MyriadHub: Efficiently Scaling Personalized Email Conversations with Valet Crowdsourcing

Nicolas Kokkalis[*1], Chengdiao Fan[*1], Johannes Roith[1], Michael S. Bernstein[1], Scott Klemmer[2]
[1]Stanford University, [2]UC San Diego
{nicolas, chengdiao, jroith, msb}@cs.stanford.edu, srk@ucsd.edu

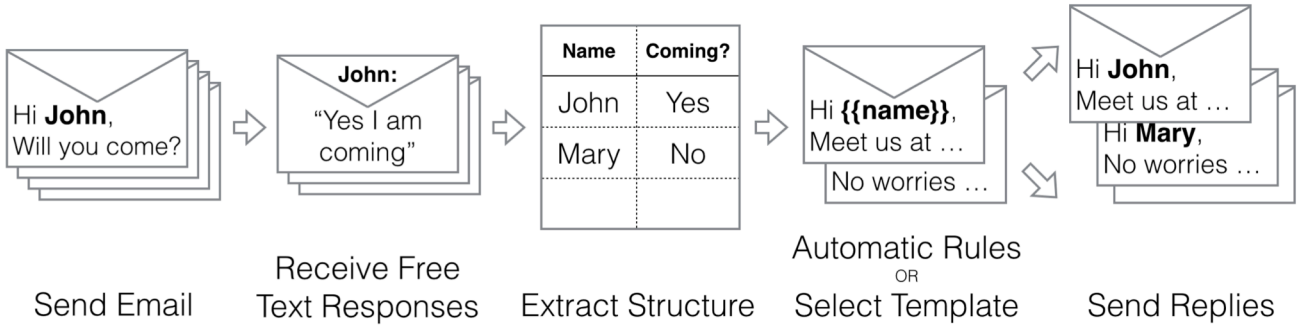

Figure 1. MyriadHub enables users to manage many parallel conversations in a personalized way. To enable reuse, it extracts implicit structure from incoming emails, matches on rules built from previous conversations, then responds.

ABSTRACT
Email has scaled our ability to communicate with large groups, but has not equivalently scaled our ability to listen and respond. For example, emailing many people for feedback requires either impersonal surveys or manual effort to hold many similar conversations. To scale personalized conversations, we introduce techniques that exploit similarities across conversations to recycle relevant parts of previous conversations. These techniques reduce the authoring burden, save senders' time, and maintain recipient engagement through personalized responses. We introduce MyriadHub, a mail client where users start conversations and then crowd workers extract underlying conversational patterns and rules to accelerate responses to future similar emails. In a within-subjects experiment comparing MyriadHub to existing mass email techniques, senders spent significantly less time planning events with MyriadHub. In a second experiment comparing MyriadHub to a standard email survey, MyriadHub doubled the recipients' response rate and tripled the number of words in their responses.

Author Keywords
Mail Merge; Email Overload; Valet Crowdsourcing.

ACM Classification Keywords
H.5.3. Information Interfaces and Presentation (e.g. HCI): Group and Organization interfaces.

CHI 2017, May 06 - 11, 2017, Denver, CO, USA

ACM 978-1-4503-4655-9/17/05…$15.00
DOI: http://dx.doi.org/10.1145/3025453.3025954

* Both authors have equal contribution in this work.

INTRODUCTION
Online communication scales our reach: with email, we can broadcast messages to thousands [29]. Unfortunately, online communication does not equivalently scale our ability to carry on those thousands of conversations [26]. For example, suppose a designer wanted to ask users for feedback on a new beta feature. Today this process means managing hundreds or thousands of parallel conversations in a single inbox [24]. Complex task coordination overloads email and hinders coordination effectiveness [11]. Forms and surveys can handle larger numbers of respondents, but they can feel impersonal and have low response rates [3, 19, 31]. Consequently, communicators are forced to choose between personalized contact with a few, or impersonal broadcasts at scale.

However, many conversational turns share content and structure. For example, many customer service requests share nearly-identical responses and follow-up requests [26]. Even discussion forums contain redundancy [34]. Likewise, event planning, student requests, and user feedback each involve far fewer conversational paths than recipients. If it were possible to structure these replies and conversational paths and re-apply them in new threads, a user might be able to hold individual conversations with far more people than possible today.

This paper introduces *MyriadHub* (Figure 1). MyriadHub exploits cross-email commonality to recycle elements of previous conversations in similar new ones. It uses *valet crowdsourcing* [22] to extract metadata from emails so that previous replies can be recycled when new email threads reach the same state. These metadata-annotated examples enable MyriadHub to generate rules and templates. Users,

valet crowd workers, or algorithms apply these rules to respond quickly to new incoming messages.

For example, suppose a student sends email requesting an assignment extension for medical reasons. A valet crowd worker [22] extracts the reason (*reason: medical*) from the email, and MyriadHub suggests a response from another thread: ask for the doctor's letter. When the student responds with such a letter, the extracted result (*excused: true*) prompts a positive response. Had the student instead asked for an extension with no reason (*reason: unjustified*), the prompted response might instead be a generic "each student gets two free late days."

The MyriadHub system demonstrates this conversational recycling, helping users see and manage massive parallel email conversations. MyriadHub uses valet crowdsourcing [22] to give crowd workers restricted, accountable access to users' inboxes so they can extract metadata from specific threads and enable the system to suggest relevant replies. MyriadHub, as of writing, includes 336 users, 1,107 campaigns created, and 104,946 emails sent. We describe three such campaigns in detail, including startup founders' introduction requests and student internship applications.

This paper presents two experimental studies of MyriadHub's effectiveness. First, to examine its success in managing responses, we performed a within-subjects experiment comparing MyriadHub to traditional mail merge when planning a simulated event. Participants spent significantly less time and made fewer errors with MyriadHub. Second, to verify that MyriadHub can emulate personalized messages, we performed a between-subjects experiment randomizing whether recipients interacted with MyriadHub or an email survey. Recipients assigned to the MyriadHub condition responded at double the rate and with three times as much in-depth content measured by the number of words in their responses.

MYRIADHUB COMBINES SCALE & PERSONALIZATION

MyriadHub sits atop existing web-mail clients and monitors specific threads to create reusable replies and rules. The user can deploy these rules to send contextualized replies when the next conversation reaches the same state.

Example scenario: Asking users for feedback

Christina is a user experience researcher for a mobile video application. To get feedback on a recent feature launch, she imports 500 users of the application, as well as how many times each person has used the feature, into a Google Spreadsheet. Christina opens MyriadHub in her web browser, links a new campaign to the Google Spreadsheet and authors a query in MyriadHub to select users who have used the new feature at least five times. She writes an initial email asking these users for their feedback. In a separate email, she encourages those who have only used the feature once to try it again and give feedback.

When Christina returns from meetings later that day, forty users have responded. Her valet crowd worker for this email campaign has extracted each user's positive and negative feedback, grouped them into themes and tagged each email thread with those themes in MyriadHub *(Structure extraction with valet crowdsourcing)*. Christina creates a few quick histograms based on extracted data stored in her spreadsheet, then writes a search query *(*which the system proactively stores as *a rule)* to identify users with the most prevalent negative feedback: wanting to share a subset of a video rather than the whole recording. She composes a template message to those users pointing them to the advanced option that allows them to do so, and asks them if they have tried it *(Scaling conversation through templates and rules)*. She writes replies similarly for other popular responses. Lastly, she groups the long tail of uncommon responses together into a template reply to share all the other popular feedback and ask which they agree with most. MyriadHub learns rules from these searches and responses that it can apply to future messages.

When Christina returns in the morning, she finds many recipients have responded further, and many others responded to her original message. Christina's crowd assistant has extracted metadata from these new responses, and MyriadHub used that data to trigger rules that it captured from her previous responses and propose replies *(Scaling conversation through templates and rules)*. Christina agrees and sends those messages. Short on time, she then skims a few responses to her most recent reply with no template yet, and replies to those users directly using her usual mail client *(Flexible integration with existing tools)*. Her crowd assistant takes those replies and generalizes them into new rules for similar messages. Through MyriadHub, Christina collected large-scale user feedback with similar benefits to holding many personalized conversations in parallel, yet in much less time. Based on the feedback received and structured, she can discover and start new directions of inquiry.

Personalized email conversations at large scale

This scenario illustrates the creation and reuse of an ad-hoc message repository [1] for large-scale email conversations. With MyriadHub, users can recycle past messages to quickly to author contextually-appropriate responses. Additional example scenarios include course staff fielding students' logistical questions, or event planners handling conversations that come up with client after client.

In summary, users log into MyriadHub with their Google account, and begin creating a MyriadHub campaign by connecting a Google spreadsheet of recipients and drafting the initial message templates. The MyriadHub inbox shows the status of who has replied (Figure 3). The email viewing interface has two properties of note. First, along the right is a metadata extraction pane (Figure 2): the user or valet can

extract field values. Second, when the user replies, below the clipped bottom of Figure 2, they have the option of saving the current reply as a new template. This template gets saved into the conversation tree (Figure 5). From the MyriadHub homepage, the user or valet can click to see a list of templates, and set up rules that will trigger each template response (Figure 4).

Structure extraction with valet crowdsourcing

For MyriadHub to reuse conversations, it needs to have a computer-understandable representation of each conversation. For example, it needs to know whether each person replied with an agreement to serve on the program committee, or the date by which they have said they can decide. In MyriadHub, email structure refers to the metadata contained in the meanings of incoming email responses, such as an answer of "yes" or "no" to the invitation or the reason for asking an assignment extension. Email structure in MyriadHub is not fixed; rather it can be as flexible as the contents of conversations change, and as the information that different senders care about in the responses varies. For example, in the scenario above, the structure is represented by the fields defined in Christina's spreadsheet and values extracted from her user responses. Email structure extraction is possible via machine learning and natural language processing [10], but it can be noisy, and difficult to bootstrap off a small number of initial messages.

MyriadHub uses valet crowdsourcing to flexibly extract semantic themes from messages. People feel a tension recruiting assistants for managing complex, personal information: they want help, but have reasonable concerns about giving others unfettered access to their personal accounts. *Valet crowdsourcing* [22] provides a limited-access window to personal information (such as email), accompanied with an audit trail of every action for accountability. Its principles can be applied to crowdsourcing (crowd of assistants), outsourcing (one or a few expert assistants), or even local employees or co-workers.

Like a valet car key, valet interfaces seek a dual objective of *Privacy* (through parsimonious access control) and *Accountability* (through transparent access boundaries and activity logging). For privacy, a valet interface parsimoniously gives assistants just enough access to help. For accountability, it makes access boundaries transparent so users have an accurate model of what the assistants can and cannot do, and it causes transgressions to leave fingerprints (for instance, by logging valet assistants' actions). MyriadHub logs all of a valet's actions, and limits their access only to messages labeled with the relevant Gmail label; valets do not have broad access. Because people may be unaware of the valet's existence, it is advised that such systems provide enough *Disclosure* to all relevant parties, for example by adding explanatory text to outgoing emails.

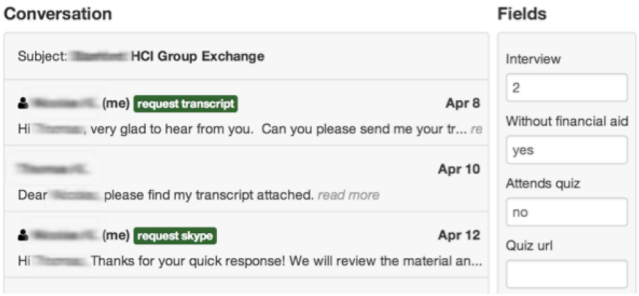

Figure 2. Crowd workers extract fields (right) from the raw thread text (left).

MyriadHub recruits valets from online crowdsourcing marketplaces such as Upwork (www.upwork.com), where workers have experience in relevant domains, such as administrative assistance and data entry. Users pay valets based on their posted rates, often around $10 per hour. A new assistant can be hired for each campaign, or previous ones can be re-hired. MyriadHub users can also recruit valets from within their organization—such as an administrative assistant or TA—or play the valet role themselves.

The valets' primary role is extracting fields and values from each incoming email (Figure 2). Most fields are determined by the user at campaign creation. More fields can be added by a user or valet as needed. Fields might include the recipient's most salient worry about a feature, their favorite and least favorite design alternative, or positive/negative votes for each proposed conference theme.

To facilitate large-scale conversational tracking and data extraction, MyriadHub separates each recipient into a separate thread. By contrast, many email clients (*e.g.*, Gmail) aggregate everyone who responds to the same initial message into one thread. While this thread aggregation makes sense for many common email tasks, it makes following distinct interleaved conversations difficult.

Scaling conversation through templates and rules

With each message's structure extracted, MyriadHub enables the user to draw on previous messages that were sent in similar circumstances (Figure 3). Crowd assistants generalize one-off responses into templates that can be reapplied to future conversations, building the conversational scaffold through demonstration.

When a message's extracted data matches a search query

Name	Your latest message	Status Resend failed 7	Emails	Web	Rails	F
	Skype Convo Soon	You need to read.	15	7	0	y
	Thanks for taking the quiz	You need to reply.	16	0	0	
	Thanks for taking the quiz	You need to reply.	14	4	5	
	Quiz URL	They need to reply.	3			
	request transcript	They need to reply.	1			
	Thanks for taking the quiz	They need to reply.	9	4	0	
	rejection	Finished	4			
	rejection	Finished	7	2	0	

Figure 3. MyriadHub signals recipients' positions in the conversation tree and information extracted

or rule created from a previous message, MyriadHub prepares a reply, inserting values into the template. For example, if a recipient responds to an invitation by accepting the invitation and asking what food to bring, and such a response has been previously sent for the properties [coming: true] and [question: food], MyriadHub can automatically reply with the same response as before or batch the response for the user to customize.

Figure 4 demonstrates rules and associated templates for a campaign. Search queries might filter on a column (e.g., [skype_handle: unknown]), the most recent template sent (e.g., [Last sent message: request transcript]), or status (e.g., [Status = They need to reply]). The user can specify a template to respond with when a rule fires. New matches to a rule who had yet to receive the associated template accrue to a count in a green label in the New Matches column. MyriadHub defaults to asking the user to confirm before responding. Alternatively, the user can opt to allow MyriadHub to auto-respond once the rule matches.

Each message in MyriadHub can be a template and include variables from data extracted using {{variable name}} in the text. Using these variables, a single template message can become many different replies depending on the information extracted from the recipient's previous message.

If MyriadHub has no rule or template for the current message, the user creates one. The user does this through demonstration: replying directly to a message using their normal email client. The crowd assistant generalizes this message by creating field placeholders such as {{name}} and {{paper title}}. They then author a rule for when the template would be applicable (*e.g.*, [agree to review: true]). MyriadHub can also create a new rule whenever the user sends messages to the resulting recipients of a search query. This rule can later be automatically or manually reapplied when new matches appear.

Visualization of Conversation State

Holding multiple conversations at the same time requires tracking the state of each conversation, like playing multiple simultaneous games of chess. To help users with this, MyriadHub visualizes each campaign as a tree (Figure 5). Sibling templates in the tree are alternative paths that conversations took, and child templates indicate responses. Labeled numbers next to each template give the user an overview of how many recipients are in each state. While not all conversations have tree structures and other conversational visualizations, such as funnels and milestones, are possible and can be explored in the future, we chose tree structure for now to make it simple for users to understand.

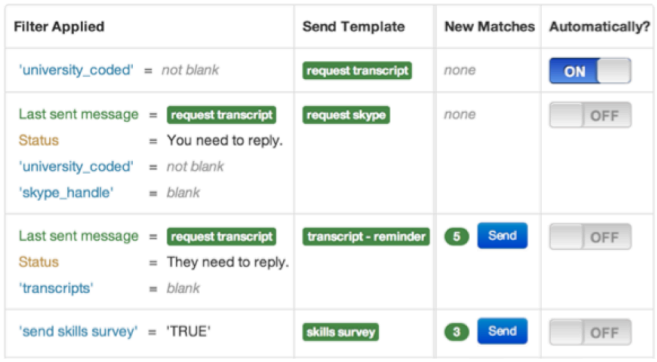

Figure 4: Rules can suggest responses based on the structure extracted from the conversation, then trigger automatically or in batch by the user's request. These four rules are a subset of the twelve used to manage the entire research intern interview campaign.

Instead of a traditional read/unread status, MyriadHub denotes the conversation status of each recipient using four states: "you need to read", "you need to reply", "they need to reply" and "conversation finished" (Figure 3). For example, if the user is running a campaign about an event, she would mark all the recipients who declined to come to the event as finished. For each individual recipient, MyriadHub also shows how many emails got exchanged and the last template they received.

Flexible integration with existing tools

MyriadHub integrates with existing tools through synchronization with the user's Google or Google Apps account. Each MyriadHub campaign appears in the user's Gmail account as a single email thread and is associated with a Google Sheet that stores all extracted metadata and can be downloaded or edited. Rows in the spreadsheet represent recipients and columns contain data about each recipient.

While a user typically begins a campaign, MyriadHub integrates with Gmail's labeling system to enable campaigns that are pushed to the user. For example, a researcher may receive ongoing questions about graduate admissions or internships throughout the year. The researcher can create a campaign for these types of messages, tag incoming email in Gmail with the MyriadHub campaign label to associate the message with the campaign, and let MyriadHub take over.

HOW PEOPLE USE MYRIADHUB

As of this writing, MyriadHub has 336 users, 1,107 campaigns created, and 104,946 emails sent. To illustrate the breadth of conversation recycling and how it helps users communicate at scale, we report on three MyriadHub deployments: coordinating student interviews, administrative requests, and personal introduction requests. These three campaigns were chosen because they are representative of how MyriadHub lowers the threshold and increases the sophistication of end-user programming for communication tasks.

Illustrated examples

Selecting student research interns

Dealing with tens or hundreds of emails about an open position for a student summer internship requires effort in

tracking each applicant's status and repetitive personalized messaging, e.g. soliciting resume and transcripts, interview instructions and scheduling. MyriadHub lowers the threshold to scale straightforward or formulaic email conversations. A researcher used MyriadHub to recruit and manage the interviews for summer intern applicants. He began by emailing an advertisement to a group of students. The resulting conversations produced a total of 226 emails. The researcher created 33 MyriadHub message templates: 19 were used multiple times; 14 were used once to handle exceptional requests.

Figure 5 shows the evolution of the campaign's conversation tree, including the popular paths and one-offs. The cascading feature of the template tree demonstrates the progress of the application/admission process: greeting, request for transcript and resume, rejection, further interviews, and finally offers. These templates were built and used on different subgroups of the campaign recipients and organized through rules created as the campaign developed. The campaign began with each student submitting an application through a Google Form. As students applied over time, MyriadHub used an automatic rule applied to the form spreadsheet to initially greet each student and request their transcript and resume. It also handled reminder emails when participants forgot to respond. Integration with Google Docs was used to administer an online quiz by sending a custom quiz URL to each applicant. The quiz results were uploaded into a new field, then later used to send different messages to students with high and low scores. The researcher had three assistants handle the entire campaign. He only wrote seven of the main emails and the rest was fully handled by the assistants on his behalf.

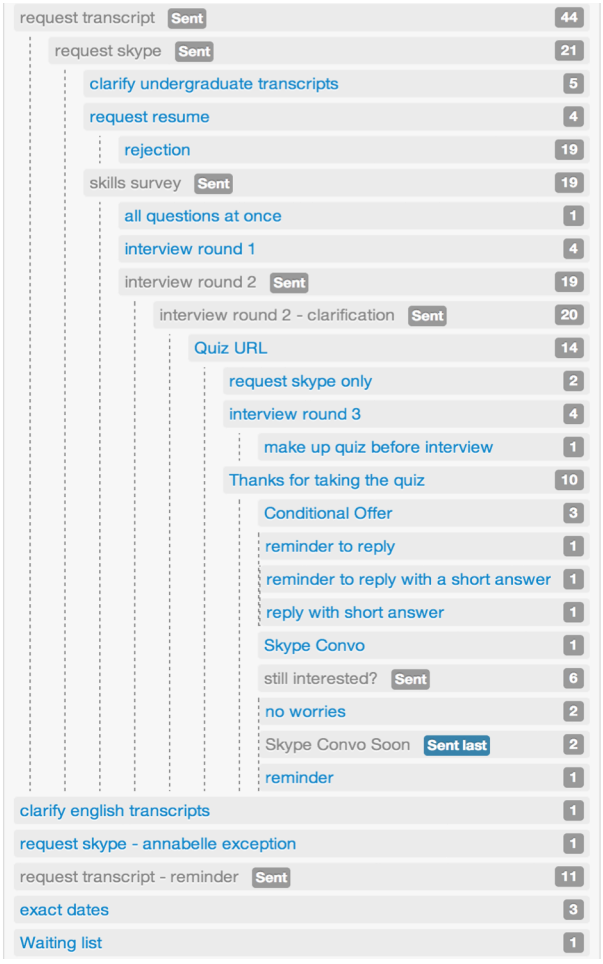

Figure 5. A conversation tree covering over two hundred emails to manage intern interviews, including highly reused templates and branching-out one-offs. Labeled numbers indicate how many recipients received that template most recently.

Figure 4 demonstrates a subset of the rules in this campaign. For example, one rule triggers when the "interview" field contained values "no", "1", "2" and "3", corresponding to rejection, first-round interview, second-round interview and third-round interview. This rule sent templates to inform the recipients of their rejection or instructions for the upcoming rounds of interviews. Likewise, the rule [send skills survey: true] has three new matches who have not received the template "skills survey". The user can send these new matches the appropriate messages by clicking the blue button "Send".

The researcher used rules to deal with the irregular timing of new applicants over a period of weeks. For example, when the bulk of the applicants were already in the second round of interviews, new applicants were still arriving. The user routed new applicants into the correct stage of the interview cycle using the spreadsheet; MyriadHub matched them to a rule to send the stage-appropriate message.

Handling incoming student requests

MyriadHub integrates with general-purpose tools and supports recycling of conversations to handle similar incoming requests in daily email. To help with a class that had heavy student administrative requests, a professor used MyriadHub to manage the responses. Unlike the previous case, where the initiator of the conversation was the user, here the user did not start the conversation: the students initiated each thread, but many asked similar questions. The professor assigned each incoming request a Gmail label that MyriadHub created for the campaign, either manually or through automatic Gmail filters. Some labels led to automatic responses. For example, MyriadHub responded to questions about the participation requirements and office hours automatically with an existing MyriadHub template based on a rule. Others were processed by TAs and then generalized into new templates.

Whenever the teaching assistants had insufficient information to respond, they asked the professor for the information, saved his input in a new message template and reused the updated template later when necessary. The

message templates and campaign structure also served as a starting point for a FAQ for the next run of the class.

Personal introductions

The previous examples show how MyriadHub lowers the threshold to simple conversations; here, we focus on how MyriadHub raises the ceiling by supporting sophisticated email interactions at scale. A director of a Silicon Valley startup accelerator needed to make introductions for the startups in the program to potential investors. This task involved multiparty coordination: the director needs to collect each founder's targeted investors, determine whether the investor and the founder are a good fit, decide whether he will make the introduction, edit the startup's supplied text so that it is relevant to the potential investor, and send a personalized email to the investors with corresponding relevance to make the introduction. With tens of startups and investors, and each connection requiring different messages for the introduction, this is heavy work.

MyriadHub helped this director with collecting the initial lists from different startup founders, editing the data collected, and sending the final personalized introduction to investors. First, he started a MyriadHub campaign to ask the founders to submit the list of targeted investors through a Google spreadsheet. In the spreadsheet, founders were asked to enter their own names and startups, the investor names and basic information, how relevant the investor was to them, and their company descriptions that they wanted to sent to specific investors. The assistant of this director helped him manage this campaign, collect the incoming lists and compile these spreadsheets into a master spreadsheet.

The director then started a new campaign based on this master spreadsheet. In this spreadsheet, he had an overview of which founder asked for which investor, and he could quickly make a decision whether he would introduce these two by entering *yes* or *no* into a new column. In the meantime, he edited the text written by the founders—the relevance between the company and the funder, and the company description—in his own wording.

When decision-making was finished, he used MyriadHub to query only the connections he agreed to introduce. He drafted a generic email template and personalized them via template variables, e.g. the customized relevance and company description per connection. He made the introductions for all founders all at once, and at the same time.

This process repeats three times a year, when the director makes introductions for about 20 startups. Each startup's list of targeted investors ranges from 10 to 50 people. He reported to us, "MyriadHub saves me at least 20 hours each time I have to make these personalized introductions in batches." In this example, MyriadHub supported multiparty coordination that required personalization at scale.

EVALUATION

Does MyriadHub save time?

The previous examples give some indication of the breadth of scenarios that conversational recycling can cover. MyriadHub was designed to help by accelerating common email flows. However, MyriadHub introduces a fixed cost of creating a template, adding variable fields to it, filtering and sending (which creates and applies rules), within an unfamiliar interface. Given these costs, is the benefit worth the trouble, and by how much? In a within-subjects experiment comparing MyriadHub to the Gmail interface, we measured how much time participants saved when handling a sample conversation. We also measured how many errors users made, for example sending the wrong response or forgetting to respond.

Method: Field Experiment

A within-subjects experiment enrolled 12 participants composed of undergraduate and graduate students at a private West Coast American university and young technology professionals. Most had engineering backgrounds. A scenario asked each participant was to organize a potluck with ten invited guests, finding out whether each person could come, and if so, what food they will bring.

In both conditions, there was an initial mail merge sent on the participant's behalf to the ten guests asking them if they would like to come to a potluck. We created a server-side email script that automatically simulated all guests' responses from a pre-labeled set of possible responses. This design ensured that the scenario was identical across participants and conditions. This script responded almost instantly to participants' emails, allowing us to measure participants' active time rather than confound it with time waiting for responses.

Guests who replied that they couldn't come were to be thanked. Guests who replied positively were to be asked what dish they will bring to the potluck. Once guests responded with a dish, they were to be thanked with a message that confirmed the dish. Participants were instructed to always personalize messages with the guests' first name.

In the *Control* condition (N=12), participants used Gmail to respond to all guests. Gmail grouped all responses into a single thread, and would notify users whenever there was a new response to the thread. In the *MyriadHub* condition (N=12), participants used MyriadHub to respond to all guests. In this condition, we also simulated the valet assistant's extraction via a pre-written script, so that we could measure the time that participants spent handling responses—which represents the end user experience—rather than also measuring the assistants' time. This simulated assistant extracted the columns *coming* (yes/no) and *dish* for the participants. So, participants could use all the filtering and template recycling functionalities of MyriadHub directly,

but they had to author all templates and apply rules themselves, just like they would in practice.

We measured the time it took participants to complete their tasks. Specifically, we measured the time between the first invitation and the last email response that the participants sent in each condition.

We asked participants to communicate with a relatively small number of people: only ten, vs. the hundreds or thousands that MyriadHub is designed for. MyriadHub provides time savings that scale as a function of the number of recipients. However, MyriadHub introduces a fixed cost of setting up templates and rules. So, with a small number of recipients, this experiment is a conservative measurement of MyriadHub's benefit. A null hypothesis would suggest that the one-time fixed cost would equal or outweigh the per-conversation benefits with only ten recipients. We hypothesized that, even with a small number of recipients, MyriadHub would result in time savings. If MyriadHub outperforms the Control condition even in this conservative setup, its benefits would be even larger as the number of guests increases, because more redundancy can be reduced using conversational recycling.

Results

MyriadHub saved people time when handling a sample scenario with high redundancy. A pairwise t-test comparing the email processing time of the 12 participants was significant ($t(11)=5.56$, $p<0.001$): participants in the MyriadHub condition spent 32% less time ($\mu=5.75$ mins, std. dev=1.82) to respond to the emails, compared to the Control condition ($\mu=8.5$ mins, std. dev=2.94).

All participants completed the tasks in MyriadHub conditions without any errors, while 7 of 12 participants in the Control condition made mistakes when responding to emails. A two-proportion z-test confirms that the difference is significant ($p<0.05$). Mistakes in the control conditions included (i) sending the wrong response to the wrong scenario (e.g. saying "thanks for coming" to someone who said they are not coming), (ii) forgetting to respond to some guests, and (iii) failing to personalize the message.

Both conditions followed up with a questionnaire to understand participants' experiences and how they spent their time. 6 of 12 participants answered the questionnaire. Control participants reported difficulty tracking guests' status: "[I spent my time] mostly figuring out whether I had already responded to an email or not. It quickly got messy in the long thread… [It's a] very tedious process"; "[I spent my time] checking who I haven't replied to (because the responses don't get grouped together by person [like in MyriadHub]). Realistically, if I were to manage the party like this at this scale, I would create my own spreadsheet to keep track of things, similar to what MyriadHub offers."

Participants' main limiting factor using MyriadHub was creating filters and templates. As one participant reported, "When I found the commands that I needed, the rest was straightforward and fast". MyriadHub clearly had a learning curve: "I think MyriadHub will be better once you get the hang of it." Participants also recognized that templates were not as deeply personalized as individual emails: for example, one simulated guest could not attend due to illness. In the Control condition, many of the participants showed concern for the recipient's health, mentioning something like "hope you feel better soon." Participants tended not to do this when using MyriadHub. With MyriadHub, users either have to handle one-offs in traditional ways or compromise personalization to achieve scale. MyriadHub does not remove the tradeoff between personalization and scale, but it does mitigate this tension. How much it can mitigate depends on how many conversational similarities MyriadHub can exploit and how much redundancy it can reduce.

Limitations

For consistency, the experiment employed simulated responses rather than real conversations. For efficiency, guests' responses arrived immediately. Also, the sample size of this study is relatively small.

Does MyriadHub yield more engagement than surveys?

Myriad's benefits for authors are of little use if recipients don't engage with the messages. To investigate this, a between-subjects experiment examined whether MyriadHub recipients are more likely to respond, and respond at length, than recipients of a typical email survey.

Method: Field Experiment

The second experiment enrolled 172 undergraduate and graduate students, all currently or recently enrolled in a one-unit seminar featuring visiting speakers. Most had engineering backgrounds. The seminar organizer wanted students' feedback on a proposed change to the class requirements that would require each student to participate as a discussant after the main lecture, once during the term.

Participants were randomly divided into two conditions. The stimuli for both conditions were emails sent at the same time, with the same wording except the response method for the instructor's questions. In the *control* condition (N=87), students received an email describing the proposed change and requesting that they fill out a short Google Form. The survey asked three open-ended questions with free text response, a yes/no question asking whether they support the change, and a field to enter their name. Respondents' names were recorded and visible to the instructor in both conditions.

In the *MyriadHub* condition (N=85), MyriadHub sent an email describing the proposed change and then asked the three open-ended questions directly in the email body. Re-

sponses were thematically grouped using MyriadHub and responded to via a template. The instructor responded to students based on the extracted structure from their feedback (*e.g.*, one response for everyone suggesting that they would need to see the paper beforehand). As the campaign developed, the researchers extracted message structure, built templates and rules, and applied them as appropriate.

Both participants in the survey and MyriadHub conditions only received one email if they never acted. The content of these questions and requests were identical, except whether the questions were embedded inside a linked survey or requested as a direct response. So, all differences are due to the requested medium for feedback.

Effectiveness measurements included response rate and response length (word count); cost measures included Upwork assistant pay and time. Response rates indicate how compelling the request was; word count suggests how engaged the participant was in the conversation.

For the MyriadHub condition, response rate is calculated by dividing the number of participants who responded to the initial email by the total number of recipients. Word count is calculated by concatenating the body text of each participant's responses. For the survey condition, the response rate was the number of surveys submitted divided by the total number of recipients. Word count is calculated by concatenating all answers to the form. The cost measurements mainly focused on the cost and time of crowd workers hired in the MyriadHub condition to extract metadata from raw email text. Other than that, in both conditions, the user had to spend about equal amount of time on the initial setup (drafting the survey vs. creating a MyriadHub campaign) with no economic cost. We measured MyriadHub crowd worker costs by hiring two remote workers from Upwork to independently, from scratch, extract values for the predetermined fields. To test accuracy, we compared their results against participants' results.

We used chi-squared tests to compare response rates and t-tests to compare word counts. Each condition had one outlier whose word count was several standard deviations longer than the mean (642 words in MyriadHub; 280 in the control). The analysis presented here, as is common, omits these outliers. Omission did not affect the results' significance. Two researchers jointly examined responses to better understand their relevance, quality and style.

Results

Thirteen of 87 participants in the control condition filled out the form, a response rate of 15% (Figure 6). By contrast, 27 out of 85 participants in the MyriadHub condition responded by email, doubling the response rate to 32%. A chi-squared test confirmed that MyriadHub recipients responded significantly more often ($X^2(171)=5.9$, $p<0.05$). Combining all the form text fields, the average word count in the control condition was 39 words (std. dev = 27), whereas the average word count in the MyriadHub condition's emails was three times that—114 words (std. dev.= 69). A t-test comparing the response lengths of the 38 respondents was significant ($t(36)=3.57$, $p < 0.05$): participants in the MyriadHub condition were not only more likely to respond, they also wrote longer responses. This length comparison is notable in that some respondents never got to the second and third questions in the script. We used response rates here not to show a predictable result already established by previous studies [19]: more personalized messages receive higher response rates. Rather, we emphasize that MyriadHub replicates these benefits with less effort than prior work.

Text content, quality and styles also diverged between two conditions. When reacting to the proposed addition to the course requirements, survey participants all replied "no" via a radio button located in the middle of other open-ended questions. The free-text answers generally focused on objections, were negative and blunt in style. For example, "this idea would deter future students from enrolling and they would instead enroll in other seminars"; "I would probably drop the class"; "please don't do this!" In contrast, MyriadHub responses were more positive, constructive and creative. Some MyriadHub respondents liked the proposal. Some expressed their concerns but also supplied specific suggestions on how to modify the proposal to make it more effective. For example, "adding the discussant can be very good for the learning experience in this class. However, I have two pieces of suggestions regarding to the specific implementation of the discussant…." Many of them were even open for further discussions after writing a long message, such as "These are just some initial thoughts, and I'd be happy to discuss them in more detail if you'd like." While it is possible that response bias would positively inflect participants' responses, participants in the MyriadHub condition engaged more with the idea.

In terms of cost, the first Upwork worker charged $11.11/hour, and spent 100 minutes on completing the data extraction for all email threads, for a total of $18.52. The second worker charged $10.00/hour, and spent 60 minutes on the same process, totaling $10.00. These results correspond to $0.37 and $0.69 per thread, typically involving at

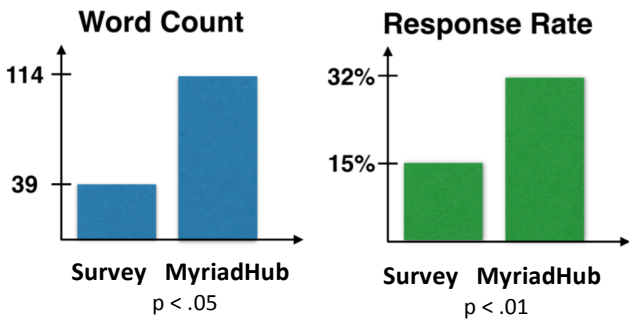

Figure 6. MyriadHub recipients were twice as likely to respond, and responded at three times the length.

least two responses per thread. The first worker agreed with the researchers' manual extractions in 96% of the cases and the second worker agreed 97% of the time. From these results, we suspect that MyriadHub's benefits can be attained using on-demand part time crowd assistants.

Limitations

Surveys are not the only way one can scale feedback solicitation. Custom websites, mailing lists, mobile apps for example could achieve some of that. The study does not prove that personalization is the only cause of the increased response rate and word count; there could be other factors, too. Word count only indirectly measures the depth of response contents. But, we believe that it captures the construct to a first approximation, and a qualitative analysis helps triangulate the effect.

DISCUSSION

Table 1 compares the relative merits of MyriadHub's techniques, surveys and email. In sum, our evaluations suggest that MyriadHub scales online communication without deeply compromising personalization, and increases response rate with a small amount of valet assistance.

The time it takes users to handle email responses with traditional email clients increases linearly with the number, N, of *total* messages they send. Using MyriadHub, the time is a function with complexity dominated by how many *unique* messages the user writes. This is a sub-linear function f(N) that depends on the redundancy that exists in the email responses (*e.g.* could be log(N)). In other words, MyriadHub is O(f(N)) while Gmail is O(N), where f(N) is a sub-linear function. This means that the higher the redundancy, the more time MyriadHub saves its users. In the first experiment, potluck planning with 10 guests, this translated to a 32% improvement. This experiment had high communication redundancy, but also few recipients.

MyriadHub's conversational recycling doubled the response rate and tripled the text written by correspondents. The personal conversation also changed the tone of the discussion: students in the MyriadHub condition were more positive about the proposal to add work to the seminar course than those who filled out a survey. The perceived level of personal touch is the most likely mechanism: MyriadHub emails were originally written by the end user and customized to each circumstance. Mail merge would have only allowed the initial email to be personalized—making follow-ups difficult. However, if it becomes obvious that the conversation is automated, the honest signal [12] would disappear, and this effect might lessen.

	Email	Surveys	Myriad
Communication Scale	↓	↑	↑
Personalization	↑	↓	↑
Response Rate	↑	↓	↑

Table 1. Custom email is highly personalized and attracts higher response rates but is difficult to scale. Surveys are easy to scale but are impersonal and suffer from low responses rates. MyriadHub combines these merits.

Does conversational recycling lie to the recipient about the level of personal attention they are getting? MyriadHub messages often look like they were authored only for the recipient, and this may well contribute to the system's benefits. However, triggering a MyriadHub response rule without tweaking the response can produce an uncanny valley of personalization, where the message looks like it is from a person but is triggered in confusing contexts. MyriadHub offers a middle ground, holding each message as a pending response so the user can tweak it to avoid any potential missteps. This approach retains many benefits of recycling while minimizing the risk of a mistake.

As with many collaborative systems, exceptions are common. Without a mechanism to handle unanticipated situations, users might abandon the system [14]. These unusual circumstances are not actually all that unusual: 14 of 33 templates in the internship interview scenario were used only once (Figure 5).

For MyriadHub to scale, it can't require individually authoring a response. For common queries and responses, the valet/rule can respond with a message from the library. The best way we have observed for MyriadHub users to engage with 'long-tail' responses is through redirection, rather than direct engagement. This is a similar parlor trick to how the 1960s AI psychoanalysis program ELIZA worked: respond to a question with a question. For example, when students provided off-script feedback on the class proposal, the user avoided answering directly by listing other common feedback and asking if any of those ideas resonated.

Conversational remixing layers a lightweight structure onto email conversation, making it useful even for ad-hoc needs. Flexible use of existing tools can beat rigid predesigned systems, especially as rapid prototyping testbeds for information needs. The goal is to ease a transition to custom-built software if the campaign remains popular and the scripts ossify. For example, people often escape structured personal information management tools and instead use *todo.txt* files on their desktop [8], non-profits create their own ad-hoc "homebrew databases" [30], and internet users author lightweight collections of items ranging from cereal boxes to historical timelines [7]. With academic paper reviewing, systems such as EasyChair, Precision Conference, and ManuscriptCentral formalize some parts of the conversation. However, what's formalized is relatively limited – reviewers still do a lot of semi-repetitive work. Furthermore, changing this process is much more difficult than in MyriadHub. Because academic reviewing software is bespoke, making even a small change often requires significant software development.

Future work will focus on supporting more natural responses at larger scales. MyriadHub has no straightforward way to manage emails that trigger components of multiple previous responses. One solution would be to enable a response to inherit from multiple response templates, like a *mix-in* in object-oriented programming. In addition, large campaigns such as the interview deployment can become unwieldy. More automation (*e.g.*, interactive machine learning) could support quicker decisions. In addition, parallelizing multiple crowd assistants might allow individuals to specialize in sub-parts of the campaign structure.

RELATED WORK

Email senders who need to reach a large audience have to choose among impersonal mass messages (*e.g.*, surveys [19, 29] or mailing lists [35]), personalized conversations with only a manageable subset of their audience [26, 3] or intractable, costly parallel conversations with everyone [11, 24]. At the same time, impersonal mass messages make it worse [26] for already overloaded email recipients [13, 15, 33] and suffer from low response rates [3, 31].

A major risk of scaling email communication is overload. Sources of email overload include the volume, email organization, and email handling strategies [13, 11, 32, 33]. MyriadHub mitigates email overload by improving email organization and handling strategies. It threads by recipient instead of by conversation, structures email content and reuses templates and rules with a small amount help from crowd workers, to compensate for the large volume of incoming responses in scaling.

Valet crowd assistants can help users extract structure from users' inboxes as in EmailValet [22]. MyriadHub extends its valet crowdsourcing philosophy. While EmailValet extract the content of choice (tasks embedded in users' inbound emails), MyriadHub adopts the "Programming by a Sample" model [18], in which crowd workers extract metadata from inbound emails and create templates based on users' outbound email examples in the same process.

Structure extraction and summarization are important components of MyriadHub. Crowds can aid with categorization and clustering of qualitative text-based data [2] or extracting tasks from email [22]. Alternatively, machine learning can help summarize messages [28] or forward and reply to messages [19]. MyriadHub demonstrates how thread-specific structure extraction can enable more effective communication: these tasks may be possible for microtask crowds or machine learning in the future.

Commercially available mass email systems (for example MailChimp.com and iContact.com) generally focus on personalizing the *initial* outgoing email of a campaign, optionally segmenting recipients based on their properties. Their segmentation structure is usually a fixed data input into campaigns, *e.g.* types of customers, rather than flexible structures created as conversations grow in MyriadHub. MyriadHub generalizes this concept to support responses into an entire conversation through recycling relevant conversational parts, as a form of programming by demonstration. This is valuable because targeted, personal emails produce higher engagement [5] and higher perceived personal attention [4, 20].

Customer support software (such as zendesk.com, desk.com, and kayako.com) focuses on building collective reusable knowledge [9] and applying it to specific issues [16]. Publishing past conversations on sites such as on StackOverflow or Quora can transfer one-off knowledge into public memory [23, 25]. However, these organizational memory systems [1] work when the conversation has no conditional branching and the only follow-ups are for clarification [21, 24, 28].

CONCLUSION

Email extends the number of people we can reach, but not our conversational bandwidth. To combine the personal touch of one-on-one email with the large scale of email surveys, we explore MyriadHub: extracting the structure of conversations using valet crowdsourcing and re-applying previous responses when a similar situation recurs with a new conversational partner. MyriadHub allows users to hold conversations with large numbers of people by adapting previous responses to new situations. Holding conversations as usual can bootstrap conversation recycling: crowds generalize each thread and then only ask the user to contribute when a new branch of the conversational tree has been encountered.

Today's narrative around systems research in communication tools focuses on enhancing users' personal information management techniques. This framing highlights email overload [15, 33], email-based tasks [6, 22], and mining communication patterns [17, 27]. Instead, tools might make us better communicators. Could they help us say what we want to say more effectively and with fewer misunderstandings? Could they help us understand when to respond? MyriadHub helps amplify our voice. Future systems might help us improve that voice as well.

ACKNOWLEDGMENTS

We thank Ludwig Schubert for his work on an early draft of this paper, Valentin Costet for his technical contributions and Anika Raghuvanshi for her contributions to the video figure. This work was supported by National Science Foundation award IIS-1351131.

REFERENCES

1. Mark Steven Ackerman and Thomas W. Malone. 1990. Answer Garden: A tool for growing organizational memory. In *Proceedings of the ACM SIGOIS and IEEE CS TC-OA conference on Office information systems* (COCS '90).

2. Paul André, Aniket Kittur and Steven P. Dow. 2014. Crowd synthesis: Extracting categories and clusters from complex data. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing* (CSCW '14). ACM New York, NY, USA. 989-998.
3. William G. Axinn, Thomas E. Fricke and Arland Thornton. 1991. The Microdemographic Community-Study Approach Improving Survey Data by Integrating the Ethnographic Method. In *Sociological Methods & Research* 20.2 (1991): 187-217.
4. Greg Barron and Eldad Yechiam. 2002. Private e-mail requests and the diffusion of responsibility. In *Computers in Human Behavior* 18.5 (2002): 507-520.
5. Gerard Beenen, Kimberly Ling, Xiaoquin Wang, Klarissa Chang and Dan Frankowski. 2004. Using social psychology to motivate contributions to online communities. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work* (CSCW '04). ACM New York, NY, USA, 212-221.
6. Victoria Bellotti, Nicolas Ducheneaut, Mark Howard and Ian Smith. 2003. Taking email to task: the design and evaluation of a task management centered email tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '03). ACM New York, NY, USA, 345-352.
7. Edward Benson and David R. Karger. 2014. End-users publishing structured information on the web: an observational study of what, why, and how. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM New York, NY, USA, 1265-1274.
8. Michael Bernstein, Max van Kleek, David Karger and MC Schraefel. 2008. Information scraps: How and why information eludes our personal information management tools. In *ACM Transactions on Information Systems* (TOIS) 26.4 (2008): 24.
9. John H. Boose. 1990. Knowledge acquisition tools, methods and mediating representations. In *Knowledge Acquisition for Knowledge-Based Systems* (Motoda, H. et al., Eds), IOS Press, Tokyo (1990): 123.
10. Simon H. Corston-Oliver, Eric Ringger, Michael Gamon and Richard Campbell. 2004. Integration of Email and Task Lists. In Proc. CEAS.
11. Laura A. Dabbish and Robert E. Kraut. 2006. Email overload at work: an analysis of factors associated with email strain. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work* (CSCW '06). ACM New York, NY, USA, 431-440.
12. Judith Donath. 2007. Signals in social supernets. In *JCMC* 13.1 (2007): 231-251.
13. Danyel Fisher and A.J. Brush. 2006. Revisiting Whittaker & Sidner's email overload ten years later. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative* work (CSCW '06). ACM New York, NY, USA, 309-312.
14. Fernando Flores, Michael Graves, Brad Hartfield, and Terry Winograd. 1988. Computer systems and the design of organizational interaction. In *ACM Transactions on Information Systems* (TOIS) 6.2 (1988):153-172.
15. Catherine Grevet, David Choi, Debra Kumar and Eric Gilbert. 2014. Overload is overloaded: email in the age of Gmail. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM New York, NY, USA, 793-802.
16. Christine A. Halverson, Thomas Erickson and Mark S. Ackerman. 2004. Behind the help desk: evolution of a knowledge management system in a large organization. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work* (CSCW '04). ACM New York, NY, USA, 304-313.
17. Sudheendra Hangal, Monica S. Lam, and Jeffrey Heer. 2011. Muse: Reviving memories using email archives. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (UIST '11). ACM New York, NY, USA, 75-84.
18. Björn Hartmann, Leslie Wu, Kevin Collins and Scott R. Klemmer. 2007. Programming by a sample: rapidly creating web applications with d. mix. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM New York, NY, USA, 241-250.
19. Dirk Heerwegh, Tim Vanhove, Koen Matthijs and Geert Loosveldt. 2005. The effect of personalization on response rates and data quality in web surveys. In *International Journal of Social Research Methodology* 8.2..
20. Adam N. Joinson and Ulf-Dietrich Reips. 2007. Personalized salutation, power of sender and response rates to Web-based surveys. In *Computers in Human Behavior* 23.3 (2007): 1372-1383.
21. William Jones. 2007. Personal information management. In *ASIST* 41.1 (2007): 453-504.
22. Nicolas Kokkalis, Thomas Köhn, Carl Pfeiffer, Dima Chornyi, Michael S. Bernstein and Scott R. Klemmer. 2013. EmailValet: Managing email overload through private, accountable crowdsourcing. In *Proceedings of the 2013 conference on Computer supported cooperative work* (CSCW '13). ACM New York, NY, USA, 1291-1300.
23. Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak and Björn Hartmann. 2011. Design lessons from the fastest Q&A site in the west. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM New York, NY, USA, 2857-2866.

24. Donna McAuliffe. 2003. Challenging methodological traditions: Research by email. In *The Qualitative Report* 8.1 (2003): 57-69.
25. Kevin Kyung Nam, Mark S. Ackerman, and Lada A. Adamic. 2009. Questions in, knowledge in?: a study of naver's question answering community. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM New York, NY, USA, 779-788.
26. Albert Oosterhof. 2000. Creating individualized e-mail to students. In *Journal of Computing in Higher Education* 11.2 (2000): 75-90.
27. Adam Perer, Ben Shneiderman, and Douglas W. Oard. 2006. Using rhythms of relationships to understand e-mail archives. In JASIST 57.14 (2006): 1936-1948.
28. Owen Rambow, Lokesh Shrestha, John Chen, and Chirsty Lauridsen. 2004. Summarizing email threads. In *Proc. NAACL*.
29. Kim Sheehan, and Sally J. McMillan. 1999. Response variation in e-mail surveys: An exploration. In *Journal of Advertising Research*.
30. Amy Voida, Ellie Harmon and Ban Al-Ani. 2011. Homebrew databases: Complexities of everyday information management in nonprofit organizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM New York, NY, USA, 915-924.
31. Herbert F. Weisberg, Jon A. Krosnick, and Bruce D. Bowen. 1996. *An introduction to survey research, polling, and data analysis*. Sage.
32. Steve Whittaker, and Julia Hirschberg. 2001. The character, value, and management of personal paper archives. In TOCHI 8.2 (2001): 150-170.
33. Steve Whittaker and Candace Sidner. 1996. Email overload: exploring personal information management of email. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '96). ACM New York, NY, USA, 276-283.
34. Amy X. Zhang, Lea Verou and David Karger. 2017. Wikum: Bridging Discussion Forums and Wikis Using Recursive Summarization. In Proc. CSCW. ACM New York, NY, USA.
35. Amy X. Zhang, Mark S. Ackerman, and David R. Karger. 2015. Mailing Lists: Why Are They Still Here, What's Wrong With Them, and How Can We Fix Them?. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM New York, NY, USA, 4009-4018.