2007-01-06

MailScape: A Visual Approach To Email Management

Amanda J. Jamin
Worcester Polytechnic Institute

MailScape: A Visual Approach To Email Management

by

Amanda Jamin

A Thesis

Submitted to the Faculty 8 of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

January 2007

APPROVED:

Professor Matthew Ward, Thesis Advisor

Professor Robert Lindeman, Thesis Reader

Professor Michael Gennert, Head of Department

Abstract

Email provides the average person with the ability to instantaneously communicate with people across the planet. However, the ease of this communication not only allows people to exchange information, but can also generate a large volume of messages. The methods of interacting with this repository currently in use are far from adequate.

This thesis focuses on the development of a new interface for managing one's inbox and mail folders. The approach provides the user with a data visualization of an overview of the entire mailbox, or a chosen subset, at once. The tool includes interactions that allow a user to focus his or her attention on specific email and delve deeper into it in a simple intuitive fashion. A user study confirmed the usefulness of the resulting system.

Acknowledgements

I would like to thank my advisor, Matt Ward, for all of his support over the course of both my undergraduate and graduate careers. Without his guidance and our walks through the park, neither my thesis nor my coursework would have progressed as smoothly as they did.

I would also like to thank my reader, Rob Lindeman, for his input throughout my project. His suggestions led to the improvement of my project.

Additionally, I would like to thank my family and my friends. I would especially like to thank my parents for their support of my career choices and encouraging me to follow my dreams. I would also like to acknowledge Tim Walsh for helping me to debug portions of my program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Email has evolved from a tool used by a small portion of the scientific community into a popular means of communication for the average person. In the past few decades email has emerged as an essential tool for businesses, a means of communicating with friends and family, and even a new venue for advertising. As the use of email has increased, so has the number of email messages a single person receives and the number of contacts with whom a given person interacts via email. This creates a large amount of electronic data for a single person to manage.

1.1 Volume of Email

ClearContext [2], a company that develops email management tools, conducted a survey in 2006 in order to determine email trends over the previous year. Their survey revealed that the majority of people receive several email messages daily, with most reporting having fifteen to one-hundred new email messages a day. Some respondents assert to having received between two-hundred fifty and five-hundred email messages daily. A comparison of these results to those of a similar survey conducted in 2005 [1], reveals that these results are not anomalous. People have

consistently received a large volume of email and current trends lead one to believe this will remain true in the years to come.

Although the amount of new email per day has remained relatively constant, the reported number of hours spent reading email per day has increased. In 2005, 45% of participants reported spending one to two hours per day dealing with their email. Thirty-three percent replied spending two to four hours, 8.3% answered four to six hours, and 5.7% claimed that email consumed more than six hours of their day. In 2006 the percentage of people saying they spent one to two hours decreased to 34%, the percent answering two to four hours remained constant, while those reporting four to six hours and more than six hours increased to 12% and 7% respectively.

In addition to an increase in the time spent reading email each day, the amount of email saved in a person's inbox has also increased. In 2005, 23.7% of participants stated they kept zero to ten email messages in their inbox, 27.3% answered ten to fifty, with the rest (49%) answering in one of the options in the fifty to three-thousand plus range. In 2006, the percentage of people claiming to have zero to ten email messages decreased to 18% and those having ten to fifty decreased to 21%. Both of these decreases led to an increase in the percentage of respondents having fifty to three-thousand email messages in their inbox.

Overall, the increase in time spent reading email in conjunction with an increase in the size of a person's inbox has led to people feeling frustrated with email in general. In 2005, approximately one third of respondents stated that email overwhelmed them. However, in 2006, the majority of participants agreed with the statement that they felt they spent too much time with their email. Additionally, a majority answered that email cut into the time they wished to spend on other tasks.

1.2 The Use of Email in Today's Society

Email provides a medium for a variety of tasks. In addition to being used as a mode of communication, email is a method of information management and a tool that enables coordination and collaborating between members of an organization or team [5].

Email began as a means of communication. The first message, sent in 1971, served as a method for a scientist on the ARPARNET project to leave a message for another on a distant terminal [11]. By 1973, email accounted for 75% of the traffic across the ARPARNET. As computers and internet connectivity become commonplace, the use of email as a mode of correspondence has become commonplace. Today email is vital both for business and personal communication.

In addition to being used for communication, email is used as a method of accomplishing information management. Generally, email users keep messages regarding tasks they need to complete and long messages that they have not had time to read. The first set of email is similar to a "To Do" list. The presence of the email itself serves as a reminder of the task. The second type of email generally is informational. However, despite their importance, their length often prevents the recipient from reading them immediately due to a lack of time [20].

Email is also used as a means of collaboration. Today, email is one of the primary means of file transfer. Users of email send meeting agendas and assign tasks to others via email. Additionally many document their activities electronically through email attachments. Many choose to use email for these purposes as these tasks are generally related to communication. Typically email applications allow for a fluid method of accomplishing activities without having to switch between multiple applications [5].

1.3 Problems With Today's Email Clients

Today's email clients have taken advantage of the development of graphical interfaces. Although still used in programs such as Pine, pure text based email interfaces are rarely seen outside of academia. However, although applications that utilize graphics and pointer devices have replaced these shell based clients, the interface layouts have, for the most part, remained the same.

The main issue with this style of email client is the feeling of frustration that a user feels when using it. The majority of people who use email experiences overwhelmed with the experience [20]. The process of organizing, and later finding an email within the filing system, is futile as often this process causes them to lose email messages instead. Although typical email clients allow one to create and maintain a folder hierarchy, often times users have problems with this interface. Both the creation and the maintenance of this hierarchy are time consuming, and often difficult. Furthermore, it has been concluded that users want immediate access to information and due to this desire, the depth of a usable folder hierarchy is limited [5].

A second feature of the typical email client is automatic filtering. This allows users to set up a system that places new email in folders, other than the inbox, when received. This feature is intended to limit the amount of filing that users have to actually do themselves. The first issue with filtering is simply the process of setting it up. The majority of users either do not know how to use filters, or feel the majority of their email cannot be filtered [5]. Among those users who successfully set up filters many often do not notice that the email has been received. This causes them to lose email messages before even knowing they exist [5].

1.4 Proposed Solution

The goal of this thesis is to develop a solution that will take advantage of data visualization more than any other client on the market today. Users will be presented with a "wave form" graph of their inbox or mail folder. In this visualization, each *wave* will correspond to a single thread. For the purposes of our application, a thread is defined as a group of email that have the same value for a user selected attribute.

This graph will be fully interactive, allowing the users to visually sort and filter their email. Choosing a particular wave will generate a *glyph view* of the email messages belonging to that thread. Clicking on a particular glyph will present the user with the underlying email message, allowing the user to preform all of the typical email interactions. It is our belief that users will be able to locate particular email messages more easily when they are presented information in this manner, as the email they desire is more likely to visually stick out.

Furthermore, we believe that our graphical interface will provide users with abilities that are currently missing from existing email systems. As the wave form view will display time, trends in one's mailbox will be visible. Users will be able to determine the frequency at which they receive email from a particular person or on a given topic. It is our belief that providing users with access to this information will lead to more efficient email management.

Sorting and filtering will cause the graph view to change, causing email messages which meet the users' requirements to be grouped together and colored similarly. This will not only allow the users to find email sent by the same person or with the same topic, but to see how these characteristics change. This will allow users to identify contacts with whom they have not recently exchanged email. It will also

reveal cyclic topics.

Chapter 2

Related Work

In order to create a useful application to visualize the contents of a person's email collection, we researched existing software hoping for insight into how our application should be designed. Our search revealed existing applications that visualize email as well as other visualization styles that could be applied to our problem. Each of these systems is outlined below.

2.1 Data Visualization of Email

Although the majority of email management tools have not focused on visualizing email, there have been several applications that have. These applications generally follow Shneiderman's mantra of "overview first, zoom and filter, then details on demand," [15]. The user is initially presented with a visualization that serves as a overview of the underlying inbox. Interacting with the visualization allows the user to find and read a specific email.

2.1.1 ReMail

IBM's TJ Watson Laboratory has explored methods of improving the user experience of interacting with email. The cumulative research has led to the development of the ReMail application [10]. As part of this mail client, IBM integrated email visualizations into the application's interface. When an email is selected, a thread view of the message is displayed on the left pane of the window. This visualization is based on IBM's previous research on Thread Arcs [9]. In Thread Arcs, each message in a conversational thread is displayed as a circle. The circles are ordered in a manner that places the circle corresponding to the oldest email on the left and the newest on the right. There are arcs which connect a message to the message for which it is a reply as shown in Figure 2.1. Thread Arcs allows a user to optionally

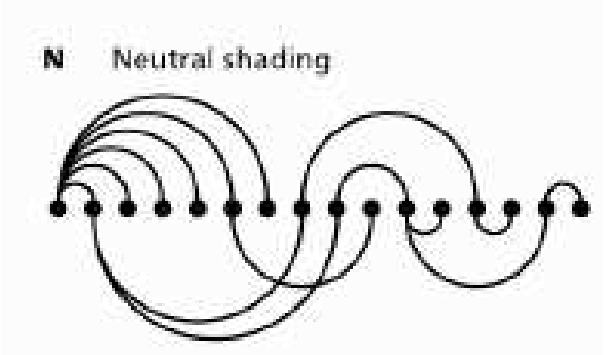


Figure 2.1: Thread Arcs [9]

color code the visualization by author. In this scheme, all messages generated by a single author are the same color. Additionally, Thread Arcs supports a variety of highlighting schemes allowing the contribution of a single email to a thread to become apparent.

A second visualization integrated into ReMail is a Correspondence Map (see Figure 2.2). This visualization displays all of the contacts within a person's mailbox. Each contact is drawn as a rectangle inside which the person's name is written. The

names are ordered by the amount of email they have sent. The person who sent the most email has their respective square drawn at the top of the visualization. Color is mapped to the age of the most recent message from that contact. If any of the messages are unread, the contact's square is colored black. Otherwise, the square is colored a shade of gray. The particular shade of gray is determined by interpolating the shade in relationship to the oldest email (shaded white) and newest message in the mailbox. An example Correspondence Map is shown in Figure 2.2.

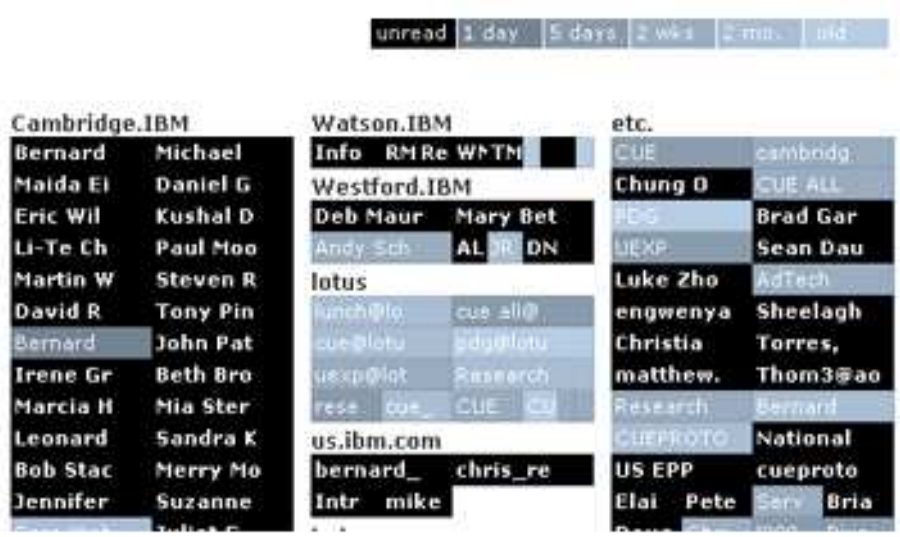


Figure 2.2: An Example Correspondence Map [10]

The contacts are visualized by the folder in which the email messages they sent are organized. The squares towards the top of the visualization, such as Michael and the first Bernard (shown in the top left) are the contacts in their respective folder (Cambridge, IBM) from whom the largest amount of email has been received. The black squares in this visualization indicate that the user has new, unread email from the respective contact. In the Cambridge, IBM folder, there is a new email from most of the contacts. One of the contacts, a second Bernard (seventh contact from the top in the far left column), is shown as a dark gray color. There is not a new

message from this contact. However, the most recent email from them is newer than any of the email messages in the lotus folder. Each contact square in the lotus folder is colored a light grey.

ReMail's final visualization is a Message Map [13]. As shown in Figure 2.3, a Message Map visualizes every message in the user's mailbox at once. Each message is drawn as a rectangle. The message currently selected is drawn in blue. Messages that belong to the same thread as the selected message are colored with a lighter shade of blue. From this view, a user can perform a limited search of the messages. During the search process, the user selects an email attribute (same author, unread, sent, attachments) and chooses to mark messages that match the criteria. The messages can be marked either by being colored orange or marked with a *dog ear*. A dog ear is a black triangle "folded" over the corner of the square analogous to how a dog's ear flops over his head. Messages that do not match the criteria are shaded grey.

2.1.2 TheMail

MIT's Media lab developed a visualization for exploring the underlying themes that exist within an email collection. Their application, appropriately entitled TheMail, renders email topics as words. There are two types of topics visualized, yearly topics and monthly topics. The yearly topics are displayed in the background in light grey. These words represent the themes that appeared within email with the greatest frequency over the course of the entire year. The monthly topics are positioned along the x axis according to their month and all of the topics for that month rise vertically along the y axis. Figure 2.4 shows a screenshot of TheMail. In this screenshot, the large light gray words in the background, such as "bhangra" and "please" are yearly topics for the application. The columns of smaller words are

Figure 2.3: A Message Map [10]

the monthly themes. At the bottom of each column, the month and year that the column represents is indicated. The size of the font in which the theme is rendered corresponds to the frequency and the uniqueness of a topic. A large font indicates not only that the message occurred often over the course of the month, but that the theme tended to appear in messages from a small number of senders. Generally, this is a sign of a single large thread. An example of this, is the word "performance" in Figure 2.4.

In addition to the words displayed, TheMail draws a circle for each message in the dataset. The size of the circle corresponds to the size of the underlying email. Larger circles represent physically larger email messages. The color of the circle represents whether the email was an incoming or outgoing message. In the

Figure 2.4: A Screenshot of the TheMail Application [17]

previously mentioned screenshot (Figure 2.4) these circles are drawn in red and white, with red circles indicating received messages and white circles representing email messages that have been sent.

TheMail allows users to filter the displayed dataset. In addition to limiting the themes displayed to those that begin with a specific substring, the date range that is displayed can be adjusted with a scroll bar. Clicking on a particular theme creates a listing of all of the email that contain that theme. This column can be seen on the right hand side of visualization in Figure 2.4. From this listing, the user can choose a particular email to read. This email is then shown on the left side of the visualization. Words that indicate the selected theme are enlarged in the message, such as the word performance in the gray box on the left of Figure 2.4.

2.2 History Flow Visualizations

History flow visualizations provide a method of visualizing broad trends in history while preserving the details of the dataset [18]. A history flow visualization is constructed from a time series dataset. In this dataset, for a finite number of points in the time interval, the frequency of each of the possible values of the data are known. The data points for each value are then connected, allowing the trends over time to be more apparent. For example, Wattenberg et al. developed a history flow

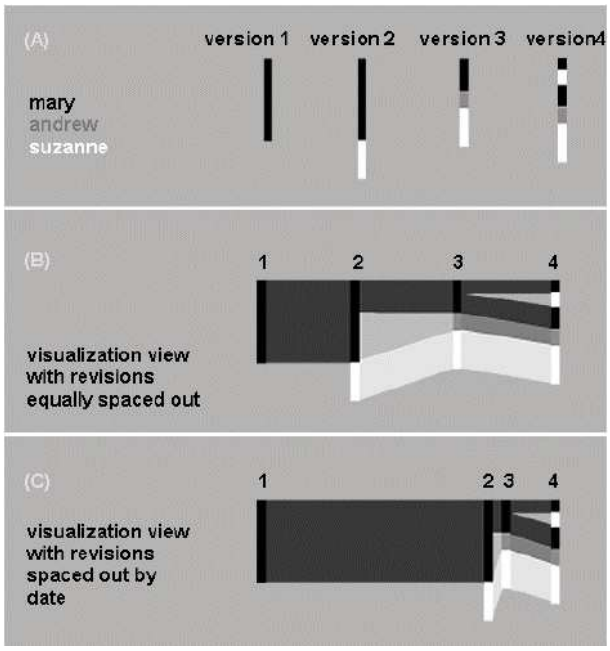


Figure 2.5: An Example History Flow Visualization [18]

visualization that focuses on the changes, and the user who writes said changes, for a Wiki site [18]. Wikis are webpages whose contents can be authored by any of the page's users. Figure 2.5 displays a history flow diagram for a hypothetical series of contributions. In segment A, each of the vertical lines represents a time at which revisions were made to a given Wiki site. The height of the bar represents the total length of the document. The shades in the band indicate which person

authored a given segment of the document and the height of a shaded bar suggests the percentage of the document he or she has authored.

In segment B, the time between revisions has been interpolated in order to reveal trends within the revision history. You can see that between Version Two of the document and Version Three, Andrew deleted a substantial percentage of the original information written by Mary. This portion of the Wiki was replaced by a relatively small amount of information.

Prior to segment C, the times between revisions are evenly spaced. In segment C, the space between the revisions is proportional to the amount of time between the revisions. This allows a user to identify how rapidly a Wiki changes. Additionally, using this version of the visualization a user can group changes that occur within the same time period.

2.2.1 NameVoyager

The primary inspiration for our approach to email management is NameVoyager, an application for visualizing the trends of naming children [19]. NameVoyager creates a band for each name representing the popularity of that name over time. The bands are then stacked upon each other forming a history flow visualization. Time flows along the x-axis while the y axis represents the frequency for all of the names currently displayed. The visualization of the entire baby name dataset is displayed in figure 2.6. NameVoyager allows users to explore the provided dataset via filters. There are two filters implemented, one which filters the set based upon the name and a second that filters by gender. The name filter is implemented as a keystroke based filter. As letters are typed, the visualization is updated so that only names which start with the particular character sequence are displayed. The gender filter is implemented simply as a radio button, allowing a user to select whether he or she

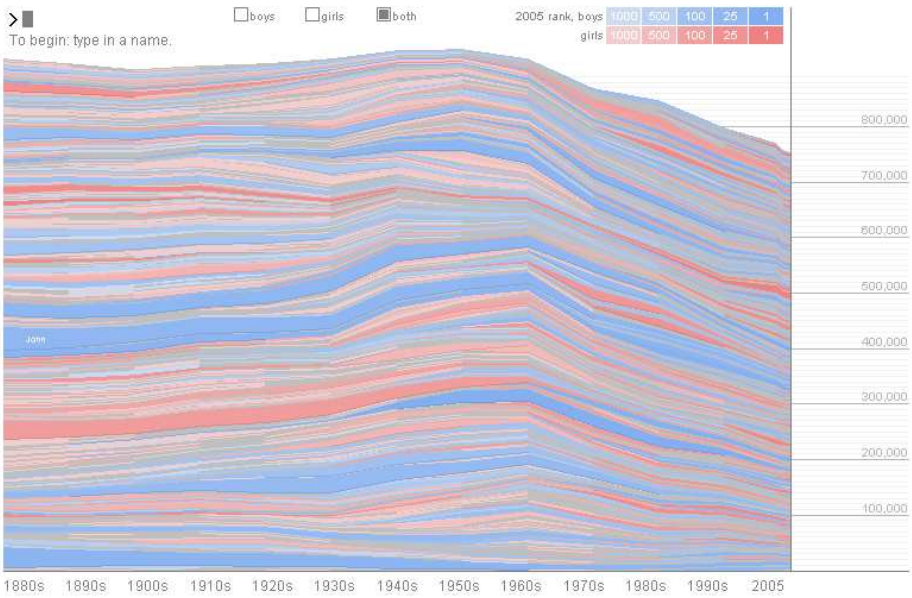


Figure 2.6: NameVoyager [19]

would prefer to see female names, male names, or both.

In addition to filters, NameVoyager allows one to discover detailed information regarding each name. Hovering over a band causes information related to the band to appear. Specifically, the name corresponding to the band as well as the popularity of the name in the year mapped to the cursor's x position are displayed. Additionally, the entire band is highlighted making the trend of the selected name more apparent.

NameVoyager proved to be of interest even to users who had no interest in the underlying dataset. The majority of the users seemed to be outside the target audience, with some even claiming to hate babies and children. The users of the system self proclaimed it to be addictive. The system influenced not only individuals, but social groups as well, with some users claiming that use of the system affected the productivity of their workplace. The popularity of NameVoyager is attributed not to its motivation, but to the design and implementation of the final visualization.

2.3 Document Collection Visualizations

Visualizing email datasets is not a unique problem. Email itself is a collection of messages with a defined order, the date on which it was sent. This feature of email allows one to apply existing techniques for visualizing document collections to the application of email visualization.

2.3.1 ThemeRiver

ThemeRiver [7] is a prototype system whose goal is the visualization of the underlying themes, and changes in those themes, of a collection of documents. The design of the system is based upon a single metaphor. The documents form a river. Each theme is a current in the river, growing in strength when the theme has a strong presence and diminishing in size when the theme grows weak. ThemeRiver presents

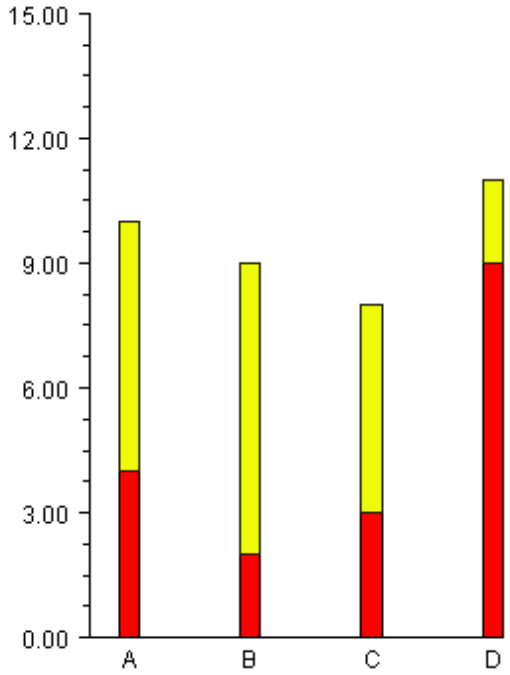


Figure 2.7: A histogram alternative to ThemeRiver

an alternative visualization to a histogram. In a stacked bar histogram, like the one shown in figure 2.7, each current becomes a color. The same color is used for each value at each time interval (A, B, C, D) in order to allow the user to analyze trends in time. Variations in the height indicate the strength of the value. Although this visualization is capable of displaying the same information as ThemeRiver, it is not as effective. Trends over time must be deciphered by the user, they are not intuitive. Additionally, the position of each color within the bar varies from time interval to time interval. This can make the visualization difficult to read. ThemeRiver solves

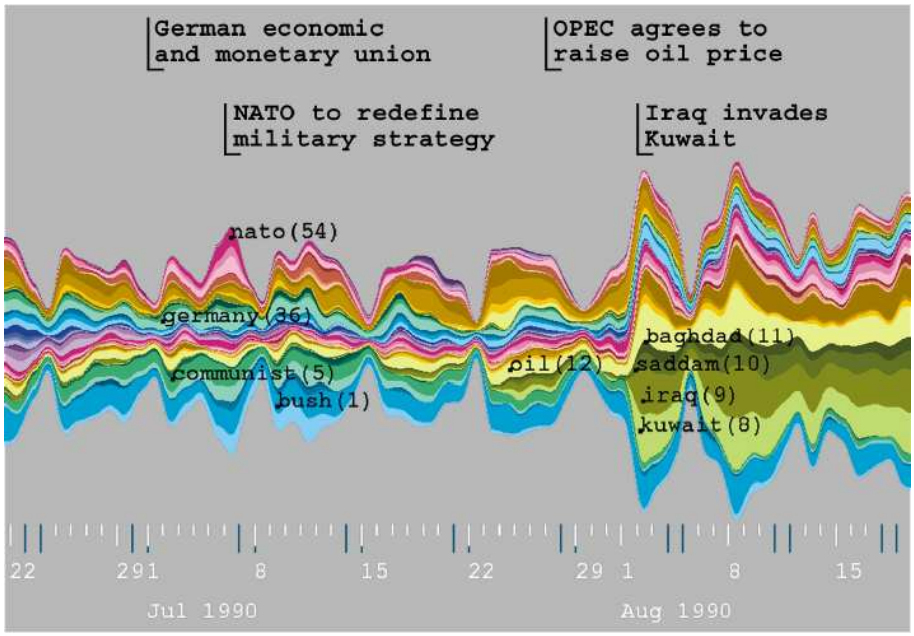


Figure 2.8: ThemeRiver [7]

both of these problems. As the currents are continuous, trends in the themes can be intuitively analyzed. Furthermore, it is easy to find each trend's location at each point in time. Although the position of a theme's current can change over time, the order of the themes remains the same. Additionally, the continuous nature of the visualization enables the user to find the theme even when its position changes. Evaluations of the system revealed ThemeRiver to be very effective at revealing

macro trends. However, the application tends to de-emphasize small values causing minor trends to be missed. The main complaint of users of the system is that ThemeRiver does not allow the user to access the document from the visualization.

2.4 TreeMaps

TreeMaps are a type of visualization that spatially organize data into rectangles. The size, color, and position of the rectangle can all be mapped to separate attributes. Originally, TreeMaps were designed as a method of turning a tree into a planar space-filling map [16]. Since their initial development, the applications of TreeMaps have been extended beyond datasets with an obvious tree structure. One such application is the visualization of compressors at oil plants [12]. At the Society

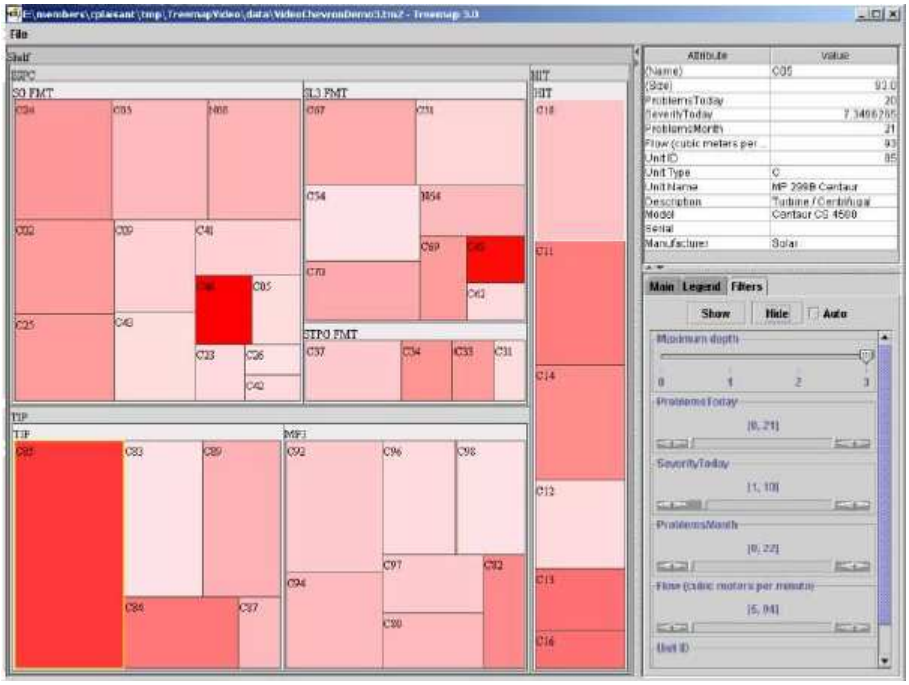


Figure 2.9: TreeMaps applied to visualizing petroleum compressors [12]

of Petroleum Engineers conference in 2003, Plaisant et. al. presented a TreeMap

visualization of the productivity of a set of oil compressors. A screenshot of this visualization is shown in Figure 2.9. The dataset displayed by the figure is comprised of sixty-eight compressors. In the visualization, they are grouped together in a hierarchy first by the field in which they are located, then by the asset team, and finally by the region. A rectangle in the visualization maps to a compressor, with the size being proportional to the flow rate. The shade of the rectangle is mapped according to the number of abnormal compressor parameters.

The visualization in Figure 2.9 makes the analysis of an oil field easier on an engineer. For example, compressors that need to be looked at are shown as large red rectangles. Altering the parameters of the visualization, mapping different values to attributes of the rectangle, allows the same visualization to be used for other daily needs. The visualization can be modified to reveal "lost" production as well as which oil wells produce the largest number of barrels. Such a visualization would be useful when writing daily production reports.

Like most visualizations, the shown TreeMap allows users to interact with the data. This application allows the user to both zoom in on the dataset and filter the displayed data. The shown TreeMap allows users to filter via sliders. These sliders can be seen on the right side of the visualization in Figure 2.9. As users adjust the position of the bar, the rectangles that fall outside of the range become grayed out. This method of filtering is both dynamic and intuitive.

Chapter 3

Solution Design

It is our conjecture that an email management tool that effectively utilizes visualization techniques will prove to be a better system for filtering and sorting through a large inbox or mail folder. To prove that this hypothesis is valid, we created a proof-of-principle application. This prototype, which we named MailScape, will allow users to manage their inbox in a completely graphically manner, including the reading and deletion of an email message. After development, we will validate our claim via a user study.

3.1 Application Goals

The primary goal of our initial implementation is to create a visualization of an email dataset. This visualization should reveal a user's email habits and how they change over time. For example, a user should be able to determine when they receive the largest amount of email by volume. Additionally, a user should be able to discover cyclic topics. A cyclic topic is a group of email with the same general subject that are received at approximately the same time period either each day, week, month, or year. Examples of cyclic topics are weekly meeting reminders and holiday greetings.

Another goal of our visualization is the easy identification of important conversational threads. It is the nature of email that some messages, and the responses to said messages, are more significant than others. At the same time, users tend to receive important email less frequently than messages of minor importance. This causes the crucial email to get lost within an user's archives. Making the search for these email messages simplistic will enable users to better utilize the time they spend using their mail client.

On the reverse side of the spectrum as our previous goal, we believe enabling users to find and recognize junk email to be a necessary feature of our system. As we previously discussed, one main problem with the use of email today is the sheer volume of messages, both received and stored. Allowing users to discover messages that are either unimportant, or simply garbage, will empower users to delete these messages which, will in turn, decrease the volume of their inbox.

In addition to allowing users to handle their inbox from an abstract visual perspective, we believe users need to be able to handle their messages on an individual basis. In order to make users want to use our application, it is fundamental that users be able to look at an individual message. A user should be able to read the message, delete it, and reply to it. Additionally, a user should be able to write and send a new message. Without these primary interactions that are integrated into even the most basic email applications, we hobble our users' ability to use email as a means of communication.

3.2 Potential Solutions

In designing our application, alternative approaches were discussed. These approaches, for one reason or another, were determined to be less adequate than our

final solution. Each of these approaches are outlined in the sections below.

3.2.1 TreeMap Approach

The primary design decision for our application was how to present users with an overview of their inbox. One approach that was discussed was based upon TreeMaps [14]. TreeMaps are a form of space-filling visualizations for displaying tree structures. TreeMaps have, however, been applied to applications that do not have an obvious tree structure, such as visualizing coffee varieties and personalities [16]. An example TreeMap visualization is shown in Figure 3.1. In this approach to email visualization,

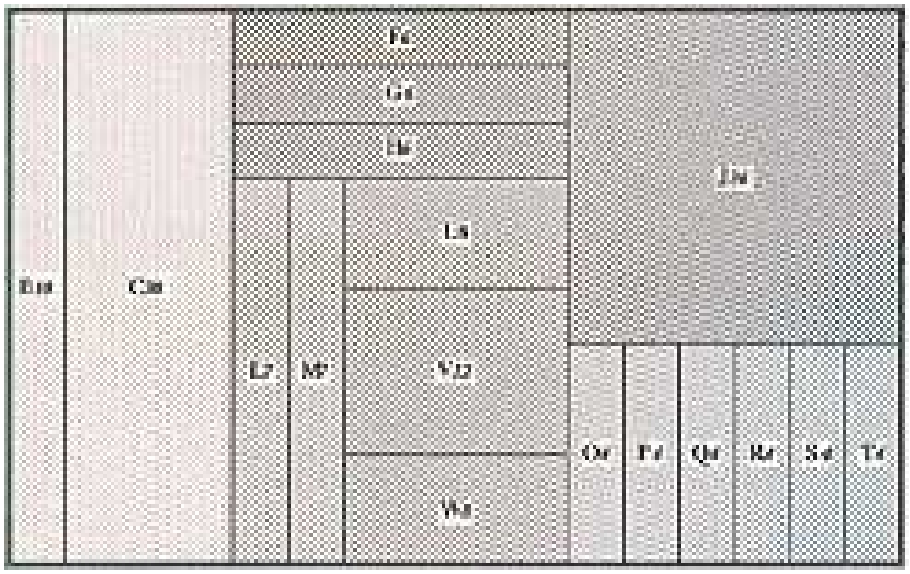


Figure 3.1: A TreeMap Example [8]

email would be divided into boxes based upon their similarities. For example, the email may first be divided horizontally into columns by their date. These columns would themselves be divided vertically by their subjects. The visualization would recursively continue to subdivide the screen into squares based upon different email attributes. The number of recursive steps would have to be limited based upon the

size of the smallest division. It would be useless to visualize squares that are too small to be seen. Zooming in on the visualization and filtering the dataset would allow for additional recursive levels to be displayed.

The benefits of applying TreeMaps to the visualization of email is the flexibility of the visualization. It would be easy to allow the user to control what each email attribute is mapped to. This would create a completely customizable application. Generally, configurable systems receive positive evaluations from their users. Additionally, the ability to customize the application would allow users to mine through their email in a variety of methods.

Although applying TreeMaps to email visualization initially seemed a natural approach to solving the problem, the solution does not seem to be scalable. An email dataset with a lot of dissimilarity would lead to a visualization with a large number of divisions at each recursive level. In this application, the resulting rectangles would be small. The visualization would be as cluttered as traditional email clients.

3.2.2 Spire Chart Approach

A second alternative that we considered was a spire chart visualization [3]. A spire chart visualization displays a histogram, where each column represents a week during the time interval shown. Rather than represent each value as a bar, as in a traditional histogram, each value is shown as a sphere. The size of the sphere is relative to the size of the underlying data point. An example of a spire chart is shown in Figure 3.2. Applying the spire chart visualization to the email dataset, each sphere would represent a email thread. The size of the sphere would be mapped to the number of messages in the thread. Each sphere would be positioned either according to the date of the oldest email in the thread, the date of the newest thread, or the average date within the thread. As shown in Figure 3.2, the spheres are distinguishable from

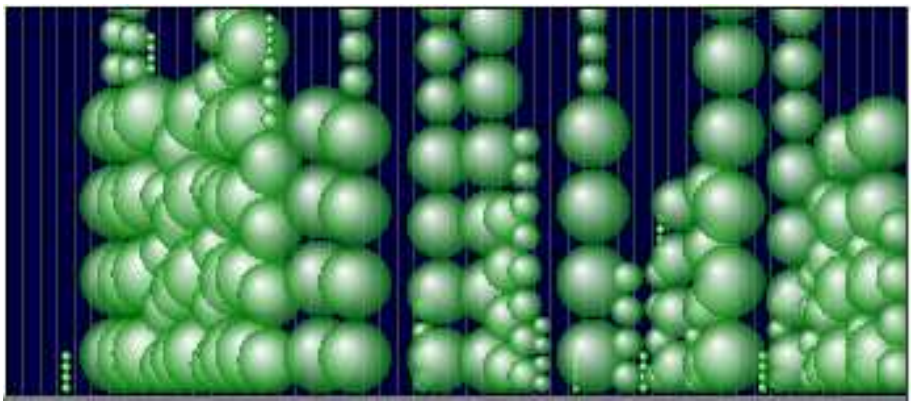


Figure 3.2: A Spire Chart Visualization [4]

each other without them being separate colors. As color is not needed to separate threads from each other, color can be used to indicate if the thread contains a new message.

The spire chart approach would successfully display an overview of an email archive. One would be capable of revealing patterns in their email habits as they would be able to compare weeks to one another. However, users would only be able to analyze their email by thread. As a thread would be positioned by a single date , a user would not be able to determine when each email within the thread was received. Additionally, the growth and lifespan of a given thread would be unknown. Trends, such as which day a user receives the most email, would be difficult to decipher.

3.3 Our Solution

With the aforementioned goals in mind, we designed our prototype application, MailScape. Our design presents the user with three views to their inbox: a history flow view, a glyph view, and a message view. Of these three perspectives, the history flow view will act as the highend visualization, providing a summary of the inbox contents. The glyph view and message view will each, in turn, provide more

information regarding each email.

3.3.1 History Flow Window

The initial view of our application should present a visualization that provides an overview of the contents of the user's inbox. Inspired by the work of NameVoyager, this view will be a history flow representation of the email dataset. The bands, or threads, in this visualization will be created by grouping together like email. As email messages are complicated objects, with each email having several attributes, the determination of what makes two email messages similar enough to group will be defined by the user. In our initial system, email can be classified as being similar if they either share the same read status, the same subject, the same sender, or the same recipient. When grouping messages by a string based field, such as the subject, the sender, or the recipient, only exact matches are considered the same. An exception to this policy occurs when grouping messages by the subject. In this scenario not only will the exact matching subject be considered the same, but subjects that are identical except for the prepended 're:' string that indicates a reply.

Each band will be assigned a color allowing one to distinguish between adjacent threads. The color of a given thread will be chosen from a list of thirteen colors in a manner such that no two adjacent bands are initially the same color. At some point in the graph, two adjacent bands may be the same color if all of the threads that were between them have a width of zero during that interval. As we could not prevent this from occurring, we included the ability to highlight the entirety of a band by hovering over a section of the band. This will allow user to distinguish between adjacent bands of the same color.

Unlike in NameVoyager, where there was a natural manner of sorting the bands

(alphabetically), there is no such instinctive ordering involved with the email threads. Due to this, we will provide the user with two different options of sorting the bands – alphabetically and by the number of messages belonging to the thread. In the alphabetical sorting of the bands, the field by which the threads are grouped together is used for the purpose of sorting. When sorted according to the number of messages in the thread, the thicker bands fall towards the bottom of the graph. These bands, due to their width, are "heavier" than other bands in the graph.

The x axis of this visualization, as in NameVoyager, will represent time. However, our visualization will allow users to modify the amount of time, and the number of time divisions, displayed. Initially, the visualization will represent all email from the oldest to the newest, with the amount of time between these two dates divided into four equal divisions. In our system the user can change the minimum date and maximum date shown. Additionally, they can change the number of divisions into which the time span is divided, with a maximum of thirty. Imposing this maximum enforces an upper bound on the drawing algorithm. The more divisions in which the time interval is divided, the more details are revealed by the visualization. However, as the lifespan of a thread is generally short lived, increasing the divisions will also increase the number of threads whose width becomes zero for one or more intervals.

In addition to being able to alter the date interval displayed, the user will also be capable of filtering the bands by the attribute by which they are grouped. In this filtering process, unlike in the grouping process, partial matches are acceptable. For example, when the messages are grouped by subject and a user filters by the string "soccer", all bands whose subject value contains "soccer" as a substring are displayed.

The History Flow perspective of MailScape allows for the partial fulfillment of our requirements. As it displays an overview of the email dataset, email trends

are more easily visible than in traditional systems. The ability to modify the time displayed allows one to further analyze trends they may see in the original display. Control over the time displayed grants users the ability to make each division equal to a single month, week, day, or even hour. Users can compare the email patterns of multiple time intervals simply by altering the minimum and maximum dates while leaving the number of interval divisions the same.

Additionally, the ability to change the grouping values and filter messages from the Message View allows a user to focus the visualization. These actions decrease the number of messages displayed. If a user is looking for a specific important message, they will be able to limit their search to a subset of their inbox quickly.

The History Flow view is not without its limitations. For users who are not the typical user and keep a relatively small amount of email, this visualization will be ineffective. A small email dataset has few threads, regardless of the grouping. As these users often delete older email messages, these datasets do not span a long period of time. Generally, the time interval they span is relatively recent, making it difficult to discover email trends.

3.3.2 Glyph Window

Selecting a single band from the History Flow view creates a second visualization. In designing our system, we knew we needed a method of allowing users to explore the contents of a single thread. Without such a view, it would be difficult to provide the user with a manner of interacting with a single email, a vital component of our application.

Design Process

The visualization for this view would need to focus on the properties of each message within the thread. Only by focusing on the email properties would a user be capable of identifying the message they desired to read. Analyzing the properties of an email, we determined the sender, recipient, and the subject to be necessary attributes to visualize. In addition to these attributes, we thought it would be beneficial to be able to display other properties such as the new status, junk status, whether the email has attachments or not, and whether or not the email has been replied to. It was also important to us that, if it was desired, the visualization be expandable to allow more attributes to be shown in future implementations.

While designing this portion of MailScape, our original thought was that the second visualization should be similar to the first. Implementing a second History Wave view would display threads within a thread. For example, when a user selects a band representing a group of email with the same subject in the initial History Flow view, they would then be capable of grouping these messages by sender in the second view, enabling them to analyze the participants in the conversation. However, further investigations into implementing a History Flow visualization as the second visualization revealed some problems. The primary issue is that the visualization would not focus on a single email as we intended it to. Second, we believe that the visualization would have broken down for small threads. This would prevent a user from reading an email until it became a part of a larger thread. We determined this to be unwanted and unacceptable requirement.

Following our decision to implement a visualization other than a History Flow, we investigated implementing a glyph based visualization. In this view, each email would be represented by a single glyph. Initially, we considered designing a glyph based upon a generic shape, such as a circle or a square. However, the majority

of shapes did not have enough characteristics to visualize each of our required attributes. Those that did, we did not consider able to be expanded to include the visualization of additional email attributes. We also considered implementing glyphs based on images, such as an envelope. However, as we began to map email attributes to envelope characteristics, we determined these images to be too complicated for the typical user to understand. Finally, we considered star glyphs. As star glyphs allow for multiple points to be displayed, we considered them to be expandable. Additionally, as the graphic of a star is simplistic, we did not feel the image itself would be difficult for a user to comprehend.

Final Design

In our glyph visualization, a single glyph is drawn per email in the thread. These glyphs are positioned according to the date on which they are sent. Initially, the visualization is drawn so that all of the glyphs are visible. We believe that, as each glyph is drawn in a manner that reveals the properties of the underlying email, that users will be able to determine details of the email prior to choosing to read it.

Each of the glyphs will be colored according to a provided key. Glyphs that are drawn in red will indicate a new email. The lines that make up these glyphs will also be thicker. This will allow unread messages in a thread to stand out. Glyphs that are colored white will indicate a message that has been read previously. Similarly, yellow glyphs will indicate a message that has been marked for deletion.

In complying with our requirements, the glyph view allows a user to identify an email as junk simply by the drawing. This ensures that a user does not have to open the message to check that it is junk. One can double check that the message is junk by looking at the subject and sender, both of which are visible by hovering over the glyph in question.

Threads that are composed of a large number of messages tend to cause the glyph perspective to lose its usefulness. The large number of messages within the thread map to a large number of glyphs in the visualization. This creates a visualization that is cluttered. Additionally, as all of the glyphs in the thread are displayed, the glyphs themselves can be small in large threads. Although zooming in on the visualization is possible, the zooming process equally zooms in on each glyph. A user can choose to move undesired glyphs around the screen, however this process can be time consuming.

3.3.3 Message Window

Selecting a glyph causes a third window to appear. Within this window, the contents of the email message are displayed. In addition to being capable of reading the message, from this window the user can reply to the message or mark it for deletion.

Chapter 4

Results

We initiated development with the implementation of the History Flow visualization. This was followed by implementation of the glyph view and, finally, the implementation of the message view. In the end, the entirety of the proposed solution was developed.

4.1 Method of Implementation

The code base was implemented as two systems, the backend and the GUI. Although physically separated, portions of the frontend are entirely dependent upon the backend. For this reason, the backend was implemented and fully tested prior to beginning to implement the frontend of the system.

4.1.1 Backend Implementation

The backend consists of objects that are designed to enable the drawing of the visualization to run efficiently. Four classes were designed and implemented to represent the underlying objects involved in the visualization. These classes model

a *graph*, a *thread*, a *thread segment* and a *message* respectively. The graph object, in addition to general information about a graph, contains a list of threads. Each thread object was designed to contain the attribute value that each message in that thread shares as well as a list of thread segments of which it is comprised. A thread segment represents the messages that belong to a thread for a given time interval. Multiple thread segments make up a single continuous thread. Within each thread segment object is the time interval that it represents in addition to a list of messages that belong to it. The message object contains the subject, sent date, and all other attributes of an email. The message object does not contain the body of the email. For efficiency purposes, it has been decided that we should not retrieve the body for every email message. Instead, we retrieve only the email bodies for messages that users request to read.

In addition to the aforementioned objects, the backend consists of a global list of messages and a mapping of attributes to messages. The global list of messages is generated once upon loading the application. By communicating with the server, each message in the user's mailbox is retrieved, constructed into a message object and then added to this list. After all messages are loaded, this list of messages is used to construct a set of mappings of attributes to messages. There is one mapping per grouping in the visualization. In the current version of MailScape, there are four mappings: one to map subjects to messages that have that subject, one to map senders to messages that have that subject, one for mapping new status to messages, and similarly, one to map recipients to messages. These mappings, although not essential to the application, speed up the switching between groupings.

In the initial implementation, these mappings were not physically stored. Instead, each time a user switched between groupings, these mappings had to be determined. This required iterating through the list of messages, finding the distinct

values for the selected attribute, and grouping the messages with this value together. For large mailboxes with a lot of dissimilarity, this was a time consuming process that was evident in the execution of the application. In the initial view, the messages are always grouped by subject. To draw the visualization the first time with this implementation, the runtime is m*n(m+1) where m is the number of distinct subjects in the email archive and n is the amount of email. The possible values for m range from 1 to n. Each time one switches between groupings, the implementation requires a setup time prior to drawing the graph. The setup time depends on the grouping chosen. The setup runtime is m*n(m+1) to switch to a group by subject view. The variables in this equation are as defined previously. When switching to a group by sender visualization the runtime is (j*n(j+1))/2. The variable j represents the number of unique senders in the message list and n the amount of email in the archive. When grouping the messages by recipient the runtime is (k*n*L(k+1))/2 where k is the number of unique recipients, n the amount of email, and L is the average number of recipients per email. To switch the grouping such that messages are grouped by their new status, the runtime is simply n where n is the amount of email. Experimentation with this implementation revealed this additional setup time to be noticeable by the user.

The second implementation calculates the mappings for each of the possible attributes once, when the list of messages is created. These mappings are then stored in memory. This process creates an overhead at the application's startup time. Rather that the initial runtime of m*n(m+1) as in the previous implementation, the runtime is n(m(m+1) + j(j+1)/2 + k*n*L(k+1) + 1) where each of the variables are defined as mentioned previously. This is a one time increase in the execution time. In this implementation, the runtime necessary to switch between groupings requires no additional setup time, opposed to the incremental increases to the runtime in

the previous method. The version of the application does, however, require a larger memory footprint. It is our belief, that the improvement in the user experience is worth the memory increase.

4.1.2 Frontend Implementation

The frontend of the application contains the functions and algorithms necessary to draw each of the visualizations. There is a set of functions used to draw the History Flow view, a second set for the glyph view, and a third that are used to render the message when a user chooses to read it. In addition to drawing all of the visualizations, the frontend handles all of the user interactions with the system. These include key presses and mouse movements.

In order to draw the History Flow visualization, the frontend maintains a single graph object. When a user changes the date range, the group by values, the interval divisions, or any other variable that alters the manner in which the visualization is drawn, the appropriate value is changed in this object. When the grouping value is changed, the back end uses the appropriate mapping to generate all of the underlying thread and thread segment objects. Once all of the values in the graph object are created, or simply updated, the screen will refresh and redraw the visualization.

The actual drawing of the History Flow visualization simply iterates through each thread in the graph object and draws that thread. To draw each thread, each thread segment is drawn. A thread segment has a particular x value and a y minimum and y maximum value. The y minimum value is equal to the total height at the current x coordinate of all threads drawn below the current thread. The y maximum is the y minimum plus the height of the thread at this moment. In order to draw a thread segment, the x position of the previous thread segment, as well as that segments y minimum and y maximum, must be known. The function remembers these values.

For each thread segment, a filled-in polygon is drawn connecting the four points (x1, y1), (x1, y2), (x2, y3), (x2, y4). In the preceding coordinates, x1 is the x position of the previous segment, x2 the x position of the current thread segment, y1 and y3 are the y minima of their respective segments, and similarly y2 and y4 are the y maxima of their respective segments. The amount of time required to draw the History Flow visualization is quadratic proportional to the number of threads in the current grouping and the number of interval divisions.

The threads are scaled in the visualization. The tallest peak of the visualization is drawn such that it covers the entirety of the window height. All additional peaks are scaled accordingly ensuring that the height for each email is the same throughout all bands. When filtering actions are preformed on the dataset, the graph is scaled again. Once filtering has occurred, the local maximum is drawn to the full height of the window.

The glyph view visualization only takes into consideration the messages in the selected thread. Upon being called, the function concatenates each of the message lists within each thread segment into a single list. Once this list has been created the algorithm determines first whether the thread spans years, months, days, or hours. This allows the glyphs to be positioned according to their date. If the email thread spans months, but are all in the same year, the x axis is divided into twelve columns. Each glyph is drawn within the column representing the month in which it has been sent. The glyph is then ordered within the column chronologically from the bottom to the top.

Each glyph itself is drawn in a manner to reflect the underlying email. A glyph is composed of six lines, each one representing an attribute of the email – subject, sender, recipient, attachment, replied to status, and junk status. Some of these attributes, such as attachment, replied to status, and junk status are boolean values.

For these lines, they are either drawn or not. For example, if the email has an attachment there will be a line, otherwise there will not be. Subject, sender, and recipient are a little more complicated. Each of these attributes is a string, or in the case of recipient, a list of strings. These string values must be mapped to integer values which in turn are used to determine the length of the lines.. To convert a string to an integer, we devised an equation inspired by positional numeral systems [6]. Each character in the string is first converted to its ASCII numeral value. This value is then multiplied by a power of ten corresponding to its position. For example, the integer value of the word "cat" would be (99 * 100) + (97 * 10) + (116 * 1), or 12669. Although characters that occur towards the beginning of the string tend to dominate the integer value of the word, this system prevents alternative sequences of the same characters, such as cat and tac, from being mapped to the exact same value. We believe that ensuring that most strings map to an unique number is more important than the dominance problem, as alternatives would confuse users into thinking messages have the same string value for an attribute when in actuality they do not. Once each message has had its string value mapped to an integer, the maximum integer value is found. The maximum value is mapped to the length of fifty. Every other integer value is mapped to a length between one and fifty in a manner that maintains the relative ratio. An example glyph can be seen in figure 4.1.

The rendering of the message window is relatively simple. The frontend connects to the server to obtain the body of the email and then just renders the text. Overall, there is nothing complicated in this portion of the system, nor are there any details worth mentioning here.

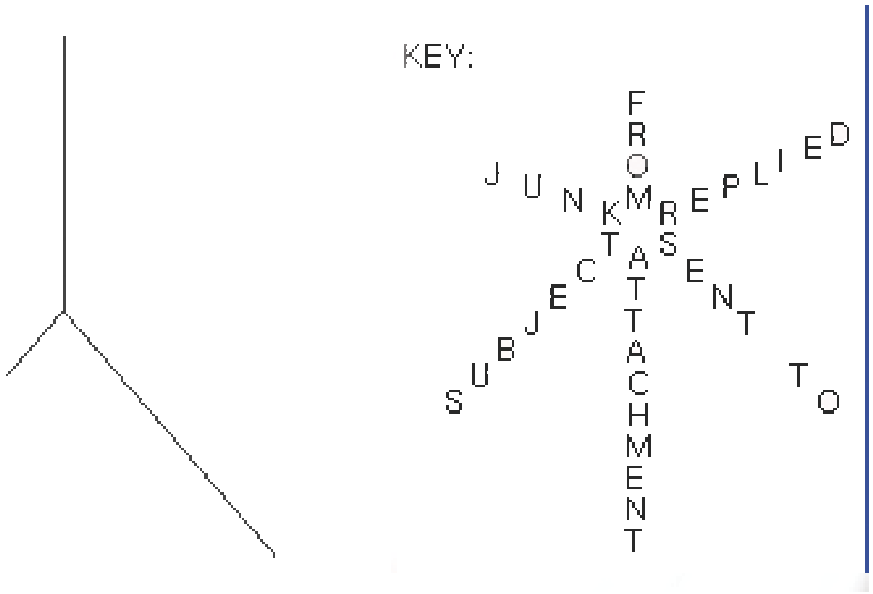


Figure 4.1: A visual explanation of a glyph

4.1.3 General Implementation

The source code for MailScape was written in C++. Implementing the source code in C++ enabled the use of pointers. This ensured that a single copy of each message was maintained. Although several lists of messages were mentioned previously, the majority of these lists simply stored references to the necessary messages. From a memory standpoint, this decreases the overall footprint of the system as the reference to a message requires less physical space than a copy of a message. From the programming perspective, this implementation required an increase in the number of tests in order to verify that no memory leaks existed in the application.

Graphically, MailScape utilizes the OpenGL libraries. OpenGL is portable to multiple operating systems, and although the current version of MailScape is not capable of running on multiple platforms, the design decision to use OpenGL allows future versions to be portable. In addition to using OpenGL, MailScape uses the GLUI toolkit to implement some of the user interface, such as the radio buttons and

text boxes in the History Flow view.

In order to obtain our email dataset, we communicate with an email server via Simple Mail Transfer Protocol (SMTP). This allows us to retrieve the current status of the user's inbox each time our application starts up. SMTP allows us ask the server for each message's id. Once we have the id, we can receive from the server the envelope information, email size, and the flags associated with the message. The envelope information must be parsed into the subject, sender, recipient, and sent date. Once this information is parsed, we place the values in our message object. Separate from the envelope information is the flag field. In the SMTP protocol, the flag information stores information such as whether or not the email has been read, replied to, or marked as junk. All of the attributes that are valid for a single email are contained in a single response from the server and must be parsed out of the response.

4.2 Resulting Application

The implemented version of MailScape opens in the default view of the History Flow visualization with the email grouped by the subject. The controls for filtering and sorting the messages run along the right hand side of the window. A screenshot of the initial view of MailScape for a sample dataset is shown in Figure 4.2.

The user is allowed to modify this view with any of the aforementioned interactions. Changing the grouping, the sort-by value, and the date range all alter the appearance of the visualization. The same dataset shown in Figure 4.2 is also shown in Figure 4.3, however in the second figure the messages have been grouped by the sender instead of the subject.

A user can also filter the dataset by a particular value. As previously described,

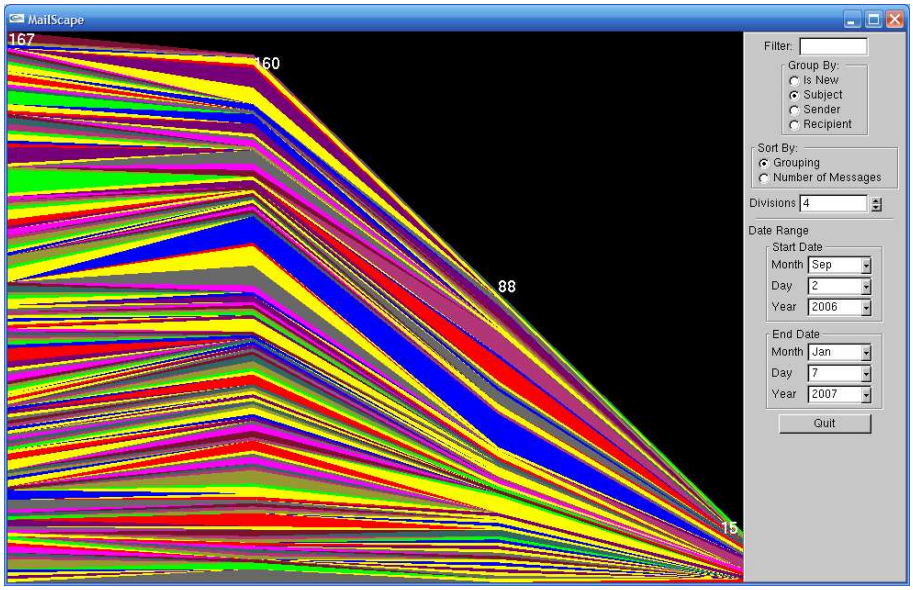


Figure 4.2: The initial view of MailScape

this action limits the messages that contribute to the visualization to those that match the filter criteria. The dataset shown in Figure 4.2 is again shown in Figure 4.4, however in the second visualization the email have been filtered to show only the messages where the subject contains "soccer."

In addition to filtering email and altering how they are grouped, a user can also hover over a thread in order to reveal the value for that thread. An example of the ability to hover is displayed in Figure 4.5.

To enter the glyph view, a user selects a given thread. Figure B is the glyph view for the thread "Soccer Today" when the messages from the same dataset are grouped by subject. These glyphs form two columns due to their dates. All of the glyphs on the left hand side were sent on October 8th, 2006, while all of the glyphs on the right were sent on November 8th, 2006. Figure 4.7 also shows an example of the glyph view. This screenshot shows the glyphs for all messages sent by the address goog@cs.wpi.edu. The figure allows one to see the color coding of glyphs.

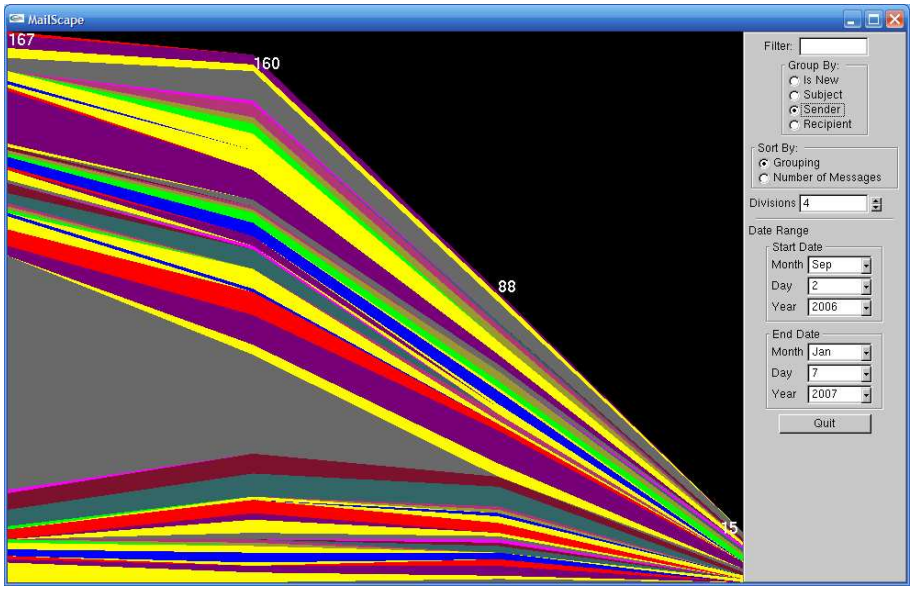


Figure 4.3: MailScape with messages grouped by sender

The red glyphs represent unread messages while the white ones show previously read email.

Finally, figure 4.8 shows MailScape's message view. Although not elaborate, this view provides the user with all of the necessary information from the email header and body. It is also from this view that the user replies to an email and, if desired, marks it for deletion. Each of these interactions are based upon key presses with reply being 'r' and delete being 'd'.

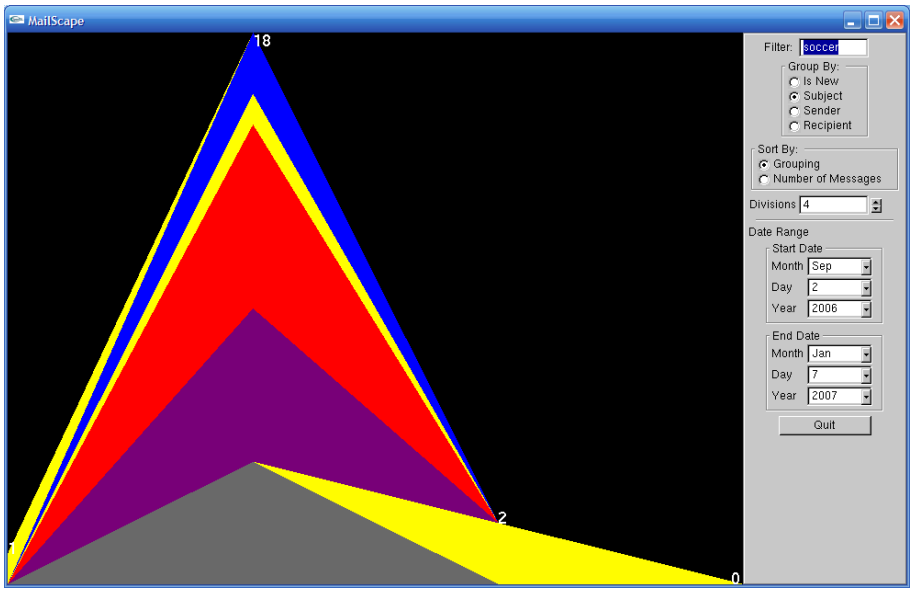


Figure 4.4: MailScape showing messages filtered by subjects about soccer

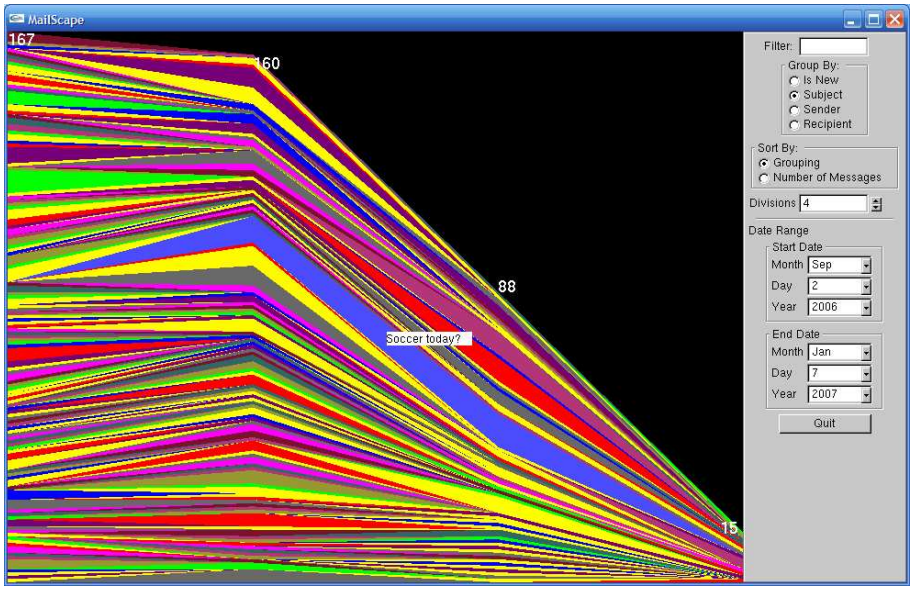


Figure 4.5: The hover ability of the History Flow view

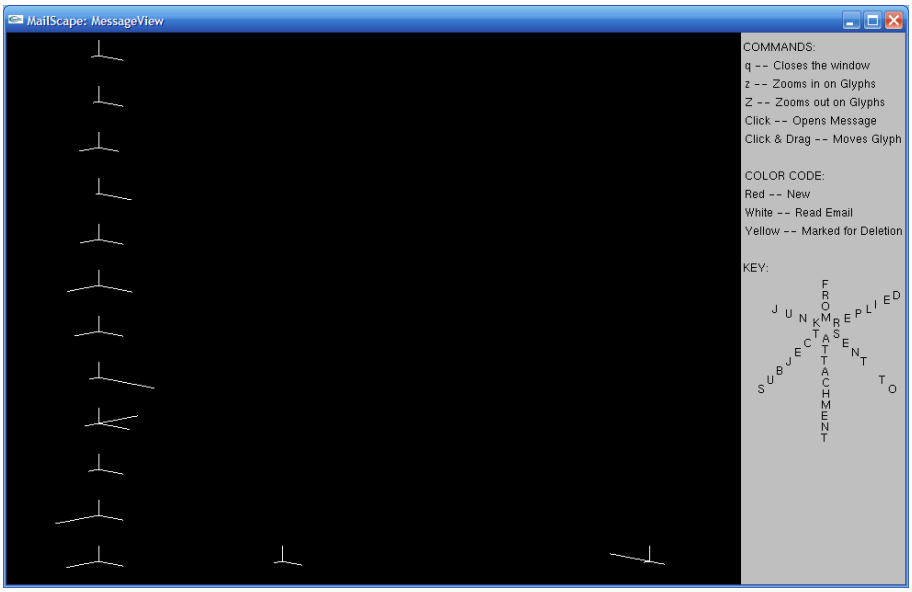


Figure 4.6: An example version of the glyph view

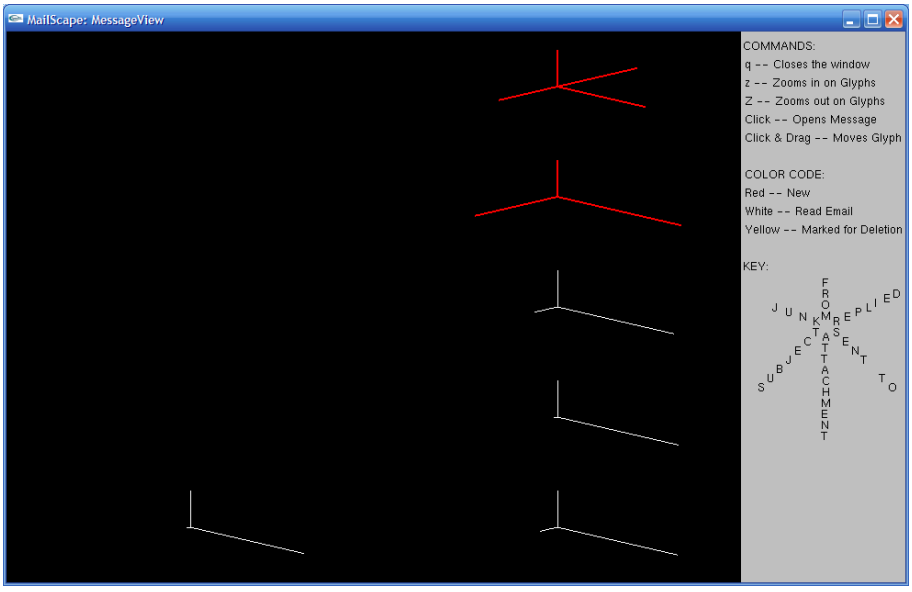


Figure 4.7: A glyph view showing both new and old email

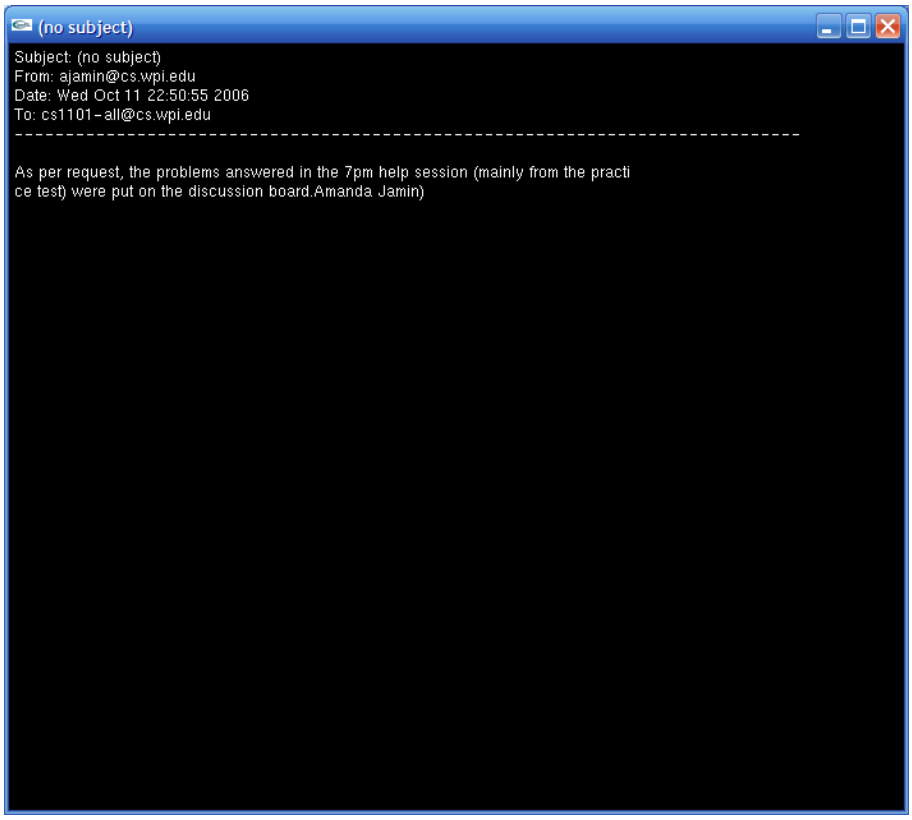


Figure 4.8: MailScape's message view

Chapter 5

User Study

In order to validate our results, we designed and conducted a user study. It was our belief that users would be able to easily understand the design of our email system. Additionally, we felt that the typical user would be able to navigate the system in a manner that allowed him or her to efficiently find a specific email.

5.1 Design of the User Study

The first portion of the user study was an informational PowerPoint presentation. The motivation behind the PowerPoint presentation was to provide each participant with the identical set of instructions. The instructions provided not only demonstrated how to use the interface, but visually showed how each of the provided interactions altered the displayed data. The dataset for this portion of the user study was a relatively small dataset, containing approximately fifty email messages.

After completing the instructional portion of the user study, each participant was given a list of 4 tasks. These tasks were:

1. Check the new email for any junk messages and mark these messages for

deletion

2. Find an email from hammelg@cs.wpi.edu regarding TA/SA assignment

3. Find the latest email regarding soccer

4. Find an email sent to CsSystems@cs.wpi.edu and GraduateStudents@cs.wpi.edu about parking and reply to it with the message "Thanks"

For each of these tasks, the user provided information about the email – such as the subject, date, or sender. This information was designed to allow one to determine if the expected email was found. Additionally, each user ranked the difficulty of the task on a scale of one (very difficult) to five (very easy). We monitored each user during the completion of the tasks, allowing us to take note of common actions and mistakes as well as how long each person took to complete a given task. The dataset for this segment of the user study was a relatively large dataset, containing 472 email messages.

After completing the previous two stages of the user study, the user was asked a set of questions regarding his or her experience. In this set of questions, the user was asked to rank the level of difficulty in both understanding and using the system. Additionally the user was asked if he or she personally would use this type of system regularly to check email. Finally all users were asked what features our client had that their typical client lacked and vice versa.

For each user we collected demographic information. This information included personal information, such as the individual's gender, age, area of study, and whether or not he or she was color deficient. The demographics also contained questions regarding each participant's typical email practices. These questions included the typical program the person used to check his or her email, the amount of email he

or she received in a day, and the amount of email typically saved. A final version of the user study questions can be found in Appendix A.

5.2 Demographics

Over the course of a week, twenty-four people participated in the user study. Ten of these participants were male with the remaining fourteen being female. The ages of the participants ranged from nineteen to twenty-six with the average age being twenty years and eight months, the median age being twenty years and six months, and the standard deviation of the ages being one year and six months. The users came from a diverse set of fields, with participants coming from thirteen different fields of study. The highest number of people, five (21 percent), identified themselves as civil engineers, with the next largest percentage (17 percent) coming from the mechanical engineering field.

Participants identified seven different mail clients – AOL, AppleMail, Evolution, Gmail, Microsoft Exchange, Microsoft Outlook, and SquirelMail (a web based client used on WPI's campus). The most popular mail application among participants was Gmail, with ten of the respondents using the program. Outlook and Exchange were next with six and four people respectively. SquirelMail had three users, and each of the other mail clients had one participant report using them each.

Those participating in the user study estimated that they received between one and seventy-five new email messages daily. The average number of new email messages was 22.91, with a median of 20.0 and a standard deviation of 17.62. Users retained between zero - four-thousand of these messages in various folders. 947.39 was the average number of saved messages, with a median of 600.0 and a standard deviation of 1,069.16.

5.3 User Study Task Results

Overall, users were able to successfully complete the tasks given during the study. Observations, as well as the data itself, indicated that there was a noticeable learning curve associated with using the application. Observing the users, we noticed that during the first task users tended to be slightly confused, either by how to proceed or how the application itself worked. Some knew what they wanted to do, however, could not remember from the instructional portion of the study exactly how to do it. This confusion tended to be alleviated by the second task. These observations are supported by the data itself. As shown in table 5.3, the time necessary to complete each task decreased significantly from the first task to the second task.

Task	Percent Right	Percent Wrong	Average Time	Median Time	Standard Deviation
1	65	35	03:02	03:10	00:55
2	78	22	02:06	01:47	00:58
3	74	26	01:22	01:06	00:50
4*	86	14	01:26	01:26	00:38

Table 5.1: User Study Task Results (Time measured in minutes)

*Values for adjusted version of Task 4

Looking at the answers recorded for the questions associated with each of the tasks, it can be concluded that the majority of users were able to successfully complete each of the tasks. Task One had the lowest success rate, with only sixty-five percent of users answering correctly. This low percentage can be partially attributed to users who, not yet familiar with the system, gave up on Task One due to frustration. These users, excluding one, proceeded with the rest of the tasks. Time performance on Task One was also worse than the other tasks. On average, Task One took 03:02 minutes, with a maximum time of 04:04 minutes and a minimum time of 01:07

minutes. Task One had a median time of 03:10 minutes with a standard deviation of 00:55 minutes.

Task Two and Task Three both had similar success rates with, respectively, seventy-eight and seventy-four percent of participants correctly responding. Task Three took a little less time to complete than Task Two, both according to the average and the median. However, it was difficult to determine if this was due to an increased familiarity with the system or due to the nature of the tasks themselves.

Task Four, at first glance, seemed to be an anomaly of the study. Although being the final task, it had a low percentage of users who got it correct, fifty-seven percent. However, notes from the observations of the users indicated that some users were confused by the question itself. The task instructed users to find a single email sent to both system@cs.wpi.edu and grads@cs.wpi.edu. However, as users noted, the task did not indicate that the email had to be sent solely to these two addresses. If email addressed to these two recipients and additional recipients that were about parking were also considered correct responses, the percentage of users completing Task Four correctly increased dramatically to eighty-six percent. Additionally, including these answers improved the time performance of the task. Without these users' answers being marked correct, Task Four had an average completion time of 01:46:00 and a median time of 01:41:00. The standard deviation was 00:31:00. When these users' times were taken into account, the average time is adjusted to 01:25 and the median to 01:26 with a standard deviation of 00:38:00.

In addition to the success rate and the time necessary to complete each task, each user assessed the difficulty of the task. The difficulty level of a given task was based upon a scale from one to five, where one was considered very difficult and five very easy. A summary of these results can be found in table 5.3. Task One was perceived as the most difficult task with an average difficulty ranking of 2.55. Task

One, in addition to having the lowest average, was the only task that did not have a single user rank it as having a difficulty of five (very easy). The difficulty rankings for Task One were also more spread out across the range than the other tasks. This is indicated by the high standard deviation for Task One in comparison to the other tasks.

Task	Min Ranking	Max Ranking	Ave Ranking	Median Ranking	Standard Deviation
1	1	4	2.545	2.0	1.157
2	1	5	3.348	4.0	0.914
3	1	5	3.348	3.0	0.865
4	1	5	3.350	3.0	0.853

Table 5.2: Difficulty Rankings of Tasks

Task One's difficulty ranking was affected by users being unfamiliar with the system. Confusion and uncertainty on how to use the software was believed to have contributed to low rankings for this task. A second user study, using the same questions in a different order, could confirm this belief. The three later tasks received approximately the same difficulty ranking as each other, with both Task Two and Three having an average of 3.35 and Task Four having an average of 3.35. The standard deviation of these three tasks' rankings reveal that the rankings were clustered together more as the users progressed though the tasks, although the difference between Task Three's and Four's standard deviation is minute.

In addition to ranking the difficulty level of each of the tasks, the users ranked the overall difficulty in understanding the system and the difficulty in using the program. The average ranking of understanding the system was 3.0. Given the fact that the entire concept behind the system was unfamiliar to all of the users, this was a good sign that this type of program could be used by the typical email user. The

interface of MailScape received a worse difficulty ranking than the concept of the program with an average ranking of 2.75. User comments indicate that this was due to the lack of a button based interface. Many of the interactions, such as composing and replying to an email, were based upon key presses. Users tended to forget these keys, and desired graphical buttons. The difficulty ranking of the interface suffered for this.

Overall, the tasks were not considered more difficult to complete than the average software program. For the majority of the tasks, the difficulty as ranked by the users was on the easy half of the scale. Overall, these rankings indicate that the system was relatively simple to learn. Furthermore, the fact that users felt the method of completing the tasks was easy indicates that they were not frustrated with the use of the program.

5.4 User Comments

Participants were also asked for their general opinions about the system. Specifically, they were asked to indicate if they personally would use this type of email system for daily use, to provide a list of features that this system had that their current client was lacking, and to provide a list of features this system was lacking. Nine users said that they would use this system over their current client, ten said they would not, four were undecided, and one did not respond. Many of those that stated that they preferred this system, cited the ease of filtering and finding email as their primary reason. Even users who preferred their current client, stated they liked the efficiency of searching with MailScape.

The primary reason given by users who did not want to switch systems was that they were content with their current client. A few of these participants were simply

just unwilling to take the time to setup and become familiar with a new program. Additionally, some people missed features that their clients had that MailScape did not implement. An integrated calendar, the ability to attach files, and the ability to render email containing html were the features most often missed.

The few undecided respondents each gave a detailed explanation of why they were unsure of which interface type they preferred. This was not by the request of the facilitator, but a coincidence. Generally, the users cited that they really liked features of both of the systems and desired to be able to manage the same email account with both clients. It was suggested by one of these participants that the tool be integrated as a plugin to an existing system. By integrating the two email client styles, users would be able to take advantage of the benefits of both systems. Users would have the familiarity of traditional email interfaces, but the ability to search and sort their email using our new design. Additionally, this would allow users to utilize many of the existing email client features, such as an integrated calendar, without having to reinvent the wheel.

5.5 Summary

The majority of users seem satisfied with the system. Within a few minutes of using the system themselves, all but one user was able to use the full features of the system. The one user who could not admitted to not reading the instructional slides. In addition to being able to use the system, both user comments and observations revealed that users enjoyed using the system. Some users even requested to continue to "play around" with the system after completing the task. Overall, our newly designed interface is a successful method of managing an email account. Although improvements are necessary, there is definite potential for this manner of email client

in the future.

Chapter 6

Future Work

MailScape is a prototype, designed to be a proof of concept system. As is the nature of such systems, many features, believed to be non-vital to the performance of the program, were left unimplemented in version 1.0. Due to the already proven success, and remaining raw potential, we believe implementing these remaining features would be beneficial to the growth of the system. In the following sections, we have outlined these improvements.

6.1 Improvements to MailScape

MailScape currently allows for many filtering and grouping techniques at the History Flow level. There are additional interactions, such as the ability to join threads together, that can be implemented for this portion of the system. Additionally, the current glyph view implementation has very few interactions. Interactions, such as filtering, can be implemented at this level to aid users in finding and reading specific email.

6.1.1 History Flow View Changes

One of the primary improvements planned for MailScape is the ability to join threads together from the History flow perspective. This feature will allow the user to inform the system that two threads, initially believed to be separate threads, are really both part of the same thread. In the current implementation only when two email messages have an exact match in the selected attribute are they considered to part of the same thread. When email are grouped by sender, this causes two email messages sent by the same person from different email addresses to form separate threads. The ability to join threads together will allow a single thread to be formed from all of the email addresses of a person.

In addition to being able to join threads together, it has been proposed that a user should be able to mark an entire thread for deletion. Deletion by thread will create a more efficient manner of cleaning up the contents of a person's mailbox. It will allow a large amount of email, such as a group of email regarding an event that has passed, to be deleted with a single action. We believe this feature will decrease the amount of email people retain in their inbox.

Finally, we believe future versions of MailScape should allow for more methods of grouping email together. The History Flow visualization currently allows users to create threads based on their subject, sender, recipients, and new status. To help users in finding email by other attributes, additional groupings should be added. Currently, grouping by whether the email has an attachment or not, the junk status of the email, whether the email is flagged or not, and if the email has been replied to are proposed. These additions would allow users to quickly divide their email. Additionally, once delete by thread is implemented, these new features would allow one to do such things as deleting all of the junk email in the inbox at once.

The final proposed improvement involves altering the underlying dataset visual-

ized by MailScape. Currently MailScape visualizes the entire contents of a person's inbox. As many users already have their email organized into folders, it would be beneficial to allow one to limit this visualization to a specific folder. Implementing this feature will allow one to maintain the existing organization of his or her mailbox. With the implementation of this feature, the ability to create and organize email into folders based upon the visualization should be allowed. This will enable users to continue to maintain their folder hierarchy.

6.1.2 Glyph View Changes

In addition to making changes to the History Flow view, we have proposed improvements to the glyph view. The first improvement is based solely upon user comments during our user study. Future versions of MailScape should allow the user to filter messages from the glyph view. This filter would be similar to the existing filter in the message view, allowing a user to filter by subject, sender, recipient, as well as other attributes. Implementing this feature would allow users to filter their dataset twice, once prior to selecting their thread and once after. For large threads, this will enable a user to find a specific email within a thread quickly. Currently, the only alternative is to hover over, or analyze, each glyph in order to determine which one represents the desired message.

A second improvement to the glyph view would be the implementation of a date filter. Currently, a user can only change the range of dates viewed on the History Flow view. If users realize on the glyph view that they would only like to see a select range, they would have to return to the History Flow view, alter the dates and start their search over. It would be an improvement if the dates could additionally be modified from the glyph view.

Finally, a user should be able to modify the color code of the glyph view. Cur-

rently, a color is defined for new mail, old mail, and mail that is marked for deletion. A user should be able to alter which color is mapped to each of these attributes. Additionally, we would allow users to choose additional attributes that email messages could be color coordinated by. This would provide users with flexibility, allowing them to control the visualization and the information it reveals.

6.2 Integrating Email Tasks

As an email client, MailScape is incomplete. Currently a user can only compose, delete, and reply to a message. It is vital to the success of MailScape that it either becomes a complete email client itself or be implemented as a plugin to an existing client. Allowing a user to attach files, save and read attachments that were sent to them, forward a message, cc and bcc additional recipients to the message and other such features that have become commonplace in today's email clients is essential. Additionally, the current implementation of MailScape does not update the user's inbox status except upon startup. Without implementing the aforementioned features, users will not use the system for daily use.

There are two approaches to implementing these features, implement each of these features into the system or integrate MailScape into a complete email client. There are benefits, and disadvantages, to each of these plans of attack. Implementing each of the features in our own stand-alone application would be time consuming. Improvements to the visualization portion of MailScape would take a back seat to MailScape's evolution into a complete mail client. However, MailScape itself would remain a complete application on its own. MailScape would not have to be dependent on a second development team and schedule, nor would its ability to work be affected by changes made in a separate code base.

Integrating MailScape as a plugin to an existing client would make MailScape a complete email client. All of the email tasks that are currently not implemented would be available. The benefits of a plugin architecture, which were outlined previously during our discussion of the user study, make it an appealing solution. However, many of the email clients today do not support a plugin architecture. We would either be limited in our choice of clients to integrate with or we would have to actively collaborate with a client's development team to integrate it in another manner. All of the aforementioned benefits of MailScape remaining an independent application would be negative side effects of MailScape's dependency on a second program.

The two above approaches must both be further researched. The benefits, as well as the disadvantages, of each must be weighed. Once analyzed, a decision will be made regarding which method is best for the purposes of MailScape allowing the missing features to be implemented.

Chapter 7

Conclusions

MailScape is an application that provides a data visualization approach to email management. It allows users to interact graphically with their email. Additionally, it enables users to manage their inboxes not email by email, but multiple email messages at a time. This speeds up the process of searching through the inbox and also enables users to organize their email quickly. With the improvements to the system that were previously outlined, MailScape has the potential to handle the email overload that currently exists.

The History Flow perspective of MailScape enables the user to analyze trends within their inbox. Additionally it provides a method of interacting with groups of similar email messages rather than looking at each one individually. It is also the medium through which a user can filter and sort their messages.

MailScape's glyph view allows users to manage their inbox from an individual message perspective. Messages can be compared and contrasted in a graphical manner. Additionally, this view allows one to open an individual email providing a method of reading, deleting and replying to a message.

Overall, our hypothesis regarding a graphically email client was proven true.

Our user study revealed the MailScape system's design to be intuitive and users believed it be easy to learn. Additionally, users seemed to enjoy using the email client. Future development work on MailScape, and applications with a similar design, should yield systems that are equally usable.

Bibliography

[1] ClearContext. Clearcontext survey on the use of email in 2005, 2005. Online; accessed 6-December-2006 from http://www.clearcontext.com/survey/2005/raw_survey_summary.html.

[2] ClearContext. Clearcontext survey on the use of email in 2006, 2006. Online; accessed 6-December-2006 from http://www.clearcontext.com/survey/2006/survey_results.html.

[3] Microsoft Corporation. Intellectual property licensing data visualization tools. Online; accessed 3-January-2007 from http://www.microsoft.com/about/legal/intellectualproperty/search/details.mspx?ip_id=IDAE Development&ipCat=Any&feeStructure=Any&keywords=&ipVenture=false.

[4] Microsoft Corporation. Microsoft research data visualization components. Online; accessed 3-January-2007 from http://research.microsoft.com/community/dataviz/.

[5] Nicolas Ducheneaut and Victoria Bellotti. E-mail as habitat: an exploration of embedded personal information management. *Interactions*, 8(5):30–38, 2001.

[6] Chaya Gurwitz. Conversion between different number systems. Online; accessed 2-January-2007 from http://www.sci.brooklyn.cuny.edu/ğurwitz/core5/convtool/convexp.html.

[7] Susan Havre, Beth Hetzler, and Lucy Nowell. Themeriver: Visualizing theme changes over time. In *Information Visualization '00: Proceedings of the IEEE Symposium on Information Vizualization 2000*, pages 115–124, Washington, DC, USA, 2000. IEEE Computer Society.

[8] Brian Johnson and Ben Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *VIS '91: Proceedings of the 2nd conference on Visualization '91*, pages 284–291, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[9] Bernard Kerr. Thread arcs: An email thread visualization. volume 00, pages 27–38, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[10] Bernard Kerr and Eric Wilcox. Designing remail: reinventing the email client through innovation and integration. In *CHI '04: Extended Abstracts on Human Factors in Computing Systems*, pages 837–852, New York, NY, USA, 2004. ACM Press.

[11] University of Maryland Department of Computer Science. A brief history of email, 2005. Online; accessed 7-December-2006 from http://www.cs.umd.edu/class/spring2002/cmsc434-0101/MUIseum/.

[12] C. Plaisant, G. Chintalapani, C. Lukehart, D. Schiro, and J. Ryan. Using visualization tools to gain insight into your data. In *Proceedings of Society of Petroleum Engineering Annual Technical Conference, 2003*, 2003.

[13] IBM Research. Reinventing email collaborate user experience group, 2003. Online; accessed 14-December-2006 from http://www.research.ibm.com/remail/index.html.

[14] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[15] Ben Shneiderman. Information visualization: Dynamic queries, starfield displays, and lifelines, 1996. Online; accessed 2-January-2007 from http://www.cs.umd.edu/hcil/members/bshneiderman/ivwp.html.

[16] Ben Shneiderman. Treemaps for space-constrained visualization of hierarchies, 2006. Online; accessed 2-January-2007 from http://www.cs.umd.edu/hcil/treemap-history/.

[17] Fernanda B. Viegas, Scott Golder, and Judith Donath. Visualizing email content: portraying relationships from conversational histories. In *CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 979–988, New York, NY, USA, 2006. ACM Press.

[18] Fernanda B. Viegas, Martin Wattenberg, and Dave Kushal. Studying cooperation and conflict between authors with history flow visualizations. In *CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 575–582, New York, NY, USA, 2004. ACM Press.

[19] Martin Wattenberg and Jesse Kriss. Designing for social data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):549–557, 2006.

[20] Steve Whittaker and Candace Sidner. Email overload: exploring personal information management of email. In *CHI '96: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 276–283, New York, NY, USA, 1996. ACM Press.

Appendix A

User Study Questions

Part I Demographics:

Age Field

Gender Normal Email Client

Are you color blind or color deficient? If so, how.

Rate your experience with Data Visualization. Data Visualization is the graphical presentation of information, with the goal of providing the viewer with a qualitative understanding of the information contents.

No Experience Beginner Intermediate Expert

Estimate how much email (read and unread) do you typically keep in your inbox.

Estimate how much new email you receive on a daily basis.

Part II Tasks: The following questions are based on a scale from 1 to 5 where the scale is defined as follows:
1 - very difficult
2 - difficult
3 - average
4 - easy
5 - very easy

1. Check your new email for any junk email. Mark these email messages for deletion and list their subjects below.

1b. Rate the difficulty of this task.

2. Look for an email from ghamel@cs.wpi.edu. The particular email you are looking for is regarding TA/SA assignment. Write down who this email was sent to below.

2b. Rate the difficulty of this task.

3. Find the most recent email about soccer. Write down the subject of the email and the date at which it was sent below.

3b. Rate the difficulty of this task.

4. There is an email sent to system@cs.wpi.edu and grads@cs.wpi.edu. The email is about parking. Write the sender of this email below, and reply to this email with the message "Thanks".

4b. Rate the difficulty of this task.

Part III Post Study Questions:

The following questions are based on a scale from 1 to 5 where the scale is defined as follows:
1 - very difficult
2 - difficult
3 - average
4 - easy
5 - very easy

Rate the difficulty level of using this software program.

Rate the difficulty level of understanding this software program.

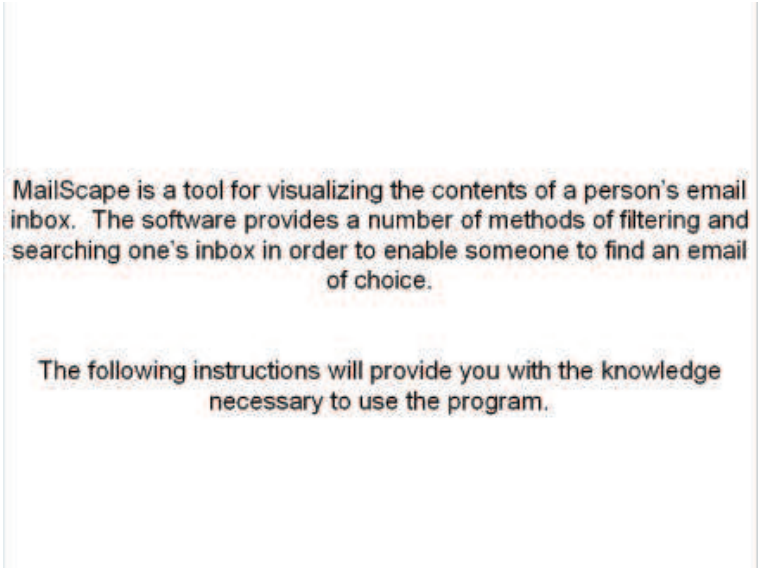
Would you consider using a program designed like this one for your daily email needs? If so, why do you prefer this program? If not, what do you dislike?

What abilities does this program have, if any, that your typical email client does not support?

What abilities does your typical email client have, if any, that this program does not support?

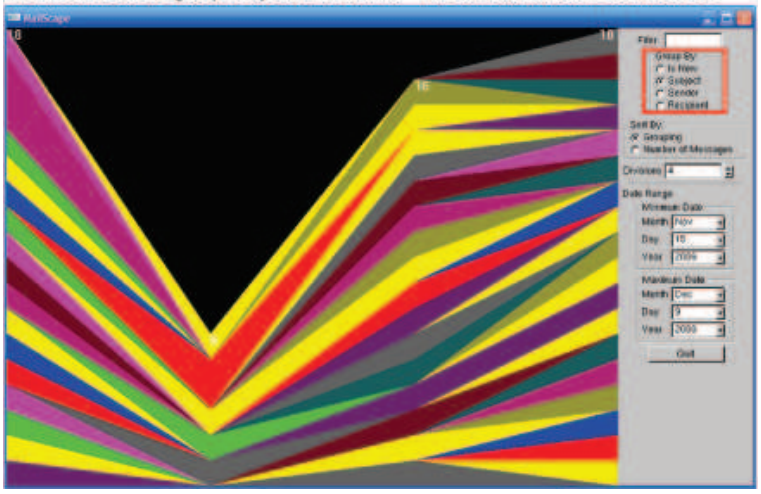
Appendix B

User Study Instructions

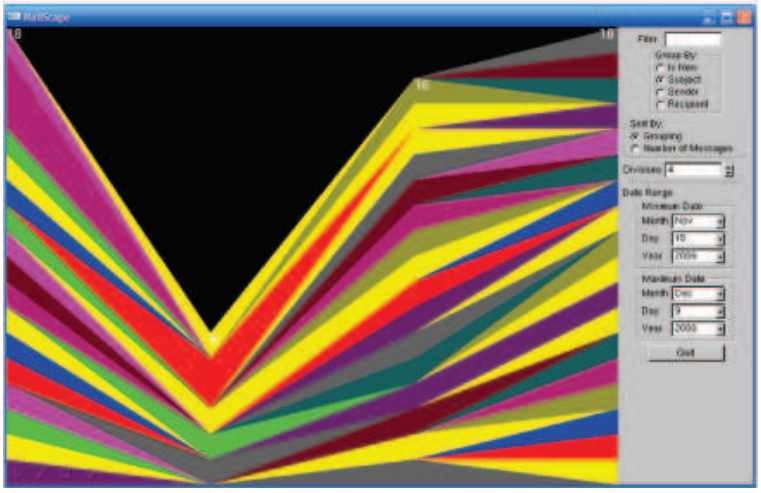


Bands are created by grouping like emails together. Currently emails are grouped together if they have the same Subject.

Emails can also be grouped by other attributes. The current attribute is defined here.

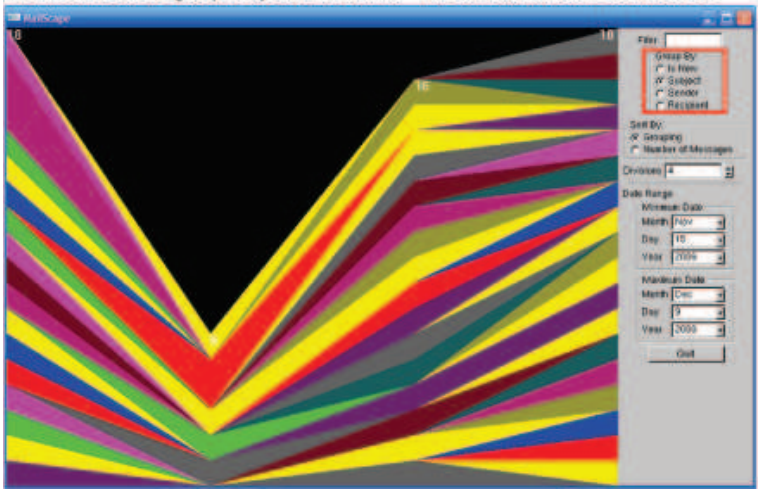


Bands are created by grouping like emails together. Currently emails are grouped together if they have the same Subject.



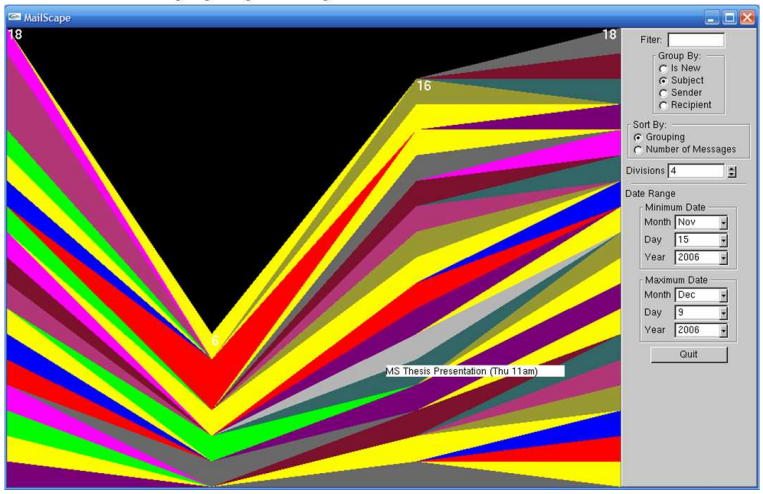
Bands are created by grouping like emails together. Currently emails are grouped together if they have the same Subject.

Emails can also be grouped by other attributes. The current attribute is defined here.

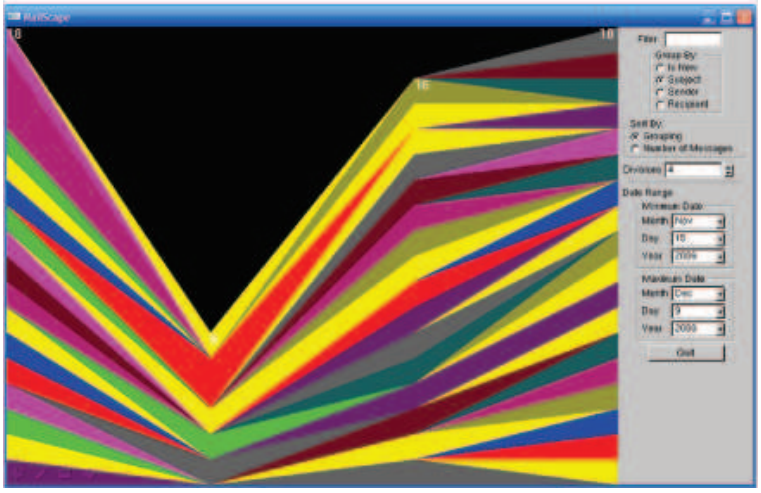


Hovering over a band causes the band to be highlighted. The value of the selected attribute is displayed for that band.

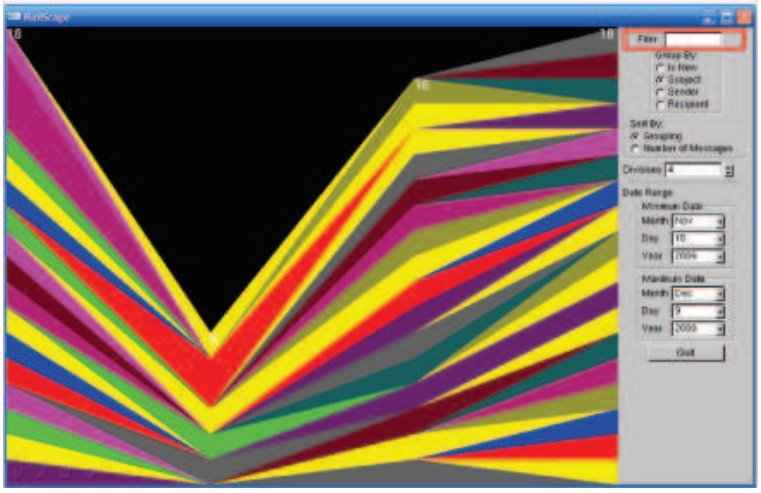
Below shows the highlighting of a single thread.



The width of the band at any point represents the number of emails received at that time.

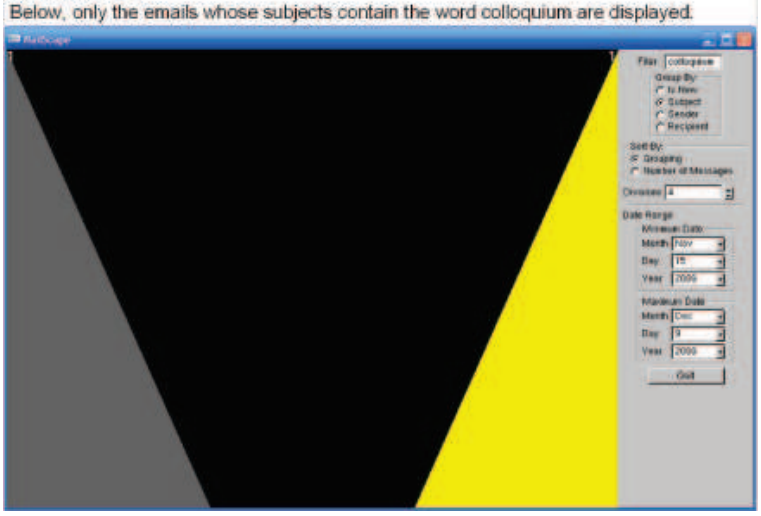


Emails can be filtered by the value of the current attribute by typing in the filter box and pressing enter.

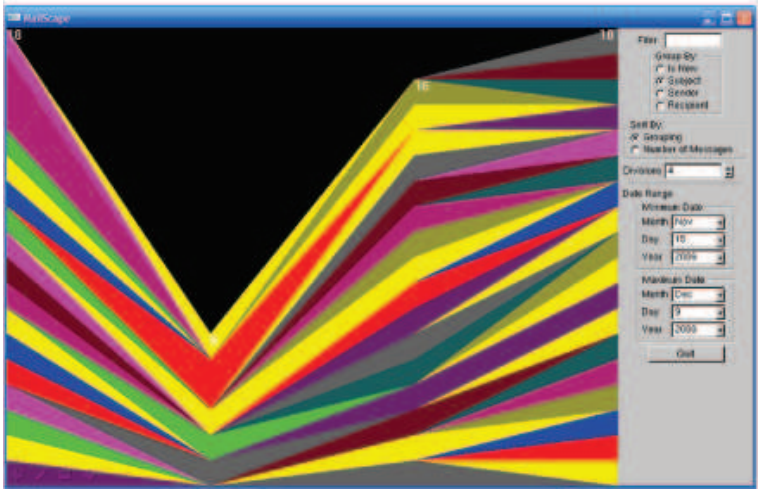


Emails can be filtered by the value of the current attribute by typing in the filter box and pressing enter.

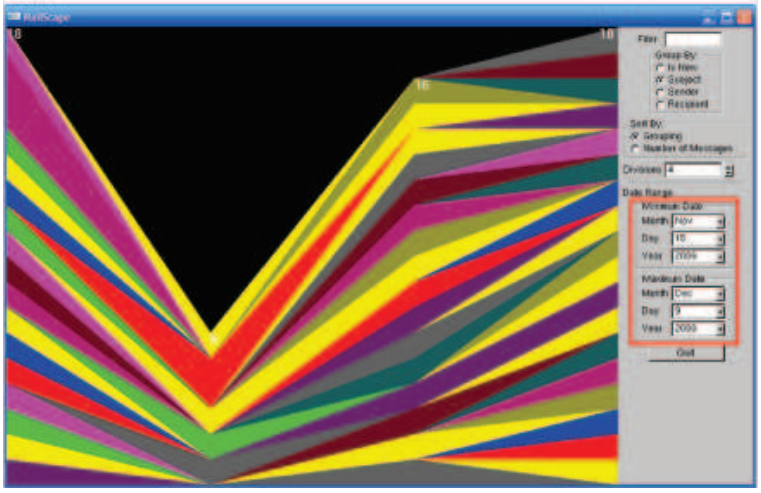
Below, only the emails whose subjects contain the word colloquium are displayed.



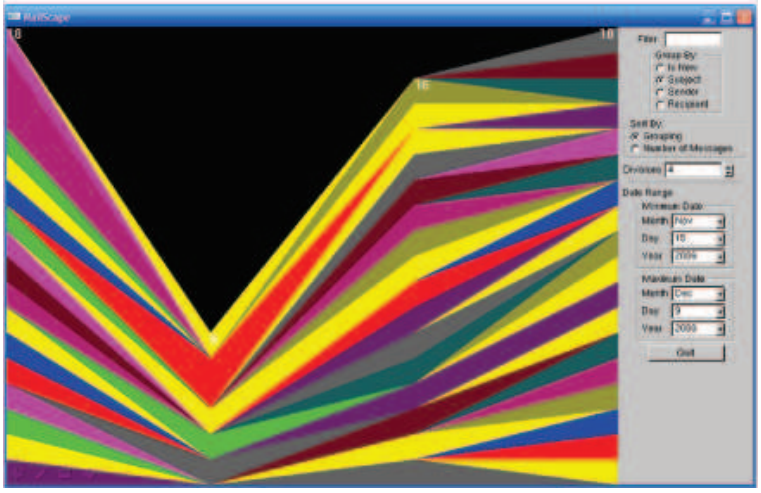
By default, the visualization shows all emails from the oldest email to the newest email.



By default, the visualization shows all emails from the oldest email to the newest email. These values can be changed by adjusting the minimum and maximum values located here.

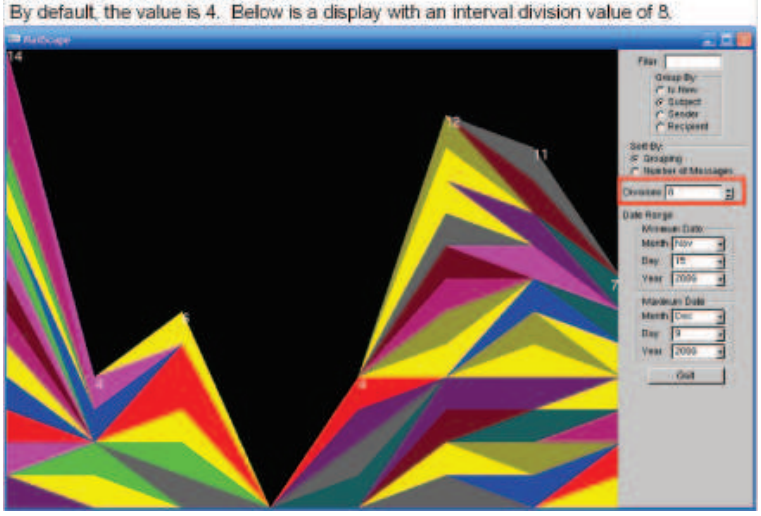


In addition to changing the maximum and minimum dates, the user can change the number of divisions within the date range.



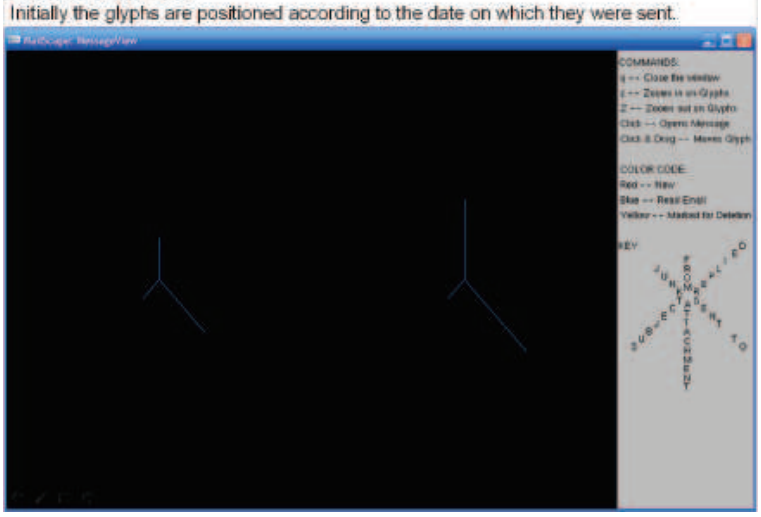
In addition to changing the maximum and minimum dates, the user can change the number of divisions within the date range.

By default, the value is 4. Below is a display with an interval division value of 8.

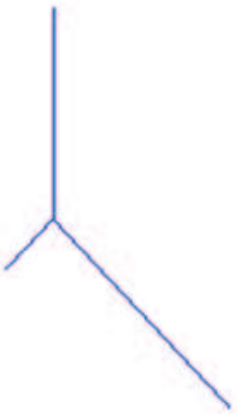


Clicking on an band creates a glyph view of the thread. For each message in the band, there is a single glyph.

Initially the glyphs are positioned according to the date on which they were sent.

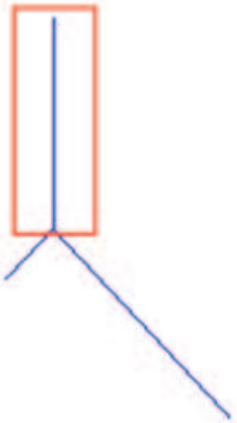


Each glyph is drawn according to the key shown on the right.



Each glyph is drawn according to the key shown on the right.

This line represents the From value of the email.

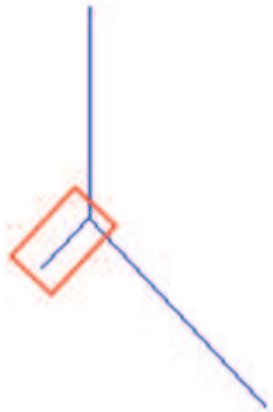


Each glyph is drawn according to the key shown on the right.

If the Email is marked as junk there will be a line like the red one below. If no line is present it is not marked as junk.

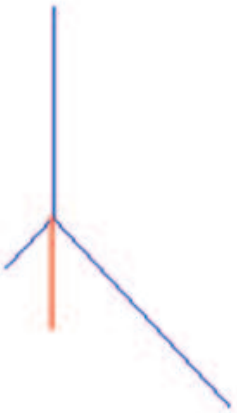
Each glyph is drawn according to the key shown on the right.

The subject of the email is mapped to the line highlighted below.



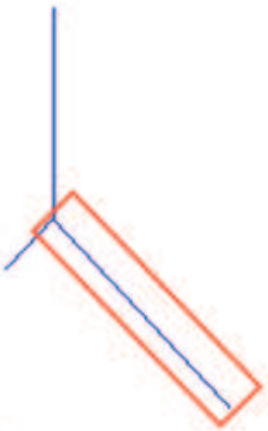
Each glyph is drawn according to the key shown on the right.

If the email has an attachment, there will be a line as shown below in red.



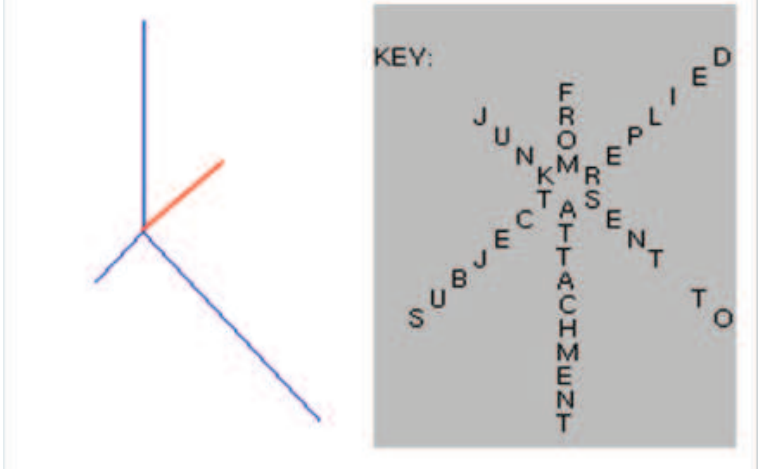
Each glyph is drawn according to the key shown on the right.

Who the email is sent to is mapped to the highlighted line below.

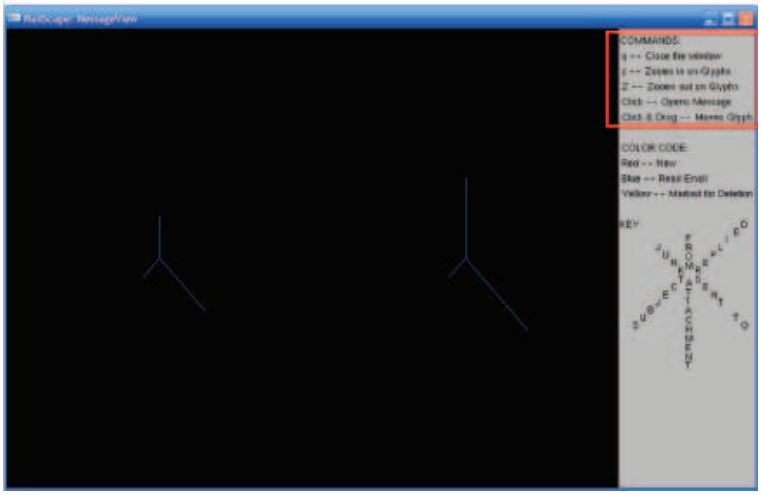


Each glyph is drawn according to the key shown on the right.

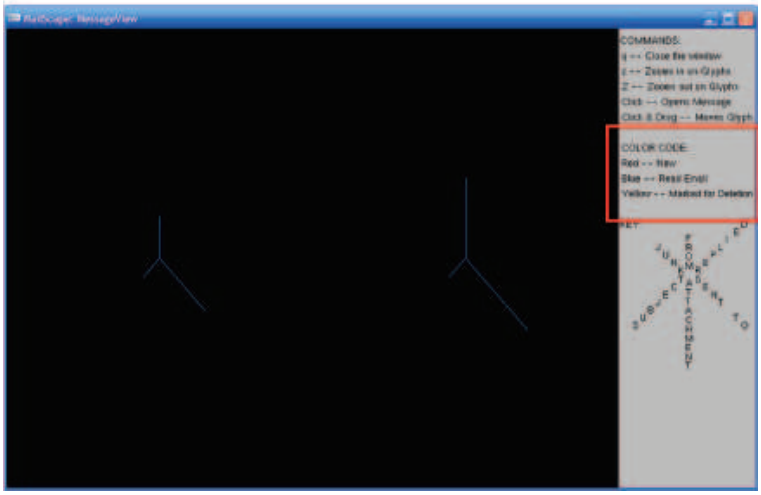
If you have replied to the email, there will be a line where the red one is below.



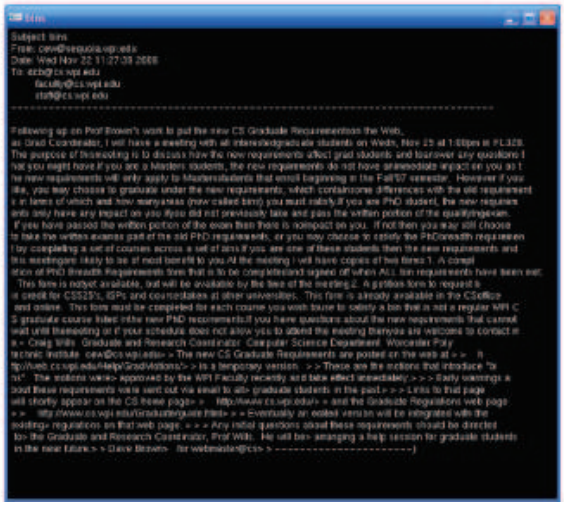
There are interactions available on the glyph view. They are listed on the left hand side of the window.



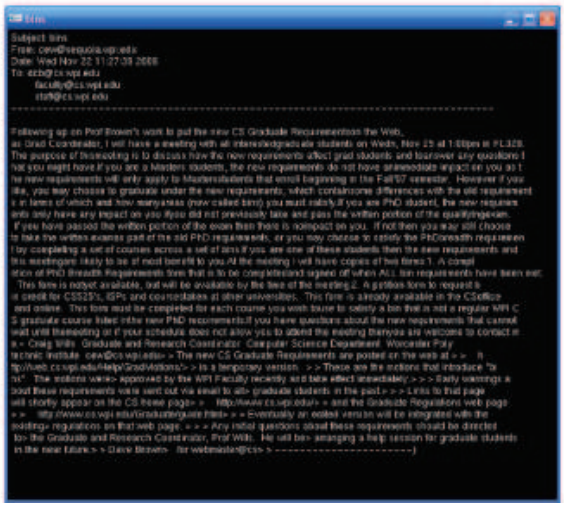
Additionally, each glyph is color coded. The key to the color code is also on the left.



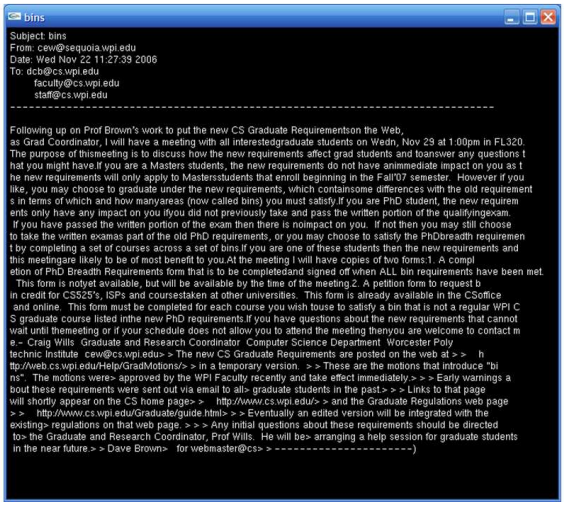
Clicking on a glyph displays the email message.



From this view, you can press 'r' to reply to the email



Pressing 'd' marks the email for deletion.



Pressing 'q' brings you back to the glyph view.