WearMail: On-the-Go Access to Information in Your Email with a Privacy-Preserving Human Computation Workflow

Saiganesh Swaminathan[1] Raymond Fok[2] Fanglin Chen[1] Ting-Hao (Kenneth) Huang[1]
Irene Lin[1] Rohan Jadvani[1] Walter S. Lasecki[2] Jeffrey P. Bigham[1]

[1] Carnegie Mellon University
Human-Computer Interaction Institute
{saiganes, fanglinc, tinghaoh, iwl, rjadvani, jbigham }@cs.cmu.edu

[2] University of Michigan
Computer Science and Engineering
{rayfok, wlasecki}@umich.edu

ABSTRACT

Email is more than just a communication medium. Email serves as an external memory for people—it contains our reservation numbers, meeting details, phone numbers, and more. Often, people need access to this information while on the go, which is cumbersome from mobile devices with limited I/O bandwidth. In this paper, we introduce *WearMail*, a conversational interface to retrieve specific information in email. WearMail is mostly automated but is made robust to information extraction tasks via a novel privacy-preserving human computation workflow. In WearMail, crowdworkers never have direct access to emails, but rather *(i)* generate an email filter to help the system find messages that may contain the desired information, and *(ii)* generate examples of the requested information that are then used to create custom, low-level information extractors that run automatically within the set of filtered emails. We explore the impact of varying levels of obfuscation on result quality, demonstrating that workers are able to deal with highly-obfuscated information nearly as well as with the original. WearMail introduces general mechanisms that let the crowd search and select private data without having direct access to the data itself.

Author Keywords

Email; Crowdsourcing; Human Computation; Privacy; Mobile; Information Extraction

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION

Our email increasingly serves as an extension of our own finite memory, making it a repository where much of our personal information resides. Unfortunately, email search is known to be hard [10]. It lacks the network structure that has been leveraged to improve areas like Web search [4], and current automated systems are not well equipped to address problems, such as email search via a user's natural language query. Colloquial references and lack of context render typical natural language processing on an isolated user query near impossible. Privacy is another concern when considering email information retrieval, and people are wary of search methods that are too intrusive. While there do exist a few large corpora of email, email use remains very personal, and the average user's email in 2017 is very different than that of, say, Enron employees in 2004 [19, 21]. One example reason for this difference is that a large portion of current email usage is now done on mobile devices [33].

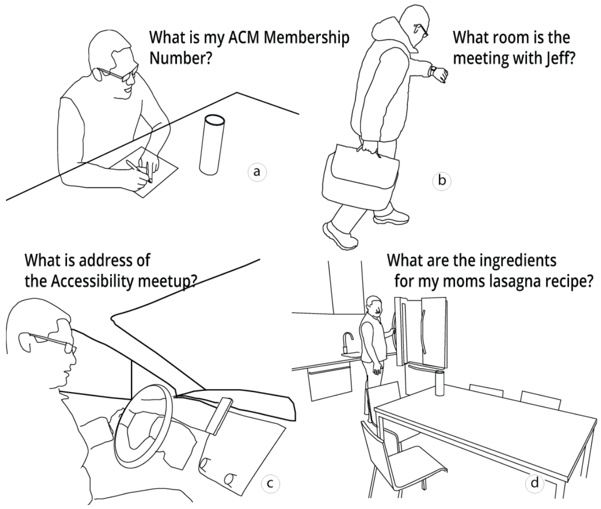

What is my ACM Membership Number?
a
What room is the meeting with Jeff?
b
What is address of the Accessibility meetup?
c
What are the ingredients for my moms lasagna recipe?
d

Figure 1. The answers to information requests that users have on-the-go can often be found in the user's email. WearMail presents a privacy-preserving human computation workflow to extract this information from the user's email based on a spoken query.

In this paper, we introduce WearMail, a system that extracts information from email via dialogue. WearMail is mostly automated but uses human computation to adapt to new kinds of information extraction tasks, which allows it to be more robust to the long tail of possible queries than prior systems.

To accomplish this, WearMail introduces a new human computation workflow that protects privacy by *(i)* having crowd

UIST '17, October 22–25, 2017, Quebec City, Canada
© 2017 ACM. ISBN 978-1-4503-4981-9/17/10...$15.00
DOI: https://doi.org/10.1145/3126594.3126603

workers interact only with heavily-obfuscated email metadata, and *(ii)* generating regular expressions by example to extract information from a sensitive dataset. The crowd leverages large public corpora (*e.g.*, via web search) to help construct sets of examples that can be used by programming by demonstration tools to create custom extractors. This generalized approach can help mitigate end user privacy concerns, a major hurdle of crowdsourcing systems [25, 28].

This paper makes the following four contributions:

- **Study of Mobile Email Search Behavior:** a study with 200 users that characterizes how people search email from mobile devices, providing evidence that commonly search for specific information while on-the-go;
- **WearMail:** a system for reliably extracting information from email by leveraging a generalizeable, privacy-preserving human computation workflow;
- **Study of Metadata Obfuscation:** a study that demonstrates crowd workers can effectively find emails likely to contain correct answer to users' queries under varying levels of information obfuscation;
- **Study of WearMail Performance:** a study that characterizes the performance of WearMail on 30 information extraction queries, and demonstrates its potential to help users find information they need within their email.

RELATED WORK

WearMail builds upon an extensive prior literature on systems designed to improve access to email. The use case that it targets is partially motivated by prior studies of how people use email, and is complemented by our study on mobile email search behavior. The technical approach that WearMail uses to extract information from email based on a user's conversational query builds upon prior work in *(i)* email use and search *(ii)* question answering and information extraction, and *(iii)* privacy-sensitive crowdsourcing workflows.

Email Use and Search

Email has been one of the primary uses of networked computing devices [9], and social scientists have long examined how people use it. Sproull *et al.* [35] provide a summary of early works on the social and organizational aspects of email. Later work is dedicated to developing theories [6, 38] and systems (*e.g.*, prioritizing emails [11], and filtering junk emails [34]) to alleviate the problem of email overload. Recently, Kokkalis *et al.* [20] introduced EmailValet to tackle email overload by asking crowd workers to help convert emails into tasks on a "TODO" list. In EmailValet, privacy is preserved by asking users to specify rules that expose only a specific subset of their emails to the crowd. In contrast, WearMail *never* directly exposes the contents of emails to the crowd workers.

Although numerous systems have been designed to facilitate the search experience, studies [36] have shown users oftentimes cannot specify their information needs, and need to take multiple steps to get their desired items. WearMail offloads searching to the crowd in order to meet information retrieval needs through a wearable email interface.

Question Answering and Information Extraction

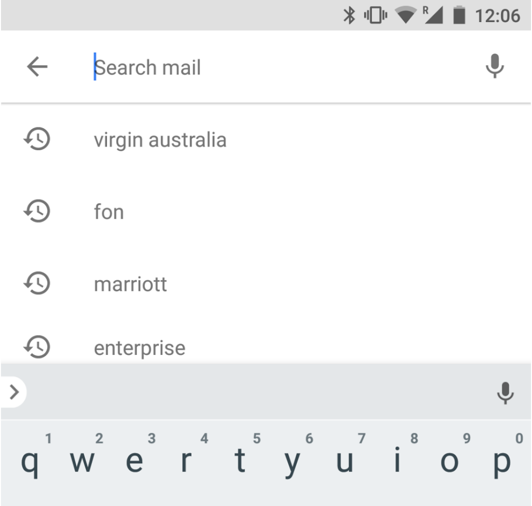
Researchers have previously used crowdsourcing to validate the correctness of regular expressions, but had the crowd do so by viewing strings generated from the regular expressions rather than the regular expressions themselves [5]. Researchers have used the crowd to perform more thorough searches without time limits [37], to extract structured information [2, 29], and to improve answers provided from a social network [14]. Parameswaran *et al.* [32] used the crowd to perform query expansion and refinement by providing an interface for viewing query results. In contrast, we consider cases where query results may be sensitive. Kim *et al.* [17] also explored using crowdsourcing to perform query expansion. Demartini *et al.* [7] used crowdsourcing within a semi-automatic pipeline that generated structured queries, but only in order to validate tagging of natural language queries and identifying relationships between entities in those queries. One method for protecting user privacy that has been proposed is to build privacy preservation directly into the workflow [16]; WearMail expands on this prior work with new privacy-preserving mechanisms for use with textual content when structured meta data is available.

Privacy-Sensitive Workflows

Researchers have developed algorithms for privacy-preserving information retrieval [8]. These works primarily focus on methods for preventing or making unlikely the analytic recovery of information that has been aggregated or otherwise made anonymous. On the other hand, literature had little to say about privacy-sensitive human computation workflows, despite privacy being one of the reasons that crowd-powered systems are not used [18]. One system to protect sensitive information is CrowdMask, which iteratively shows workers progressively comprehensive versions of content so that they can recognize potentially-sensitive information before it is fully understandable in context, thus reducing the risk of revealing private or sensitive information [16]. Glance [24], Legion-AR [27], and Zensors [22] all ask the crowd to examine video that may contain sensitive information in order to identify events/behaviors. Prior work has explored how obfuscation can be traded off against recognition accuracy in these video-based tasks, and presented tools for requesters to tune system to their own domain and risk tolerance [25]. WearMail obfuscates the inputs that the workers use to identify the "filterset", and extractions occur without workers seeing personal information.

Interactive Crowd-Powered Systems

Prior work that has shown crowds can be leveraged on demand [3] and continuously to power interactive applications [26, 12]. Crowd-powered systems are able to have conversations with users to answer their questions [23]. WearMail demonstrates the feasibility of having crowds engage in a workflow for information finding that preserves privacy by using the crowd to perform just-in-time training of a system,



Search mail

virgin australia

fon

marriott

enterprise

Figure 2. Participants in our study of mobile email search were asked to submit a screenshot of their mobile search history, and to describe the context around their search for three of the queries.

instead of providing direct responses to a query. While beyond the scope of this work, we believe our approach could be integrated into interactive systems in the future.

CHARACTERIZING MOBILE EMAIL SEARCH

While email use has been studied extensively in prior work [9, 33], relatively little work has specifically studied the goals of mobile email use. To better understand what WearMail should support, we first conducted a preliminary study seeking to better understand: “What do people search for on-the-go in their email?” We recruited 200 participants on Amazon’s Mechanical Turk platform, and restricted to those who use the Gmail email application on an Android phone. The latter requirement was included because the iPhone Gmail search history displays both mobile and desktop searches.

Participants were asked to take a screenshot of their recent searches (Figure 2) from their email search box on the mobile app and upload the image. Then, participants were asked to explain what they searched for and why, and try to recall where they were and what they were doing. Each participant provided 2-4 queries. Participants were required to sign a consent form before completing the survey and had the choice to stop participating in the survey if they felt that their search history held overly private or sensitive information.

Of the 200 responses, we discarded invalid responses, such as ones reporting general Google searches and other searches that didn’t result in exact answers. From the 608 queries collected, 253 queries were valid. In addition to search histories, we also collected demographic information about our participants. The results of the study are categorized in Table 1.

Most searches emerged from the long tail of private information requests by the users on-the-go. Below, we provide a few examples of the real-world information requests collected for each of these types:

- **Order**: orders from Amazon, Ebay, FedEx, Target, etc.
- **Code**: coupons for Costco, Bento, Pizza Hut, Embry, etc.
- **Link**: URLs to blogs, games, password reset links, etc.
- **Event**: dinner details, due dates, appointment details, etc.
- **Contact Info**: details for Asus, Big Sunny, Ventura, etc.
- **Account**: user/pw to Chase, PokerStars, Skype, ESPN, etc.
- **Finance**: Paypal transaction, MTurk, Billing dates, etc.
- **Name**: author, tax consultant, soccer camp org., etc.

This study shows that over 30% of the valid 253 queries were made while the user was “in-transit,” highlighting the frequency of on-the-go scenarios. Overall, 85% of queries were made by users between ages 20-40. The results are summarized in Table 2.

WEARMAIL

WearMail consists of multiple crowd interface components connected with a chatbot that runs on a user’s smartwatch (Figure 3). Users can ask questions on-the-go using the built-in speech recognition software available on the smartwatch. Once the chatbot receives the user’s query, it processes the query and forwards the appropriate tasks to the crowd.

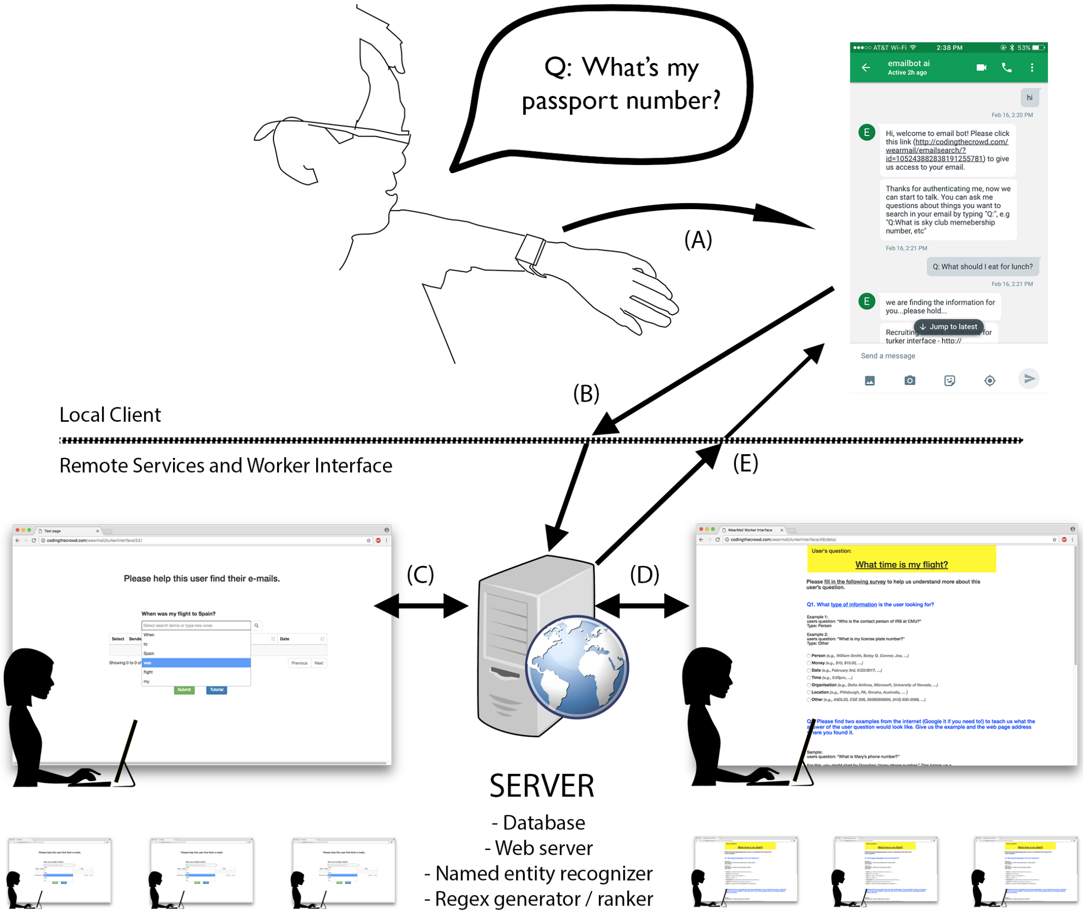
The WearMail privacy-preserving human computation workflow for extracting information consists of two main steps:

- **Step 1 – Email Filtering:** In the email filtering interface, crowdworkers are asked to select—through obfuscated metadata only—relevant emails that pertain to the user’s query. After three workers finish selecting a set of emails, the intersection of these emails are then stored as a *filterset* for use in the extraction step.
- **Step 2 – Information Extraction:** After creating the filterset, crowdworkers provide examples of the information searched for by the query, and select the data type of the question being asked. For example, when the user asks a question “What is my passport number?”, crowdworkers search the web to find examples of passport numbers (e.g., J8369854, D84648344). We then use the examples to generate a robust extractor using regular expressions and an existing named entity recognizer (NER). This extractor is run against the email filterset to produce a result for the

Type	Percentage
Order	16.86
Code	13.03
Link	11.88
Event	11.11
Contact Info	9.58
Account	9.2
Binary Queries[1]	8.43
Finance	7.66
Travel	4.21
Name	3.45
Other	2.68
Location	1.92

Table 1. Types and percentages of user queries. A majority of queries target specific information located in the user’s emails.

[1] Binary Queries are user queries that have has a “yes“ or “no“ answer (*e.g.*, “Is my Delta membership confirmed?”)



hi

Feb 16, 2:20 PM

Hi, welcome to email bot! Please click this link (http://codingthecrowd.com/wearmail/emailsearch/?id=105243882838191255781) to give us access to your email.

Thanks for authenticating me, now we can start to talk. You can ask me questions about things you want to search in your email by typing "Q:", e.g "Q:What is sky club memebership number, etc"

Feb 16, 2:21 PM

Q: What should I eat for lunch?

Feb 16, 2:21 PM

we are finding the information for you...please hold...

Recruiting turkers...click here for turker interface - http://

codingthecrowd.com/wearmail/turkerinterface/52/

Please help this user find their e-mails.

When was my flight to Spain?

Select search terms or type new ones

When
to
Spain
was
flight
my

Select | Sende | Date

Showing 0 to 0 of

codingthecrowd.com/wearmail/turkerinterface/48/data/

User's question:

What time is my flight?

Please fill in the following survey to help us understand more about this user's question.

Q1. What type of information is the user looking for?

Example 1:
users question: "Who is the contact person of IRB at CMU?"
Type: Person

Example 2:
users question: "What is my license plate number?"
Type: Other

○ Person (e.g., William Smith, Betsy Q. Conner, Joe, ...)
○ Money (e.g., $10, $15.02, ...)
○ Date (e.g., February 3rd, 5/23/2017, ...)
○ Time (e.g., 5:23pm, ...)
○ Organisation (e.g., Delta Airlines, Microsoft, University of Nevada, ...)
○ Location (e.g., Pittsburgh, PA, Omaha, Australia, ...]
○ Other (e.g., A9DL23, CSE 209, 28390208920, (412) 830-2099, ...)

Q2. Please find two examples from the internet (Google it if you need to!) to teach us what the answer of the user question would look like. Give us the example and the web page address where you found it.

Sample:
users question: "What is Mary's phone number?"

Figure 3. WearMail accepts a user's verbal query (for instance, from a smartwatch) and securely extracts the queried information from the user's email. WearMail is partially automated, but crowdworkers help make it more robust to new types of requests. Workers identify a filterset of emails likely to contain the information requested by examining obfuscated email metadata. Rather than directly extracting information from email content, workers choose between an existing Named Entity Recognizer or submit examples that are used to build information extractors that are then run automatically.

Location	Percentage
Home	51.82
In transit	30.03
Work	10.56
Other	3.96
Waiting	2.15
Store/Restaurant	1.49

Age	Percentage
< 20	2.19
20-30	48.63
30-40	38.25
40-50	9.29
50-60	1.64

Table 2. Distribution of locations where users asked their queries, and age demographics. 30% of participants were "in-transit" while querying, and a majority of users were in the 20-40 age group.

user. This approach is similar to prior work on programming by example [30]. Two interesting differences are that the examples are provided by a third party other than the user (the crowd), and the examples are drawn from a public corpus (the Web) with the intent to run the resulting program on a private corpus (email).

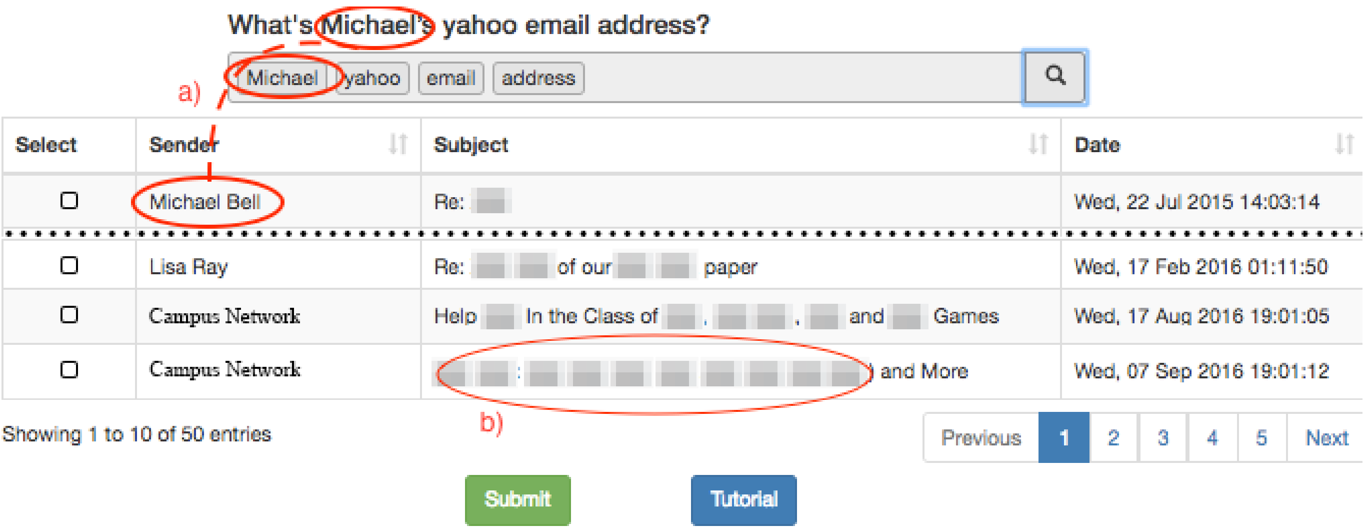
Implementation

WearMail communicates with the user via Google Hangouts, which allows us to leverage existing applications on various mobile and wearable platforms. After sending a question to the chatbot through Google Hangouts, Human Intelligence Tasks (HITs) are automatically created and posted on Amazon's Mechanical Turk (MTurk) platform, where workers are recruited to find the email the user was looking for and generate examples to aid in extraction.

Once workers accept a HIT, they choose the most relevant terms from the user's query. WearMail users authenticate their Gmail account prior to messaging the chat bot, and each subset of terms is queried with the Gmail API. The system then has full access to the user's email, but unlike other crowd-powered email management tools, WearMail restricts workers to viewing only obfuscated email metadata when performing their task.

Finding a Good Filter Set

As shown in Figure 4, WearMail first presents an interface for crowd workers to filter user email headers based on the given query. Crowd workers input keywords into the search bar to iteratively narrow down the headers, select the headers that they think might contain the requested information, and finally, submit the selected set of headers to the system.



What's Michael's yahoo email address?

Michael yahoo email address

Select	Sender	Subject	Date
□	Michael Bell	Re:	Wed, 22 Jul 2015 14:03:14
□	Lisa Ray	Re: of our paper	Wed, 17 Feb 2016 01:11:50
□	Campus Network	Help In the Class of , , and Games	Wed, 17 Aug 2016 19:01:05
□	Campus Network	: and More	Wed, 07 Sep 2016 19:01:12

Showing 1 to 10 of 50 entries

Figure 4. Crowdworkers identify the emails likely to contain the information asked for in the query by picking emails in a obfuscated view of email metadata. *a)* shows the obfuscation of person names that maps names mentioned in the query to different but consistent names in the Sender column, and *b)* shows the obfuscation of textual information in the subject based on general word frequency. Workers are allowed to change the search query, but can only use words mentioned by the user.

The filtering interface preserves end user privacy by design. Although less privacy-sensitive than an email body, an email header still contains personally identifying information (PII) like names, account numbers, addresses, etc., WearMail never exposes email addresses to crowdworkers and obfuscates all names of people in the sender field. We use a named entity recognizer to detect names, and then map each name to a new randomized name. When the query contains a person name, we replace the original person name in the query with the matching randomized name in the sender field (Figure 4(a)).

Given the complex semantics in the subject field, we wanted to explore the possibility of a generalized obfuscation method. In the obfuscation algorithm, we scrub any words which are not listed in Google's corpus of top 10,000 common words [15]. Ideally, we want to adequately obfuscate any personal information while maintaining enough information for crowd workers to select the right email headers. The obfuscated textual information is shown as blurred rectangles on the worker interface. Since blur effect is only a CSS style, we also replace sensitive text with random characters.

To investigate effects of obfuscation level, and thus determine an appropriate dictionary size for it, we performed an additional study involving the crowd. We used 24 queries from 6 participants at 19 different obfuscation levels. In the first 10 obfuscation levels, we varied the dictionary with a step size of 100 words, starting with a dictionary containing only the top 100 most common words (obfuscation level 1). In later obfuscation levels, we increased the step size to 1000 words. For each HIT, we recruited 10 crowdworkers for 1 query at 1 obfuscation level, and we recorded the top 3 most commonly selected email headers from the crowdworkers at each level. We found that crowdworkers only achieved 37.5% accuracy at obfuscation level 1, a significant accuracy increase after level 10, and stayed around 80% as the dictionary size continued to increase.

Finding Extractions

To find the user-queried information, we use extractors that match likely results from the bodies of a user's emails. The user emails that are searched are given by the previously found email filterset. The first layer of information extraction uses the Stanford Named Entity Recognizer (NER) [13]. Crowd workers are given a query, and are first asked to classify the type of information the user was likely searching for. If this classification matches any of the seven NER-recognized categories (Location, Person, Organization, Money, Percent, Date, Time), then the NER is run against the email filterset to produce a candidate list of results for the user. However, given its restriction to the seven aforementioned entity classes, the NER is insufficient as a standalone extractor for WearMail.

Hence, to overcome this, we introduce an additional layer of information extraction using regular expressions (regexes) that targets the data types that the NER fails to recognize. Prior work in generating regexes have taken a genetic programming and/or active learning approach. But these methods rely on a large initial training set and extensive computation time, rendering them impractical for use in Wearmail. Our approach uses a small set of crowd-generated examples to construct regexes that parse the email filterset and return an appropriate answer. For each query, crowd workers collect 20 examples of what the queried information may look like. For example, given the query "What is my DELTA confirmation number?", a crowd worker could search for and then respond with "DL1370", "JPX4QR", and so on.

The 20 worker-generated examples are first analyzed for character/whitespace density, variations in length, and relative character positioning. In Step 1 of Figure 5, a user asks the query "What was my Panera Bread order number?", and 20 crowd-generated examples are collected for the query: 13 incorrect and 7 correct. Examples with large deviations from the median length are discarded, and a "base" regex is derived from the remaining examples. This regex is based on various properties of the example set: most-frequent length, ratio of numeric to alphabetic characters, and the exsistence of (or lack of) non-alphanumeric characters.

To account for variability in the examples, we generate permutations of the base regex by relaxing and constraining features such as length and sequential character density. In Step 2 of Figure 5, various regexes are generated from the same example set. The first regex in Step 2 is the base regex, matching any 8 digit alphanumeric string in which at least 5 of the 8 characters are numbers. The next regex is generated by relaxing the length condition while holding the other conditions constant, thus matching strings with the same conditions as the base regex but between 2 and 15 characters long. Each subsequently generated regex is less constrained than the previous, and matches a wider range of strings. The final regex in Step 2 represents the last expression typically generated, matching any alphanumeric string with 5 or more characters.

Ranking Extractions

Each regex is given a *regex rank* based on its level of constraint. The most constrained regex would accept the fewest results and is given the highest rank (rank 1), and each level of constraint relaxation would accept a wider range of results and have a diminishing rank (down to rank 10).

We run each of the regex permutations against the crowd-generated filterset of emails to construct a preliminary list of matches. To more accurately differentiate between correct matches and false positives, we analyze the text in the email surrounding the matched result. Stop words (a, for, and, how, etc.) are stripped from each user query, and the remaining non-stop words are searched for in the neighboring text of each match. A *context rank* is given to each match based on the number and frequency of these non-stop words found in the surrounding text (Step 3 of Figure 5). A final ranking of results is determined by a combination of the regex rank and the context rank, and the top 5 results are returned to the user via the chat interface.

EVALUATION

To understand the performance and tradeoffs of our approach, we conducted several different evaluations of WearMail and its components. To benchmark our system as a whole, we invited 5 users to post a total of 30 specific queries. For each query, we collected ground truth answers for the information users were searching for, and the emails that contained this information. The 30 queries we tested closely matched the common categories of items derived from the mobile email search study. The most common were codes (9) like a Microsoft account security code, and a Domino's order number. A few others focused on contact info (7) like addresses, phone numbers, passport , etc. Travel-specific queries (3) included a Virgin Australia confirmation number, and an Enterprise confirmation number. There were a total of 5 queries relating to money (2), time (1) and date (2), all of which were NER-recognizable. The rest of the queries consisted of room numbers (3), order numbers (2) and an address (1).

Finding the Right Emails

We ran a study with 30 email queries to compare the performance of our two proposed filterset generation methods: one crowd-powered versus one automated. The crowd-powered approach asked workers to identify a filterset of emails that were likely to contain the information requested by the user in the query. The automated approach first removed stop words from the query, and then ran the remaining keywords through the Gmail search function. The top 3 emails returned became the automated email filterset. We also collected a set of ground truth emails from the user.

After generating the filterset using both approaches, we compared the filtersets against the ground truth answers (emails). We used HIT@3 (a standard metric in information retrieval) to evaluate each approach. HIT@3 calculates the total number of ground truth emails found in the first three results from each filterset, across all 30 queries. The crowd-powered approach produced 25 filtersets containing the ground truth email (HIT@3 = 0.833), and the automated approach produced 17 filtersets containing the ground truth email (HIT@3 = 0.567). Thus, over the 30 queries, the crowd-powered approach performed notably better at constructing a filterset that contained the ground truth email. This is necessary for any subsequent step to have a chance to succeed.

Information Extraction

Information extraction was performed with a combination of NER and a custom crowd-powered extractor. Applying the extraction pipeline to the crowd-generated filterset for each query, a correct result was found for 20 of the initial 30 queries (HIT@3 = 0.66).

Of the 30 end-user generated queries, five of them were recognizable by NER, and the remaining were passed to the second stage of the pipeline. Of the 20 successful results returned to the user, only three were extracted using NER, while 17 (85%) of them were extracted using the crowd-generated regex extractor.

For the crowd-powered extractor, matches were ranked by their corresponding regex rank and contextual relevance, and the 5 highest ranked extraction candidates were returned to the user. Of the 17 successful matches, 10 were returned as top ranked result, 6 were returned as the second ranked result, and 1 was returned as the fourth ranked result, achieving an MRR[2] (mean reciprocal rank) score of 0.779 for the successful crowd-powered extraction matches.

For the entities where workers found it difficult to generate perfect examples, our results showed that some workers were able to come "close" to the desired answer. This is particularly evident in the 30 queries that we tested, with 20 examples generated for each. Of the 17 correct results that were extracted from the users' emails, 6 queries had over 18 correct examples, 5 queries had between 6 and 12 correct examples, and 6 queries had 4 or fewer correct examples.

[2] Mean reciprocal rank is the sum of the inverse of the rank position of each query result over the total number of queries.

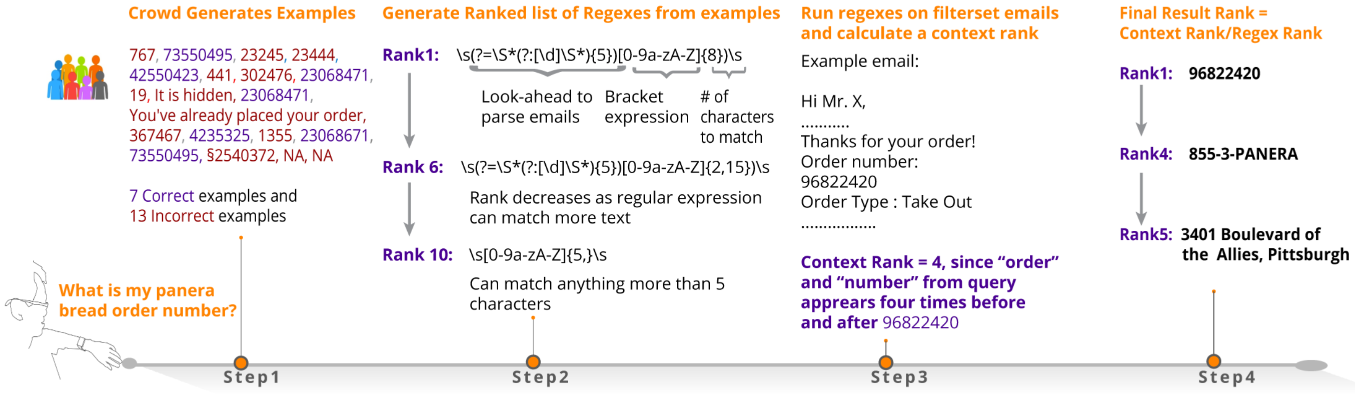

Crowd Generates Examples
767, 73550495, 23245, 23444, 42550423, 441, 302476, 23068471, 19, It is hidden, 23068471, You've already placed your order, 367467, 4235325, 1355, 23068671, 73550495, §2540372, NA, NA
7 Correct examples and
13 Incorrect examples
What is my panera bread order number?
Generate Ranked list of Regexes from examples
Rank1: \s(?=\S*(?:[\d]\S*){5})[0-9a-zA-Z]{8})\s
Look-ahead to parse emails
Bracket expression
# of characters to match
Rank 6: \s(?=\S*(?:[\d]\S*){5})[0-9a-zA-Z]{2,15})\s
Rank decreases as regular expression can match more text
Rank 10: \s[0-9a-zA-Z]{5,}\s
Can match anything more than 5 characters
Run regexes on filterset emails and calculate a context rank
Example email:
Hi Mr. X,
....
Thanks for your order!
Order number:
96822420
Order Type : Take Out
....
Context Rank = 4, since “order” and “number” from query apprears four times before and after 96822420
Final Result Rank = Context Rank/Regex Rank
Rank1: 96822420
Rank4: 855-3-PANERA
Rank5: 3401 Boulevard of the Allies, Pittsburgh
Step1
Step2
Step3
Step4

Figure 5. A crowd-powered workflow to build WearMail's custom regular expression extractor. Details of each step are as follows: 1) 10 workers generate a total of 20 examples of the queried information. 2) The examples are used to construct an ordered list of regexes, each of which are given a *regex rank* based on its level of constraint. 3) Each regex is run against the crowd-generated email filterset and a list of matches is recovered. A *context rank* is given to each match, equal to the number of non-stop words found in the surrounding email body text of the match. 4) The final results are ranked based on both the regex rank and context rank.

DISCUSSION

Failure Modes

A post-hoc analysis of failed queries highlights several reasons why WearMail sometimes failed to find the correct user information. First, the user's query could lack sufficient specificity to allow crowdworkers to accurately search for candidate examples. For example, consider the failed query “Where was the RERC meeting last Friday held?” The phrasing of the query is specific enough for the user and anyone aware of the “RERC meeting”, but workers are likely unaware of where such a meeting would be held (*i.e.*, building name). Second, the previous example shows another common scenario: users use acronyms in their queries by habit, but create situations in which it is difficult for crowd workers to accurately find examples. Third, spelling mistakes in query formulations (*i.e.*, DCMA notice number instead of DMCA) misled crowdworkers. In the future, WearMail may guide users to give more useful information or update incorrect information in their queries.

Recovery Strategies

In the future, WearMail may include more robust recovery strategies for users who don't receive their intended results. For example, the filterset generation stage could support collection of additional metadata, such as “from”, “to”, and “date” to give better context to crowd workers. If the extraction stage fails, WearMail could return the crowd-generated filtersets to the user.

Latency

Another metric that influences WearMail's feasibility is latency. On average across the 30 queries, crowd workers completed the email filtering task in 2min 18s, and the information extraction task in 6min 16s. While WearMail currently does not specifically optimize for time, future work may look into streamlining the system and reducing overall latency. In deployment, we would use a retainer [1] to recruit workers, and expect many of the data types and examples to be reusable in future queries, reducing latency and cost.

FUTURE WORK

WearMail suggests a number of opportunities for future work. One is making more complex tasks, like email searching, easier to accomplish on constrained mobile and wearable devices. The task of searching emails on-the-go from wearable devices provides a pretense for complex tasks like composing email from smartwatches, drawing from crowd-powered systems like WearWrite [31]. WearMail also suggests approaches for using the crowd to tackle other privacy-preserving tasks. Future systems may explore how the crowd can contribute to more complicated queries on sensitive data. For instance, the query “How many times have I taken an Uber to work in the last month?” could be answered by systems that combine information extractors with logical operators. Users may also want to ask questions that require subjective interpretation, *e.g.*, “Of the students in my class who emailed me last semester, how many seemed frustrated?” These queries may be feasible with more complicated workflows and a controlled relaxation of obfuscation to show more of the message content to crowdworkers.

CONCLUSION

In this paper, we have introduced WearMail, a crowd-powered system designed to extract information from a user's email, which was a core mobile use case identified in our study. Unlike prior crowd-powered systems with email access, a primary contribution of WearMail is in the privacy-preserving workflow that combines human work and automation, using both obfuscated metadata and programming by example. Both the privacy-preserving workflow, and WearMail as a system, present a number of insights that can benefit future crowd-powered systems.

ACKNOWLEDGEMENTS

This work was supported by Yahoo! and the University of Michigan. Aspects of this work were also developed under a grant from the National Institute on Disability, Independent Living, and Rehabilitation Research (NIDILRR grant number 90DP0061). NIDILRR is a Center within the Administration for Community Living (ACL), Department of Health and Human Services (HHS). The contents of this paper do not necessarily represent the policy of NIDILRR, ACL, HHS, and you should not assume endorsement by the Federal Government. We thank Lydia Chilton and Jonathan Bragg for work on the initial idea for WearMail. Finally, we thank our user study participants and workers on Amazon Mechanical Turk without whom this research would not have been possible.

REFERENCES

1. Bernstein, M. S., Brandt, J., Miller, R. C., and Karger, D. R. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 33–42.

2. Bernstein, M. S., Teevan, J., Dumais, S., Liebling, D., and Horvitz, E. Direct answers for search queries in the long tail. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems* (2012), 237–246.

3. Bigham, J. P., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R. C., Miller, R., Tatarowicz, A., White, B., White, S., et al. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM (2010), 333–342.

4. Brin, S., and Page, L. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems 30*, 1 (1998), 107–117.

5. Cochran, R. a., D'Antoni, L., Livshits, B., Molnar, D., and Veanes, M. Program Boosting: Program Synthesis via Crowd-Sourcing. In *POPL* (2015), 677–688.

6. Dabbish, L. A., Kraut, R. E., Fussell, S., and Kiesler, S. Understanding email use: predicting action on a message. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2005), 691–700.

7. Demartini, G., Trushkowsky, B., Kraska, T., and Franklin, M. J. CrowdQ: Crowdsourced Query Understanding. In *6th Biennial Conference on Innovative Data Systems Research (CIDR '13)* (2013).

8. Doudalis, S., Mehrotra, S., Haney, S., and Machanavajjhala, A. Releasing True Data with Formal Privacy Guarantees. In *Privacy-Preserving IR Workshop at SIGIR* (2016).

9. Dumais, S., Cutrell, E., Cadiz, J. J., Jancke, G., Sarin, R., and Robbins, D. C. Stuff i've seen: A system for personal information retrieval and re-use. *SIGIR Forum 49*, 2 (Jan. 2016), 28–35.

10. Elsweiler, D., Baillie, M., and Ruthven, I. What makes re-finding information difficult? a study of email re-finding. In *European Conference on Information Retrieval*, Springer (2011), 568–579.

11. Horvitz, E., Jacobs, A., and Hovel, D. Attention-sensitive alerting. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc. (1999), 305–313.

12. Huang, T. H., Lasecki, Walter S., Azaria, A. and Bigham, J.P. "Is there anything else I can help you with?": Challenges in Deploying an On-Demand Crowd-Powered Conversational Agent. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2016)*, (2016).

13. Jenny Rose Finkel, T. G., and Manning, C. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)* (2005), 363–370.

14. Jeong, J. W., Morris, M. R., Teevan, J., and Liebling, D. A crowd-powered socially embedded search engine. *Proceedings of the 7th International Conference on Weblogs and Social Media, ICWSM 2013* (2013), 263–272.

15. Kaufman, J. 10,000 most common english words. https://github.com/first20hours/google-10000-english.

16. Kaur, H., Gordon, M., Yang, Y., Bigham, J. P., Teevan, J., Kamar, E., and Lasecki, W. S. Crowdmask: Using crowds to preserve privacy in crowd-powered systems via progressive filtering. In *Proceedings of the AAAI Conference on Human Computation (HCOMP 2017).*, HCOMP '17 (2017).

17. Kim, Y., Collins-thompson, K., and Arbor, A. Crowdsourcing for Robustness in Web Search. In *TREC* (2013).

18. Kittur, A., Nickerson, J. V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., and Horton, J. The future of crowd work. In *Proceedings of the 2013 conference on Computer supported cooperative work (CSCW 2013)*, 1301–1318.

19. Klimt, B., and Yang, Y. Introducing the enron corpus. In *CEAS* (2004).

20. Kokkalis, N., Köhn, T., Pfeiffer, C., Chornyi, D., Bernstein, M. S., and Klemmer, S. R. 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, 2013), 1291–1300.

21. Koutra, D., Vogelstein, J. T., and Faloutsos, C. Deltacon: A principled massive-graph similarity function. In *Proceedings of the 13th SIAM International Conference on Data Mining (SDM)*, SIAM (2013), 162–170.

22. Laput, G., Lasecki, W. S., Bigham, J. P., Wiese, J., Xiao, R., and Harrison, C. Zensors: Adaptive, rapidly deployable, human-intelligent sensor feeds. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2015).

23. Lasecki, W. S., Wesley, R., Nichols, J., Kulkarni, A., Allen, J. F. and Bigham, J. P. Chorus: A Crowd-powered Conversational Assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST 2013)*, 151–162.

24. Lasecki, W. S., Gordon, M., Koutra, D., Jung, M. F., Dow, S. P., and Bigham, J. P. Glance: Rapidly coding behavioral video with the crowd. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, ACM (New York, NY, USA, 2014), 551–562.

25. Lasecki, W. S., Gordon, M., Leung, W., Lim, E., Bigham, J. P., and Dow, S. P. Exploring privacy and accuracy trade-offs in crowdsourced behavioral video coding. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, ACM (New York, NY, USA, 2015), 1945–1954.

26. Lasecki, W. S., Murray, K. I., White, S., Miller, R. C., and Bigham, J. P. Real-time crowd control of existing interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 23–32.

27. Lasecki, W. S., Song, Y. C., Kautz, H., and Bigham, J. P. Real-time crowd labeling for deployable activity recognition. In *Proceedings of the 2013 conference on Computer supported cooperative work*, ACM (2013), 1203–1212.

28. Lasecki, W. S., Teevan, J., and Kamar, E. Information extraction and manipulation threats in crowd-powered systems. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '14, ACM (New York, NY, USA, 2014), 248–256.

29. Lasecki, W. S., Weingard, L., Ferguson, G., and Bigham, J. P. Finding dependencies between actions using the crowd. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2014), 3095–3098.

30. Myers, B. A., Ko, A. J., and Burnett, M. M. Invited research overview: End-user programming. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, ACM (New York, NY, USA, 2006), 75–80.

31. Nebeling, M., To, A., Guo, A., de Freitas, A. A., Teevan, J., Dow, S. P., and Bigham, J. P. Wearwrite: Crowd-assisted writing from smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, ACM (New York, NY, USA, 2016), 3834–3846.

32. Parameswaran, A., Teh, M. H., Garcia-Molina, H., and Widom, J. DataSift: An expressive and accurate crowd-powered search toolkit. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing* (2013), 112–120.

33. Qingyao Ai, Susan T. Dumais, N. C., and Liebling, D. Characterizing email search using large-scale behavioral logs and surveys. In *Proceedings of the World Wide Web Conference (WWW 2017)* (2017).

34. Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, vol. 62 (1998), 98–105.

35. Sproull, L., and Kiesler, S. New ways of working in the networked organization. *Cambridge, MA* (1991).

36. Teevan, J., Alvarado, C., Ackerman, M. S., and Karger, D. R. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2004), 415–422.

37. Teevan, J., Collins-Thompson, K., White, R. W., Dumais, S. T., and Kim, Y. Slow Search. In *Proceedings of the Symposium on Human-Computer Interaction and Information Retrieval - HCIR '13* (2013), 1–10.

38. Thomas, G. F., King, C. L., Baroni, B., Cook, L., Keitelman, M., Miller, S., and Wardle, A. Reconceptualizing e-mail overload. *Journal of Business and Technical Communication 20*, 3 (2006), 252–287.