Summarizing Email Threads

Owen Rambow Columbia University New York, NY, USA

Lokesh Shrestha Columbia University New York, NY, USA

John Chen Microsoft Research Asia Beijing, China

Chirsty Lauridsen Columbia University New York, NY, USA

rambow@cs.columbia.edu, lokesh@cs.columbia.edu
v-johnc@msrchina.research.microsoft.com, christy@columbia.edu

Abstract

Summarizing threads of email is different from summarizing other types of written communication as it has an inherent dialog structure. We present initial research which shows that sentence extraction techniques can work for email threads as well, but profit from email-specific features. In addition, the presentation of the summary should take into account the dialogic structure of email communication.

1 Introduction

In this paper, we discuss work on summarizing email threads, i.e., coherent exchanges of email messages among several participants.[1] Email is a written medium of asynchronous multi-party communication. This means that, unlike for example news stories but as in face-to-face spoken dialog, the email thread as a whole is a collaborative effort with interaction among the discourse participants. However, unlike spoken dialog, the discourse participants are not physically co-present, so that the written word is the only channel of communication. Furthermore, replies do not happen immediately, so that responses need to take special precautions to identify relevant elements of the discourse context (for example, by citing previous messages). Thus, email is a distinct linguistic genre that poses its own challenges to summarization.

In the approach we propose in this paper, we follow the paradigm used for other genres of summarization, namely sentence extraction: important sentences are extracted from the thread and are composed into a summary. Given the special characteristics of email, we predict that certain email-specific features can help in identifying relevant sentences for extraction. In addition, in presenting the extracted summary, special "wrappers" ensure that the reader can reconstruct the interactional aspect of the thread, which we assume is crucial for understanding the summary. We acknowledge that other techniques should also be explored for email summarization, but leave that to separate work.

[1]The work reported in this paper was funded under the KDD program. We would like to thank three anonymous reviewers for very insightful and helpful comments.

2 Previous and Related Work

Muresan et al. (2001) describe work on summarizing individual email messages using machine learning approaches to learn rules for salient noun phrase extraction. In contrast, our work aims at summarizing whole threads and at capturing the interactive nature of email.

Nenkova and Bagga (2003) present work on generating extractive summaries of threads in archived discussions. A sentence from the root message and from each response to the root extracted using *ad-hoc* algorithms crafted by hand. This approach works best when the subject of the root email best describes the "issue" of the thread, and when the root email does not discuss more than one issue. In our work, we do not make any assumptions about the nature of the email, and learn sentence extraction strategies using machine learning.

Newman and Blitzer (2003) also address the problem of summarizing archived discussion lists. They cluster messages into topic groups, and then extract summaries for each cluster. The summary of a cluster is extracted using a scoring metric based on sentence position, lexical similarity of a sentence to cluster centroid, and a feature based on quotation, among others. While the approach is quite different from ours (due to the underlying clustering algorithm and the absence of machine learning to select features), the use of email-specific features, in particular the feature related to quoted material, is similar.

Lam et al. (2002) present work on email summarization by exploiting the thread structure of email conversation and common features such as named entities and dates. They summarize the message only, though the content of the message to be summarized is "expanded" using the content from its ancestor messages. The expanded message is passed to a document summarizer which is used as a black box to generate summaries. Our work, in contrast, aims at summarizing the whole thread, and we are precisely interested in changing the summarization algorithm itself, not in using a black box summarizer.

In addition, there has been some work on summarizing meetings. As discussed in Section 1, email is different in important respects from (multi-party) dialog. However, some important aspects are related. Zechner (2002), for example, presents a meeting summarization system which uses the MMR algorithm to find sentences that are most similar to the segment and most dissimilar to each other. The similarity weights in the MMR algorithm are modified using three features, including whether a sentence belongs to a question-answer pair. The use of the question-answer pair detection is an interesting proposal that is also applicable to our work. However, overall most of the issues tackled by Zechner (2002) are not relevant to email summarization.

3 The Data

Our corpus consists of 96 threads of email sent during one academic year among the members of the board of the student organization of the ACM at Columbia University. The emails dealt mainly with planning events of various types, though other issues were also addressed. On average, each thread contained 3.25 email messages, with all threads containing at least two messages, and the longest thread containing 18 messages.

Two annotators each wrote a summary of the thread. We did not provide instructions about how to choose content for the summaries, but we did instruct the annotators on the format of the summary; specifically, we requested them to use the past tense, and to use speech-act verbs and embedded clauses (for example, *Dolores reported she'd gotten 7 people to sign up* instead of *Dolores got 7 people to sign up*). We requested the length to be about 5% to 20% of the original text length, but not longer than 100 lines.

Writing summaries is not a task that competent native speakers are necessarily good at without specific training. Furthermore, there may be many different possible summary types that address different needs, and different summaries may satisfy a particular need. Thus, when asking native speakers to write thread summaries we cannot expect to obtain summaries that are similar.

We then used the hand-written summaries to identify important sentences in the threads in the following manner. We used the sentence-similarity finder SimFinder (Hatzivassiloglou et al., 2001) in order to rate the similarity of each sentence in a thread to each sentence in the corresponding summary. SimFinder uses a combination of lexical and linguistic features to assign a similarity score to an input pair of texts. We excluded sentences that are being quoted, as well as signatures and the like. For each sentence in the thread, we retained the highest similarity score. We then chose a threshold; sentences with SimFinder scores above this threshold are then marked as "Y", indicating that they should be part of a summary, while the remaining sentences are marked "N". About 26% of sentences are marked "Y". All sentences from the email threads along with their classification constitutes our data. For annotator DB, we have 1338 sentences, of which 349 are marked "Y", for GR (who has annotated a subset of the threads that DB has annotated) there are 1296 sentences, of which 336 are marked "Y". Only 193 sentences are marked "Y" using the summaries of both annotators, reflecting the difference in the summaries written by the two annotators. The kappa for the marking of the sentences is 0.29 (recall that this only indirectly reflects annotator choice). Thus, our expectation that human-written summaries will show great variation was borne out; we discuss these differences further in Section 5.

4 Features for Sentence Extraction

We start out with features that are not specific to email. These features consider the thread as a single text. We call this feature set **basic**. Each sentence in the email thread is represented by a feature vector. We shall call the sentence in consideration *s*, the message in which the sentence appears *m*, the thread in which the sentence appears *t*, and the entire corpus *c*. (We omit some features we use for lack of space.)

- thread_line_num: The absolute position of *s* in *t*.
- centroid_sim: Cosine similarity of *s*'s TF-IDF vector (excluding stop words) with *t*'s centroid vector. The centroid vector is the average of the TF-IDF vectors of all the sentences in *t*. The IDF component is derived from the ACM Corpus.
- centroid_sim_local: Same as centroid_sim except that the inverse document frequencies are derived from the thread.
- length: The number of content terms in *s*.
- tfidfsum: Sum of the TF-IDF weights of content terms in *s*. IDF weights are derived from *c*.
- tfidfavg: Average TF-IDF weight of the content terms in *s*. IDF weights are derived from *c*.
- t_rel_pos: Relative position of *s* in *t*: the number of sentences preceding *s* divided by the total number of sentences in *t*. All messages in a thread are ordered linearly by the time they were sent.
- is_Question: Whether *s* is a question, as determined by punctuation.

Ann.	Feature set	**ctroid**	**basic**	**basic+**	**full**
DB	Recall	0.255	0.315	0.370	0.421
DB	Precision	0.298	0.553	0.584	0.607
DB	F-measure	0.272	0.401	0.453	0.497
GR	Recall	0.291	0.217	0.193	0.280
GR	Precision	0.333	0.378	0.385	0.475
GR	F-measure	0.311	0.276	0.257	0.352

Figure 1: Results for annotators DB and GR using different feature sets

We now add two features that take into account the division of the thread into messages and the resulting dialog structure. The union of this feature set with **basic** is called **basic+**.

- msg_num: The ordinality of *m* in *t* (i.e., the absolute position of m in t).
- m_rel_pos: Relative position of *s* in *m*: the number of sentences preceding *s* divided by the total number of sentences in *m*.

Finally, we add features which address the specific structure of email communication. The full feature set is called **full**.

- subject_sim: Overlap of the content words of the subject of the first message in *t* with the content words in *s*.
- num_of_res: Number of direct responses to *m*.
- num_Of_Recipients: Number of recipients of *m*.
- fol_Quote: Whether *s* follows a quoted portion in *m*.

5 Experiments and Results

This section describes experiments using the machine learning program Ripper (Cohen, 1996) to automatically induce sentence classifiers, using the features described in Section 4. Like many learning programs, Ripper takes as input the classes to be learned, a set of feature names and possible values, and training data specifying the class and feature values for each training example. In our case, the training examples are the sentences from the threads as described in Section 3. Ripper outputs a classification model for predicting the class (i.e., whether a sentence should be in a summary or not) of future examples; the model is expressed as an ordered set of if-then rules. We obtained the results presented here using five-fold cross-validation. In this paper, we only evaluate the results of the machine learning step; we acknowledge the need for an evaluation of the resulting summaries using word/string based similarity metric and/or human judgments and leave that to future publications.

	DB only	GR only	avg	max
Recall	0.421	0.280	0.212	0.268
Precision	0.607	0.475	0.406	0.444
F-measure	0.497	0.352	0.278	0.335

Figure 2: Results for combining two annotators (last two columns) using **full** feature set

We show results for the two annotators and different feature sets in Figure 1. First consider the results for annotator DB. Recall that **basic** includes only standard features that can be used for all text genres, and considers the thread a single text. **basic+** takes the breakdown of the thread into messages into account. **full** also uses features that are specific to email threads. We can see that by using more features than the baseline set **basic**, performance improves. Specifically, using email-specific features improves the performance over the **basic** baseline, as we expected. We also give a second baseline, **ctroid**, which we determined by choosing the top 20% of sentences most similar to the thread centroid. All results using Ripper improve on this baseline.

If we perform exactly the same experiments on the summaries written by annotator GR, we obtain the results shown in the bottom half of Figure 1. The results are much worse, and the centroid-based baseline outperforms all but the **full** feature set. We leave to further research an explanation of why this may be the case; we speculate that GR,as an annotator, is less consistent in her choice of material than is DB when forming a summary. Thus, the machine learner has less regularity to learn from. However, we take this difference as evidence for the claim that one should not expect great regularity in human-written summaries.

Finally, we investigated what happens when we combine the data from both sources, DB and GR. Using SimFinder, we obtained two scores for each sentence, one that shows the similarity to the most similar sentence in DB's summary, and one that shows the similarity to the most similar sentence in GR's summary. We can combine these two scores and then use the combined score in the same way that we used the score from a single annotator. We explore two ways of combining the scores: the average, and the maximum. Both ways of combining the scores result in worse scores than either annotator on his or her own; the average is worse than the maximum (see Figure 2). We interpret these results again as meaning that there is little convergence in the human-written summaries, and it may be advantageous to learn from one particular annotator. (Of course, another option might be to develop and enforce very precise guidelines for the an-

1	IF centroid_sim_local $\geq$ 0.32215 AND thread_line_num $\leq$ 4 AND isQuestion = 1 AND tfidfavg $\geq$ 0.212141 AND tfidfavg $\leq$ 0.301707 THEN Y.
2	IF centroid_sim $\geq$ 0.719594 AND numOfRecipients $\geq$ 8 THEN Y.
3	IF centroid_sim_local $\geq$ 0.308202 AND thread_line_num $\leq$ 4 AND tfidfmax $\leq$ 0.607829 AND m_rel_pos $\leq$ 0.363636 AND t_rel_pos $\geq$ 0.181818 THEN Y.
4	IF subject_sim $\geq$ 0.333333 tfidfsum $\leq$ 2.83636 tfidfsum $\geq$ 2.64262 tfidfmax $\leq$ 0.675917 THEN Y.
5	ELSE N.

Figure 3: Sample rule set generated from DB data (simplified for reasons of space)

Regarding ”acm home/bjarney”, on Apr 9, 2001, Muriel Danslop wrote: Two things: Can someone be responsible for the press releases for Stroustrup?
Responding to this on Apr 10, 2001, Theresa Feng wrote: I think Phil, who is probably a better writer than most of us, is writing up something for dang and Dave to send out to various ACM chapters. Phil, we can just use that as our ”press release”, right?
In another subthread, on Apr 12, 2001, Kevin Danquoit wrote: Are you sending out upcoming events for this week?

Figure 4: Sample summary obtained with the rule set in Figure 3

notators as to the contents of the summaries.)

A sample rule set obtained from DB data is shown in Figure 3. Some rules are intuitively appealing: for example, rule 1 states that questions at the beginning of a thread that are similar to entire thread should be retained, and rule 2 states that sentence which are very similar to the thread and which have a high number of recipients should be retained. However, some rules show signs of overfitting, for example rule 1 limits the average TF-IDF values to a rather narrow band. Hopefully, more data will alleviate the overfitting problem. (The data collection continues.)

6 Postprocessing Extracted Sentences

Extracted sentences are sent to a module that wraps these sentences with the names of the senders, the dates at which they were sent, and a speech act verb. The speech act verb is chosen as a function of the structure of the email thread in order to make this structure more apparent to the reader. Further, for readability, the sentences are sorted by the order in which they were sent. An example can be seen in Figure 4. Note that while the initial question is answered in the following sentence, two other questions are left unanswered in this summary (the answers are in fact in the thread).

7 Future Work

In future work, we will perform a qualitative error analysis and investigate in more detail what characteristics of DB’s summaries lead to better extractive summaries. We can use this insight to instruct human annotators, and to improve the automatic extraction. We intend to learn predictors for some other thread aspects such as thread category and question-answer pairs, and then use these as input to the sentence extraction procedure. For example, identifying question-answer pairs appears to be important for generating “complete” summaries, as illustrated by the sample summary. We also intend to perform an evaluation based on human feedback.

References

William Cohen. 1996. Learning trees and rules with set-valued features. In *Fourteenth Conference of the American Association of Artificial Intelligence*. AAAI.

Vasileios Hatzivassiloglou, Judith Klavans, Melissa Holcombe, Regina Barzilay, Min-Yen Kan, and Kathleen McKeown. 2001. SimFinder: A flexible clustering tool for summarization. In *Proceedings of the NAACL Workshop on Automatic Summarization*, Pittsburgh, PA.

Derek Lam, Steven L. Rohall, Chris Schmandt, and Mia K. Stern. 2002. Exploiting e-mail structure to improve summarization. In *ACM 2002 Conference on Computer Supported Cooperative Work (CSCW2002), Interactive Posters*, New Orleans, LA.

Smaranda Muresan, Evelyne Tzoukermann, and Judith Klavans. 2001. Combining Linguistic and Machine Learning Techniques for Email Summarization. In *Proceedings of the CoNLL 2001 Workshop at the ACL/EACL 2001 Conference*.

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

Paula Newman and John Blitzer. 2003. Summarizing archived discussions: a beginning. In *Proceedings of Intelligent User Interfaces*.

Klaus Zechner. 2002. Automatic summarization of open-domain multiparty dialogues in diverse genres. *Computational Linguistics*, 28(4):447–485.