AI Communications 25 (2012) 271–283
DOI 10.3233/AIC-2012-0516
IOS Press

Detection of imperative and declarative question–answer pairs in email conversations

Helen Kwong [a] and Neil Yorke-Smith [b,*]
[a] *Stanford University, Stanford, CA, USA*
E-mail: helenkwong@gmail.com
[b] *American University of Beirut, Beirut, Lebanon and*
SRI International, Menlo Park, CA, USA
E-mail: nysmith@aub.edu.lb

Abstract. Question–answer pairs extracted from email threads are valuable in constructing summaries of the content of the thread, as well as in providing data for semantic-based assistance with email. Previous work dedicated to extracting question–answer pairs from email threads considers only questions in interrogative form. We extend the scope of question and answer detection and pairing to encompass 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. On multiple benchmark email corpora, we show that our methods balance precision and recall in extracting question–answer pairs, while maintaining a modest computation time.

Keywords: Email, question–answer pairing, information extraction, NLP

1. Introduction

The ubiquitous tool of email has become a tyrant upon our lives. The "tyranny of email" negatively impacts work performance and tangibly weighs upon organizations and even national economies [8]. Various software tools have been developed to ameliorate email overload [41]. Some seek to refactor the role and workflow of email – for example, blurring the concepts of email message, to-do item, and calendar entry [29]; interweaving email with explicit task management [1, 15]; or semantically understanding actions pertaining to a message and assisting the user to perform them [14,33]. Others seek to improve email client software with nuggets of Artificial Intelligence (AI) – for example, auto-filing messages into folders, detecting missing attachments, or summarizing email threads [11]. Parts of these latter developments, e.g., attachment detection, are now seen in standard email clients. Still others seek to exploit email's ubiquity – such as to mine email corpora for social network analysis [21,32].

The nature of email as asynchronous, unstructured, written, multi-party communication is one reason for its ubiquity. Research has sought to automatically derive summaries of email threads and embed them in email clients [25,38]. Summaries can be *extractive*, i.e., selected fragments of the thread [9,25,28], or *generative*, ranging from as sophisticated as a high-level synopsis [39] 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).

This article considers the two related problems of (1) identifying questions in an email thread (possibly between multiple parties) – *question identification* – and (2) linking the identified questions to possible answers later in the thread – *question–answer pairing*. Our work arose as part of an effort to endow an email client with a range of usable smart features by means of AI technology [14].

Prior work that studies these twin 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 can often be expressed in non-interrogative forms: declarative questions such as "I was wonder-

*Corresponding author: Neil Yorke-Smith, American University of Beirut, Beirut 1107-2020, Lebanon and SRI International, Menlo Park, CA 94025, USA. E-mail: nysmith@aub.edu.lb.

ing 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 [20] found that about one in five questions are non-interrogative in form. Further, besides focusing on interrogative question forms, previous work for email threads has operated at the paragraph level, thus potentially missing more fine-grained conversation.

In this article we propose an approach to extract question–answer pairs that include imperative and declarative questions. Our main contributions are fourfold. First, analysis of prior question–answer (Q–A) pairing algorithms on recognized email corpora (e.g., the Enron corpus). Second, an improved *non-learning* method for question detection. Third, extension of both supervised learning and non-learning methods for answer detection to include non-interrogative questions and to operate at sentence-level fidelity. Fourth, analysis of performance of the algorithms with more stringent metrics.

2. Related work

Given the ubiquity of email, it is little surprise that a wide spectrum of email research has appeared in communities from computational linguistics to machine learning and organizational behaviour to human-computer interaction. Laclavik and Maynard [22] survey email communication and processing research that targets business applications.

With the aim of classifying the intent of an email sender, Cohen et al. [6] defined verb and noun categories for "email speech acts". They developed a supervised learning method to recognize such speech acts in email threads. Separately, Dhillion et al. [10] defined speech acts for meeting transcripts.

Question detection and answer detection is a form of speech act detection. Shrestha and McKeown [34] 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 instance, the scope of their question detection does not include declarative questions such as "I was wondering if you are free today".

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. In a supervised learning approach, the classifier is trained and tested on a private corpus of messages from an ACM student committee, annotated by a pair of human annotators.

Academic attention has been growing for online forums and discussion boards, perhaps due to the commercial value and the accessibility of annotated corpora. 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.

Cong et al. [7] examine question and answer detection and pairing for online forums. 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. For online forums, they find the best performance with a graph-based method employing KL-divergence.

Yang et al. [42] build on CWLSS and other work dedicated to online forums, focusing on extracting question contexts and answers in forums. They develop a learning method based on structural Support Vector Machines (SVMs), over features including textual similarity, forum post structure and grammatical measures. Hong and Davison [16] apply information retrieval methods to question detection and question–answer pairing in forums, noting the importance of non-content, structural features. For Chinese language forums, Wang et al. [40] develop rules for question detection, supplemented with structural features for answer pairing.

Ravi and Kim [31] detect unanswered questions in online forums through speech act classification, achieved by employing a linear SVM over N-gram features. They apply a set of rules to detect threads that contain unanswered questions; specific question–answer pairing is not the goal of their work.

Jeong et al. [17] apply semi-supervised learning to recognize speech acts in email threads and online forums at the sentence level. They model lexical features as subtree patterns, learning over a transcribed telephone speech corpus. They transfer from that domain to the domains of email and online forum, by using bootstrapping and semi-supervised boosting. The speech acts they detect, a modified subset of those of Dhillon et al. [10], include rhetorical question, or/or-clause question, Wh-question (5W words), yes-no question and (unclassified) open-ended question. Lacking fidelity for question answers, these speech acts represent a subset of the speech acts necessary for Q–A detection and pairing, since indeed Q–A pairing is not the goal of their work.

Related to the task of Q–A detection and pairing in online forums is the task of answer extraction in FAQ (Frequently Asked Questions) documents. Archetypal approaches to the latter focus on the use of information retrieval techniques guided by machine learning, most often operating at the sentence level. For example, Soricut and Brill [35] contrast a web-based predictive approach based on statistical translation, a retrieval approach based on a language model, and a hybrid approach that combines statistical chunking and classical information retrieval.

Marom and Zukerman [24] study automation of help-desk queries. Their goal of providing intelligent user assistance is akin to the motivation for smart email assistance that frames our work. They seek to assist users – help-desk operators in this case – by generating suggested help-desk responses, and possibly automatically sending the responses. This allows human operators to focused on atypical queries. Help-desk dialogues differ from general email conversations in several ways: the focused objective of the customer, the restricted domain of intercourse, and the usually two-party dialogue.

Marom and Zukerman employ a unsupervised learning approach, leveraging a large proprietary corpus of request- response email threads between customers and operators at Hewlett-Packard. The focus of this and similar works is to attain high precision. They examine document-level and sentence-level operation of retrieval-based and predictive-based methods. They combine the methods with a ensemble learning approach. In contrast to Q–A extraction from email threads, using syntactic features of the help-desk dialogue did not improve the results.

Just as the information obtained from mining help-desk email dialogues has been applied to assist users, the information gained by accurately extracting question–answer pairs has been used in higher-level tasks, such as to inform extractive thread summarization [25], and automated task extraction [14].

Besides email threads, online forums, FAQs and help desks, there also has been work on extracting question–answer pairs in multi-party dialogue [19]. While extraction of questions in meeting transcripts could be similar to question extraction 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. Nonetheless, concepts from discourse analysis can inform dedicated methods for email threads.

A still more challenging task 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. For example, Lamjiri et al. [23] found that using semantic similarity information is helpful for closed-domain question answering.

As noted, the work reported in this article arose as part of a larger effort to endow an email client and other desktop productivity software with a range of usable smart assistive features by means of AI technology [12,14]. A baseline question detection algorithm had already been implemented in our email assistant when we began our work; we describe it in the sequel. For a description of the evaluation of the larger project, we refer to Steinfeld et al. [36].

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 [43] (the antonymous task is text segmentation [37]); we assume this procedure has been performed.

3.1. Question detection

Given our purpose of question–answer pairing, for us question identification serves the greater purpose

of linking questions with answers. We consider three algorithms for question detection that operate 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 a question-detection 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 when we began our work. 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: i.e., a sentence is classified as a question if it commences with Who, What, Where, When, Why or How. The performance of Algorithm Naïve being unlikely to improve much with 5W-1H detection, we focussed on evaluating alternate methods.

S&M. The state-of-the-art in question detection in email threads is the work of Shrestha and McKeown [34]. Algorithm S&M uses Ripper [5] to learn a model that classifies each sentence as a question or a non-question, based on parts-of-speech features. In that prior work, the training data was a transcribed telephone speech corpus [18]. The scope of the S&M algorithm is limited to detection of questions in interrogative form.

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, before considering building a question detector based on learning.

Regex. Like the Naïve question detector, algorithm Regex is also based entirely on regular expressions. A sentence is detected as a question if it fulfils any of the following conditions; otherwise, it is classified as a non-question:

- It ends with a question mark, and is not a URL or a filename.
- 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" begins with the words "can you", 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. An alternative approach is to acquire patterns by learning; CWLSS take this approach, and present a generalized learned classification model to acquire patterns.

3.2. Answer detection and question–answer 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 [3], 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 then 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 an-

[1] An improvement to S&M, discussed below, 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.

swer 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 earlier questions are answered earlier than later questions; however, 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 an apartment?"). Speech acts corresponding to these question types are akin to those of Jeong et al. [17]. We emphasize however that our goal is not speech act detection in itself, but question–answer linkage, and our approach is not founded on a causal speech act analysis but directly tackles the Q–A pairing task. When the Naïve algorithm looks for an answer to a question, only a subset of the types of answers are captured; namely, 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 we remarked earlier, the features that the S&M algorithm uses include standard textual analysis features from information retrieval, such as the cosine similarity between the question and candidate answer paragraphs, features derived from the thread structure, such as the number of intermediate messages between the two paragraphs, and features based on comparison with other candidate paragraphs, such as the number of earlier candidate answers.

Heuristic Q–A. We developed 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-selected 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 each content sentence for questions, 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] and (3) not one of the detected question sentences. Condition (3) prevents the algorithm from considering the possibility that a question could be answered by a further question.

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 (1) train answer detectors, and (2) 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 default value 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 feature 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).

[3] That is, if Bill has already replied to Alice's question email, then if Bill sends another email, we exclude sentences in the second message as being candidate answers to Alice's question, at the cost of potentially reduced precision. This assumption is notably not plausible for online forums, but it is for email threads.

(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 sentence 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 [43] – 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 [13,30].

The most difficult feature to capture is the last: expected answer type. The algorithm Naïve detects answer type crudely by the means of regular expressions. This purely syntactic approach suffices for yes/no questions, but is insufficient for other types of answers. We developed a named entity recognizer that identifies and classifies proper names, places, temporal expressions, currency and numbers, among other entities. Based on static and learned patterns on N-grams and lexical features, 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 did not attempt to detect by type essay, what and choice answers.

A further means of capturing type is to obtain the semantic similarity between the classification *tags* for the expected answer types and the candidate answer text. For example, a *when* question would be represented by the string tags `time` and `date`; a candidate answer that contains the word `month` would receive the string tag `date`. This computation gave a modest improvement.

4. Empirical analysis

We undertook a set of 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 Quad 2.33 GHz machine with 3.25 GB memory, running Windows XP.

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 judgements. In our experiments we follow prior works in information retrieval to train (where relevant) and test algorithms on human-annotated datasets.

In contrast to email summarization and to question–answering, where standardized datasets now exist [27, 38], there are unfortunately no email corpora annotated adequately 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 (i.e., the lack of available corpora) differs also to online forums: typically in forums, many community members rate posts to a thread, thus providing ready and large datasets [7,16].

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, and they are often tagged with parts-of-speech. We used the widely-studied Enron corpus [20] and the CSpace corpus [26]. The former corpus results from senior management at Enron. The cleaned version of the corpus contains about 30,000 threads, with messages from about 150 users. The latter corpus was collected from a management class at CMU in which MBA students were organized in teams and ran simulated companies over a 14-week period. It contains 2739 email threads. Messages from about 275 users are contained in the corpus; the messages tend to be very task-oriented in nature.

The more recently released BC3 email corpus [38] contains 40 labelled email threads from the TREC W3C corpus [27]. Each thread is annotated by three different annotators to include extractive and generative summaries, and speech acts regarding meeting arrangements. Like the other email corpora, BC3 is not annotated with question–answer pairings.

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 examined 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. 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 recall 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 proportion of non-questions as false positives.

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 (i.e., the questions marked as imperative, declarative and other were taken out), 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 [34], i.e., we did not need to train it. Thus all 700 sentences were used for testing purposes.

We maintained the same training methodology and dataset as the original in order to be able to directly compare results. If trained on POS-tagged email data, the precision of S&M can be expected to improve.

Results. The question detection 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 observation that its precision is so low is at first surprising, because it is reported that the S&M algorithm achieves high precision [7,34].

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".

Table 1
Question detection, interrogative questions only

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 2
Question detection, including non-interrogative questions

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

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.

The results for the two regular expression algorithms, Naïve and Regex, 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 18.5 ms. 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 0.185 ms, respectively (full dataset, including non-interrogative questions). S&M exhibits considerable variance of 1750 ms.

Table 3 reports results from [7]. Although these results are on a different dataset and for online forums rather than email threads, we give them for indicative comparison. It can be seen that their 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

Table 3
Question detection on online forums, including non-interrogative questions [7]

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**

Note: Results given for comparative purposes only.

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.

Taking the results together, we found 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 [4] 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. 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 that both annotated.

Additional data annotation would allow us to distinguish interrogative and non-interrogative questions, since different annotators marked question/non-question detection and question–answer pairing. It would be interesting to ask Q–A pairing annotators to distinguish between interrogative/non-interrogative questions.

Table 4
Question–answer pairing methods

Algorithm	Abbreviation
S&M original	SMQA
S&M with Regex	SMQA-regex
S&M with Regex and highest probability	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

We used two metrics to evaluate the quality of the question–answering pairing. Clearly, a prerequisite for an algorithm to perform well at the pairing is that it is able to identify questions and possible candidate answers with sufficient accuracy. We find that the Regex algorithm for question detection serves the question detection adequately.

The first metric we employed is the paragraph-based metric used by Shrestha and McKeown [34], which we will call SM. This metric is most suited to when messages are segmented 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 shorter 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.[4]

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 of being the answer paragraph.

[4] CWLSS consider three metrics suited to Q–A forums: mean reciprocal [answer] rank, mean average [answer] precision and precision of first ranked answer.

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, instead of then scoring each candidate and taking the highest scoring one for each question, 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. Experiments were replicated 50 times. 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. Of the three variants, Heuristic-LR is exhibits slightly better performance.

While SMQA exhibits superior recall on Annotator 2's data both LCS and SM metrics, its F_1score is badly affected by poor precision. Comparing with our two SMQA variants and also with Naïve suggests that the poor precision of SMQA is due to the rate of false positives of its question detection in the presence of non-interrogative questions. On the other hand, Naïve exhibits reasonable precision but poor recall, suggesting that the algorithm rarely can find question–answer pairs.

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. Results are similar on the datasets from the remaining two annotators.

Our algorithms fare well on the intersection and union sets (described in [34]), with the three Heuristic variants yielding essentially the same performance, as shown in Tables 7 and 8. On the intersection set, Heuristic is insignificantly best, while on the union set, it is Heuristic-LR. Our Heuristic variants exhibit a clear separation in performance for both metrics over our SMQA variants.

Comparing the datasets for the two annotators reveals that Annotator 1 tended to mark short, single-sentence question and answer segments, whereas An-

Table 5
Question–answer pairing, Annotator 1

Algorithm	LCS			SM		
	Precision	Recall	F_1-score	Precision	Recall	F_1-score
SMQA	0.1014	0.4703	0.1653	0.0995	0.5125	0.1654
SMQA-regex	0.5016	0.4746	0.4830	0.4926	0.5208	0.5015
SMQA-highest	0.4656	0.4606	0.4616	0.5062	0.5047	0.5034
Naïve	0.4733	0.1047	0.1694	0.5378	0.1332	0.2105
Random	0.3209	0.3130	0.3157	0.4862	0.4956	0.4899
Heuristic	0.5123	0.4816	0.4951	0.6100	0.5789	0.5922
Heuristic-C	0.4920	0.4643	0.4760	0.6089	0.5801	0.5920
Heuristic-LR	**0.5150**	**0.4857**	**0.4982**	**0.6211**	**0.5882**	**0.6021**

Table 6
Question–answer pairing, Annotator 2

Algorithm	LCS			SM		
	Precision	Recall	F_1-score	Precision	Recall	F_1-score
SMQA	0.1222	**0.5465**	0.1972	0.1018	**0.4134**	0.1607
SMQA-regex	0.4818	0.4546	0.4587	0.4272	0.3651	0.3852
SMQA-highest	0.5273	0.4597	0.4890	0.4815	0.3450	0.3985
Naïve	0.5375	0.1050	0.1728	0.4874	0.0865	0.1445
Random	0.4660	0.4084	0.4330	0.4477	0.3274	0.3752
Heuristic	0.6072	0.5039	0.5483	0.5567	0.3768	0.4459
Heuristic-C	0.5595	0.5018	0.5439	0.5547	0.3784	0.4463
Heuristic-LR	**0.6236**	0.5240	**0.5670**	**0.5725**	0.3908	**0.4597**

Table 7
Question–answer pairing, intersection set

Algorithm	LCS			SM		
	Precision	Recall	F_1-score	Precision	Recall	F_1-score
SMQA	0.0820	0.5514	0.1403	0.0804	0.5404	0.1375
SMQA-regex	0.3804	0.5597	0.4420	0.3803	0.5595	0.4419
SMQA-highest	0.4514	0.6490	0.5286	0.4364	0.5597	0.4867
Naïve	0.4896	0.1807	0.2583	0.4896	0.1807	0.2583
Random	0.3958	0.5603	0.4603	0.3906	0.5177	0.4417
Heuristic	0.5293	**0.7149**	**0.6043**	**0.5118**	**0.6222**	**0.5575**
Heuristic-C	0.5283	0.7118	0.6021	0.5104	0.6175	0.5546
Heuristic-LR	**0.5300**	0.7138	0.6041	0.5131	0.6204	0.5571

Table 8
Question–answer pairing, union set

Algorithm	LCS			SM		
	Precision	Recall	F_1-score	Precision	Recall	F_1-score
SMQA	0.1128	0.4815	0.1812	0.1104	0.4682	0.1772
SMQA-regex	0.5324	0.4402	0.4471	0.5320	0.4398	0.4767
SMQA-highest	0.5976	0.4505	0.5108	0.5928	0.3995	0.4746
Naïve	0.5610	0.0907	0.1544	0.5610	0.0907	0.1544
Random	0.5584	0.4212	0.4777	0.5572	0.3874	0.4546
Heuristic	0.7016	0.5055	0.5845	0.6961	0.4467	0.5414
Heuristic-C	0.7053	0.4963	0.5795	0.6983	0.4383	0.5358
Heuristic-LR	**0.7126**	**0.5125**	**0.5935**	**0.7074**	**0.4542**	**0.5506**

notator 2 tended to mark longer segments (our algorithms extract only sentences as questions and answers, not longer pieces of text). Marom and Zukerman [24] remark on the potential of extracting multi-sentence answers, in contrast to whole message or single sentence extraction; this is an area of our future work.

We experimented with permutations of Heuristic's methodology. Removing its second condition, i.e., allowing it to consider more than 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 9, 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.

Table 9
Question–answer pairing computation times (ms)

Algorithm	Mean	Variance	Median
SMQA	1800	5480	2590
SMQA-regex	37.4	9.70	305
SMQA-highest	33.8	7.5	86.0
Naïve	1.10	0	0
Random	1.70	0	0
Heuristic	241	341	945
Heuristic-C	320	508	1340
Heuristic-LR	316	488	1320

Learning and features. Figures 1 and 2 show the learning curve of that is representative of Heuristic-C

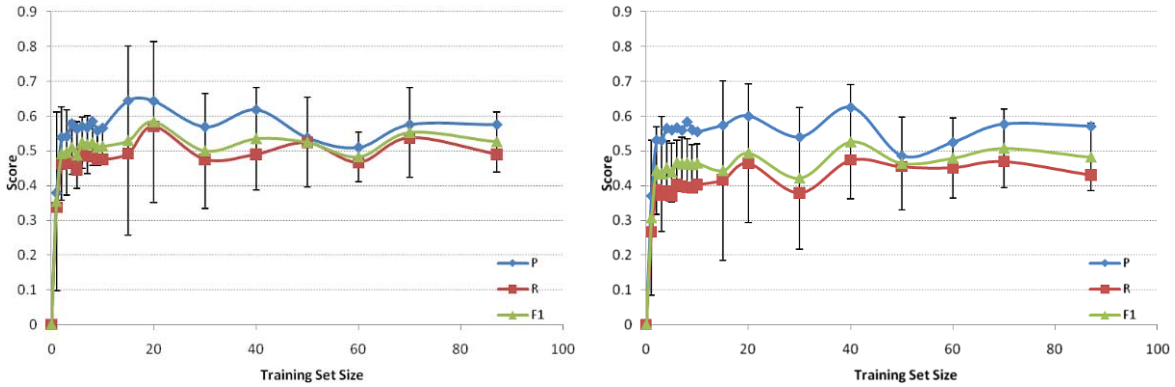

Fig. 1. Learning curves for Heuristic-C. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-2012-0516.)

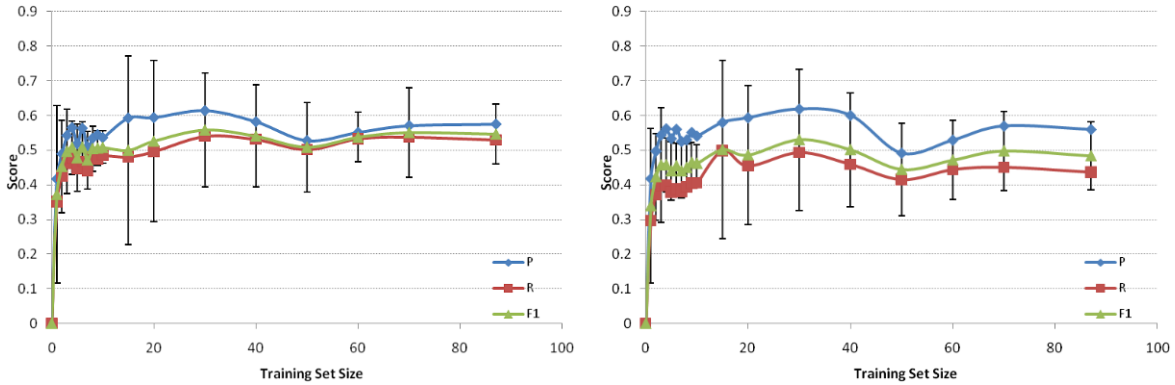

Fig. 2. Learning curves for Heuristic-LR. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-2012-0516.)

and Heuristic-LR (the error bars indicate standard deviations). Under both SM and LCS metrics, we see that learning converges for both variants 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 the 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 conclusion. 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 the features indicates that, on our benchmark corpora, the features which 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

The primary tasks that people perform upon their email are identified by Whittaker et al. [41] as allocating attention, deciding actions, managing tasks, and organizing messages and folders. This article presents an extension of question–answer identification and pairing techniques to encompasses imperative and declarative as well as interrogative questions forms. The resulting algorithms have been embedded into a smart email system that assists its user, to differing degrees, with all four of the identified email-related activities [12,14]. Although the system itself is not the subject of this article, we remark that evaluation of the system reported improved task performance and user satisfaction compared to commercial off-the-shelf tools [12].

For question detection, on threads drawn from the Enron and CSpace corpora, we found that the learning algorithm of Shrestha and McKeown, designed for interrogative questions, suffers from poor precision when exposed to non-interrogative questions. A gen-

eralized expression matching algorithm performs adequately with very low runtime.

Second, for answer detection and question–answer pairing, we presented a generalization of Shrestha and McKeown's feature-based algorithm. Empirical results indicate that our heuristic-based method, using linear regression, balances precision and recall while maintaining a modest computation time.

Summarized, the specific contributions in our approach are to (1) consider non-interrogative questions, (2) work at the sentence level not paragraph level, (3) improve (albeit with a simple method) question detection, (4) remove greetings and signatures, (5) reassess S&M features and expand upon them, (6) look at the highest-scoring answer instead of the first answer and (7) consider non-learning and linear regression approaches as well as classification approaches. We obtain significant improvement in F_1 score as a result. Further, our empirical investigations use standard corpora and operate at the sentence level.

The value in automated and semi-automated analysis of email conversations points to further work. We have followed a methodology of supervised learning. A radical alternative is unsupervised learning to attempt to automatically mine and exploit patterns in the corpora data. Such an approach has been applied, with some success, to online forums by Marom and Zukerman [24]. Perhaps more directly applicable to our interest, Jeong et al. [17] bootstrap from non-email corpora to the email domain using transfer learning. This is one means to obtain training data without the need for corpora annotated with question and answer information.

The Heuristic algorithms for Q–A pairing extract sentences as questions and answers. Future work is to investigate the possibility of extracting multi-sentence phrases, particularly as answers [24]. Specifically for question detection, future work is to investigate the promise of the question detection method of Cong et al. [7] to email conversations. For answer detection and question–answer pairing, future work is in multiple directions. First, to examine further the interplay of sentence-level and paragraph-level features. Second, to more fully exploit named entity extraction. Third, to consider explicitly thread structure by means of the induced graph of message relationships, together with quoted material (compare [2,37]).

Acknowledgements

The authors express appreciation to the volunteers who annotated the thread corpora, and to Karl Pichotta and Prateek Tandon for implementation assistance. We thank Michael Freed, Michael Wimble, the anonymous reviewers, and the reviewers of a preliminary version of this article. The first author thanks Andrew Ng. This material is based upon work supported by DARPA under Contract No. FA8750-07-D-0185, performed while the first author was at SRI International. Any opinions, findings and conclusions or recommendations expressed are those of the author(s) and do not necessarily reflect the views of DARPA.

References

[1] 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) (2005), 89–138.

[2] G. Carenini, R.T. Ng and X. Zhou, Summarizing email conversations with clue words, in: *Proc. 16th Int. Conf. on World Wide Web*, Banff, Canada, 2007, pp. 91–100.

[3] G. Carenini, R.T. Ng and X. Zhou, Summarizing emails with conversational cohesion and subjectivity, in: *Proc. 46th Annual Meeting of the Association for Computational Linguistics*, Columbus, OH, 2008, pp. 353–361.

[4] J. Carletta, Assessing agreement on classification tasks: the kappa statistic, *Computational Linguistics* **22**(2) (1996), 249–254.

[5] W.W. Cohen, Learning trees and rules with set-valued features, in: *Proc. 13th National Conf. on Artificial Intelligence*, Portland, OR, 1996, pp. 709–716.

[6] W.W. Cohen, V.R. Carvalho and T. Mitchell, Learning to classify email into "speech acts", in: *Proc. Conf. on Empirical Methods in Natural Language Processing*, Barcelona, Spain, 2004, pp. 309–316.

[7] G. Cong, L. Wang, C.-Y. Lin, Y.-I. Song and Y. Sun, Finding question–answer pairs from online forums, in: *Proc. 31st Int. ACM Conf. on Research and Development in Information Retrieval*, Singapore, 2008, pp. 467–474.

[8] L. Dabbish and R. Kraut, Email overload at work: An analysis of factors associated with email strain, in: *Proc. Conf. on Computer Supported Collaborative Work*, Banff, Canada, 2006, pp. 431–440.

[9] A. Dalli, Y. Xia and Y. Wilks, Adaptive information management: FASiL email summarization system, in: *Proc. 20th Int. Conf. on Computational Linguistics*, Geneva, Switzerland, 2004, pp. 23–27.

[10] R. Dhillon, S. Bhagat, H. Carvey and E. Shriberg, Meeting recorder project: dialog act labeling guide, Technical report, International Computer Science Institute, 2004.

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

[12] A. Faulring, B. Myers, K. Mohnkern, B. Schmerl, A. Steinfeld, J. Zimmerman, A. Smailagic, J. Hansen and D. Siewiorek, Agent-assisted task management that reduces email overload, in: *Proc. 14th Int. Conf. on Intelligent User Interfaces*, Hong Kong, 2010, pp. 61–70.

[13] C. Fellbaum, ed., *WordNet: An Electronic Lexical Database*, MIT Press, Cambridge, MA, 1998.

[14] 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. 23th National Conf. on Artificial Intelligence*, AAAI Press, Menlo Park, CA, 2008, pp. 1287–1293.

[15] J. Gwizdka, TaskView: design and evaluation of a task-based email interface, in: *Proc. IBM Centers for Advanced Studies Conf.*, Toronto, Canada, 2002, pp. 136–145.

[16] L. Hong and B.D. Davison, A classification-based approach to question answering in discussion boards, in: *Proc. 32nd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Boston, MA, 2009, pp. 171–178.

[17] M. Jeong, C.-Y. Lin and G.G. Lee, Semi-supervised speech act recognition in emails and forums, in: *Proc. Conf. on Empirical Methods in Natural Language Processing*, Singapore, 2009, pp. 1250–1259.

[18] D. Jurafsky, E. Shriberg and D. Biasca, Switchboard SWB-DDAMSL shallow-discourse-function annotation coders manual, Draft 13, Technical Report 97-02, Institute of Cognitive Science, University of Colorado, Boulder, CO, 1997.

[19] A. Kathol and G. Tur, Extracting question/answer pairs in multi-party meetings, in: *Proc. 33rd IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Las Vegas, NV, 2008, pp. 5053–5056.

[20] B. Klimt and Y. Yang, The Enron corpus: a new dataset for email classification research, in: *Proc. 15th European Conf. on Machine Learning*, Pisa, Italy, 2004, pp. 217–226.

[21] M. Laclavik, S. Dlugolinsky, M. Kvassay and L. Hluchy, Email social network extraction and search, in: *Proc. Int. Conf. on Web Intelligence and Intelligent Agent Technology*, Lyon, France, 2011, pp. 373–376.

[22] M. Laclavik and D. Maynard, Motivating intelligent email in business: an investigation into current trends for email processing and communication research, in: *Proc. CEC'09 Workshop on Emails in e-Commerce and Enterprise Context*, Vienna, Austria, 2009, pp. 476–482.

[23] A.K. Lamjiri, L. Kosseim and T. Radakrishnan, Comparing the contribution of syntactic and semantic features in closed versus open domain question answering, in: *Proc. 1st IEEE Int. Conf. on Semantic Computing*, Irvine, CA, 2007, pp. 679–685.

[24] Y. Marom and I. Zukerman, An empirical study of corpusbased response automation methods for an e-mail-based helpdesk domain, *Computational Linguistics* **35**(4) (2009), 597–635.

[25] K. McKeown, L. Shrestha and O. Rambow, Using questionanswer pairs in extractive summarization of email conversations, in: *Proc. 8th Int. Conf. on Computational Linguistics and Intelligent Text Processing*, Mexico City, Mexico, 2007, pp. 542–550.

[26] E. Minkov, R. Wang and W.W. Cohen, Extracting personal names from emails: applying named entity recognition to informal text, in: *Proc. Conf. Human Language Technology and Conf. on Empirical Methods in Natural Language Processing*, Vancouver, Canada, 2005, pp. 443–450.

[27] National Institute of Standards and Technology, Text Retrieval Conference (TREC), available at: http://trec.nist.gov/, 2010.

[28] A. Nenkova and A. Bagga, Facilitating email thread access by extractive summary generation, in: *Proc. Int. Conf. on Recent Advances in Natural Language Processing*, Borovets, Bulgaria, 2003, pp. 287–296.

[29] Open Source Applications Foundation, Chandler project, available at: http://www.chandlerproject.org/, 2010.

[30] G. Pirrò and N. Seco, Design, implementation and evaluation of a new similarity metric combining feature and intrinsic information content, in: *Proc. 7th Int. Conf. on Ontologies, DataBases, and Applications of Semantics*, Monterrey, Mexico, 2008, pp. 1271–1288.

[31] S. Ravi and J. Kim, Profiling student interactions in threaded discussions with speech act classifiers, in: *Proc. 13th Int. Conf. on Artificial Intelligence in Education*, Los Angeles, CA, 2007, pp. 357–364.

[32] R. Rowe, G. Creamer, S. Hershkop and S.J. Stolfo, Automated social hierarchy detection through email network analysis, in: *Proc. 9th WebKDD and 1st SNA-KDD Workshop on Web Mining and Social Network Analysis*, San Jose, CA, 2007, pp. 109–117.

[33] S. Scerri, G. Gossen, B. Davis and S. Handschuh, Classifying action items for semantic email, in: *Proc. 7th Int. Conf. on Language Resources and Evaluation*, Valletta, Malta, 2010.

[34] L. Shrestha and K. McKeown, Detection of question–answer pairs in email conversations, in: *Proc. 20th Int. Conf. on Computational Linguistics*, Geneva, Switzerland, 2004, pp. 889–895.

[35] R. Soricut and E. Brill, Automatic question answering using the Web: beyond the factoid, *Information Retrieval* **9**(2) (2006), 191–206.

[36] A. Steinfeld, R. Bennett, K. Cunningham, M. Lahut, P. Quinones, D. Wexler, D. Siewiorek, J. Hayes, P. Cohen, J. Fitzgerald, O. Hansson, M. Pool and M. Drummond, Evaluation of an integrated multi-task machine learning system with humans in the loop, in: *Proc. 7th NIST Workshop on Performance Metrics for Intelligent Systems*, Washington, DC, 2007, pp. 182–188.

[37] N. Stokes, J. Carthy and A.F. Smeaton, SeLeCT: a lexical cohesion based news story segmentation system, *AI Communications* **17**(1) (2004), 3–12.

[38] J. Ulrich, G. Murray and G. Carenini, A publicly available annotated corpus for supervised email summarization, in: *Proc. AAAI'08 Workshop on Enhanced Messaging*, Chicago, IL, 2008, pp. 77–81.

[39] S. Wan and K. McKeown, Generating overview summaries of ongoing email thread discussions, in: *Proc. 20th Int. Conf. on Computational Linguistics*, Geneva, Switzerland, 2004, pp. 549–555.

[40] B. Wang, B. Liu, C. Sun, X. Wang and L. Sun, Extracting Chinese question–answer pairs from online forums, in: *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics*, San Antonio, TX, 2009, pp. 1159–1164.

[41] S. Whittaker, V. Bellotti and J. Gwizdka, Everything through email, in: *Personal Information Management*, W. Jones and J. Teevan, eds, University of Washington Press, Seattle and London, 2007, pp. 167–189 (Chapter 10).

[42] W.-Y. Yang, Y. Cao and C.-Y. Lin, A structural support vector method for extracting contexts and answers of questions from online forums, in: *Proc. Conf. on Empirical Methods in Natural Language Processing*, Singapore, 2009, pp. 514–523.

[43] J.-Y. Yeh and A. Harnly, Email thread reassembly using similarity matching, in: *Proc. 3rd Conf. on Email and Anti-Spam*, Mountain View, CA, 2006.