ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/228708104

Detection of question-answer pairs in email conversations

Article · January 2004

DOI: 10.3115/1220355.1220483

CITATIONS
49

READS
27

2 authors, including:

Lokesh Shrestha
Apple Inc.
3 PUBLICATIONS **124** CITATIONS

SEE PROFILE

All in-text references underlined in blue are linked to publications on ResearchGate, letting you access and read them immediately.

Available from: Lokesh Shrestha
Retrieved on: 18 September 2016

Detection of Question-Answer Pairs in Email Conversations

Lokesh Shrestha and **Kathleen McKeown**
Columbia University
Computer Science Deparment
New York, NY 10027,
USA,
lokesh@cs.columbia.edu, kathy@cs.columbia.edu

Abstract

While sentence extraction as an approach to summarization has been shown to work in documents of certain genres, because of the conversational nature of email communication where utterances are made in relation to one made previously, sentence extraction may not capture the necessary segments of dialogue that would make a summary coherent. In this paper, we present our work on the detection of question-answer pairs in an email conversation for the task of email summarization. We show that various features based on the structure of email-threads can be used to improve upon lexical similarity of discourse segments for question-answer pairing.

1 Introduction

In this paper, we discuss work on the detection of question and answer pairs in email threads, i.e., coherent exchanges of email messages among several participants. Email is a written medium of asynchronous multi-party communication. This means that, 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 responders need to take special precautions to identify relevant elements of the discourse context (for example, by quoting previous messages). Thus, email is a distinct linguistic genre that poses its own challenges to summarization.

With the increasing popularity of email as a means of communication, an increasing number of meetings are scheduled, events planned, issues resolved, and questions answered through emails. As the number of emails in one's mailbox increases, information from past conversations becomes increasingly inaccessible and difficult to manage. For example, a number of emails can be used in scheduling a meeting, and a search for information on the meeting may retrieve all the intermediate emails, thus hindering one's access to the required information. Access to required information could be dramatically improved by querying summaries of email conversations

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 1: Sample summary obtained with sentence extraction

While summarization of email conversations seems a natural way to improve upon current methods of email management, research on email summarization is in early stages. Consider an example summary of a thread of email conversation produced by a sentence extraction based email thread summarization system developed at Columbia (Rambow et al., 2004) shown in Figure 1. While this summary does include an answer to the first question, it does not include answers to the two questions posed subsequently even though the

answers are present in the thread. This example demonstrates one of the inadequacies of sentence extraction based summarization modules: namely, the absence of discourse segments that would have made the summaries more readable and complete. A summarization module that includes answers to questions posed in extractive summaries, then, becomes very useful.

Further, questions are a natural means of resolving any issue. This is especially so of email conversations through which most of our issues, whether professional or personal, get resolved. And, the asynchronous nature of email conversation makes it possible for users to pursue several questions in parallel. In fact, in our corpus of email exchanges, we found that about 20% of all email threads focus primarily on a question-answer exchange, whether one question is posed and multiple people respond or whether multiple questions are posed and multiple responses given. For these type of email exchanges, a summary that can highlight the main question(s) asked and the response(s) given would be useful. Being able to distinguish questions pertaining to different issues in an email thread and being able to associate the answers with their questions is a necessity in order to generate this type of summary.

In this paper, we present our work on the detection of question and answer pairs in email conversations. The question-answer detection system we present will ultimately serve as one component of a full email summarization system, providing a portion of summary content. We developed one approach for the detection of questions in email messages, and a separate approach to detect the corresponding answers. These are described in turn below.

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 is 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 try to learn strategies to link question and answer segments for summarization.

(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. Because the summaries are extractive in nature, this approach still suffers from the possibility of incomplete summaries.

(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 and Lavie, 2001), for example, presents a spoken dialogue summarization system that takes into consideration local cross-speaker coherence by linking question answer pairs, and uses this information to generate extract based summaries with complete question-answer regions. While we have used a similar question detection approach, our approach to answer detection is different. We get back to this in Section 4.

(Rambow et al., 2004) show that sentence extraction techniques can work for summarizing email threads, but profit from email-specific features. In addition, they show that the presentation of the summary should take into account the dialogic structure of email communication. However, since their approach does not try to detect question and answer pairs, the extractive summaries suffer from the pos-

sibility of incomplete summaries.

3 Automatic Question Detection

While the detection of questions in email messages is not as difficult a problem as in speech conversations where features such as the question mark character are absent, relying on the use of question mark character for identifying questions in email messages is not adequate. The need for special attention in detecting questions in email messages arises due to three reasons. First, the use of informal language means users might use the question mark character in cases other than questions (for example, to denote uncertainty) and may overlook using a question mark after a question. Second, a question may be stated in a declarative form, as in, "I was wondering if you are free at 5pm today." Third, not every question, whether in an interrogative form or in a declarative form, is meant to be answered. For example, rhetorical questions are used for purposes other than to obtain the information the question asked, and are not required to be associated with answer segments.

We used supervised rule induction for the detection of interrogative questions. Training examples were extracted from the transcribed SWITCHBOARD corpus annotated with DAMSL tags.[1] This particular corpus was chosen not only because an adequate number of training examples could be extracted from the manual annotations, but also because of the use of informal language in speech that is also characteristic of email conversation. Utterances with DAMSL tags "sv" (speech act "statement-opinion") and "sd" (speech act "statement-non-opinion") were used to extract 5,000 negative examples. Similarly, utterances with tags "qy" ("yes-no-question"), "qw" ("Wh-question"), and "qh" ("rhetorical-question") were used to extract 5,000 positive examples. Each utterance was then represented by a feature vector which included the following features:

- POS tags for the first five terms (shorter utterances were padded with dummies)
- POS tags for the last five terms (shorter utterances were padded with dummies)
- length of the utterance

[1]From the Johns Hopkins University LVCSR Summer Workshop 1997, available from http://www.colorado.edu/ling/jurafsky/ws97/

Scheme	Ripper	Ripper+
Recall	0.56	0.72
Precision	0.96	0.96
F_1-score	0.70	0.82

Table 1: Test results for detection of questions in interrogative form

- POS-bigrams from a list of 100 most discriminating POS-bigrams list.

The list of most discriminating POS-bigrams was obtained from the training data by following the same procedure that (Zechner and Lavie, 2001) used.

We then used Ripper (Cohen, 1996) to learn rules for question detection. 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 speech acts extracted from the SWITCHBOARD corpus as described above. Ripper outputs a classification model for predicting the class (i.e., whether a speech act is a question or not) of future examples; the model is expressed as an ordered set of if-then rules. For testing, we manually extracted 300 questions in interrogative form and 300 statements in declarative form from the ACM corpus.[2] We show our test results with recall, precision and F_1-score[3] in Table 1 on the first column.

While the test results show that the precision was very good, the recall score could be further improved. Upon further investigation on why the recall was so low, we found that unlike the positive examples we used in our training data, most of the questions in the test data that were missed by the rules learned by Ripper started with a declarative phrase. For example, both "I know its on 108th, but after that not sure, how exactly do we get there?", and "By the way, are we shutting down clic?" begin with declarative phrases and were missed by the Ripper learned rules. Following this observation,

[2]More information on the ACM corpus will be provided in Section 4.1. At the time of the development of the question detection module the annotations were not available to us, so we had to manually extract the required test speech acts.

[3]F_1-score = $\frac{2PR}{P+R}$, where P=Precision and R=Recall

we manually updated our question detection module to break a speech act that was not initially predicted as question into phrases separated by comma characters. Then we applied the rules on the first phrase of the speech act and if that failed on the last phrase. For example, the rules would fail on the phrase "I know its on 108th", but would be able to classify the phrase "how exactly do we get there" as a question. In doing this we were able to increase the recall score to 0.72, leading to a F_1-score of 0.82 as shown in Table 1 in the second column.

4 Automatic Answer Detection

While the automatic detection of questions in email messages is relatively easier than the detection of the same in speech conversations, the asynchronous nature of email conversations makes detection and pairing of question and answer pairs a more difficult task. Whereas in speech a set of heuristics can be used to identify answers as shown by (Zechner and Lavie, 2001), such heuristics cannot be readily applied to the task of question and answer pairing in email conversations. First, more than one topic can be discussed in parallel in an email thread, which implies that questions relating to more than a single topic can be pursued in parallel. Second, even when an email thread starts with the discussion of a single topic, the thread may eventually be used to initiate a different topic just because the previous topic's list of recipients closely matched those required for the newly initiated topic. Third, because of the use of "Reply" and "ReplyAll" functions in email clients, a user may be responding to an issue posed earlier in the thread while using one of the email messages subsequent to the message posing the issue to "reply back" to that issue. So, while it may seem from the structure of the email thread that a person is replying back to a certain email, the person may actually be replying back to an email earlier in the thread. This implies that when several persons answer a question, there may be answers which appear several emails after the email posing the question. Finally, the fact that the question and its corresponding answers may have few words in common further complicates answer detection. This is possible when a person uses the context of the email conversation to ask questions and make answers, and the semantics of such questions and answers have to be interpreted with respect to the context they appear in. Such context is readily available for a reader through the use of quoted material from past email messages. All of these make the task of detecting and linking question and answer pairs in email conversations a complicated task. However, this task is not as complicated a task as automatic question answering where the search space for candidate answers is much wider and more sophisticated measures than those based on lexical similarity have to be employed.

Our approach to automatic answer detection in email conversations is based on the observation that while a number of issues may be pursued in parallel, users tend to use separate paragraphs to address separate issues in the same email message. While a more complicated approach to segmentation of email messages could be possible, we have used this basic observation to delineate discourse segments in email messages. Further, because discourse segments contain more lexical context than their individual sentences, our approach detects associations between pairs of discourse segments rather than pairs of sentences.

We now present our machine learning approach to automatic detection of question and answer pairs.

4.1 Corpus

Our corpus consists of approximately 300 threads of about 1000 individual email messages 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. Threads were constructed from the individual email messages using the "In-Reply-To" header information to link parent and child email messages.

Two annotators (DB and GR) each were asked to highlight and link question and answer pairs in the corpus. Our work presented here is based on the work these annotators had completed at the time of this writing. GR has completed work on 200 threads of which there are 81 QA threads (threads with question and answer pairs), 98 question segments, and 142 question and answer pairs. DB has completed work on 138 threads of which there are 62 QA threads, 72 question segments, and 92 question and answer pairs. We consider a segment to

be a question segment if a sentence in that segment has been highlighted as a question. Similarly, we consider a segment to be an answer segment if a sentence in that segment has been paired with a question to form a question and answer pair. The kappa statistic (Carletta, 1996) for identifying question segments is 0.68, and for linking question and answer segments given a question segment is 0.81.

4.2 Features

For each question segment in an email message, we make a list of candidate answer segments. This is basically just a list of original (content that is not quoted from past emails)[4] segments in all the messages in the thread subsequent to the message of the question segment. Let the thread in consideration be called *t*, the container message of the question segment be called *mq*, the container message of the candidate answer segment be called *ma*, the question segment be called *q*, and the candidate answer segment be called *a*. For each question and candidate answer pair, we compute the following sets of features:

4.2.1 Some Standard Features

(a) number of non stop words in segment *q* and segment *a*;
(b) cosine similarity[5] and euclidean distance[6] between segment *q* and *a*;

4.2.2 Features derived from the structure of the thread *t*

(c) the number of intermediate messages between *mq* and *ma* in *t*;
(d) the ratio of the number of messages in *t* sent earlier than *mq* and all the messages in *t*, and similarly for *ma*;
(e) whether *a* is the first segment in the list of candidate answer segments of *q* (this is true if a segment is the first segment in the first message sent in reply to *mq*);

4.2.3 Features based on the other candidate answer segments of *q*

(f) number of candidate answer segments of *q* and the number of candidate answer segments of *q* after *a* (a segment *x* is considered to be after another segment *y* if *x* is from a message sent later than that of *y*, or if *x* appears after *y* in the same message);
(g) the ratio of the number of candidate answer segments before *a* and the number of all candidate answer segments (a segment *x* is considered to be before another segment *y* if *x* is from a message sent earlier than that of *y*, or if *x* appears before *y* in the same message); and
(h) whether *q* is the most similar segment of *a* among all segments from ancestor messages of *ma* based on cosine similarity (the list of ancestor messages of a message is computed by recursively following the "In-Reply-To" header information that points to the parent message of a message).

While the contribution of a single feature to the classification task may not be intuitively apparent, we hope that a combination of a subset of these features, in some way, would help us in detecting question-answer pairs. For example, when the number of candidate answer segments for a question segment is less than or equal to two, feature (e) may be the best contributor to the classification task. But, when the number of candidate answer segments is high, a collection of some features may be the best contributor.

We categorized each feature vector for the pair *q* and *a* as a positive example if *a* has been marked as an answer and linked with *q*.

4.3 Training Data

We computed four sets of training data. Two for each of the annotators separately, which we call **DB** and **GR** according to the label of their respective annotator. One taking the union of the annotators, which we call **Union**, and another taking the intersection, which we call **Inter**. For the first two sets, we collected the threads that had at least one question and answer pair marked by the respective annotator. For each question that has an answer marked

[4] While people do use quoted material to respond to specific segments of past emails, a phenomenon more common is discussion lists, because the occurrence of such is rare in the ACM corpus we decided not to use them as a feature.

[5]
$$cosine\_sim(x,y) = \frac{\sum_{i=1}^{N}(c_{x^i} * c_{y^i})}{\sqrt{\sum_{j=1}^{N} c_{x^j}^2 * \sum_{j=1}^{N} c_{x^j}^2}}$$
where c_{x^i} is the count of word i in segment x, and c_{y^i} is the count of word i in segment y.

[6]
$$euclidean\_dis(x,y) = \sqrt{\sum_{i=1}^{N}(c_{x^i}^2 - c_{y^i}^2)}$$
where c_{x^i} is the count of word i in segment x, and c_{y^i} is the count of word i in segment y.

Data Set	**DB**	**GR**	**Union**	**Inter**
datapoints	259	355	430	181
positives	89	139	168	59
questions	72	98	118	52
threads	62	81	97	46

Table 2: Summary of training data: number of instances

Data Set	Precision	Recall	F_1-score
DB	0.6	0.27	0.372
GR	0.629	0.439	0.517
Union	0.61	0.429	0.503
Inter	0.571	0.407	0.475

Table 3: Baseline results

(some of the highlighted questions do not have corresponding answers in the thread), we computed a list of feature vectors as described above with all of its candidate answer segments. For the union set, we collected all the threads that had question and answer pairs marked by either annotator, and computed the feature vectors for each such question segment. A feature vector was categorized positive if any of the two annotators have marked the respective candidate answer segment as an answer. For the intersection set, we collected all the threads that had question and answer pairs marked by both annotators. Here, a feature vector was labelled positive only if both the annotators marked the respective candidate answer segment as an answer. Table 2 summarizes the information on the four sets of training data.

4.4 Experiments and Results

This section describes experiments using Ripper to automatically induce question and candidate answer pair classifiers, using the features described in Secion 4.2. We obtained the results presented here using five-fold cross-validation.

Table 3 shows the precision, recall and F_1-score for the four datasets using the cosine similarity feature only. We use these results as the baseline against which we compare the results for the full set of features shown in Table 4. While precision using the full feature set is comparable to that of the baseline measure, we get a significant improvement on recall with the full feature set. The baseline measure predicts that the candidate answer segment whose similarity with the question segment is above a certain threshold will be an actual answer segment. Our results suggest that lexical similarity cannot alone capture the rules associated with question and answer pairing, and that the use of various features based on the structure of the thread of email conversations can be used to improve upon lexical similarity of discourse segments. Further, while the results do not seem to suggest a clear preference for the data set **DB** over the data set **GR** (this could be explained by their high kappa score of 0.81), taking the union of the two datasets does seem to be better than taking the intersection of the two datasets. This could be because the intersection greatly reduces the number of positive data points from what is available in the union, and hence makes the learning of rules more difficult with **Inter**.

Data Set	Precision	Recall	F_1-score
DB	0.69	0.652	0.671
GR	0.68	0.612	0.644
Union	0.698	0.619	0.656
Inter	0.6	0.508	0.55

Table 4: Summary of results

Finally, on observing that some questions had at most 2 candidate answers, and others had quite a few, we investigated what happens when we divide the data set **Union** into two data sets, one for question segments with 2 or less candidate answer segments which we call the data set **Union_a**, and the other with the rest of the data set which we call the data set **Union_b**. **Union_a** has, on average, 1.5 candidate answer segments, while **Union_b** has 5.7. We show the results of this experiment with the full feature set in Table 5. Our results show that it is much easier to learn rules for questions with the data set **Union_a**, which we show in the first row, than otherwise. We compare our results for the baseline measure of predicting the majority class, which we show in the second row, to demonstrate that the results obtained with the dataset **Union_a** were not due to majority class prediction. While the results for the other subset, **Union_b**, which we show in the third row, compare well with the results for **Union**, when the results for the data sets **Union_a** and **Union_b** are combined, which we show in the fourth row, we achieve better results than without the splitting,

Data Set	Precision	Recall	F_1-score	positives	datapoints
Union_a	0.879	0.921	0.899	63	79
Baseline for Union_a	0.797	1.0	0.887	63	79
Union_b	0.631	0.619	0.625	105	351
Combined	0.728	0.732	0.730	168	430
Union	0.698	0.619	0.656	168	430

Table 5: Summary of results with the split data set compared with a baseline and the unsplit data set

shown in the last row.

5 Conclusion and Future Work

We have presented an approach to detect question-answer pairs with good results. Our approach is the first step towards a system which can highlight question-answer pairs in a generated summary. Our approach works well with interrogative questions, but we have not addressed the automatic detection of questions in the declarative form and rhetorical questions. People often pose their requests in a declarative form in order to be polite among other reasons. Such requests could be detected with their use of certain key phrases some of which include "Please let me know...", "I was wondering if...", and "If you could....that would be great.". And, because rhetorical questions are used for purposes other than to obtain the information the question asked, such questions do not require to be paired with answers. The automatic detection of these question types are still under investigation.

Further, while the approach to the detection of question-answer pairs in threads of email conversation we have presented here is quite effective, as shown by our results, the use of such pairs of discourse segments for use in summarization of email conversations is an area of open research to us. As we discussed in Section 1, generation of summaries for email conversations that are devoted to question-answer exchanges and that integrate identified question-answer pairs as part of a full summary is also needed.

6 Acknowledgements

We are grateful to Owen Rambow for his helpful advice. We also thank Andrew Rosenberg for his discussion on kappa statistic as it relates to the ACM corpus. This work was supported by the National Science Foundation under the KDD program. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Jean Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254.

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

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*.

Owen Rambow, Lokesh Shrestha, John Chen, and Christy Lauridsen. 2004. Summarizaing email threads. In *Proceedings of HLT-NAACL 2004: Short Papers*.

Klaus Zechner and Alon 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.