentence, retrieving a single sentence. Rule 2 attempts to encode the intuition that the issue sentence is likely to occur early in the email, however, not usually at the top of the email. Finally, we use the Centroid Method as a default because it is less prone to errors arising from low vocabulary sizes found in shorter threads. For such threads, we found that SVD approaches tend not to perform so well.

The second oracle again relies on a majority vote. However, it relaxes the constraint of just returning a single sentence if the majority is the first sentence of the email. Since we tend not to find issue sentences in at the very top of emails, we return all possible issue sentences in rule 1.

1. If a majority exists then return it; **UNLESS** $i = 1$ in which case, return all choices
2. If tie then retrieve the lowest index number *i*, where $i \neq 1$
3. If all methods return different answers, then choose Centroid Method's answer

Figure 5. Oracle 2 heuristic rules

Finally, as a baseline, the third oracle returns all the possible issue sentences identified by all of the contributing methods.

## 5 Extracting the Responses to the Issue

To extract the responses to the issue, we simply take the first sentence of the replies of each responding participant. We make sure to only extract one response per participant.

An alternative solution analogous to that of issue detection was also considered. In this solution, we applied the issue detection algorithm to the reply email in question. However, it turns out that most of the tagged responses occurred at the start of each reply email and a more complex approach was unnecessary and potentially introduced more errors.

## 6 Evaluation of Issue Detection Algorithms

### 6.1 The Test Data

The test data used was a portion of the Columbia ACM Student Chapter corpus. This corpus included a total of 300 threads which were constructed using message-ID information found in the header. On average, there were 190 words per thread and 6.9 sentences in the first email.

Threads longer than two emails[2] were categorized manually. We identified discussions that supported a decision-making process. For these, we manually annotated the issue of the thread and the responses to the issue. Although we do not currently use this information, we also classified the responses as being either in agreement or disagreement. According to the assumptions listed in Section 4, we discarded those threads in which the issue was not found in the first email. In total, we identified 37 discussion

[2] Longer threads offered a great chance of identifying a discussion.

threads, each of which forms a test case. A manual annotation of the discussion issues was done by following the instruction: *Select the sentence from the first email that subsequent emails are responding to.*" These annotated issue sentences formed our gold standard.

Our approach was designed to operate on the new textual contributions of each participant. Thus, the emails underwent a limited preprocessing stage. Email headers, automatically embedded "reply context" text and static signatures were ignored.

## 6.2 Evaluation Framework and Results

The evaluation was designed to test if our methods which use dialogue structure improve sentence extraction results. We used the recall-precision metric to compare the results of a system with the manually annotated gold standard. In total, we tested 6 variations of our issue detection algorithms. These included the Centroid method, the SVD Centroid method and the SVD Key Sentence method and the 3 oracles.

For each test case, the approach being tested was used to extract one or more sentences corresponding to the issue of the discussion, which was then compared to the gold standard. The baseline used was the first *n* sentences of the first email as a summary, where *n* ranged from 1 to 3 sentences.

The recall-precision results of the evaluation are presented in Table 1. On average, the chance of correctly choosing the correct sentence randomly in a test set was 21.9%.

We used an ANOVA to test whether there was an overall effect between the various methods for recall and precision. We rejected the null hypothesis, that is, the choice of method does affect recall and precision ($\alpha$=0.05, $df_{numerator}$= 8, $df_{denoinator}$= 324).

To determine if our techniques were statistically significant compared to the baselines, we ran pair-wise two-tailed student t-tests to compare the three methods and the first oracle to the n=1 baseline since these all returned a single sentence. The results are presented in Table 2. Similarly, Table 3 shows the t-test comparisons for the oracle and oracle baseline against the n=3 baseline.

Except for the SVD Key Sentence method, all the methods were significantly better than the n=1 baseline. However, a useful recall score was only obtained using the oracle methods. When comparing the oracle methods which returned more than one sentence against the n=3 baseline, we found no significant difference in recall. However, when comparing precision performance we found that the difference between the precision of Centroid method and the three oracles were significantly different compared to the baseline.

| Method | Ave.Recall % | Ave. Prec. & |
|---|---|---|
| Centroid | 62.2 | 62.2 |
| SVD Centroid | 48.6 | 48.6 |
| SVD Key Sent | 37.8 | 37.8 |
| Oracle 1 | 62.2 | 62.2 |
| Oracle 2 | 70.3 | 62.7 |
| Oracle Baseline | 83.8 | 45.1 |
| Baseline n=1 | 24.3 | 24.3 |
| Baseline n=2 | 48.6 | 24.3 |
| Baseline n=3 | 64.0 | 21.6 |

Table 1. Average recall and precision values for each method.

| Method | Prob(Recall) | Prob(Prec.) |
|---|---|---|
| Centroid | 0.0016 | 0.0016 |
| SVD Centroid | 0.0187 | 0.0187 |
| SVD Key Sent | 0.1601 | 0.1601 |
| Oracle 1 | 0.0004 | 0.0004 |

Table 2. Pair-wise t-test scores comparing each method to the n=1 baseline (df = 36). The values show the probability of the obtained t value.

| Method | Prob(Recall) | Prob(Prec.) |
|---|---|---|
| Oracle 2 | 0.5998 | 0.0001 |
| Oracle Baseline | 0.1686 | 0.0108 |

Table 3. Pair-wise t-test scores comparing each method to the n=3 baseline (df = 36). The values show the probability of the obtained t value.

The recall and precision statistics for the Centroid method was the most impressive of the three methods proposed, far outperforming the baseline. The results of comparisons involving the oracles, which combine the three methods, showed improved performance, suggesting that such techniques might potentially be useful in an email thread summary. Whilst there was little difference between the recall values of the three oracles and the baselines, the benefit of using a more involved approach such as ours is demonstrated clearly by the gain in precision performance which will impact the usefulness of such a summary. It is also interesting to note that the performance of the oracles was achieved by simply using simple rules without any corpus training.

## 7 Conclusion and Future Work

The methods described in this paper would form part of a larger email thread summarizer able to identify task boundaries and then initiate the appropriate summarization strategy for that task. We have addressed the sub-problem of summarizing the decision-making processes which have been supported by discussions over email. Despite the preliminary nature of our investigation, our findings are encouraging and lend support to the view that a combination of simple word vector approaches with singular value decomposition approaches do well at extracting discussion issues. Such methods, even with only a simple notion dialogue structure achieve a useful level of recall and precision. We would like to conduct extrinsic experiments to test our assumptions about the usefulness of these summaries. Further investigations will also focus on examine issues of scalability, with regard to group size, and domain independence. We would also like to investigate how issue detection might be integrated with a more complete solution to email thread summarization.

## 8 Acknowledgements

The research described in this paper was partially supported by a grant provided through NSF's Knowledge Discovery and Dissemination Program. We would like to thank the NLP groups of Columbia University, Macquarie University and CSIRO for feedback received on this work.

## References

J. Alexandersson, P. Poller, M. Kipp, and R. Engel. 2000. Multilingual Summary Generation in a Speech-To-Speech Translation System for Multilingual Dialogues. In *Proc. of INLG-2000*, Mitzpe Ramon, Israel.

B. Simpson-Young, N. Ozkan, C.Paris, C. Chung, J. Brook, K. Yap. 2000 Video Messaging: Addressing the Characteristics of the Medium. In the *Proceedings of the Euromedia 2000 Conference*. Antwerp, May 2000.

Borko, H., and Bernier, C. 1975 *Abstracting Concepts and Methods*. New York: Academic Press.

F.Y.Y. Choi. 2000 Advances in domain independent linear text segmentation. In the *Proc. of the North American Chapter of the Association. for Comp. Linguistics,* pp. 26-33.

Clark, H.H. and S.E Brennan. 1991. Grounding in Communication" In *Readings in Groupware and Computer-Supported Collaborative Work,* R.M. Baeker, ed. Morgan Kaufmann, California, 222-223.

Nicolas Ducheneaut, Victoria Bellotti 2001. E-mail as habitat: an exploration of embedded personal information management, interactions. in *Communications of the ACM* v.8 n.5, p.30-38.

Dustin Hillard, Mari Ostendorf, and Elizabeth Shriberg 2003. Detection Of Agreement vs. Disagreement In Meetings: Training With Unlabeled Data. in the *Proc. HLT-NAACL Conference*, Edmonton, Canada, May 2003.

Gong Y., and Liu, X. 2001. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. In the *Proceedings SIGIR 2001*: pages 19-25.

M. Hearst. 1994. Multi-paragraph segmentation of expository text. In the *Proc. of the 2nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, NM.

Hiroyuki Murakoshi, Akira Shimazu, and Koichiro Ochimizu. 1999. Construction of Deliberation Structure in Email Communication. In Proceedings of the *Pacific Association for Computational Linguistics (PACLING'99)*, pages 16--28, Aug.

T. Hofmann. 1999. Probabilistic latent semantic analysis, in the *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence,* Morgan Kaufmann Publishers, San Francisco, CA, pp. 289-296.

Lam, Derek and Rohall, Steven L. and Schmandt, Chris and Stern, Mia K. 2002. *Exploiting E-mail Structure to Improve Summarization*. Technical Paper at IBM Watson Research Center #20-02

Ani Nenkova and Amit Bagga. 2003. Facilitating email thread access by extractive summary generation. In Proceedings of RANLP, Bulgaria.

Newman and Blitzer, Paula Newman and John Blitzer. 2002. Summarizing Archived Discussions: a Beginning. In the *Proceeding of Intelligent User Interfaces*.

G. Salton and M. J. McGill. 1983. *Introduction to modern information retrieval,* McGraw-Hill, New York.

Owen Rambow, Lokesh Shrestha, John Chen and Christy Laurdisen. 2004. Summarizing Email Threads. In the Proc. of HLT-NAACL 2004: Short Papers.

Stephen Wan, Mark Dras, Cécile Paris, Robert Dale. 2003. Using Thematic Information in Statistical Headline Generation. In the *Proceedings of the Workshop on Multilingual Summarization and Question Answering at ACL 2003*, July 11, Sapporo, Japan

K. Zechner and A. Lavie. 2001 Increasing the coherence of spoken dialogue summaries by cross-speaker information linking. In *Proceedings of the NAACL-01 Workshop on Automatic Summarization*, Pittsburgh, PA, June, 2001.