MUSE: Reviving Memories Using Email Archives

Sudheendra Hangal  
Computer Science Department  
Stanford University  
hangal@cs.stanford.edu

Monica S. Lam  
Computer Science Department  
Stanford University  
lam@cs.stanford.edu

ABSTRACT
Our email archives often silently capture our actions and thoughts, virtually every day, as they come up in our communication. Email can therefore be viewed as a convenient form of life-logging that provides millions of users an excellent source of material to remember and reflect upon their lives.

This fact motivated us to develop Memories Using Email, or Muse, a mining and browsing system that lets users easily and rapidly browse long-term email archives. We incorporated a collection of data mining techniques, including organization of content by groups and text mining. We also developed novel presentation and interaction techniques, such as using an on-screen jog dial to quickly scan through a sequence of messages, automatic hyperlinking between significant terms in messages and a date-aware diary area for clipping terms into.

The collective result of these techniques – as confirmed by our users – is that sense-making of email archives is much easier than with a conventional email client. We discuss users’ experiences with Muse and how they would like to use our system.

Author Keywords
Life-logging, E-mail, Data mining, User interfaces

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous—Optional sub-category

INTRODUCTION
Email is one of the Internet’s most enduring “killer applications” with over 1.8 billion users and 2.9 billion accounts worldwide [1]. Given the availability of free email services with virtually unlimited storage, we expect that millions of mainstream users will amass large email repositories, perhaps spanning 50 years or more, over their lifetimes. Moreover, since email archiving (of sent messages in particular) is virtually automatic, we expect that they will have stored a reasonably large fraction of their email records. It is already not uncommon to find people, especially in universities, who have email records going back 20 years or more.

Email as Life-Logging
In much of the wired world, email is used for many daily activities, for everything from setting up meetings to processing business workflow; from making purchases on the Internet to sending emotional messages of love, joy, and condolence; from sending oneself reminders to sharing an interesting web link with friends, and so on. Unlike blogging or keeping a diary or journal, email silently captures our experiences and thoughts, virtually every day, week, month, or year, in situ, as they come up in our communication. Indeed, email has become a defacto tool of record: many people consciously deposit information important to them into email, knowing they can look it up later, and thereby use their email account as an informal backup device. In corporations, people routinely write up emails to colleagues containing information that has already been conveyed, as a way to maintain a communication trail. We surmise that for a significant fraction of computer users, more characters are typed into their email application than into any other, and processing these characters, as accumulated over a lifetime, can provide a powerful window into history.

The history of one’s life is not only interesting to the user, but also to his or her family and friends, and eventually to historians and digital archaeologists in the future. We envision that, in future, it should be possible for a 30-year old to recall a field trip he made in kindergarten, along with all the details associated with planning the event that are in his parents’ email archives. People should be able to put their archives to work when sharing their fond memories at events like weddings and 50th birthdays. And it should be easy for ordinary users, not just famous celebrities, to write their memoirs, which would be of great interest at least to their immediate friends and family.

Email Mining Tools
Our work attempts to answer the question of how people might use long-term archives of email and what interfaces are needed. Our research on mining email archives was motivated in the wider context of the increasing use of life-logging techniques [6]. Ironically, while the pulse of the entire Internet can be captured with sophisticated tools like Google Zeitgeist or Twitter trending topics, and email service providers routinely mine email for advertising purposes, there are relatively few tools to help individuals make sense of their own collection of digital archives. As Petrelli et al remark [12]: This may be the fate of lifelog data, stored somewhere and ignored, if the owner is not given tools for sorting, clearing and distilling what is of value.

Typical email clients today present messages to users one by
one, and let them issue search queries based on keywords and message meta-information. This usage model is inadequate for archives spanning years or decades, and often running into tens of thousands of messages. Further, it is ill-suited to the purposes of browsing, where a user does not know exactly what to look for, but where rich serendipitous discoveries are waiting to be made. We have built and publicly released a system called Muse, which attempts to do this by performing mining to organize social landscapes, and summarizing the contents of the messages themselves. The summaries are excellent ways to jog the users memory and are used as entry points into a browsing system that lets users rapidly scan messages.

The Themail program [16] is an email visualization tool that lets users visualize their relationships with their contacts, one at a time, by visualizing high-scoring single words occurring frequently in communication with that contact across time. We started our research project by implementing a system based on ideas loosely similar to Themail. We took an iterative prototyping approach to understanding user needs. We observed users’ reactions to the system, refined it based on their feedback, explored various design choices, and iterated this process numerous times until users find the system engaging and acceptable for processing large archives.

We believe that better understanding of user needs through fieldwork and contextual inquiry (for example, approaches such as Kirk and Sellen’s on understanding human archiving practices and needs [7] or Elsweiler’s work on understanding human memory [4]), as well as concrete implementations such as ours that can be deployed in the real world will all contribute to successful systems in the future. Since Muse brings together techniques related to several different areas, we defer our survey of related work until after we have described our system.

Contributions

This paper makes the following contributions to the topic of using email archives for life-browsing.

1. With the help of user feedback, we have identified a number of important ideas useful for browsing long term email archives and implemented them in a working system.
   - Because email archives accumulated over decades are very large and span thousands of contacts, it is important to let users separate conversations from different facets of their lives. We have developed a novel group mining algorithm automatically detects groups of people in the email corpus, based on email patterns and organizes results using these groups.
   - It is useful to identify new multi-word terms at around the time that they were first introduced. Our term mining algorithm adds the notion of time-based incrementality to the TF-IDF metric to locate such terms. It finds key phrases and not just single keywords, which tend to be ambiguous.
   - Interactive aids are needed to help users rapidly scan multiple email messages. One such aid that we have introduced is an on-screen jog dial that enables a user to scan through long sequences of messages without the need for clicking.
   - We discovered novel uses of email analysis during our user studies. For example, while users are generally sensitive about email privacy, they also like to selectively share with others the memories revived by browsing their archives. There is also a lot of individual variability in what users would like to use this system for.

2. Virtually all users in our user study prefer to use Muse for the purpose of browsing their archives to their current email client like Thunderbird or Gmail. In one experiment, users could mine their email to recall about 38 significant events in a year of their life in 15 minutes or less.

3. We have made the Muse system publicly available for download at the (cloaked) URL http://bit.ly/cE5Bv0. Our system has been used by over 25 users, and has been used on archives of up to 50,000 messages.

While we have not completely evaluated all the features of our current system, detailed evaluation of each feature is planned as part of future work. We will briefly describe 2 formal studies in later sections.

The current implementation of Muse focuses on email; however, our general techniques have varying degrees of applicability to other forms of communication that are primarily text-based, and optionally include supplementary data files. Examples of such communication include SMS or MMS messages, instant messages, newsgroups posts and Facebook wall posts.

![Figure 1. A screenshot of the Muse Control Panel showing 2 sources of email: folders in a Gmail account and files in a local directory. Some details are blurred to preserve user privacy.](image-url)
THE MUSE SYSTEM

In this section, we provide a brief description of how Muse works and present some of the more interesting user interface aspects. We focus on techniques needed to scale to large-scale email archives of ordinary consumers, which tend to be accumulated over many years, loosely organized, perhaps with duplicates, stored in multiple places etc.

To use Muse, the user launches the application which in turn starts an internal web server and brings up a window in a standard web browser pointed at the server. The user provides details of all online email accounts to be accessed (via IMAP or POP access) and any file system directories that may contain saved email folders.

Once the sources of email are identified, the user selects specific folders to be mined and if desired, chooses to filter messages based on whether the message was sent or received, a person’s name or email address, keywords, or by date range. (See Fig. 1.)

We typically recommend that users analyze their sent messages, which usually reflect the user’s thoughts and actions more accurately than incoming messages. Users often receive several times (2 to 20 times in our experience) as many emails as they send. Much incoming email is from mailing lists, which are sometimes read casually, in batches, or even ignored entirely.

To identify significant terms per year or month, we score terms based on a modified TF-IDF metric described in the next section. The top scoring terms per time unit are displayed on a card, and the cards are arranged in a calendar view.

Within a card for each time window, all terms are assigned to the group that they are most closely related to. Terms assigned to the same group (and therefore in a particular color) are displayed together, and are further sorted by score. Documents not assigned to any group are colored gray and displayed last. The color encoding makes it easy to follow at a glance all terms related to a particular category, such as work, family, soccer group, spouse, etc. (or to avoid them: in the words of one user, “I can gaze over all the boring exchanges with my secretary.”)

See Fig. 3 for an example of a card for the same dataset, shown in three different configurations, which were evaluated in one of our user studies. The default configuration is the one on the right, which employs both color coding by group and displays multi-word phrases.

Organization by groups

A key principle in Muse is the organization of people into logically cohesive groups. We find this preliminary step essential when analyzing a long-term archive, since email across even a few years typically involves thousands of contacts. Given the large number of contacts, it is difficult for users to review messages, topics, etc. one individual at a time, or for that matter, to manually arrange contacts into various groups.

Muse automatically discovers likely groups by analyzing co-occurrence patterns and frequencies in email archives. It employs a novel group mining algorithm (described in the next section) that satisfies several properties we found to be important in social contexts. First, the algorithm can infer groups of people who do not all appear together in any single message. Consider a user’s extended family which is loosely clustered, with people joining and leaving across time; no single message reaches all of them. Second, within the groups, there may be important subgroups which have a significant identity of their own. For example, siblings may be a cohesive subunit of the extended family. Next, the same person could belong to multiple groups: e.g. a colleague at work may also be part of a hiking group. And finally, a significant “group”, for the purposes of organization, could consist of just one very important person with a high communication volume, like a spouse.

While the groups inferred automatically by our algorithm are an excellent starting point, they may need to be tuned since email communication patterns may not always reflect real-life relationships. Therefore we also provide users a drag-and-drop editor with which can move contacts between groups, create or delete groups, etc. (See Fig 2). After the user has applied any edits, we select the top 20 groups and assign each of them a different color chosen from a palette of distinguishable colors. These colors are then used consistently throughout the interface to represent that group. We use the W3C Contacts API, if available in the browser, to access pictures and supplementary information about the user’s friends gathered from social network sites like Facebook, LinkedIn and Gmail. (Currently this API is implemented by Mozilla’s Contacts addon for Firefox.) The groups editor uses this plugin if installed to display people’s profile pictures for drag and drop in addition to names or email addresses.

Identifying representative terms

To identify significant terms per year or month, we score terms based on a modified TF-IDF metric described in the next section. The top scoring terms per time unit are displayed on a card, and the cards are arranged in a calendar view.

Within a card for each time window, all terms are assigned to the group that they are most closely related to. Terms assigned to the same group (and therefore in a particular color) are displayed together, and are further sorted by score. Documents not assigned to any group are colored gray and displayed last. The color encoding makes it easy to follow at a glance all terms related to a particular category, such as work, family, soccer group, spouse, etc. (or to avoid them: in the words of one user, “I can gaze over all the boring exchanges with my secretary.”)

See Fig. 3 for an example of a card for the same dataset, shown in three different configurations, which were evaluated in one of our user studies. The default configuration is the one on the right, which employs both color coding by group and displays multi-word phrases.
Figure 3. A comparison of term displays for the same month: using just words ranked by score (L); words sorted first by group (M); and phrases sorted by group (R).

Users typically start browsing the archive from the top terms view in the cards. The displayed phrases themselves often evoke relevant memories for the user, but they can also be clicked as entry points into browsing the actual messages in the archive. When a term is selected, a message view is launched in a new tab for displaying the content of the messages with that term. The message view includes the message header, message text and thumbnails of and links to attachments, if any. Similarly clicking on a time unit, a group or a person launches a message view with the corresponding selection of messages.

Rapid scanning with an on-screen jog dial

Our original implementation of the message view just displayed all the messages in the view, one below the other. We found this tended to create long pages, and when there were more than about 10-15 messages, users would get bored and stop scanning messages part of the way down the page. Moreover, in large archives, some views (e.g. all messages for a group or person) can consist of thousands of messages and make the page slow to load.

To alleviate the tedium of scrolling down a long page, we download multiple pages into the browser but display only one message at a time in the message view. This has the advantage that it fixes the locations on the screen of the message headers and the beginning of the message contents. To enable rapid scanning of a sequence of messages in the view, we provide a translucent on-screen circular jog dial that is brought up on the spot and dismissed by clicking anywhere in the message view. See Fig. 4 for a screenshot of the message view when the dial is visible. The operation of the dial is similar to the physical dial on iPod music players and is thus familiar to most users. Moving clockwise to the next octant causes the view to display the next message; moving counter-clockwise displays the previous message. Apart from being somewhat playful, the dial allows fast interactive performance due to the use of Javascript – in our experience, users can easily flip through about 3 pages a second in rapid scanning mode, an experience similar to quickly flipping through the pages of a book to get an overview of its contents. The dial affords fine-grained control, since users can stop when they see something interesting and slowly move backwards or forwards. It also lets them travel relatively long distances (scanning through 200 messages is not uncommon) without the need for mouse clicks, key presses, adjusting timeline zoom, repositioning the cursor, or switching gaze from the message view.

It is important to manage browser load for a message view with thousands of messages. Muse maintains a sliding window of pages around the page currently being displayed, and keeps only the pages in that window loaded in the browser. As the user moves along, the Javascript on the page “pages in” new messages to maintain the window around the current page, while retiring pages outside the window. In practice, we find that a default window size of 100 pages (60 pages ahead and 40 pages backward) is adequate for good user experience with no stalls and is easily handled by current browsers.

The user also has the option of tabbing to quickly advance or revert to the first message in the next or previous time unit (e.g. the next or previous month). The user can also use the forward and back arrow keys for scrolling across messages.

While the current features of our jog dial are fairly simple, recent implementations like Cyclostar have demonstrated the effectiveness and versatility of elliptical gestures [8]. The jog dial works best with a trackpad such as the one on a laptop computer; it is usable with a conventional mouse though not as convenient.

Hyperlinks between message views

Our goal is to automatically create a rich hypertext system that permits exploratory browsing of the archives. Inside the message view, we hyperlink high scoring terms in the message that are also present in other messages to the message view for that term; clicking on the link launches messages for that term in a new tab. This makes it easy for users to find interesting cross-references as they are browsing, fork off another thread of exploration, etc. In Fig. 4, the message has two inserted hyperlinks in its body.

Throughout the interface, clicking on the name, email address or picture of a person, or the description of a group, can be used to launch a message view containing all the associated messages. Similarly it is possible to view all attach-
ments for the selected set of messages, all web links embedded in them, or to see graphs of incoming and outgoing message activity. When viewing an attachment, one can go to all messages which had that attachment; or from the links view to messages containing the link.

Inside a Muse message view, we also offer a simple filter-based browsing interface, so users can launch searches for new terms, restrict the view to a particular time period, or messages involving a particular person, etc. Systems like Phlat [3] already provide sophisticated forms of filtering and search, which could be usefully employed by Muse. An advantage Muse has over these systems is that its mining of representative terms lets it link between documents without the need for structured metadata.

**Date-Aware Diary**

In the top terms view, we allow users to effortlessly copy terms that they deem valuable to a scratchpad area representing a diary (Fig. 5). The diary area is date aware and knows where to place incoming terms regardless of the order in which they are selected. Users can edit the diary area directly, which is kept extremely simple and represented in plain text, and can be saved, e-mailed or printed. One of our explicit goals is that users should be able to take something away from the Muse system, as opposed to seeing a cool visualization once which they cannot revisit or share easily.

![Figure 5. A date-aware diary area which serves as a clipboard for interesting terms that the user can then edit.](image)

**Web browser based implementation**

As mentioned already, Muse is implemented in Java and Javascript and is used via standard web browsers. The application uses Java Webstart, so that users can launch it on their own machines with a single click, yet use a familiar web browser based interface to the program. The browser interface has the advantage that Muse automatically benefits from browser capabilities: Users can open multiple windows or tabs, use browser history, seamlessly follow HTTP links, enable gesture recognizers and other plugins, and so on. Mozilla’s Contacts is a good example of a plugin that we were able to leverage very easily.

In terms of implementation, the web-server based approach also makes it easy for us to run the service transparently in the cloud (if necessary) or on users’ local machines. However, before we had implemented the Webstart mode, we used to run Muse via a hosted web server, and many email users were concerned about privacy, so much so that they would ask us to set up a server on their own machine before they would agree to use it. One user backed out of a planned study in somewhat of a panic after realizing that there were highly confidential financial documents in her email.

**Browsing attachments**

People often use email archives as an informal file repository for important files they can find later. This is because it fixes well-known problems associated with storing files on a computer: naming a file is unneeded overhead, directories are inadequate as an organizing device, archiving is not automatic, and it is not easy to access or share data from everywhere [5]. While email archives avoid some of these problems, they introduce another problem: it is not easy for users to browse all of their attachments with current email clients.

Muse lets users browse attachments efficiently by extracting all pictures as well as by converting the first page of PDF documents to thumbnail images. (We are evaluating options for doing the same for Microsoft Office documents.) All these images are then displayed using one of two interfaces, a picture page with thumbnails, or the PicLens 3D photo wall (from cooliris.com) which presents a time-based, aesthetically pleasing, 3-dimensional zoomable and draggable interface. Users can download and open the attachment or bring up all messages containing the attachment. Attachments that cannot be easily converted to an image representation are assigned a generic thumbnail, but the user can still view the name of the attachment and open it using a native application.

**External links**

Email is often used for passing around hyperlinks to useful information on the web. To allow users to review these links in one place, we extract all HTTP links from message text, along with the date of the message. We list these links on the links page with appropriate hyperlinks to the original page. However, over a long period of time, it is possible that some hyperlinks would have gone dead. For showing the user the page as it may have existed on the original date of communication, we point the link to the Wayback Machine of the Internet Archive (http://archive.org). The Wayback Machine takes snapshots of public web pages every two months and stores all historical revisions, allowing queries for the contents of a page as of a specified date.

**DATA MINING TECHNIQUES IN MUSE**

In this section, we describe how Muse identifies significant groups and representative phrases from the email archive. Since the user already has some familiarity with the content, Muse can operate hand-in-hand with the user’s memory and expertise by offering useful suggestions and letting the user complete the job easily and effectively.

Note that Muse is usually downloaded onto the user’s laptop or desktop and run locally for privacy reasons; hence
we do not make any assumptions about running the mining on server class hardware. We therefore use relatively lightweight, yet effective techniques that make efficient use of memory and CPU cycles.

**Automatic Group Extraction**

Our group extraction algorithm analyzes co-recipients of email to derive a number of groups from the corpus. As discussed earlier, the groups extracted may not occur as co-recipients of any single message. Furthermore, as the groups extracted may be overlapping or nesting, traditional clustering algorithms like k-means and modularity based methods cannot be used.

Our algorithm starts by associating with each co-occurrence of recipients a value reflecting the group’s frequency and size. The value increases with the number of messages received and the number of people in the group. The goal of the algorithm is to derive a relatively small number of groups representing as much value as possible. The algorithm iteratively reduces the number of groups with one of three moves: dropping a group, merging two groups, and intersecting two groups. In each step, it picks the move that reduces the total value of all groups by the smallest amount until the desired number of groups is met. When two groups are merged, the value of the new group is the sum of their original values minus a penalty proportional to the over-sharing of messages across the two groups. When two groups are intersected, there is no over-sharing penalty, but the new group’s value is reduced proportional to the lost coverage.

**Entity Resolution**

Since our goal is to tackle email corpora built over years or decades, it is important to handle identities correctly. Specifically, email addresses as well as the way names are spelt are prone to change over time. We handle this issue by unifying names and email addresses in email headers when either the name or email address is equivalent. Name equivalence is tested by ignoring case differences and equating commonly used variations in naming, e.g., with or without a middle initial, “Firstname Lastname”, “Lastname, Firstname” and “Firstname Lastname - Department”.

**Tokenization and Indexing**

We first strip each message of attachments and other extraneous characters such as HTML tags and identify the text of the message, as well as its metadata like sender, recipients and date. We can also, at the user’s option, filter out quoted messages (which are often included when a message is replied to), forwarded messages etc.

We cluster email messages in user-specified time units (typically by month or by year). All messages in the unit are put in a single cluster. Subject headers usually carry highly representative words so they are given a higher weight by boosting their frequency (by default, we boost them by a factor of 2); generic terms like “Hi” will automatically get a low score due to the fact that they are common across the entire corpus.

Next, we tokenize the input, forming a postings list and index each cluster of messages, by identifying sentences and forming unigrams and bigrams. Bigrams are useful for deriving understandable phrases as described below, and therefore bigrams spanning sentences are not considered. Connecting words like ‘to’, ‘a’, ‘on’, are skipped over for the purposes of creating bigrams; however, proper indices are maintained so that when selected terms are displayed to the user, they include all the join words, original capitalization and punctuation.

**Incremental TF-IDF**

To identify meaningful terms from a set of email messages, we use a metric based on TF-IDF (Term Frequency times Inverse Document Frequency) scores on unigrams and bigrams, using the *nnt* variation described by Manning et al [9]. High TF-IDF scores reflect highly weighted terms that are specific to that time interval.

An important innovation we have incorporated into TF-IDF scoring is the computation of incremental TF-IDF. Traditionally, the number of documents containing the term across the entire corpus is considered while computing the IDF factor. However, we compute the inverse document frequency for a term $T$ in a document $D$ based on the number of documents preceding $D$ that contain $T$. Incremental IDF computation can be done fairly easily in the same pass as indexing by processing documents in increasing order of time, immediately computing the IDF factor for each term once a time quantum is over, and computing TF-IDF for the document’s terms. Traditional IDF computation would wait till the entire corpus is indexed before scoring any of the terms, and then assign an IDF score to each term.

This scoring method captures the fact that email accumulates chronologically, and therefore it makes sense for the analysis to extract terms on a time based replay of messages. It allows us to highlight a newly emergent term such as the name of a newborn family member, a newly acquired hobby, or a name like Obama, that subsequently becomes commonplace in the archive. As time goes by with repeated use of the new term, the IDF score slowly reduces, making the term less prominent. A side benefit of the incremental TF-IDF measure for interactivity (though one that we do not yet exploit) is that the scores for terms in a time unit are immediately available after it is indexed, without waiting for subsequent time units to complete.

**Phrase Construction**

At the end of the indexing step, we extract unigram and bigram terms with the highest score. Since even high scoring words may be ambiguous when shown to the user, we attempt to derive longer, more meaningful phrases by combining bigrams when possible. To save memory, we restrict our indexing to bigrams, but attempt to combine them in the top scoring terms if the second word of one is the first word of the other. Trying to combine only pairs of unigrams would cause more inaccuracies as well as increase runtime. Term postings in our index also store a small hash of the positions of the term in the document, so we can probabilisti-
cally check whether the bigrams are consecutive, forming a genuine trigram. Similarly, trigrams are considered to be merged with other bigrams and trigrams. This process continues till a fixed point is reached and no new phrases can be derived. Using this approximate algorithm to combine bigrams into longer phrases avoids the memory and time overhead of directly maintaining longer phrases in the index.

Our current algorithm tries to combine phrases as long as possible; however, as we note later when discussing our second user study, some users associated negative returns with phrases beyond a length of 3 or 4.

**Phrase Selection**

One problem we observed after selecting the highest scoring terms and phrases was that a single message sometimes contributed a disproportionately large number of terms. Users would ask why we had so many terms reminding them of the same thing. To tackle this problem, we allow for each message a maximum number of selected terms that point only to that message; after this maximum is reached, other such terms are suppressed, even if they would otherwise have been selected. Empirically, we have determined that a threshold of 4 works well to prevent a single message from dominating the entire output. In addition, to suppress noise, we also suppress single-word terms that are in the list of the 100 most frequent words in the English language, even if they have high scores.

Currently, we maintain a list of connecting words and a list of most frequent words only for English. While Muse works with international languages like Chinese, Korean and Japanese, it does not yet have such words lists for other languages (though they should be easy to incorporate).

**SYSTEM IMPLEMENTATION**

Our experimentation uncovered several practical challenges, which led to a number of optimizations implemented in our system. We found that users often had poorly organized folders or multiple accounts, hence we had to implement support for logins to multiple email sources and detect duplicates and remove them before processing.

We found that users frequently spend hours on the results and visit them in multiple sessions. Therefore we have implemented a feature to save and restore a session once the mining is complete. We found that while users with local accounts from a university or corporation had no trouble accessing all their email, users with several gigabytes of email on public and free email service providers hit a bottleneck. Therefore, caching is essential for a usable system. Muse caches message data once it has been fetched from a server. For example, it may locally store just message headers, or message bodies in addition, or attachments, and it may cache computed thumbnails of attachments. We also attempted to parallelize the fetch with multiple threads, but this led to unexplained server errors with a large online service provider, so we run Muse in single-threaded mode by default.

**TWO USER STUDIES**

User studies are critical in the design of Muse because the quality of such a system is very subjective. We conducted several informal user studies during the development of the Muse system; they were instrumental in the development of many of the ideas presented in this paper.

In this section, we will briefly describe the outcomes from two small user studies that we conducted, and our learnings from them. Due to space restrictions, our discussion will be relatively brief.

**Study 1**

In this study, we wanted to evaluate how well Muse performed in its overall goal of reviving people’s memories using email archives.

**Experimental Setup**

We recruited 8 users (2 female) for the purposes of our user study. All but one were university graduate students, though several of them had work experience in a professional setting as well. The users were from a diverse cultural and linguistic background. Without exception, all the users were heavy users of email.

Users were asked to connect to an email system that would be best for them to browse their past lives, and to run Muse on messages from one year of their lives. We asked them to choose a duration of one year from as long ago as possible for which they still had a significant quantity of email. In many cases, the users’ history of archived email went back about 5 years, coinciding with the rise of free email services with virtually unlimited storage and the emergence of the “Never delete anything!” mindset. We encouraged users to select the folder containing the messages they sent, but asked them to select any other folders as well that had deep or special meaning for them.

Users were asked to use Muse to recall as many distinct topics and events from that time period that were interesting to them. They were given a relatively short time limit of 15 minutes, because our goal is to enable rapid, exploratory browsing that would scale to many-decade archive. Before the task, they were given a two-minute tour of the interface explaining the color coded terms, how to browse messages and how to select terms. They were told they had the option to record the topics they recalled any way they wanted, including using our date-aware diary area, copying and pasting the term into an editor, and jotting them down using paper and pen. All our users ran Muse on their personal laptops, which were equipped with track pads.

**Study Results**

In all cases, the automatic group identification and indexing process was quite fast: under a minute on modern dual-core laptop machines. The indexing thread for users with a large number of messages was memory intensive and was run with a 1GB heap size.

Five users took the entire allotted 15 minutes, while three users finished in 7, 8 and 14 minutes respectively. On av-
erage, Muse helped each user recollect 38 events or topics with a maximum of 59 and a minimum of 24. Cumulatively, our users spent 104 minutes recalling 306 events, at a rate of 2.94 events per minute.

7 out of our 8 users were unequivocal that they would choose a system like Muse over their current email client (Thunderbird, Gmail’s web interface, the Zimbra email client and Apple Mail) for the purpose of reviving memories. One user was split and said he thought he could also get a high-level summary of major topics in his year by looking just at the email subject lines in Thunderbird, though Muse would be more useful for smaller topics that he might have forgotten. In contrast, another user made the observation that looking only at his subject lines in his email client would work relatively well for formal emails in his work account, but would not be useful for his personal email account because “most of my friends send me email with a subject like hello or hi.” On this topic, another user independently remarked that “looking at subjects is less useful the more you know the person ... my close friends’ messages always have weird subjects because we share so much context.”

All users said that Muse revived their memories of topics that they had otherwise forgotten about. Sometimes these were topics in themselves, and sometimes they were satellite topics around other events that they did remember: “I had forgotten about that lunch we organized right before my thesis defense.” Hence Muse can add color and detail even to significant and well-remembered topics.

All our users found the convenience of clipping terms directly into the date-aware diary useful and used this method in preference to any other. While we have not made the diary interface very sophisticated beyond the date feature, this is easy for us to adjust score thresholds and thereby control the noise, we did not have an interface for doing so interactively. Interactive controls are useful where possible, so that users find the thresholds that generate results that they are comfortable with.

We also observed a wide personal variability in the tolerance of users to noisy terms. Some users were very disturbed by common words appearing in the terms list, and repeatedly pointed them out to us. Others barely noticed them. While it is easy for us to adjust score thresholds and thereby control the noise, we did not have an interface for doing so interactively. Interactive controls are useful where possible, so that users find the thresholds that generate results that they are comfortable with.

In the rest of this section, we present some insights obtained by feedback given by users in our reported studies as well as by others who have informally used our system as we have refined it. Some of these insights may provide fodder for future work, including in our system.

Summarizing work progress
Several of our users remarked after using Muse that they would find it useful to summarize their year when writing an annual report, a performance review or going in to a project review. A user commented: “I wish I had this system at project reviews to quickly scan all the project group mes-
sages since the last meeting”.

Extraction and organization
One user suggested that it would be useful to form a group of all personal contacts and use it to take personal email out with her when leaving a job. “My husband was leaving the newspaper company he worked at, and spent 2 days printing out all the personal emails in his work account.” (rolls eyes)

The user who was an entrepreneur said he saved his email mainly because it had important documents, such as “company ownership spreadsheets, benefits packages, legal agreements – stuff that’s there nowhere else”.

Family groups
A consistent pattern was that users tended to spend a lot of their browsing time with their family group(s), perhaps more than any other group. This may have been due to the long-lived nature of such relationships, which makes introspection on them particularly valuable. In addition, it is common for people to tell their family members not living with them about important milestones and events such as job promotions and new romances.

Picking up forgotten threads
A few users remarked that the messages reminded them of unfinished work or projects. Sample comments were: “I’d like to remind my friend that we were planning this trip - wonder why it got dropped and we never went.” Our hypothesis is that users may also find such life-browsing useful as a reminder of high level goals and ambitions they once had. Interestingly, one user reported that she felt a renewed sense of confidence by looking at her past achievements, an observation also made by Kirk and Sellen [7].

Resharing messages
A common theme emerging from several users was that they said they would like to re-share old messages with the people they had sent the message to some years ago. The reasons for this were for fun (Oh, it was fun to see all that discussion we had with the family about possible names for the baby; I’d like to remind them of it now!”) or nostalgia (“I would like to remind my dad of the advice he gave me when I came to the United States for the first time.”) This suggests a use for aesthetic user interfaces that enable re-sharing of email messages from the past.

Renewing relationships
Multiple users remarked after reviewing old conversations that they felt bad they were no longer in touch with people who they had been very close with some years ago. e.g. “I had forgotten that we were such close friends, but then I moved, and we stopped talking.” And from another user: “Wow! I had forgotten how nice one friend was in offering me a temporary place to stay (I ended up staying elsewhere) but she has been a little grumpy lately but I can forgive that a bit now that I remember that incident.”

Serendipitous discovery
One user when browsing messages noticed that her son’s name was part of the message and had a hyperlink. Clicking on the link brought up a view with 224 messages with her son’s name. As she scrolled through the messages, she remarked: “Wow, this offers a pretty complete history of my first son’s milestones. There is no other record of this. I’ve been trying to remember his milestones to compare with my other son’s.”

Summary
From seeing how our users reacted to our system, it was striking to us the variety of ways in which they derive all kinds of utility and benefits from a system that jogs their memory, from summarizing one’s work to renewing friendships and serendipitous discovery. Further, it is clear that remembering the past can affect the future.

RELATED WORK
We have already discussed some related work in context as we described Muse; we now expand to work in other areas.

Email Analysis
There has been much prior interest in email processing, but there are relatively few usable systems available for processing large scale email archives of ordinary consumers. Xobni (xobni.com) is an Outlook plugin that provides email frequency statistics about a contact, a list of exchanged attachments, and shows supplemental information such as the sender’s LinkedIn profile. Rapportive (rapportive.com) is a Gmail plugin with similar features. SNARF [10] and Gmail’s Priority Inbox attempt to infer importance of different senders based on communication history and use this information to alleviate email overload. There have been attempts in the topic detection and tracking (TDT) area towards clustering, automatic foldering and generating topic labels. However, much of this work (e.g. [15]) deals exclusively with clustering message texts. We believe our method of organizing messages based on groups of people is more natural in the context of reviewing life logs, and performs much better.

Life-logging
Most life-logging systems today are geared towards active life-logging, which involves deliberate actions by the user and/or use of specific hardware, software or services (see http://personalinformatics.org for an excellent list). Pensieve is another system that actively solicits input from the user by periodically sending email questions and thus attempts to create a repository of reminiscences. [11]. Our work is focused on passive life-logging which is widely applicable since it involves no extra effort from the user. Mining spending data [13] is another example that is closer to analysis of passive life-logs.

UI and Visualization
The Froggy plugin for Firefox tries to enable rapid perusal of documents by varying font sizes in the text [18]. Parallel tag clouds (PTC) [2] attempts to reveal significant absence as
well as presence of a term across similar but parallel corpora by different circuit courts. Jigsaw [14] is a system for browsing unstructured text documents using entity extraction and linked, multi-display views, especially for use by investigative analysts. ContactMap [17] provides an editable visualization of personal contacts, spatially organized and colored by group membership, but expects users to create them manually, a difficult proposition.

FUTURE WORK
Muse is already available to the public and we intend to scale our current user study by recruiting diverse users via the web. We plan to improve its usability and features based on feedback from users, and conduct detailed user studies of specific user interface elements. A limitation of our current user studies is that we don’t measure recall, i.e., we do not know what important topics and events were not picked up by Muse. A controlled experiment to identify such topics will suggest possible improvements.

CONCLUSIONS
This paper explores the use of an email archive to jog users’ memory of past events. Tools to help users browse and recall events that they may have forgotten are very different from search tools. The challenge lies in making sense of the wealth of information embedded in the gigabytes of email history.

With the help of an iterative design process, we have developed a system called Muse that has shown some success in helping users revive their memories. Muse presents phrases embedded in the email archives that are likely to represent key discussion topics for each interval over a period of time. Phrases are shown only when they first appear. They are color-coded according to groups, which are automatically mined from the email corpus, so users can selectively focus or ignore classes of messages. From the key phrases, the user can launch into an exploration of the actual messages, connected through hyperlinks, as well as the external links and attachments included in the messages. Finally, its on-screen jog dial lets users scan messages rapidly and easily. Our small user studies suggest that Muse is able to help users recall important events in less than 15 minutes.

There are many more improvements to be made to the mining algorithm and aids for data navigation, and many more user studies to be performed as we delve further into each of the aspects of this topic. We hope to see more and more systems built for the purposes of analyzing passively acquired life-logs.

REFERENCES