Detection of Imperative and Declarative Question-Answer Pairs in Email Conversations

Helen Kwong
Stanford University, USA
hhkwong@stanford.edu

Neil Yorke-Smith
SRI International, USA
nysmith@ai.sri.com

Abstract

Question-answer pairs extracted from email threads can help construct summaries of the thread, as well as inform semantic-based assistance with email. Previous work dedicated to email threads extracts only questions in interrogative form. We extend the scope of question and answer detection and pairing to encompass also questions in imperative and declarative forms, and to operate at sentence-level fidelity. Building on prior work, our methods are based on learned models over a set of features that include the content, context, and structure of email threads. For two large email corpora, we show that our methods balance precision and recall in extracting question-answer pairs, while maintaining a modest computation time.

1 Introduction

Studies of email overload show that the "tyranny of email" negatively impacts work performance and tangibly weighs upon organizations and even national economies [Dabbish and Kraut, 2006]. Various software tools have been developed to ameliorate email overload. At one extreme, some refactor the role and workflow of email (e.g., interweaving it with explicit task management [Belotti *et al.*, 2005], semantically understanding actions pertaining to a message and assisting the user to perform them [Freed *et al.*, 2008]). Others seek to improve email client software with nuggets of AI (e.g., auto-filing messages into folders, detecting missing attachments, summarizing email threads) [Dredze *et al.*, 2008].

One reason for the ubiquity of email is its nature as asynchronous, unstructured, written, multi-party communication. Research has sought to automatically derive summaries of email threads and embed them in email clients [Ulrich *et al.*, 2008]. Summaries can be *extractive*, i.e., selected fragments of the thread, or *generative*, ranging from as sophisticated as a high-level synopsis to as simple as a table of respondent frequencies. One potent form of generative summarization is identification of the questions (explicit or implicit) present in the thread and the answers to them, if any.

We consider the related problems of identifying questions in an email thread (possibly between multiple parties), and linking them to possible answers later in the thread. Our work is part of an effort to endow an email client with a range of usable smart features by means of AI technology.

Prior work that studies the problems of question identification and question-answer pairing in email threads has focused on interrogative questions, such as "Which movies do you like?" However, questions are often expressed in non-interrogative forms: declarative questions such as "I was wondering if you are free today", and imperative questions such as "Tell me the cost of the tickets". Our sampling from the well-known Enron email corpus finds about one in five questions are non-interrogative in form. Further, previous work for email threads has operated at the paragraph level, thus potentially missing more fine-grained conversation.

We propose an approach to extract question-answer pairs that includes imperative and declarative questions. Our main contributions are: (1) analysis of prior question-answer (Q-A) pairing algorithms on more diverse email corpora (e.g., Enron), (2) an improved non-learning method for question detection, (3) extension of both learning and non-learning methods for answer detection to include non-interrogative questions and to operate at sentence-level fidelity, and (4) analysis of performance with more stringent metrics.

2 Related Work

Shrestha and McKeown [2004] describe an approach for detecting question-answer pairs in email threads for the purpose of summarization. For question detection, Shrestha and McKeown (henceforth S&M) learn a classification model based on parts-of-speech (POS) features, training it on a transcribed telephone speech corpus. The focus is on detecting questions in interrogative form only.

For question-answer linkage, S&M work at the paragraph level. They learn a classification model based on features over original (i.e., non-quoted) paragraphs. The features include standard informational retrieval measures, such as stop words, cosine similarity, and euclidean distance, features from thread structure, such as the number of interleaved messages, and features based on the set of candidate answer paragraphs, such as the number of candidates. The classifier is trained and tested on an ACM student corpus annotated by two human annotators.

Cong *et al.* [2008] examine question and answer detection and pairing for online forums. These forums bear many similarities to email threads, as well as several key distinguishing

characteristics. For instance, over 90% of forum threads contain Q-A knowledge (more than for email), the number of participants is often higher, multiple questions and answers are often highly interleaved, quoting is used in different ways, and message reply relationships are usually unavailable.

The approach of Cong *et al.* (henceforth CWLSS) for question detection is based on characterizing questions and non-questions by extracting labelled sequence patterns (LSPs)—as opposed to POS analysis only—and using the discovered patterns as features to learn a classification model for question detection. For online forums, they find the LSP-based classifier outperforms the S&M POS-based classifier. CWLSS develop a graph-based propagation method for question-answer linkage, leveraging the linkage structure within forum threads, and combine it with syntactic and classification-based methods for answer identification.

Besides email threads and online forums, there also has been work on extracting question-answer pairs in multi-party dialogue [Kathol and Tur, 2008]. While extraction of questions in meeting transcripts could be similar to that in email text, the structure of email threads is very different from that of meeting speech. For example, while question detection is relatively more simple in email than in speech, the asynchronous nature of email and its rich context (e.g., quoting) makes question-answer pairing relatively more difficult.

Harder still than automated answer detection is automated question answering. While question-answering systems serve purposes very different from ours, ideas from the literature on question answering can be of benefit.

3 Algorithms

The starting point for our work is an email thread. Given a set of messages, thread reassembly is the task of extracting the individual threads [Yeh and Harnly, 2006]; we assume this procedure has been performed.

3.1 Question Detection

Question detection serves the greater purpose for us of question-answer linkage. We consider three algorithms for question detection that work at the sentence level, i.e., the algorithms look at each sentence in an email thread and classify each as either a question or not. To serve question-answer linkage, we seek an algorithm that exhibits a sufficiently high F_1 score (the geometric mean of precision and recall) on real data, coupled with a low running cost.

Naïve A baseline question detection algorithm had been implemented in our email assistant. Algorithm Naïve employs regular expressions to classify sentences that end with a question mark as questions, except for sentences that fit the pattern of a URL. A common extension is to detect 5W-1H question words (Who, What, Where, When, Why, or How).

S&M The state-of-the-art in question detection in email threads is the work of Shrestha and McKeown [2004]. Algorithm S&M uses Ripper [Cohen, 1996] to learn a model that classifies each sentence as a question or a non-question, based on parts-of-speech features. The scope of the S&M algorithm is detection of questions in interrogative form only.

There are two broad possibilities to create a more general question detector: a method based on learning, as S&M, or not, as Naïve. We chose to examine whether the performance a simpler, non-learning, method would suffice.

Regex Like the Naïve question detector, algorithm Regex is also based entirely on regular expressions. A sentence is detected as a question if it fulfills any of the following:

- It ends with a question mark, and is not a URL.
- It contains a phrase that begins with words that fit an interrogative question pattern. This is a generalization of 5W-1H question words. For example, the second phrase of "When you are free, can you give me a call" is a strong indicator that the sentence is a question.[1] This condition is designed to catch sentences that should end with a question mark but were not typed with one.
- It fits the pattern of common questions that are not in the interrogative form. For instance, "Let me know when you will be free" is one such question.

Our hand-crafted database contains approximately 50 patterns; it was assembled rapidly. The alternative approach is to learn patterns; CWLSS take this approach, and present a generalized learned classification model to acquire patterns.

3.2 Answer Detection and Q-A Pairing

Traditional document retrieval methods can be applied to email threads, by treating each message or each candidate answer as a separate document. However, as has been observed in the direct application of information retrieval literature for extractive email summarization, these methods, such as cosine similarity, query likelihood language models, and KL-divergence language models, do not on their own exploit the content, context, and structural features of email threads.

We again consider three algorithms: one initially implemented in our email assistant, one identified as state-of-the-art in the literature (S&M), and a new heuristic-based algorithm of our construction that builds upon the literature.

Naïve Q-A We again regard the Naïve question-answer pairing algorithm as a baseline method. It is based on detecting answer types, and subsequent simple matching of question and answer sentences by type. Given an email thread, the algorithm iterates through each original (i.e., unquoted) sentence[2] and determines whether it is a question using the Naïve question detector described above. For each detected question, it guesses the expected answer type based on regular expression patterns. For example, Naïve categorizes a question that begins with "Are you" as a yes/no question. For each question, the algorithm looks at each original sentence in every subsequent email in the thread for a sentence that fits the expected answer type. For example, the sentence "I think so" fits a yes/no question. The first sentence that fits the expected answer type of a detected question is naïvely paired as the answer to that question (i.e., a sentence is paired as the answer to at most one question). The underlying heuristic is that

[1] There is an improvement to S&M, discussed below, that breaks a sentence into comma-delimited phrases, in an effort to catch such cases.

[2] While quoted material is used in email to respond to segments of earlier messages, the usage of this practice varies; it is much more common in online forums.

earlier questions are answered earlier than later questions; the asynchronous nature of conversations in email threads means the assumption does not hold in general.

The classification of question types is: yes/no, essay (why and how questions), what, when, where, who, number, and choice (e.g., "Is it a house or a flat?"). However, when the Naïve algorithm looks for an answer to a question, the only types of answers captured are yes/no, essay, and what.

S&M Q-A For answer detection and question-answer pairing, S&M again use Ripper to learn a classification model. They work at the paragraph level. For training purposes, a paragraph is considered a question segment if it contains a sentence marked by the human annotator as a question, and a paragraph is considered an answer segment to a question paragraph if it contains at least one sentence marked by the human annotator as an answer to the question. The candidate answer paragraphs of a question paragraph are all the original paragraphs in the thread subsequent to the message containing the question paragraph. Candidate answer paragraphs that do not contain sentences marked as an answer by the annotator are used as negative examples.

As noted earlier, the features that the S&M algorithm uses include standard textual analysis features from information retrieval, features derived from the thread structure, and features based on comparison with other candidate paragraphs.

Heuristic Q-A We present a portfolio of heuristic algorithms that operate at *the sentence level*. The algorithms use a common set of features that extends those considered by S&M. The first algorithm variant uses hand-picked parameter values, the second (like S&M) learns a classification model (also using Ripper), while the third algorithm learns a linear regression model. We examine for questions each content sentence, i.e., each original sentence that is not classified as a greeting or signature line. (Culling greetings and signatures improves performance by as much as 25%.) Any question detector could be used; we use the Regex algorithm described earlier. For each question, we obtain a set of candidate answers according to the following heuristic. A candidate answer is a content sentence in a subsequent message in the thread that is: (1) not from the sender of the question email, and (2) an individual's first reply to the question email, (3) not one of the detected question sentences.

Our heuristic algorithms score each of the candidate answers based on a weighted set of features. In variants that employ learning, we use the same set of features (described below) to train answer detectors, and to train Q-A pairing classifiers that assign each candidate answer a probability that it is the answer to a given question. We attribute the highest-scoring or most probable candidate (appropriately) as the answer to a given question, assuming that the score or probability is above a minimum threshold (default 0.5). Finally, we limit the number of questions to which an answer can be assigned; for the experiments reported, we use a limit of two.

The set of features we use for answer detection and question-answer pairing is a combination of textual features, structural features, entity tags, and expected answer types. Let Q be a question in message m_Q and A be a candidate answer found in message m_A. The features are as follows.

1. Number of non stop words in Q and A (S&M feature a)
2. Cosine similarity between Q and A (part of S&M feat. b)
3. Cosine similarity between Q and A, after named entity tagging, stemming, and removal of stop words
4. Number of intermediate messages between m_Q and m_A (S&M feature c)
5. Ratio of the number of messages in the thread prior to m_Q to the total number of messages, and similarly for m_A (S&M feature d)
6. Number of candidate answers that come before A (similar to part of S&M feature f)
7. Ratio of the number of candidate answers that come before A to the total number of candidate answers (S&M feature g)
8. Whether m_A is the first reply to m_Q
9. Whether Q is a question addressed to the sender of m_A: for example, "Bill, can you clarify?" is addressed to a user whose name or address includes "bill"
10. Semantic similarity between the sentences Q and A
11. Whether A matches the expected answer type of Q

We thus include most of the features found to be useful in S&M. We omit S&M feature e, since it is superseded by our feature 8, and S&M feature h, since it relies on `In-Reply-To` header information which is surprisingly often not available [Yeh and Harnly, 2006] (for instance, it is not available in the Enron corpus), and moreover since the intent of this feature is superseded by taking the best-scoring match in Heuristic.

Computation of the features is mostly straightforward. We find that cosine similarity suffices and do not use also euclidean distance (unlike S&M feature b). We compute semantic similarity between question and candidate answers based on WordNet relations [Pirrò and Seco, 2008].

The most difficult feature to capture is the last: expected answer type. The algorithm Naïve detects answer type by the crude means of regular expressions. This suffices for yes/no questions. We built a named entity recognizer that identifies and classifies proper names, places, temporal expressions, currency, and numbers, among other entities. The recognizer tags phrases in message bodies. Thus, for a *when* question, a match occurs if any phrases in the candidate answer are tagged as times or dates. We do not attempt to detect by type essay, what, and choice answers.

4 Empirical Analysis

We undertook experiments to assess the behaviour of the algorithms described in the previous section. The algorithms were implemented in Java 1.6, and the experiments were performed on an Intel Core 2 Duo 2.20GHz machine with 2GB memory, running Windows Vista.

4.1 Methodology and Datasets

What makes a segment of text a question or not, and what constitutes an answer to a question, are both subjective judgments. We follow prior works in information retrieval to train (where relevant) and test algorithms in our experiments on human-annotated datasets.

Algorithm	Precision	Recall	F_1-score	Time (ms)
Naïve	**0.956**	0.918	0.936	**0.0254**
S&M	0.623	0.865	0.724	48.30
Regex	0.954	**0.964**	**0.959**	0.243

Table 1: Question detection, interrogative questions only

Algorithm	Precision	Recall	F_1-score	Time (ms)
Naïve	0.958	0.786	0.863	**0.0286**
S&M	0.662	0.823	0.734	45.90
Regex	**0.959**	**0.860**	**0.907**	0.227

Table 2: Question detection, including non-interrogative questions

In contrast to email summarization, where standardized datasets now exist [Ulrich *et al.*, 2008], there are unfortunately no annotated email corpora available for question-answer pairing. The ACM student corpus used by S&M and the annotations upon it are not available for reasons of privacy. This state of affairs differs to online forums, also, in many of which the community rate the posts to a thread, thus providing ready and large datasets [Cong *et al.*, 2008]. However, while not annotated with questions—or at least not with sufficient fidelity on question types—nor with question-answer pairings, extensive email corpora are available. We used the widely-studied Enron corpus [Klimt and Yang, 2004] and the Cspace corpus [Minkov *et al.*, 2005].

4.2 Question Detection

Data and Metrics To evaluate the question detection algorithms Naïve, S&M, and Regex (Section 3.1), we created a test dataset of sentences as follows. We randomly selected 10,000 sentences from emails from the Enron corpus and the Cspace corpus. A human annotator looked at each sentence and marked all questions until 350 questions were marked. For each question the annotator also marked its sentence form as interrogative, declarative, imperative, or other. Out of the 350 questions, about 80% were marked as interrogative, 11% as imperative, 5% as declarative, and 4% as other. In addition to these 350 question sentences, we randomly selected 350 sentences out of those that had been passed over as non-questions to obtain a collection of 700 sentences in total.[3]

We measured the precision, recall, and F_1-score for each of the three algorithms, on a restricted set with only the interrogative questions and the 350 non-questions, and on the complete set of 700 sentences. The Naïve and Regex algorithms do not require training, while the S&M algorithm had been previously trained on the transcribed telephone speech corpus [Shrestha and McKeown, 2004].[4] Thus all 700 sentences were used for testing purposes.

[3] We chose a 50-50 distribution in order to perform a similar evaluation as S&M; a more natural distribution would have many more non-questions. Increasing the number of non-questions can be expected to leave recall unchanged, since it depends more on non-interrogative *questions* than non-questions. By contrast, precision can be expected to fall across the algorithms, supposing each question-detection algorithm mis-classifies a fixed fraction of non-questions as false positives.

[4] We maintain the same training as the original so to directly compare results. If trained on email data, the precision of S&M can be expected to improve. However, its training requires POS-tagged data; we were not aware of POS-tagged email corpora. Further, available trained POS taggers are mostly trained on traditional text, whereas email data has been described as more similar to speech.

Algorithm	Precision	Recall	F_1-score
5W-1H	0.690	0.151	0.248
Naïve	**0.978**	0.780	0.868
S&M	0.873	0.873	0.871
CWLSS	0.971	**0.978**	**0.975**

Table 3: Question detection on online forums, including non-interrogative questions [Cong et al., 2008]

Results The results are shown in Tables 1 and 2. We can see that S&M performs relatively less well than the other two algorithms on both datasets. The fact that its precision is so low is at first surprising, because it is reported that the S&M algorithm achieves high precision [Shrestha and McKeown, 2004; Cong *et al.*, 2008]. The difference may be understood in that S&M tested only on questions in interrogative form and statements in declarative form, whereas we tested on questions in any form and non-questions in any form. Examples of non-questions that are not in declarative form, that S&M incorrectly detected as questions, are "Brian, just get here as soon as you can" and "Let's gear up for the game".[5]

The results for the two regular expression algorithms are more expected. Both perform very well on interrogative questions only, emphasizing that question detection is not so challenging a task. Both have lower recall scores on the overall set, since non-interrogative questions are harder to detect. Since S&M does not consider declarative or imperative questions, its performance is essentially the same on both datasets. As expected, Regex achieves higher recall than Naïve because of its greater sophistication. Although the runtime increases by one order of magnitude, the absolute runtime remains modest. The tables report the mean runtimes of the algorithms in milliseconds per sentence. The median time per sentence for both Regex and Naïve is essentially zero; for S&M it is 18.5ms. POS tagging is the main reason for the considerable extra time taken by S&M. The variance of Regex is greater than Naïve: 8.61 and 0.185ms respectively (full dataset, including non-interrogative questions). S&M exhibits considerable variance of 1750ms.

Table 3 reports results from Cong *et al.* [2008]. Although these results are on a different dataset and for online forums rather than email threads, we give them for comparison. It can be seen that Naïve based on 5W-1H words performs very poorly, while Naïve based on question marks (as our Naïve) has similar performance as to our experiments. The notable difference is S&M, which exhibits the higher precision also reported by S&M. However, Naïve continues to perform as well as S&M in these reported experiments.

The LSP-based classifier learned by CWLSS—algorithm CWLSS—detects interrogative and non-interrogative questions. It is found to outperform Naïve and S&M, and, albeit on different datasets, has higher scores than Regex. However, and again comparing unfairly across datasets, Regex still has scores better than S&M, particularly precision. The use of CWLSS for email threads is to be explored.

[5] The precision reported in CWLSS may be understood because they trained their S&M question detector on the same kind of online forum data as they were testing, instead of phone speech data as S&M and ourselves. As noted above, training S&M on email corpora was infeasible without POS tagging.

Algorithm	Abbreviation
S&M original	SMQA
S&M with Regex	SMQA-regex
S&M with Regex and highest prob.	SMQA-highest
Naïve type matching	Naïve
Heuristic hand-tuned, random selection	Random
Heuristic hand-tuned	Heuristic
Heuristic by classifier	Heuristic-C
Heuristic by linear regression	Heuristic-LR

Table 4: Question-answer pairing methods

	LCS			SM		
Algorithm	Precision	Recall	F_1-score	Precision	Recall	F_1-score
SMQA	0.0138	0.0829	0.0232	0.0761	0.4782	0.1297
SMQA-regex	0.1157	0.1144	0.1145	0.4358	0.4922	0.4609
SMQA-highest	0.2861	0.2817	0.2835	0.4981	0.5142	0.5048
Naïve	0.1690	0.0452	0.0708	0.5045	0.1276	0.1998
Random	0.2797	0.2695	0.2735	0.4738	0.4896	0.4799
Heuristic	**0.4618**	**0.4408**	**0.4538**	**0.5818**	0.5710	**0.5749**
Heuristic-C	0.4545	0.4373	0.4439	0.5747	**0.5750**	0.5715
Heuristic-LR	0.4534	0.4280	0.4379	0.5711	0.5642	0.5647

Table 5: Question-answer pairing, Annotator 1

Altogether, we find that the F_1-score of Regex of above 0.9 over all question types to be sufficient to move to consider the more challenging task of question-answer pairing.

4.3 Answer Detection and Q-A Pairing

We now turn to our primary focus, answer detection and question-answer pairing for both interrogative and non-interrogative questions in email threads.

Data and Metrics We again randomly drew email threads from the Enron and Cspace corpora. Four human annotators marked the questions and corresponding answers (if any) in a set of email threads. Two of the annotators (whom we refer to as Annotators 1 and 2) each annotated about 90 threads containing question and answer pairs; two annotated significantly fewer. There were 23 Q-A threads that every annotator annotated (the intersection set). The Fleiss kappa statistic for identifying question paragraphs was 0.69, and the kappa statistic for linking answer paragraphs with question paragraphs was 0.77. These numbers are close to the numbers reported in prior studies and indicate decent inter-annotator agreement.[6]

We used two metrics to evaluate the quality of the question-answering pairing. The first metric we employed is the paragraph-based metric used by Shrestha and McKeown [2004] (which we will call SM). This metric is fitting only if we segment messages at the paragraph level. Thus we developed a more stringent metric (which we will call LCS) based on the longest common substring of words. Two text segments are considered the same under LCS if the length of their longest common substring is greater than half the length of the longer segment. We consider two Q-A pairs to be the same if their question segments are the same by LCS and their answer segments are also the same by LCS. We measure precision, recall, and F_1 score by both SM and LCS metrics.

[6]Between Annotators 1 and 2, kappa for question detection and Q-A pairing were 0.74 and 0.83 respectively; there were 61 Q-A threads they both annotated.

	LCS			SM		
Algorithm	Precision	Recall	F_1-score	Precision	Recall	F_1-score
SMQA	0.0506	**0.2568**	0.0842	0.0934	**0.4213**	0.1519
SMQA-regex	0.2101	0.2245	0.2166	0.3832	0.3693	0.3756
SMQA-highest	0.1979	0.1665	0.1805	0.4429	0.3068	0.3582
Naïve	**0.2615**	0.0473	0.0779	0.4544	0.0909	0.1482
Random	0.1117	0.1013	0.1059	0.4187	0.3120	0.3521
Heuristic	0.2444	0.2012	**0.2195**	**0.5376**	0.3655	**0.4259**
Heuristic-C	0.2058	0.1717	0.1854	0.5238	0.3621	0.4202
Heuristic-LR	0.2396	0.1814	0.2039	0.5335	0.3459	0.4113

Table 6: Question-answer pairing, Annotator 2

Methods Table 4 summarizes the nine algorithms we studied. The first three algorithms evaluated are variants on S&M: SMQA represents the original, unimproved S&M algorithm described in Section 3.2 (i.e., S&M question detection, and S&M's learned classifier), SMQA-regex replaces the S&M question detector with our better-performing Regex question detector, and SMQA-highest makes the further refinement of taking the most probable answer paragraph instead of any candidate answer paragraph that has over 0.5 probability.

Algorithms Naïve and Heuristic were described in Section 3.2. The three variants of Heuristic employ hand-tuned weights over the features, a learned classifier, and a linear regression model, respectively. Finally, the question-answer pairing algorithm Random retrieves the list of candidate answer sentences as hand-tuned Heuristic. However, it then simply picks a random candidate answer. Thus Random allows us to determine the merits of our three pairing strategies.

Results We took the data from Annotators 1 and 2 and performed 5-fold cross validation for each annotator's dataset. Tables 5 and 6 give the precision and recall results. It can be seen that the three variants of our Heuristic algorithm perform significantly better than the S&M and Naïve algorithms on the two annotators' individual data. However, SMQA exhibits superior recall on Annotator 2's data, while Naïve has slightly winning precision under LCS. There is little difference between the three Heuristic variants. The relative performances of SMQA-highest and Heuristic-C is understood from the primary differences between them: the feature sets, and the operation of the latter at the sentence level rather than the paragraph level. The much lower LCS scores that our algorithms have for Annotator 2 can be explained in that Annotator 1 tended to mark short, single-sentence question and answer segments, whereas Annotator 2 tended to mark longer segments; our algorithms extract only sentences as questions and answers, not longer pieces of text. Thus both metrics have value in assessing the algorithms.

We experimented with permutations of Heuristic's methodology. Removing its second condition, i.e., to consider only the first reply from each individual, provides improvement in some cases, at the cost of significantly larger runtime. Varying the heuristic on the number of questions to which an answer may pertain can modestly increase F_1-score; the best overall setting on our dataset is 3. For Heuristic-LR, taking the best answer with probability over 0.8 is better than our default threshold of 0.5. As expected, taking *all* answers with probability over 0.5 (rather than the best), as S&M, does not improve the performance of any Heuristic variant.

The average processing time per *thread* is given in Table 7,

Algorithm	Mean	Variance	Median
SMQA	1450	1480000	1340
SMQA-regex	69.7	4570	62.0
SMQA-highest	57.6	4570	32.0
Naïve	0.0769	0.0710	0
Random	2.46	67.9	0
Heuristic	296	84100	172
Heuristic-C	372	130000	249
Heuristic-LR	256	62300	171

Table 7: Question-answer pairing computation times (ms)

in terms of number of milliseconds per thread. Mean thread length is 5.2 email messages; mean sentences per message is 13.9. We can see that the median amount of time taken by all the algorithms, except for SMQA, is extremely low, but that our Heuristic variants take an order of magnitude more time than our SMQA variants, but an order of magnitude less than the original S&M. This relatively higher runtime, and variance, is at the gain of the higher precision and recall. Heuristic-C is the slowest of the three variants. The bulk of the time for S&M is taken by question detection; for Heuristic, it is finding semantic similarity (feature 10). Feature computation is reflected in the runtime of Naïve versus Heuristic.

Learning and Features Learning curves (not shown for reasons of space) show that, under both SM and LCS metrics, both Heuristic-C and Heuristic-LR converge with a training set of approximately 30 threads. The comparable-or-better performance of hand-tuned Heuristic, after only modest effort in tuning the feature weights, suggests that it is identification of appropriate features that is more significant in this problem than the values of the weights. The similar performance of the two learning variants, despite their different models, lends support to this. However, automated acquisition of feature weights, since training is straightforward, does offer the flexibility of adaption to characteristics of particular datasets.

A factor analysis of the relative benefit of features indicates, on our benchmark corpora, the features that give greatest benefit are numbers 2–7. Features 1 (stop words) and 9–11 are also beneficial, but feature 8 (whether m_A is the first reply to m_Q) has surprisingly little contribution. Although visual inspection reveals that answers are frequently seen in the first reply to m_Q, we hypothesize that this occurrence is accounted for by feature 4.

5 Conclusion and Future Work

Question-answer identification and pairing provides a generative summary of an email thread. For question detection, on threads drawn from the Enron and Cspace corpora, we find that the learning algorithm of Shrestha and McKeown [2004] suffers from poor precision when exposed to non-interrogative questions. A generalized expression matching algorithm performs adequately with very low runtime. Future work is to investigate the promise of the question detection method of Cong *et al.* [2008] to email conversations.

For answer detection and question-answer pairing, we presented a generalization of Shrestha and McKeown's feature-based algorithm, and found that our heuristic-based method balances precision and recall, while maintaining a modest computation time. Future work is to (1) examine further the interplay of sentence-level and paragraph-level features, (2) more fully exploit named entity extraction, and (3) consider explicitly thread structure by means of the induced graph of message relationships, together with quoted material (compare [Carenini *et al.*, 2007]).

Acknowledgments We thank with appreciation the volunteers who annotated the thread corpora. This material is based upon work supported by DARPA under Contract No. FA8750-07-D-0185. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s).

References

[Belotti *et al.*, 2005] V. Belotti, N. Ducheneaut, M. Howard, I. Smith, and R. Grinter. Quality vs. quantity: Email-centric task management and its relations with overload. *Human-Computer Interaction*, 20(2/3):89–138, 2005.

[Carenini *et al.*, 2007] G. Carenini, R. T. Ng, and X. Zhou. Summarizing email conversations with clue words. In *Proc. of WWW'07*, pages 91–100, 2007.

[Cohen, 1996] W. Cohen. Learning trees and rules with setvalued features. In *Proc. of AAAI-96*, pages 709–716, 1996.

[Cong *et al.*, 2008] G. Cong, L. Wang, C.-Y. Lin, Y.-I. Song, and Y. Sun. Finding question-answer pairs from online forums. In *Proc. of SIGIR'08*, pages 467–474, 2008.

[Dabbish and Kraut, 2006] L. Dabbish and R. Kraut. Email overload at work: An analysis of factors associated with email strain. In *Proc. of CSCW'06*, pages 431–440, 2006.

[Dredze *et al.*, 2008] M. Dredze, V. R. Carvalho, and T. Lau, editors. *Enhanced Messaging: Papers from the 2008 AAAI Workshop*, Menlo Park, CA, 2008. AAAI Press.

[Freed *et al.*, 2008] M. Freed, J. Carbonell, G. Gordon, J. Hayes, B. Myers, D. Siewiorek, S. Smith, A. Steinfeld, and A. Tomasic. RADAR: A personal assistant that learns to reduce email overload. In *Proc. of AAAI-08*, pages 1287–1293, 2008.

[Kathol and Tur, 2008] A. Kathol and G. Tur. Extracting question/answer pairs in multi-party meetings. In *Proc. of ICASSP'08*, pages 5053–5056, 2008.

[Klimt and Yang, 2004] B. Klimt and Y. Yang. The Enron corpus: A new dataset for email classification research. In *Proc. of ECML'04*, pages 217–226, 2004.

[Minkov *et al.*, 2005] E. Minkov, R. Wang, and W. Cohen. Extracting personal names from emails: Applying named entity recognition to informal text. In *Proc. of HLT-EMNLP'05*, 2005.

[Pirrò and Seco, 2008] G. Pirrò and N. Seco. Design, implementation and evaluation of a new similarity metric combining feature and intrinsic information content. In *Proc. of ODBASE'08*, 2008.

[Shrestha and McKeown, 2004] L. Shrestha and K. McKeown. Detection of question-answer pairs in email conversations. In *Proc. of COLING'04*, pages 542–550, 2004.

[Ulrich *et al.*, 2008] J. Ulrich, G. Murray, and G. Carenini. A publicly available annotated corpus for supervised email summarization. In *Proc. of AAAI'08 Workshop on Enhanced Messaging*, pages 77–81, 2008.

[Yeh and Harnly, 2006] J.-Y. Yeh and A. Harnly. Email thread reassembly using similarity matching. In *Proc. of CEAS'06*, 2006.