Behavior-based Email Analysis with Application to Spam Detection

Shlomo Hershkop

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2006

Abstract

Behavior-based Email Analysis with Application to Spam Detection

Shlomo Hershkop

Email is the "killer network application". Email is ubiquitous and pervasive. In a relatively short timeframe, the Internet has become irrevocably and deeply entrenched in our modern society primarily due to the power of its communication substrate linking people and organizations around the globe. Much work on email technology has focused on making email easy to use, permitting a wide variety of information and information types to be conveniently, reliably, and efficiently sent throughout the Internet. However, the analysis of the vast storehouse of email content accumulated or produced by individual users has received relatively little attention other than for specific tasks such as spam and virus filtering. As one paper in the literature puts it, "the state of the art is still a messy desktop" (Denning, 1982).

The Problem: Email clients provide only partial information - users have to manage much on their own, making it hard to search or prioritize large amounts of email. Our thesis is that advanced data mining can provide new opportunities for applications to increase email productivity and extract new information from email archives.

This thesis presents an implemented framework for data mining behavior models from email data. The Email Mining Toolkit (EMT) is a data mining toolkit designed to analyze offline email corpora, including the entire set of email sent and received by an individual user, revealing much information about individual users as well as the behavior of groups of users in an organization. A number of machine learning and anomaly detection algorithms are embedded in the system to model the user's email behavior in order to classify email for a variety of tasks. The work has been successfully applied to the tasks of clustering and classification of similar emails, spam detection, and forensic analysis to reveal information about user's behavior.

We organize the core functionality of EMT into a lightweight package called the Profiling Email Toolkit (PET). A novel contribution in PET is the focus on analyzing real time email flow information from both an individual and an organization in a standard framework. PET includes new algorithms that combine multiple models using a variety of features extracted from email to achieve higher accuracy and lower false positive than any one individual model for a variety of analytical tasks.

Contents

List of Figures

List of Tables

Acknowledgments

This thesis would not have been possible without the support, direction and love of a multitude of people. First, I have been truly blessed to have a wonderful adviser guiding me on the path of scholarship, whose gave me the unique chance to explore the ramifications of my work in different research areas. Professor Salvatore J. Stolfo, has led me through the beginnings of a project that was to become my thesis and then let me lead my thesis work once I knew where I was going. His constant support and encouragement has allowed me to balance my Ph.D. career in all aspects of academia, research, family and friends.

I have been blessed to have had many people supporting my endeavors for scholarship since the very beginning of this work, playing multiple roles for which I am greatly thankful for:

My wonderful wife, Dr. Chana R. Hershkop (PharmD) who has been a constant support and inspiration during my academic career. Her unwaivering faith in my abilities and real time feedback on new ideas will always accompany me throughout my career. And to my daughter Esther Odel, who has been providing constant distraction and entertainment since the first second she was born.

My parents Eliezer and Chaya Hershkop, and my siblings for the moral

support and constant irritation of asking about when I would be finished with the dissertation;

My In-laws, Dr. Chaim and Chavi Goldberg who have always treated me as one of the their own and allowing me to marry their wonderful daughter. Also a special thanks for having the patience when I took over their computer network for my own needs during visits;

My thesis committee members: Gail Kaiser, Angelos Keromytis, Richard Segal, and Alexander Tuzhilin for their time and effort. Thank you for taking the time to critically read the thesis and offer me positive criticism and ask insightful questions that enabled me to clarify claims and contributions which needed additional coverage in the thesis;

The EMT design team and fellow students: Ke Wang, Wei-jen Lee, Charlie Hu, and Oliver Nimerskin without whose help my thesis' hypothesis would be untested and the system sorely missing the needed features.

Special thanks to the IBM Anti-Spam team for letting me work with them on developing these ideas in the real world. Specifically I would like to thank Jeff Kephart, Richard Segal, Mark Wegman, V. T. Rajan, and Joel Ossher.

My officemates over the many Ph.D. years: Eleazar Eskin, Ke Wang, and Wei-jen Lee; My fellow students Paul Blair, Phil Gross, Janak Parekh, and Suhit Gupta with whom I shared many hours of discussion on general research and computer skills.

The IDS group, both past and present: Matt Miller, Stephen Jan, Alan Lue, Ryan Ferster, Johnie Lee, John Gasner, Paolo De Dios, Zhi-Da Zhong, Ying Wang, Matthew Schultz, Kyri Sarantakos, Xue Rui, Christy Lauridsen, Eli Glanz, Sean

Escola, Andrew Howard, Andrew Honig, Tania D'Alberti, Bill Bruner, Frank Apap, Grace Zhang, Ganesh Sastry, Alvin Mai, Liu Tie, Manoj Lala, Rakan El-Khalil, Manasi Bhattacharyya, Rahul Bhan, Daniel Paluch, Edward Mezarina, Joanna Gilberti, Stephan Ortman, Edward Young, Devang Thakkar, Nick Edwards, Linh H Bui, Peter Davis, Charlie Hu, Morris Perl, Shahmil Merchant, Suhail Mohiuddin, Anna Kozynenko, and Spencer Whiteman and others whom I forgot to mention;

The Computer Science Department's front office: Rosemary Addarich, Alice Cueba, Twinkle Edwards, Pat Hervey, and formerly; Martha Zadok for pulling strings for me in times of need;

I am also grateful for the comments from the anonymous reviewers of the papers I have had the privilege of publishing in conferences and workshops.

Much of the material in this thesis is based upon work support by the National Science Foundation and DARPA. In particular NSF grant Email Mining Toolkit Supporting Law Enforcement Forensic Analysis from the Digital Government research program, No. 0429323 and the support of the Luis Morrin Foundation. Any opinions, findings, and conclusions or recommendations expressed in this dissertation are mine alone and do not necessarily reflect the views of any of the above funding agencies.

Shlomo Hershkop

COLUMBIA UNIVERSITY

February 2006

To my dearest wife Chana, All this would not be possible without you.

To my daughter, Esther Odel.

To my parents, Eliezer and Chaya Hershkop.

To my parents-in-law Dr. Chaim and Chavi Goldberg.

Chapter 1

Introduction

1.1 Email Communication

Email is one of the most popular forms of communication today. The surprisingly fast acceptance of this communication medium is best exemplified by the sheer number of current users, estimated to be as close to three quarters of a billion individuals, and growing (IDG.net, 2001). This form of communication has the simple advantage of being almost instantaneous, intuitive to use, and costing virtually nothing per message.

The current email system is based on the SMTP protocol RFC 821 and 822 (Postel, 1982; Crocker, 1982) developed in 1982 and extended in RFC 2821 in 2001 (Klensin, 2001). This system defines a common standard to unite the different messaging protocols in existence prior to 1982. It allowed users the ability to exchange messages with one another using a system based on the SMTP protocol and email addresses. These protocols allowed messages to flow from one user to another, making it practical and easy for different users to communicate independent of the

service-provider or the client application.

In 1982, Denning (Denning, 1982) wrote about the problem of working with email, asking

> "Who will save the receivers from drowning in the rising tide of information so generated?".

The organization of email files was never adequately addressed by the designers of email client applications. File organization for holding email message stores has not substantially changed in the last 25 years. Emails for the most part are held in data files or folders with no structured relationship (flat files), making anything more than a keyword search very slow. Users may choose to move messages into time-ordered sub-folders of related messages. Studies have shown that typical users quickly generate anywhere from tens to hundreds of folders in a relatively short amount of time. Finding a particular past message across these sub-folders can easily turn into a daunting task. Not only is the email the subject of search, but also the folder in which it might have been placed! Within these flat file folders, attachments are encoded in MIME format making analysis of anything other than simple filename close to impossible. Recent tools have been released which allow indexing and searching local data including emails and parts of attachments. However, there are no widely available email applications with mechanisms to compare two attachments.

Above and beyond simply sending messages, studies have shown that many users have quickly adopted email to a variety of tasks including task delegation, document archiving, personal contact list, reminder and scheduling (Ducheneaut and Bellotti, 2001; Whittaker and Sidner, 1996). For example, a typical users will

use their INBOX or main message area, as an active "to-do list", leaving current messages on the top of the list . Even for well organized users who always maintain past messages in appropriate sub-folders, there remains the possibility of down-time, and hence, over a relatively short period of time, bursts of email can quickly accumulate making organization of these new messages a slow and difficult task (Mackey, 1988).

In addition to these organization issues, the Achilles heel of the current email system is its relative ease of abuse. The protocols were based on the assumption that email users would not abuse the privilege of sending messages to each other. The misuse and abuse of the email system has taken on many forms over the years. Typical misuse include forged emails, unwanted emails (spam), fraudulent schemes, and identity theft and fraud through "Phishing" emails. Abuse includes virus and worm attachments, and email DOS attacks. The common denominator among all these categories is they exploit the email system's lack of controls and authentication of sender and recipient (an inherit problem in a decentralized system). Email is not permission based, and one can simply send a message without prior approval. Users should not be expected to pay a repair bill for simply opening an email which seemed to have originated from a friend's email address, spoofed by an abuser.

1.1.1 Problem Statement

Email has seen explosive growth of usage in the last twenty-five years. As such, there are issues and problems facing a system that has not kept abreast of technological developments.

1.1.1.1 Information Overload

The combination of low cost, high bandwidth Internet connections, the decline of the cost per megabyte of storage, and the growth in the number of email users, has resulted in an explosion of email data per user. Sorting through all of this data, and separating the good from the bad, is a daunting task. With hundreds of messages, kept in tens or hundreds of folders, sometimes across different email accounts, this is close to impossible. If the current trend continues, users will need a fast way to organize, prioritize, and search their email archives and of understanding the underlying data. Furthermore, the current set of email clients provides no facilities to manage and analyze increasing attachment sizes with varied formats, types, and many copies of edited versions of the same attachment.

To put this in perspective, some medical professionals are now using email to communicate with their patients. Especially with the rise of 'Boutique Doctors' patients are paying extra to have constant attention (Zuger, 2005). This may create a liability issue in the near future; if a doctor does not respond to an emergency email request in a suitable time frame and patient care is affected. The lack of a timely response may be treated as an act of indifference, or worse an act of incompetence. This is not an impossible scenario, either because a spam filter misclassified the patient's message, or because the flood of unwanted messages prevented the doctor from seeing the email on a timely basis.

1.1.1.2 Message Handling

In most current email implementations, the user has almost no control over which messages enter and exit their email account. A limited amount of control is some-

times provided through the use of client side filters or rapidly clicking on the delete key. Many available filters are either hard coded rules, or simple pattern matchers, directing messages to specific folder destinations (Cohen et al., 1996; Diao et al., 2000; Provost, 1999).

Some recent progress has been made on automatic methods to help the user select folders through helpful suggestions (Crawford et al., 2001; Segal and Kephart, 1999). But both works treat this approach as a simple text classification problem and do not consider the overall picture of the user's usage of the email system. In addition, some of the filters operate on a rule-based system and need to be frequently updated with new rules to remain effective. Anecdotal evidence from the security domain has shown that this updating phase is the weakest link in the chain of protection.

1.1.1.3 Message Analysis

When analyzing large sets of emails or attachments generated by a single user or group of users, the common approach is to treat the problem as if the data was one large email box. The most sophisticated analysis is to count of the number of messages in a user created sub-folder. Basic flat searches and name, date, and topic sorting are the most commonly available functions. In addition, current email clients have no analysis tools for quickly analyzing past messages or attachments within a user's email box. Profile views of the data for different tasks should be made available to the user, to enable them to understand a message in its historical context. For example an automatic list of emails which have not received responses can be generated for the user to show them any 'open issues' they might have in

their email box.

Researchers at Microsoft have also begun to work on analyzing user's email behavior. The Lifebits project (Gemmell et al., 2003; Aris et al., 2004; Gemmell et al., 2004) includes email as an important source of information to create new views to a user of their own historical data.

In addition to the issue of managing large stores of emails for analysis there is the problem of noise in the data. The propagation of spam email has increased in noticeable proportions for the last few years (Postini.com, 2004; CDT, 2003). Junk mail has evolved since the introduction of email, from being an occasional annoyance to a veritable flood. The background flood of spam is estimated at between 60 and 80 percent as reported by some ISP's (Postini.com, 2004; Leyden, 2003). This is effectively a DOS attack against the email infrastructure. Even with good filters in place, there are messages which are misidentified as either good or unwanted, which make the analysis of specific folder content harder to accomplish.

1.1.1.4 Protection

Email is a convenient medium to share files as attachments with other users in a group. Malicious attachments propagating viruses or worms are creating havoc with the email system and wasting email and IT resources. Current email service providers utilize one or more integrated anti-virus products to check and identify malicious attachments. Most current anti-virus products work on the basis of signatures. Experts encode unique signatures to identify malicious programs. Typically, these signature databases are updated weekly, daily, or hourly depending on the specific product and configuration. The advantage to a signature-based system is

the low false positive rates. Because signatures match exact patterns, they can detect known viruses with very high level of confidence. A strong shortcoming to this system is that they are based on known virus signatures, and as such, cannot detect unknown or new malicious attachments, i.e. they do not solve the zero-day virus problem. Substantial research on using heuristics and machine learning algorithms to learn virus patterns has been explored (Kolter and Maloof, 2004; Jeffrey O. Kephart and White, 1993; Schultz et al., 2001b; Tesauro et al., 1996), but slow to propagate to common deployment.

Forged emails represent another side of the same problem. Because the current email system implementation does not explicitly include any authentication mechanisms, emails can be easily forged at least on the surface level. A recent story highlighted this exact problem; multiple security warnings containing malicious attachments disguised as updates originating from the Microsoft security group were sent out. It was so believable that it was even mistakenly posted to BugTraq, a highly respected security mailing list, as a recommended update.

Finally, like credit card users, email users typically have multiple email addresses or aliases for their different needs. In addition, over time, user typically change email accounts, as they switch jobs or internet service providers. Currently, there are no systems to profile the user on one account and apply this profile to a new or shared account to provide fraud and misuse protection. Fraud detection successfully applied to both credit card and cell phone usages (Stolfo et al., 1996; Chan and Stolfo, 1998; Cahill et al., 2000; Hollmen, 1999) has not been applied to email accounts. On both the client and server level, being able to detect a fraudulent email would mitigate some virus attempts and prevent email identity

theft.

1.2 Email Modeling

Our thesis is that advanced data mining and machine learning techniques implemented for email analysis can provide the infrastructure enabling a new generation of applications that help in solving many of these problems.

In this thesis, we show that a user's email archive can be used to model the behavior of a user to protect their email account from misuse and abuse. We describe an implemented system for mining email data to build a diverse pool of models based on behavior. We present our results in the domain of spam detection. To show the general utility of the approach we also describe the use of the same methods in the framework of a forensic investigator dealing with large amounts of unclassified and or unknown emails.

The functionality provided by EMT's behavior-based models can provide new capabilities to increase the productivity of users when processing their email. This would reduce the information overload by providing advanced search capabilities for emails and attachments, provide automatic categorization using behavior as an organizing principle, and permit advanced automatic analysis to reveal typical behavior that is useful in detecting abnormal email indicative of abuse or misuse.

EMT has an implemented an anomaly detector, priority mechanism, email file system organization, and email classifier. We describe our system to combine these new models alongside traditional information retrieval models to augment email protection. We show how these techniques have utility over the current state of the art, and show why and how these models can be used in other applications.

1.3 PET: an Automatic Behavior Modeling Toolkit

EMT has been developed over time by a team of researchers. EMT primarily was designed for an analyst interested in studying email archives for any purpose. In our thesis we have designed and fully implemented a new email client application for end users of email. The Profile Email Toolkit (PET) utilizes EMT as its core engine, but provides very specific new features to demonstrate the power of email data mining.

PET extends the current email client application from being a simple reader application to a more robust analysis tool. In particular, the task of filtering out unwanted messages can be easily achieved if one were to use behavioral models to learn which messages are important to a particular user and which are not. It also allows the user to perform clustering of similar email and reordering email based on a novel priority scheme modeled on past behavior. A more detailed description of this scheme and exact definition of similarity is addressed in Chapter 7.

PET includes many of the EMT functions, wrapped in a real-time package. PET is implemented as an interface to Mozilla's Thunderbird email client and a detailed description is provided later in this Thesis.

Here we outline the core features and functions provided by PET. Screen shots are provided in Appendix A to display the user presentation of the new email client. PET is also available for downloading by contacting the author.

1.3.1 Data Organization

Most email clients store the email data in some form of a continuous flat file. We have redesigned this standard storage model to make it easier to access and

analyze the email data. Email data is stored in an underlying database scheme, which we describe in section 5.2. This allows the system to quickly find an email and also build new models based on historical email data. In addition, PET can be configured to only index the client side email data and keep basic statistics over the data it has seen. This configuration has the advantage of letting the client email system manage the email storage allowing client design changes to not affect the PET system.

1.3.2 Priority Reorganization

Functions embedded in PET , and hooked in the client side application, allows the user to reorganize emails based upon the user's prior behavior. This feature is described in Chapter 7.

1.3.3 Information Retrieval Models

In addition to behavior models, we also embed a set of standard Information Retrieval models in PET to allow the user easily train a classifier or combination of classifiers over some subset of the emails, and automatically classify incoming emails. These classification models can either be used for prioritizing email or for the user's "acceptance criteria", discussed in Chapter 3.

1.3.4 Behavior Models

We provide a brief overview of the underlying analysis of the models provided by EMT and embedded in PET.

Stationary User Profiles - Histograms of past activity are used to model the behavior of a user's email account over time. Histograms can be compared to find similar behavior or abnormal behavior within the same account (between a long-term profile histogram, and a recent, short-term histogram), and between different accounts.

Attachment Statistics - PET runs an analysis on each attachment in the database to calculate a number of metrics. These include birth rate, lifespan, incident rate, prevalence, threat, spread, and death rate. They are explained fully in (Bhattacharyya et al., 2002) and in section 3.3 and are used to detect unusual attachment communication indicating security breaches or virus propagations.

Similar Users - User accounts that may behave similarly may be identified by computing the pair-wise distances of their histograms (e.g., a set of spam accounts may be inferred given a known or suspect spam account as a model). Intuitively, most users will have a pattern of use over time, which spamming accounts will likely not follow. (Spam bots don't sleep or eat and hence may operate at times that are highly unusual.) Thus it provides one way of identifying email senders who behave like spammers, allowing for example an analyst to quickly weed out potentially uninteresting target emails.

Group Communication (Cliques) - Graph-based communication analysis identify clusters or groups of related email accounts that frequently communicate with each other. This information is used to identify unusual email behavior that violates typical group behavior. For example, intuitively it is doubtful

that a user will send the same email message across all of their social groups. Even social announcements (wedding or deaths) typically are not sent to everyone in one's email contact list. When some legitimate event does occur and an email is sent across cliques, one would like to be able to automatically reference such events when analyzing archived data. From a misuse detection point of view, a virus attacking an email account book would surely not know the social relationships and the typical communication pattern of the victim, and hence would violate the user's group behavior profile if it propagated itself in violation of the user's "social cliques".

Clique violations may also indicate internal email security policy violations. For example, members of the legal department of a company might be expected to exchange many Word attachments containing patent applications. It would be highly unusual if members of the marketing department, and HR services would likewise receive these attachments. PET can infer the composition of related groups by analyzing normal email flows and computing cliques, and use the learned cliques to alert when emails violate clique behavior.

Recipient Frequency - Another type of modeling considers the changing conditions of an email account over sequences of email transmissions. Most email accounts follow certain trends, which can be modeled by some underlying distribution. As an example of what this means, many people will typically email a few addresses very frequently, while emailing many others infrequently. Day to day interaction with a limited number of peers usually results in some predefined communication patterns. Other contacts communicated to on less

than a daily basis have a more infrequent email exchange behavior. These patterns can be learned through the analysis of a user's email archive over a bulk set of sequential emails.

Every user of an email system develops a unique pattern of email emission to a specific list of recipients, each having their own frequency. Modeling every user's idiosyncrasies enables the PET system to detect malicious or anomalous activity in the account. This is similar to what happens in credit card fraud detection (Chan and Stolfo, 1998; Fawcett et al., 1998), where current behavior violates some past behavior patterns. Changes in usage patterns are discussed in 5.4.5 and 3.2.5.

VIP Users - The recipient frequency analysis identifies the relative importance of various email users. By extending the analysis to compute a "weighted response rates" to a user's typical recipients, one can learn the relative rank ordering of various people. Within the same organization, to whom a user responds immediately to, are likely important people to either the user or the organization.

Besides this rich collection of email user behavior used by PET , which do not rely upon any content-based analysis, PET also provides the means of statistically modeling the content of email flows. Several of these are described in Chapter 3.

1.4 Contributions of this Thesis

This thesis builds upon our earlier work on EMT and makes five additional contributions to the field of Data Mining and the related field of Information Retrieval.

We briefly summarize these contributions as follows:

Email behavior models - We present new models to represent email usage. These models represent past usage patterns and calculate current pattern deviations using standard metrics. The types of models and pattern calculations are described in Chapter 3. Our research shows that unwanted messages can be detected by considering only the behavior of the email user.

Behavior Profiles - We develop the concept of a behavior data structure which stores the user behavior in a compact representation. The representation not only allows behavior comparisons to be done efficiently, but also allows a user to port their behavior profiles between machines and accounts.

Framework for mining email data - We present a data base back-end to store and analyze email data. The advantage of the database file system is that it is both fast and scalable. It allows individual features for statistical analysis to be quickly and easily calculated without having to process all the data sequentially with a custom built application.

Automatic message prioritization - We introduce a novel scheme to prioritize messages in an email collection based on past behavior in Chapter 7. The user's own behavior can be used as an indication for which messages and users are more important and hence PET reorders the list of emails based on this criteria.

Spam classifier model combinations We have developed a novel scheme for combining various spam classifiers to achieve higher detection and lower false positive rates. The combination mechanisms are described in Chapter 4.

1.5 Guide to the Thesis

The remainder of this thesis is organized as follows:

Chapter 2 examines the basic email information organization, modeling and classification, and filtering techniques. We examine the current state of the art of the research literature in each of the topics, and enumerate the current available implementations.

Chapters 3 through 7 discuss how to model email flows, how to combine models, and how to use them to organize email.

Chapter 3 describes how email is actually represented in the underlying implementation and gives an overview of the machine learning algorithms used in the system. We present the different types of models and compare them to one another.

Chapter 4 gives the theory behind the methods for combining the email models. We review current literature and theory to show how this is novel, and how it has been applied to email data.

In Chapters 5 and 6, we discuss the implementation of PET and the underlying modeling system EMT. Chapters 8 and 9 evaluate the models in use in real-world scenarios, discuss the results and offer some conclusions.

EMT an automatic email mining toolkit is described in Chapter 5. Chapter 6 covers the main contribution of the thesis, with the design of a pluggable data mining toolkit for an email client. We have implemented it specifically for Mozilla Thunderbird but this can be extended for any client. We detail the setup and organization of the application.

In Chapter 7, we develop the notion of how to group similar email. We define what similar means in different contexts and how to use it to reorganize groups of

email messages.

Chapter 8 describes the email corpus, and presents the results of our experiments over the data for the task of spam classification. We review the performance of different models and different combinations of models.

Chapter 9 concludes the thesis with supporting materials, and reviews user needs in email data mining and how PET is able to fulfill these needs. We discuss the implications of the thesis and recommend areas for future research to broaden the scope of the thesis and its possible application to forensic and behavior applications.

Appendix A contains screen-shots of the actual EMT application. We will make reference to this section throughout the thesis.

Appendix B contains a description and outline of the email parser engine.

Chapter 2

Background

This chapter presents background literature on the areas of organizing, classifying, and prioritizing emails. We also provide background on the machine learning models elaborated more fully in Chapters 3 and 4. Once the current state of the art is presented, we outline the goals of a complete system addressing the shortcomings of current email clients and illustrate the ideas in the subsequent chapters.

2.1 Mail Organization

Research on how to effectively organize email stores has not attracted a lot of attention in the research community. A survey of email clients show that for the most part they do nothing more than store emails as flat files, with the exception of using indexed data structures for faster retrieval (Gross, 2002).

Piles is a proposal for visually organizing email stores in piles instead of folder hierarchies (Mander et al., 1992). *PostHistory* allows individuals to explore their email archives over time with a unique visual approach (Viegas et al., 2004).

Other work has proposed using treemaps (Fu, 2003), self-organizing maps (Keim et al., 2005), and timelines (Mandic and Kerne, 2005) in a similar fashion. Visual manipulation of an organizing folder is interesting but the scheme might not scale to the amount of messages in a typical folder (i.e., piles can grow very high and deep). It has been found that as the number of messages grows, the folder approach seems to degrade quickly (Whittaker and Sidner, 1996). Visually summarizing the contents of a folder would be a useful tool for any email user.

Lifestream is a proposal to organize personal data including email as a time-ordered stream of information (Freeman and Gelernter, 1996), as an alternative to the primary metaphor of directory or folder organization. The authors point out that directory organization is inadequate for organizing electronic information and an alternative principle has not been implemented by any mail client vendors. An interesting observation made in (Fawcett and Provost, 1997) is the fact that there is no reason an electronic message cannot be associated to more than one folder (without maintaining multiple instances). We will expand on this later in section 5.4.1.

This thesis differs from the work on life-streams in several ways including the use of an underlying database as opposed to flat files, and the use of the behavior models. The power of an underlying database provides PET with an easy means of representing data in a more flexible and efficient manner, but also allows a variety of models to be readily implemented in the style of OLAP. Furthermore, PET reveals information about a user's behavior in far more detail than what is available by Lifestream. This is discussed in detail in Chapter 5.

REmail is an IBM prototype for reinventing email (Rohall et al., 2003; Kerr

and Wilcox, 2004). Research done has been applied to IBM's Lotus email client (IBM, 2005). Although Lotus does have an underlying database design, most users do not have it deployed on their personal machines and are not as familiar with its design and use.

Some of the literature have argued (Boardman et al., 2002) that solutions must encompass other tools outside of the email box. Personal Information Management (PIM) tools try to bridge the gap between email, scheduling, and other available tools. We believe that there is still much innovation that needs to be done before we can abandon the email box. Since users do spend so much time reading and answering emails, we need to augment the INBOX to provide more information to help the user without adding to the problem.

2.2 Message Prioritization

Message prioritization refers to the task of reordering a group of messages into an ordered list relevant to some ordering. In general most email clients will order messages by timestamp, with newest messages either first or last. Some clients also allow user to order messages by sender's or recipient's email, subject line, and size.

Multiple studies of user email habits have shown great discrepancies between the types of users and how they maintain order in their mail box (Mackey, 1988). An overview of different surveys is given in (Cadiz et al., 2001). Their conclusions are that users use email clients for all types of tasks in their daily routine outside of the basic send and receive model of message exchange. Furthermore, (Cadiz et al., 2001) studied the affects of how threaded messages can be used to dealing with email backlogs while the user is away from the system for a prolonged period

of time. They concluded that threading was helpful in dealing with the deluge of emails by helping the user focus their attention to more important messages first. Prioritization would allow messages to be reordered by importance accomplishing the same without having to learn the subtleties of a clustering system.

Work by Horvitz et al (Horvitz et al., 1999) studied the prioritization problem from a cost of interruption point of view. Using a set of email messages labeled by each user as important, they trained an SVM classifier (Vapnik, 1995) and ordered new messages based on how confident they were that they matched previously seen messages. Their goals was to decide if to interrupt the user if a new message arrives, based on the importance of the trained classifier. Because this is based on trained classifiers, it can only generalize to classify message based on a snapshot of current messages, and would have to be retrained over time, to learn new behavior of each user.

Using software agents in reducing information overload is mentioned in (Maes, 1994; Gruen et al., 1999; Lashkari et al., 1994). The approach uses a memory based learning approach (Stanfill and Waltz, 1986) to learn appropriate email based tasks such as reminders, sorting, and action suggestions. But beyond basic prototypes, user agents still present many problems before they can be actually implemented for users to use in every day tasks (Nwana, 1995; Wooldridge and Jennings, 1995). Our approach differs in the fact that we do not aim to replicate a user, but rather enhance the email experience with email client augmentations.

Studying mail servers on how to predict spam messages and introduce delays so that non spam messages are delivered more quickly is an approach taken by (Twining et al., 2004). Although a very promising approach to reduce server loads,

it ultimately ends up delivering the spam messages albeit with some calculated delay, leaving it as a problem that the end user needs to address.

2.2.1 Threading

Threading messages, has been used for years by newsgroup readers as a way of organizing message topics. They are usually based on linking subject lines or looking at the message 'reply-to' id in the email header field.

Recent work by (Venolia and Neustaedter, 2003; Kerr, 2003) on visualizing conversation threads are excellent propositions, once an important or relevant email has been located. If the user has a few hundred messages sitting in the INBOX, without priority reorganization, picking the start or middle of an interesting thread is not an easy task.

2.2.2 Reordering

ClearContext Inbox Manager for Microsoft Outlook has an add-on program to the Microsoft Outlook email client that features different organizational tools. They have implemented some priority tools, but do not provide much information on the underlying technology. They estimate contact priority by using volume as an indication of importance. We have a more sophisticated model (see Chapter 7), but can not directly compare the two technologies as they do not disclose much detail about their implementation.

Related to email organization is work on automatically organizing voice mail messages in (Ringel and Hirschberg, 2002). They use a static set of past messages to learn features and try to prioritize voice mail messages based on those features.

Our work differs in that we are using a rolling model over time, allowing messages to shift priority as new behavior is observed.

2.3 Email Classification

One way to help the user organize email is to have the email client automatically either discard or move messages into specific folders for the user's convenience.

One of the earliest systems (Pollock, 1988), called *ISCREEN*, had a rich set of rules and policies to allow the user to create rule sets to handle incoming emails. *Ishmail* (Helfman and Isbell, 1995) helped organize messages, by also providing summaries to the user on the status of what and where groups of new messages where being moved.

Related to rule-based systems, work studying outgoing email flows that may violate hard-coded policy rules are presented in (Lee and Park:, 2003; S.Vidyaraman et al., 2002).

The underlying technology here is the rule-based systems. They make the assumption that all new emails can be classified by some set of rules. The problem is two-fold. First, all the rules must be specified by the user, not an easy task for many users. Second, and more importantly, the rules must be constantly updated by the user since unwanted email authors would presumably be able to test their technology against an installed rule system, and adopt. We advocate a system of automatically learned models from past behavior (the subject matter of this thesis) rather than a rule-based system for a targeted task. For example for automatic prioritization, or to automatically archive specific types of emails (e.g. confirmations of receipt of an email) would be an ideal task for automatic models. Fraud

detection using customer profiles are currently in use in the credit card domain and cell phone systems (Hollmen, 1999; Fawcett and Provost, 1997; Cahill et al., 2000). As of yet, they have not been applied to the email domain.

2.3.1 Defining Spam

Email today is not a permission-based service, yet one may observe and model the individual user's behavior to calculate a prediction of how a user would treat a specific message. Computer algorithms can learn what types of emails the user opens and reads, and those which he/she immediately discards. For example, electronic bills or annoying forwards from friends might be unwanted, but they are not spam if the user reads them and sometimes responds. Those emails have a clear source marked on the email and a relatively easy method will stop those emails (by specifying a simple filter rule) from reoccurring if the user so desires (block forwards from user X).

For the day-to-day usage of email the biggest challenge facing users is recognizing and dealing with misuse and abuse of emails. In general this means dealing with unwanted messages, which can quickly grow and overwhelm most users. To deal with this problem, many systems are implementing spam filters to automatically move all spam messages to special spam folders.

Informally, most users seem to agree on which of the email messages they would classify as spam. They can identify spam messages without having to actually open and read the message, so why shouldn't computer algorithms be able to do the same? The envelope of the email plus the user's past behavior should provide sufficient information to decide that an email is unwanted and can be comfortably

filtered.

In a general sense, there has been a weak consensus in the research literature for an acceptable definition of spam. We employ a simple definition of spam:

Spam - is all email the user does not want to receive and has not asked to receive (Hershkop and Stolfo, 2005b).

Thus, spam is in the eye of the beholder, and is therefore all unwanted emails that the user cannot easily stop from receiving. (We distinguish this from email the user would rather have not received but accepts anyway.) Some of the recent literature has correctly arrived at the same definition, but also include UCE (unsolicited commercial email), which only includes a small portion of the overall spam problem.

Why does spam exist? Email is a very cost effective method of marketing legitimate products or services to millions of users. Physical bulk mail per recipient costs are substantially higher (about 100 times higher) than email advertisements (Mangalindan, 2002). At the same time, email can also be used to conduct scams and confidence schemes to steal information or user identities (Krim, 2003; Cranor and LaMacchia, 1998). Although only a minute percentage of email users respond to spam messages, given the low cost of distribution, it is enough to fuel the popularity and existence of spammers and spam messages (Sullivan, 2003).

Note, that the definition of spam is entirely divorced from the actual semantics of the email message or the number of recipients of the message. Current literature focuses the definition of spam relying on the contents of the email message, and upon the frequency of re-occurrence of the same email message or body text among groups of users. Generally the current work defines spam to include

unsolicited advertisements (CDT, 2003; Hall, 1999; Kolcz and Alspector, 2001; Manaco et al., 2002) , fraudulent messages (Mertz, 2002; Hidalgo and Sanz, 2000; Drucker et al., 1999) , and adult content (Provost, 1999; Sahami et al., 1998) emails.

Although all these definitions may be acceptable, all of these are content-based definitions whose category depends upon the meaning of the message but which is unknown to the filter. Deriving the meaning of an individual email is a difficult problem complicated by the fact that most messages are very short and hard to understand as a standalone document. The literature will usually include a notice by an author that their definition of spam is somewhat arbitrary based on individual users' interpretations of the contents. Our definition does not have the limitation of content, as it is defined in a sense on a per user basis, and is based upon the user's behavior, as we shall explain fully later.

It is important to note that spam is not only annoying to the individual user (Fallows, 2003), but also represents a security risk and resource drain on the system. By weeding out spam from the email stream, the user is once again empowered to use their email for what it was mean to be, a personal communication tool.

Much attention in the recent research literature has focused on the email spam problem. The current state of the art of anti-spam research and solutions are for the most part ad hoc efforts concentrating on specific areas of the problem. No formal study of the entire problem and solution set has been proposed except for a recent report outlining some data mining issues (Fawcett, 2003). We now outline current efforts and the state of the art of filtering unwanted messages. Current solutions can be outlined as detecting and filtering email spam from among the normal emails. The solutions proposed can be divided into four general approaches:

preemption, legislation, protocol reimplementation, and filtering.

2.3.2 Identity Protection

The first is preemptive protection; putting the burden of protection on the user. These methods (CDT, 2003) instruct the user to encode or hide their email address in such a way so that spammers do not harvest their email addresses. For example encoding "example@domain.com" as:

```
&#101;&#120;&#097;&#109;&#112;&#108;&#101;&#064;&#100;
&#111;&#109;&#097;&#105;&#110;&#046;&#099;&#111;&#109;
```

Other techniques suggested (Hallam-Baker, 2003) include choosing unusual email addresses, hiding them as graphics, and using multiple throwaway or one time use email accounts. Extend-able email addresses (Gabber et al., 1998) is a related solution, which works on the basis of adding a hash to the beginning of the email address and then hard coding rules as to what to do with each hash. For example, `Shlomo@cs.columbia.edu` would become `Shlomo+Xdf345@cs.columbia.edu` with the email server responsible for delivering the message to the correct recipient, and the end user running rules on how to treat each hash. An automatic system to generate alias accounts, and to time bound them is proposed in (Gburzynski and Maitan, 2004). Single purpose address (SPA) schemes encode the policy in the email user string to allow automatic policy enforcement (Ioannidis, 2003). In these scheme the SMTP protocols remain the same, but the user gains the ability of generating unlimited emails addresses.

We challenge the underlying assumptions that it is the user's problem to deal with the reality of spam and must accept inconveniences in order to use the email

system. This would be ideal if the level of spam was only for example two percent of the total email traffic, not the flood that it is. Additionally it forces users to learn unusual email extensions and makes it hard to give out email addresses without looking up an extension. A subset of this approach is also hiding schemes that try to encode email names on public Internet web pages in such a way that make it easy for users to see, but hard for spam programs to pick up. We see no reason why spam bots can not be upgraded with some simple rules or OCR techniques to recompose these hidden email addresses.

2.3.3 Legislation

The next approach proposed to dealing with the Spam problem is legislation. We address this issue only because it is being actively debated and pursued by a community who has the rather sobering view that no technological solution will suffice to solve the spam problem (Weiss, 2003). Legislation aims to solve the problem of unwanted messages by creating laws governing the use of email. Recent laws such as Can-Spam (Controlling the Assault of Non-Solicited Pornography and Marketing Act of 2003) (U.S. Senate and House of Representatives, 2004) have had little effect except to make it harder for legitimate advertisers and mailing lists to send out bulk email. Those who ignore the law will be happy to continue ignoring the law. Since going into effect on January 2004, spam levels have not in any way been affected or slightly increased (Gaudin, 2004; Fallows, 2005) and in fact, it actually legalized certain forms of spam (Lee, 2005). Some feel, that having additional laws will just have the effect of having the spammers send out email from servers that are out of jurisdiction of the law.

In addition, recent reports (ZDNet UK, 2005; Leyden, 2003) suggest 80 percent of spam is sent out from compromised hosts leaving the question of who is responsible still up in the air. Further laws might make ordinary people responsible or their hijacked machine's spam behavior.

2.3.4 Protocols

The next solution that has been proposed calls for the overhaul of the entire email system transforming it into a permission-based system. Making the assumption that a new protocol might solve the problem once and for all, designers have suggested multiple ways to fix all the security concerns and authentication mechanisms of the current system. The first problem is that the current open email system is very much entrenched, making it unrealistically hard to implement a new protocol. Second, deploying a new system across the entire Internet in the foreseeable future is a very hard task.

In this thesis we are only interested in those solutions which work within the current SMTP protocols. There are many original solutions to deal with the spam problem that unfortunately require a complete overhaul of the current protocols. For example related to protocol changes, challenge-response systems try to make sure a human is at the other end of a first email from a new user. That is a really good idea, until one realizes that the entire Internet does not necessarily speak or understand the same language. For example, a user who would want to register a product through email, might have to contend with a challenge in some foreign language. This would also require some level of sophistication from most users, something hard to imagine given past performance on other technology issues.

SPF (Lentczner and Wong, 2004) would add a caller-id like feature to email servers. Ideally this would ensure the identity of the sender. However, protocol redesign solutions are for the most part impractical at this point, as the current system is tightly integrated into many subsystems, and cannot be replaced overnight. To give a simple example, many sensors and appliances will send out alerts on the network using the email protocol. Under SPF, these messages would be prevented from being delivered, as they are not registered. Some of the larger ISP's are trying to adopt SPF as a spam solution, but SPF's own website clearly states it is only to prevent forged from lines in email headers, and not as a means to address all spam. In addition political issues between competing ISP SPF solutions have thrown other barriers in the way of implementation (TrimMail, 2005).

There are other proposals closely related, based on a payment scheme such as HashCash (HashCash.org, 2002), BondedSenders (Bondedsender.com, 2005), or email postage (GoodMail.com, 2003), which would make sending emails a costly billable service. The assumption is that by increasing the costs we can make spam too costly to send from bulk mailers (Goodman and Rounthwaite, 2004). Related to monetary charge, is a computational cost per email sent (Dwork et al., 2003). In both schemes the current protocols would have to be redesigned. Second, some of the solutions would make individuals responsible for spam sent from their compromised email address or compromised host machine on a monetary fashion. This is a problem since there is a breed of malicious programs which turn personal computers into zombies. These zombies are used for various nefarious purposes including sending spam, which would leave the owners responsible for the cost of sending spam under some of these schemes. In framing the spam problem in terms of dol-

lars and cents makes the assumption that for the right price, spam can be offered to the end user (Fahlman, 2002; Dai and Li, 2004).

2.3.5 Spam Filtering

The last approach tries to filter out spam email messages from the user's email box by identifying which messages are likely to be spam and which are not. There are three popular methods for filtering out spam: white lists, black lists, content based filtering, and various methods combining all three. Surveys can be found in (Diao et al., 2000; Mertz, 2002; Pazzani, 2000; Massey et al., 2003; Zhang et al., 2004; Allman, 2003; Sipior et al., 2004; Eide, 2003)

2.3.5.1 Lists

White lists are lists of allowable sources of emails which the destination node trusts as a legitimate source of messages. These are usually implemented at the client side, though some have proposed using white lists at the gateway to a single domain. The destination node constructs the list at some point and extends the list over time. Many systems, such as Hotmail, automatically prompt the user to add new (non white listed) email recipients to the list, as part of sending out each message.

This approach somewhat alleviates the traditional spam problem, as the source email address in the spam is not usually a member of the white list. In the real world, setting up a good white list is not trivial, and in fact has a negative impact on the convenience of using email. It is also easy for a spammer to circumvent this filter by spoofing white listed email addresses in general and specifically between users in a single organization.

Black lists are lists of those sources (usually IP address ranges or domains) from whom the user does not wish to receive email. These lists are supposed to represent known spam sources. For example, open relays are those mail servers that allow third-parties to send mail to any other third-party to essentially proxy the identity of the originating site. Because of this anonymity, they are assumed to be a frequent source of spam emails. These black lists are compiled and distributed widely; if the source email originates from an ip address in the black list, it will be discarded. Example of such lists are spamhaus (Spamhaus, 2005), dnsrbl (DNSRBL, 2005), and spews (Spews, 2005). Although historically most spam have been traced to open relays, zombies now account for more than 60% of seen spam according to many estimates (Leyden, 2003; Spring, 2005).

Systems such as Hotmail, allow the user to specify a list of users to block, and the list is limited to for example 256 email addresses. Because spammers typically forge their source emails, this list quickly fills up and becomes ineffective at stopping spam messages.

In real world deployment the black list technique has been getting negative feedback as many people have found their email servers added to a black list for no apparent reason (typically as a form of denial of service as a consequence of address spoofing). Especially on well-distributed black lists, the burden of proof is shifted to the source user, basically making it almost impossible to prove their innocence. For example, recently AHBL (AHBL, 2004) a popular black list source, decided to put all Spanish traffic on their blacklist (Knight, 2005). In addition to arbitrary inclusion, these lists require frequent updates to keep the blacklists up to date and thus suffer from lag time vulnerabilities. This implies that there is a period of time

when a new computer starts to send out spam, until the black lists users will detect and stop responding to that computer. In addition long term observations seem to imply that black lists are not careful about maintaining the lists over time to allow legitimate users to use ip once used by spammers (Jung and Sit, 2004).

2.3.5.2 Filtering

We focus attention on content-based filtering, which has garnered attention in the popular media for being the next "all-inclusive" spam solution (Graham, 2002). Because most of the recent research on improving email clients has been the focus of better filters, we address them here in depth.

One of the most popular techniques has been to filter spam emails based on the textual content part of the email. There are two general approaches. The first uses hard-coded rules that are periodically updated for and by the user (Crawford et al., 2001; SpamAssassin, 2003; Ahmed and Mithun, 2004). Each email is given a certain amount of 'spam' points based on some rule set. If a specific email exceeds some arbitrary threshold score the system typically quarantines it for later review and deletion by the user.

An approach called collaborative spam detection (Gray and Haahr, 2004; Kleinberg and Sandler, 2004) allows groups of users share information about spam content. For any message received, a local client or server checks a signature of the message against a global server list to see how frequent the message has been reported as spam. If the message has been reported as spam above a specific threshold, the user considers it spam. One way which these lists are seeded is by creating millions of fake email accounts visible only to spam programs which harvest email

addresses. Any message received by these addresses are almost always guaranteed to be spam. A shortcoming of this system is the susceptibility to mimicry attack when a legitimate message is sent to many of these fake email addresses. DCC (Rhyolite Software, 2001) and Razor (Prakash, 2005) are two examples of this approach.

There are dozens of commercial and open source spam filtering solutions for either the client side, server side, or both using any combination of the above mentioned approaches. Commercial solutions include most commercial email clients, Symantec Brightmail, Postini, CipherTrust, and Barracuda among others (Metz, 2003). Open source solutions include Bogofilter, CRM114 (Yerazunis, 2003), DSpam, Spamassassin (SpamAssassin, 2003), and Spambayes, among others (Asaravala, 2004). Some commercial solutions are designed to receive the user's email first and then forward the email to the user as either labeled email (which filtering rules can be applied) or only spam free email. Examples are Spamcop (Corporate Email Systems, 2002), Cloudmark, and BrightMail. The inherent danger to both content privacy and false positives messages need to be considered with external filtering solutions.

2.3.5.3 Machine Learning Models

The second approach uses machine-learning models, leveraging work done on text classification and natural language processing applied directly to spam. A training set of emails is created with both normal and spam emails, and a machine learning technique is chosen to classify the emails. For performance reasons, the usual method is not to have an online classifier, but rather a preset offline classifier,

which automatically classifies the emails for the user behind the scenes based on a static model.

Content-based filtering has been shown to be very accurate with some experiments claiming accuracy as high as 98.8% for certain data sets (Graham, 2002; Yerazunis, 2004; Siefkes et al., 2004). These numbers do not reflect two very important factors of the real world driven by the economics enjoyed by spam senders. The first is that the nature of spam content is dynamic and changes over time; it is constantly being updated to reflect changing social norms. The second is that there is a financial incentive for spammers to circumvent the filters so that their message can be delivered to end users. Filtering models based on content alone do not reflect the reality that spammers are clever enough and have a financial motivation to adopt their messaging to avoid having their messages easily filtered. We note, that independent tests do not always achieve such high accuracy numbers in part to these factors (Snyder, 2004).

2.3.5.4 Content Based Features

Most filtering systems using machine learned models base their feature sets on the actual contents of the message, either as a fixed length word vector or dividing the text into tokens, and scoring the message by some model applied to these vectors or tokens. Although usually not addressed by the literature, the practice is to sometimes ignore binary attachments or header data when analyzing messages. It has been shown that header information is as important in analyzing emails as the content (Zhang et al., 2004).

One of the earliest filters based on machine learning models was applied to

email flames (Spertus, 1997) and spam (Sahami et al., 1998). Sahami et al proposed using a Naïve Bayes model to filter spam. For their model, they used the 500 highest frequency tokens as a binary feature vector, 35 hand-crafted rule phrases, and 20 hand-crafted non-textual features such as the sender's domain. They make some strong assumptions, which most of the early literature assumes to be a true. The first is that spam can be detected based on textual content alone , just like any other information retrieval task. A second assumption is that they could make broad generalities on accuracy and detection rates based on their local corpus of only 2500 emails.

Later work introduced the bag of words model (Androutsopoulos et al., 2000b; Androutsopoulos et al., 2000a), i.e., treating each word as a token, basing token probabilities on those found within the email body along with adding a cost associated with either deleting or marking emails as spam. Comparison between words and tokens has shown promise in estimating priors (Peng and Schuurmans, 2003). Applying a mix of machine learning techniques including Naïve Bayes (Pantel and Lin, 1998; Provost, 1999; Sahami et al., 1998; Androutsopoulos et al., 2000c; Iwanaga et al., 2004) cost evaluation (Androutsopoulos et al., 2000b; Androutsopoulos et al., 2000a; Kolcz and Alspector, 2001), boosting trees (Carreras and Márquez, 2001), TF-IDF (Segal and Kephart, 2000; Cohen, 1996), support vectors (Kolcz and Alspector, 2001; Rios and Zha, 2004), gene analysis (Rigoutsos and Huynh, 2004), hash values (Yoshida et al., 2004), training combinations (Sakkis et al., 2001), statistical correlation (Gee, 2003), and complex systems (Segal et al., 2004) have all been tried. Even one-class detection, that is training on only spam examples has been attempted (Hershkop, 2004; Schneider, 2004).

Accuracy and recall results from the different experiments in the literature show promise (in some cases), but these filters are susceptible to attacks because of one simple reason. They are all mostly based on the contents of the message. A spammer can with little work create millions of combinations of messages without using any of the tokens seen in past messages, as shown in (Hall, 1999). In fact this is very close to what is known as mimicry attack in the security domain (Wagner and Soto, 2002). Although it is hard to judge if spammers have been taking advantage of this, one must make the assumption that eventually someone will think of trying it.

2.3.5.5 Non-content Features

An alternative to modeling the contents of each message, has been to study the behavior of the messages. Looking at group communication, attachment flows, and usage trends can all be used as features to help classify email.

There has been some reported work on finding groups of email users as discussed in our work and others (Stolfo et al., 2003b; Tyler et al., 2003). ContactMap (Nardi et al., 2002) is an application which organizes a visual representation of groups of users, but does not consider the case of emails that may violate group behavior as implemented in EMT (Stolfo et al., 2003b). Finding connections between messages based on content topics through threading has been extensively discussed in (Vel et al., 2001; Vel et al., 2002; Murakoshi et al., 2000; Lewis and Knowles, 1997; Lane and Brodley, 1998; Cohen et al., 1996). These are all published academic works, which have not made their way into common email client programs as far as we are aware.

Our work differs than what is implemented in PEA (Winiwarter, 1999) in the fact that their features are only based on the content of the messages. In addition they use an expensive "evolutionary algorithm" to model the problem. They also rate documents based on past usage to some extent, but concentrate on how to place messages into folders rather than utilizing it for other purposes. We note the fixation of 'folder only views', which we will address later in section 5.4.1.

In short most of the current research in the domain of email has been concentrated on weeding out unwanted messages and not on the bigger picture of evolving the role which email has come to play in day to day use.

2.4 Model Combinations

A considerable amount of literature exists concerning various methods for combining multiple classifiers. Combining and correlating models has been used in speech recognition, statistical pattern recognition, fraud detection, document classification, handwriting analysis and other fields. Various approaches combine models using different feature sets, other works correlate model outputs. An overview of the topic appears in (Clemen, 1989; Kittler and Alkoot, 2003; Kittler et al., 1998). Numerous studies have shown that combining classifiers yields better results than achievable with an individual classifier (Dietterich, 2000; Larkey and Croft, 1996). Some propose combining very strong classifiers (i.e., with low error rates) (Provost and Fawcett, 2000; Zheng et al., 2004) assuming that weak classifiers (high false positives) will not combine as well, or will require too many rounds of training to achieve low error rates. Measuring the "competence" of each classifier before combining them is a common approach as in (Asker and Maclin, 1997).

In (Sakkis et al., 2001) a combination of spam classifiers is proposed. That work was limited to training a group (referred to as a committee) of classifiers on a subset of labeled data, and then training a 'president' classifier using the labeled data plus the outputs from the sub-classifiers.

The combination methods described in this thesis in chapter 4 combine the output of individual classifiers, each of which outputs a confidence score associated with the output class label. The individual classifiers are computed by distinct machine learning algorithms, some of which are trained on independent features extracted from email. We detail the collection of supervised machine learning algorithms built into EMT in the later sections.

Some of the earlier literature assume a combination of classifiers with binary output (good/bad) and (Dietterich, 2000) points out that only when the classifiers have uncorrelated errors can we improve their overall assessment. We show later why the combination of confidence factors is able to achieve better results than a combination of binary classifiers.

Related work on enhancing email clients by adding meta-data to email systems exists in very limited fashion (Itskevitch, ; Macskassy et al., 1999; Manaco et al., 2002; S.Vidyaraman et al., 2002; Segal and Kephart, 1999; Winiwarter, 1999). These proposed systems, some with advanced functionality, have in some very limited extent affected the standard clients but not in any organized fashion relating to email utilization. Sometimes the underlying theme in the literature was a belief that the only useful work the user would need is a suggestion as to which folder a message should be classified (Segal and Kephart, 1999; Mock, 1999). To our knowledge no formal work has been done on user behavior modeling as an or-

ganizing principle for automatically suggesting actions the user would take for a particular message (such as deleting it).

The work presented in this thesis on spam filtering is unique in the fact that it tries to filter out spam using behavior as a model. Each email received over time, by a particular user, forms a larger picture of the individual's email account behavior. The behavior models in this Thesis consist of non-content features which help distinguish spam email from normal email. The features help identify spam without having to parse or token-ize or otherwise interpret the contents of the body of the message. Thus, we use statistical features that profile the user's behavior to provide evidence that a received message is indeed one the user would ordinarily filter. This concept of email behavior profiling using machine learning techniques (for security tasks) was first introduced in Columbia's Malicious Email Tracking (MET) and Email Mining Toolkit (EMT) systems (Bhattacharyya et al., 2002; Stolfo et al., 2003b; Stolfo et al., 2003a). In the upcoming chapters we will detail the theory behind the email models, and describe the implementation of our system.

Chapter 3

Email Models

To effectively model the information in an email collection, we need to represent it in a form amenable to analysis. In this chapter, we introduce the models used to represent email messages, email communication, and email flows. These models consist of traditional information retrieval and text classification models, and new behavior based models introduced in our work on EMT.

3.1 Classification Models

We first briefly overview the theory behind machine learning modeling. We then step through each of the specific classification and behavior models presented in the thesis. Before continuing, we will define some common terms used in the text.

Features - or attributes are the alphabet of language we are mathematically modeling. A set of attributes describe an instance, that we would like to label. For example when modeling the body of an email, a typical feature would be a word in the body of the message. These individual features are sometimes

preprocessed and converted to a specific type value which is processed by a machine learning algorithm.

Target Function - or class label is the pattern we are trying to learn. For example, in the spam detection task, given an unknown email, we would like to predict with some degree of confidence whether it is spam or not. In this case, "is it spam?" is the target function.

False Positive Rate - is the percentage of examples which our model has misidentified as the target concept. Generally our goal is to minimize this measurement while not increasing the error rate. Generally the cost associated with false positives are higher than false negatives. The false positive rate is computed as:

$$\text{FP rate} = \frac{\text{\# misidentified as target examples}}{\text{total \#non-target examples}} \tag{3.1}$$

False Negative Rate - is the proportion of target instances that were erroneously reported as non-target. When tuning the detection algorithm we must find a balance between false negatives and false positives. A threshold is used over all examples the higher this threshold, the more false negatives and the fewer false positives. The false negative rate is computed as:

$$\text{FN rate} = \frac{\text{\# misidentified as non-target}}{\text{total \# of target examples}} \tag{3.2}$$

Sample Error Rate - is the percentage of examples of the training that the model has misclassified divided by the total number of examples seen. This is one measure to estimate how well the classifier has learned the target function.

True Error rate - is the probability that the model will misclassify an example given a specific training sample and sample error rate. This measurement is hard to accurately measure, but can be approximated if the training set closely resembles the true distribution of future examples. In other words, if we train on half spam and half non-spam examples, but in reality 90% of examples will be spam, the sample error will not be an accurate measurement of the model's error rate.

Bias - is the difference between what we expect the model behave and its actual performance. Classification bias is the tendency of a machine-learned model to bias its output towards any one output value as measured during the concept training.

Training - is the process of teaching a model some target concept. During training specific examples are shown to the model and are used to tune the model's parameters.

Testing - is the process of evaluating the model classification effectiveness. If we have a labeled set of examples which are not shown to the classifier during training, and making the assumption that the testing set represents an accurate statistical sample of examples, we can measure the accuracy of a classifier to generalize the training examples.

Noise - is corrupt labeled data, that is data which for one reason or another is mislabeled. Certain algorithms are robust, i.e. unaffected by noise in the training data, while other require clean data to be able to measure ground truth. Real world data is often noisy; typically its is hard and expensive to

acquire clean data.

3.1.1 Machine Learning Theory

Supervised learning is the task of classifying unknown data using models trained over labeled data. In the email domain we have a set of labeled email and we aim to learn a classification function which will help us distinguish new unknown emails into an accurate class function. In the spam detection task, much of the research literature has cast the problem as a binary classification problem, i.e. we are trying to determine if a new email is spam or non-spam. In general there is no reason to limit it to a two class problem, but we illustrate it with binary classification to make the analysis somewhat simpler.

In general machine learning, most algorithms require clean data. Acquiring a large set of clean and accurate data is a non-trivial task in most domains. In addition, we would like the training samples to reflect the general population for which the model will be operating in, something which is not always easy to achieve.

Unsupervised learning is the other side of the coin. In this context, we seek to extract patterns from unlabeled training data and then assign new data into some set of categories. This can also be viewed as a clustering task, i.e. first grouping the training data into a number of sets (i.e clusters) and then fitting new data into similar data clusters. The exact definition of similar is algorithm dependent.

In both supervised and unsupervised learning, our goal is deduce from the training data the best pattern to describe the data in general terms. The "best" would mean the most probable pattern fitting the past observation in the training and at the same time be accurate for any future date we encounter. We define $P(C)$

to be the probability of class C before observing any data. This is also referred to as a priori probability, and we can leverage outside knowledge about a problem domain in calculating it. In general we write $P(a|b)$, which denotes the conditional probability that a occurs given b. We are also interested in $P(C|data)$, which is the probability of class C, given we have seen a specific set of data. This is known as the posterior probability because it reflects a confidence measure in the class, after seeing a specific set of data points.

A very hard question to answer is how accurately can we learn a pattern from a set of data and how well does it generalize. Fortunately for machine learning this is a well studied statistical problem of estimating a proportion of a population which have some pattern given a random sample of the population. By collecting a random sample and seeing how the model performs over the sample, we can estimate the model misclassification rate.

Measuring the performance of the model over training data is equivalent to running an experiment with a random outcome. As more random samples are measured, we can measure the amount of errors for the specific model. The error rates over the training sample is the sample error, which we are using to generalize to the true error rate, or the model's performance over future data. The model bias is the average difference between what its actual behavior is and what we had estimated it to be. It has been shown that a binomial distribution can characterize the probability of observing r errors in a data sample containing n randomly drawn samples. $\frac{r}{n}$ is our sample error rate observed during the training period. The confidence that the sample error rate closely mirrors the actual error rate can be calculated using a confidence interval. For the binomial distributions this can be

tedious, but easily found if we use a normal distribution to approximate it. As n grows larger, the binomial distribution is closely approximated by the normal distribution, which has well known statistical properties for its mean and variance.

3.1.2 Modeling Emails

Data mining in the email domain has been applied in the past to the problem of automatically classifying email and determining its proper "folder" (Cohen, 1996; Segal and Kephart, 1999). Recently has data mining been applied to studying task extraction and social network analysis (Whittaker et al., 2005; Rohall et al., 2003; Tyler et al., 2003; Johnson, 2003; Culotta et al., 2004).

Beginning with learning rules to classify emails, and then evolving into studying various machine learning algorithms for filtering spam, they both shared a common misconception. The notion that based only on the body of the email, we cam extract enough features is a strong underlying assumption. The reason this developed was for the most part to minimize the complexity of the problem and also allow general conclusions to be determined from localized email sets. In addition the problem was viewed as a simple IR task, thus ignoring a significant portion of features in the data.

It has been shown (Bhattacharyya et al., 2002; Hershkop and Stolfo, 2005b) that email contains a rich set of features, which when used in a data mining framework can provide additional models to be used for both traditional email classification and newer anomaly detection tasks. These features can be harvested from emails allowing a rich set of behaviors to be extracted. EMT has leveraged this and implemented a rich collection of models to extract and learn patterns from the

underlying data in a semi autonomous fashion.

We now present a brief overview of several commonly used machine learning algorithms. Specifically we present Naïve Bayes, N-Gram, text classifier, TF-IDF, URL, and a "Limited N-Gram". We then present the behavioral models including usage, communication, attachment, and cliques. We will provide a detailed explanation for each of the models presented.

3.1.3 Naïve Bayes

One of the most applied machine learning algorithms for the task of spam detection has been the Bayesian classification algorithm to modeling the content of email.

Bayes classifiers are based on early works by (Duda and Hart, 1973) in the field of pattern recognition. Given an unlabeled example, the classifier will calculate the most likely classification with some degree of probability. Bayes theorem is a way of calculating the posterior probability based on prior probability knowledge. Bayes theorem states:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \tag{3.3}$$

Notice that the right side of the equation can be calculated based on estimating the probabilities from the training set, making the strong assumption that the training set represents a true sample of the domain. For a set of features $f_1, f_2, \ldots, f_n$ which describe a problem instance, and a target label (L), Bayes theorem becomes:

$$P(L|f_1, f_2, \ldots, f_n) = \frac{P(f_1, f_2, \ldots, f_n|L)P(L)}{P(f_1, f_2, \ldots, f_n)} \tag{3.4}$$

The $P(L)$ can be easily estimated from the training data by simply counting the frequency of the particular target label L. Getting an accurate count for $P(f_1, f_2, \ldots, f_n|L)$ is not possible unless we gather a huge amount of data. This is because we would be required to observe every instance in the instance space many times to get a reliable estimate.

This can be addressed by making a simplifying assumption; namely that the features are conditionally independent. The classifier is known as a Naïve Bayes classifier. It is called naive because it makes a naive assumption that the tokens a statistically independent. In other words, the probability of observing the combination of $f_1, f_2, \ldots, f_n$ is simply the product of the probabilities. Although this is an oversimplification, it greatly reduces the computational costs of estimating the conditional probabilities and in practice as been found to work as well as neural networks and decision trees (Mitchel, 1997; Schneider, 2003). The naïve Bayes estimate is:

$$\arg\max_{c_i \subseteq C} P(C_i) \prod \frac{f_c + mp}{n_c + m} \tag{3.5}$$

Where C_i is the specific target class, f_c is the count of the particular feature per class, n_c is count of total unique feature tokens in the target class, m is a constant called the equivalent sample size, and p is a uniform distribution for discrete values (Mitchel, 1997).

Notice that the algorithm is based on analyzing specific feature tokens, and does not directly deal with numerical features. One simple way to deal with numerical features is to bin the values so that continuous values are mapped to discrete features. Another way of dealing with continuous values is to use a probability

distribution to map the continuous values to discrete probabilities. A standard way to deal with continuous values is to model them using a normal distribution (also called a Gaussian distribution). This is a bell-shaped distribution with the probability density function being:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{3.6}$$

A more accurate estimation as been shown using multiple Gaussian distributions with kernel estimates to map to probability values (John and Langley, 1995).

The kernel estimate $p(x)$ is simply:

$$p(x) = \frac{1}{n} \sum g(\left(\frac{x - x_i}{\sigma}\right), 0, 1) \tag{3.7}$$

The formula takes all the observed numerical features seen in the training and calculates a multiple Gaussian. n is the number of training examples, x_i is the i^{th} seen variable. We set μ to zero and σ to one to get a standard PDF which has some nice algorithmic properties as in (John and Langley, 1995). Detailed information can be found in (Hershkop and Stolfo, 2005b). A good survey of baysian machine learning can be found in (Lewis, 1998).

3.1.4 N-Gram

When analyzing text, one alternative to using words as tokens is to take subsequences of the data and use these subsequences as tokens. The advantage is that we do not need to define what the notion of a word is for us to analyze the text. This is ideal for example where some foreign languages which use characters instead of words.

An N-gram represents the sequence of N adjacent characters or tokens that appear in a document. We pass an N-character (or N-word) wide window through the entire email body, one character (or word) at a time, and count the number of occurrences of each distinct N-gram. For example for a 5-gram, the sequence "Hello world" would be mapped to tokens: "Hello", "ello ", "llo w", "lo wo", etc.

For email modeling, the algorithm works as follows. We count the number of occurrences of each n-gram for each email; this may be viewed as a document vector. Given a set of training emails, we use the arithmetic average of the document vectors as the centroid for that set. For an unknown test email, we compute the cosine distance (Damashek, 1995) against the centroid created for the training set. If the cosine distance is 1, then the two documents are deemed identical. The smaller the value of the cosine distance, the more different the two documents are. Cosine distance is defined as:

$$D(x,y) = \frac{\sum_{j=1}^{J} x_j y_j}{(\sum_{j=1}^{J} x_j^2 \sum_{k=1}^{J} y_k^2)^{\frac{1}{2}}} = \cos\theta_{xy} \tag{3.8}$$

Here J is the total number of possible N-grams appearing in the training set and the test email. x is the document vector for a test email, and y is the centroid computed from the training set. x_j represents the frequency of the j^{th} n-gram (the N-grams can be sorted uniquely) occurring in the test email. Similarly y_k represents the frequency of the k^{th} N-gram of the centroid.

3.1.5 Limited N-Gram

Tracking attachment behavior by studying its spread over time is one way to quantify anomalous behavior (section 3.3). When studying large amounts of email data, we would want to associate all messages or attachments that are similar to each other in some way, in order to track edited documents over time or evolving malicious attachments (e.g. polymorphic worm payloads). We can use an n-gram hash of parts of the message to associate similar messages with each other. We adopt an idea presented in (Manber, 1994) for approximating fingerprints. Instead of using all n-grams generated by sliding a size n window across the content, we only use those grams which have their last k bits set to zero. This translates into a large reduction in the number of n-grams we need to consider, which considerably speeds up the running time of the similarity test, at the expense of some accuracy.

3.1.6 Text Based Naïve Bayes

This algorithm uses a Naïve Bayes classifier based on simple word or token frequency as described in (Mitchel, 1997). We calculate the probability of each token as seen during training using a Naïve Bayes formula and assign a confidence score of the predicted class. The predicted class is the maximum, and the score is a normalized measure of confidence taken by dividing the maximum value by both values.

$$\text{class,score} = \max \begin{cases} P(\text{SPAM}) \prod P(word_i|\text{SPAM}) \\ P(\text{NotSPAM}) \prod P(word_i|\text{NotSPAM}) \end{cases} \tag{3.9}$$

where the $P(word_i|\text{NotSPAM})$ is the naïve Bayes estimate as explained in section 3.1.3.

3.1.7 TF-IDF

The TF-IDF algorithm (Salton and McGill, 1986) used by (Segal and Kephart, 1999) is based on using a combination of the term frequency (TF) multiplied by the inverse of the document frequency (IDF). Words that appear more frequent should have a greater impact on prediction than words which appear infrequently. We consider each email message ($\mathcal{M}$), which contain token words (w). Each group of emails denoted by $\mathcal{F}$ a folder, is represented by a weighted word frequency vector $W(\mathcal{F}, w)$. To calculate the weights we must first calculate a frequency centroid F over groups of messages (folders):

$$F(\mathcal{F}, w) = \sum_{\mathcal{M} \in \mathcal{F}} F(\mathcal{M}, w) \tag{3.10}$$

we now convert the folder centroid in the following manner. We calculate the fractional frequency FF:

$$FF(\mathcal{F}, w) = \frac{F(\mathcal{F}, w)}{\sum w' \in \mathcal{F} F(\mathcal{F}, w')} \tag{3.11}$$

now the term frequency is simply:

$$TF(\mathcal{F}, w) = \frac{FF(\mathcal{F}, w)}{FF(\mathcal{A}, w)} \tag{3.12}$$

where $\mathcal{A}$ is the set of all messages we are comparing. The document frequency is the fraction of messages which have the word w appear at least once. So the inverse document frequency used here is:

$$IDF(w) = \frac{1}{DF(w)^2} \tag{3.13}$$

and we combine the two with:

$$W(\mathcal{F}, w) = TF(\mathcal{F}, w) \times IDF(w) \tag{3.14}$$

For unknown messages, we compute the centroid of the new message and take a variation of the cosine distances between it and any centroid computed from the training data to find its classification.

3.1.8 Biased Text Tokens

Recent work by Graham (Graham, 2002) on the task of spam detection has floated the idea of a partial Naïve Bayes approach, biased towards low false positive rates.

The algorithm works as follows: We start with two training sets, spam and good (non-spam). For each word token in each collection, we count the number of times each word is seen, and store the spam counts in SC and good counts in GC. For each token w_i define:

$$x = \min(1, \frac{SC(w_i)}{\text{Number of Spam Tokens}}) \tag{3.15}$$

$$y = \min(1, \frac{2 * GC(w_i)}{\text{Number of Good Tokens}}) \tag{3.16}$$

We can now calculate the weight of each token w:

$$W(w) = \max(.01, \min(.99, \frac{x}{x+y})) \tag{3.17}$$

The .01 and .99 allow the scores to be as close but not touching 0 or 1 when we do not have a clear score. The probability that a message $\mathcal{M}$ is spam is

calculated as follows: for each $w_i \in \mathcal{M}$ calculate $W(w_i)$ and sort them by distance from 0.5 and combining fifteen of the most outlying scores in the following manner:

$$\frac{W(t_1)W(t_2)\dots W(t_{15})}{(W(t_1)W(t_2)\dots W(t_{15})) + (1 - W(t_1))(1 - W(t_2))\dots(1 - W(t_{15}))} \tag{3.18}$$

A threshold is used to determine if the new Message $\mathcal{M}$ is spam. We have used the suggested 0.9 as a threshold.

3.2 Behavioral-based Models

To model email behavior we use specific features for specific tasks. In some models we use a histogram to model the behavior of the user. A histogram represents the distribution of items in a given population of samples. A more detailed description is provided in section 3.4

We describe the algorithms used to compute email behavior models in the following sections.

3.2.1 Sending Usage Model

The usage model computes the stationary behavior of usage of an individual email account from an outgoing email point of view. The goal of this model is to quantify what it means for an email user to use their account in a typical way so anomalies can be detected.

When profiling email accounts based on an hourly division, every histogram has 24 bins that represent the 24 hours of a day. Emails are allocated to different bins, according to the time when they were sent. The value of each bin represents

the daily average value of a given feature of the emails sent out in a specific period of time. There are several features for profiling accounts including the number of attachments, the size of emails, the number of recipients. This can be extended and varied for the particular task.

An *Aligned Histogram* can also be used when comparing two different histograms. The two histograms are compared by anchoring the comparison to time bin 0. Anchoring the histograms allow us to compare real start times of an email user's 24-hour email usage period, as illustrated in the following example:

It may be the case that User A's histogram shows no activity until hour i, and User B (which might be A's second account) may shows no activity until hour j. Perhaps this is because User A is using a system in California and another system in New York City at the same time, but emails sent at the same time on different systems but analyzed together each may be associated with a different sent-time because of the difference in time zones and computers. We allow disambiguation by enabling the algorithm to anchor the histogram comparison to a single time bin.

Our approach to align histogram is as follows: we find the first non-zero bin of a day starting from hour 0 (12 a.m.). If the bin for hour 0 has value 0, which may occur if the user sends email late at night, we find the first non-zero bin which we use as the starting bin. Zero-period intuitively indicates a period of time when the user is inactive. From a histogram perspective, we treat four or more consecutive bins with zero value as a zero-period. After ascertaining the new start time, we wrap-around the rest of the histogram and use it for comparison. This provides another means of detecting "similarity" among email accounts different than raw histograms.

Histogram comparison functions described in section 3.4 allow the algorithm to choose a specific distance function. The comparison functions can be used to compare behavior of an account's recent behavior to the long term profile of that account. As explained above, the histogram comparison functions also may be run "unanchored", meaning, the histograms are shifted to find the best alignment with minimum distance, thus accounting for time zone changes.

Figure A.2 displays an example for one particular user account.

3.2.2 Similar User Model

In many cases, it would be useful to group email accounts into sets of similar behaving users, for example for classification or finding alias accounts. It has been found that some people maintain specific email accounts for either personal or business purposes. In many cases, they will use both accounts at the same time, and we would like to be able to find those two similar accounts.

The algorithm is designed to find a group of accounts that have a similar usage behavior. We compute the histogram for each account and then find similar users through comparison measurements. Different comparison methods are used, some based on the distance between a pair of histograms, and others based on statistical tests which will be explained in section 3.4.

When looking for similar accounts, a specific account is chosen as a pivot, and we compute its own distance to all other accounts. Alternatively, we can cluster accounts in a K-Nearest Neighbor fashion and return sets of similar accounts.

3.2.3 User Clique Model

Another method to model outgoing behavior is to model the collection of recipients in a single email as a set, and summarize these sets and their dynamics over time. This information is used to detect abnormal emails that violate the user's clique behavior.

Formally, email communication can be captured by a directed graph $G(V, E)$ with the set of nodes, V, being individual email accounts. A directed edge, e_{12} exists if v_1 sends an email to v_2. Viewed in this way, cliques are a certain pattern in this graph that we are trying to characterize and use as norms of communication behavior.

The user clique model is best described in terms of item sets (see Figure 3.1). An item set is a set of items associated with a transaction, such as a single purchase at a supermarket. The goal of analyzing item sets is to extract useful association rules of how items appear together (Holt and Chung, 2001). This problem has been studied in the data mining and database community and is of great commercial interest for its wide range of applications and potential predictive value that can be derived.

In the context of mining email, an email can be viewed as a transaction that involves multiple accounts, including a sender (in the FROM field) and recipient(s) in the (TO, CC and, BCC fields). If we discover the rules governing the co-appearance of these addresses, we could then use these rules to detect emails that violate these patterns. Suspicious emails may then be examined further by other models to confirm or deny that they are malicious.

The recipient list of a single email can be viewed as a clique associated with

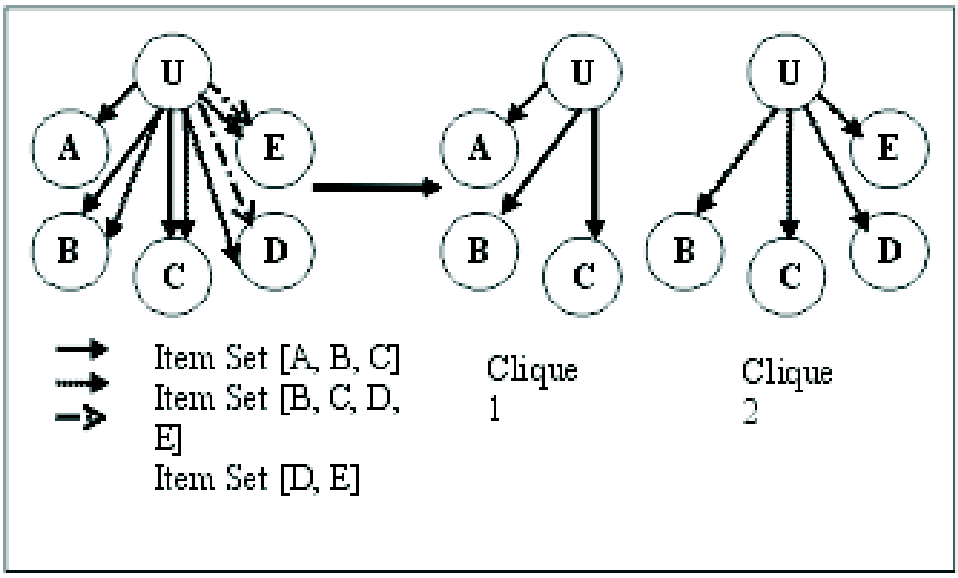

Figure 3.1: Three item sets from account U: [A, B, C], [B, C, D, E] and [D, E]. The first two sets share two nodes and the last set is subsumed by the second set. The resulting user cliques are [A, B, C] and [B, C, D, E].

the FROM account. However, using this set (or item set) directly is problematic for two reasons. First, a single user account would contain a large number of such sets and enumerating them for real-time reporting or detection tasks would be undesirable if are trying to reduce the number of emails which will have to be examined. Second, some of these sets are duplicates or subsets of one another and it would be difficult to use them directly for any purpose. For these reasons, we define a user clique as a set of recipients that cannot be subsumed by another set. Thus, we compute the most frequent email item sets that are not subsumed by another larger item set. Naturally, a single user will have a relatively small number of user cliques. As an example, suppose a user has in his/her sent-folder four emails with the following recipient lists: [A, B, C], [A, B, C], [A, B], and [A, B, D]. The user cliques belonging to this user would be [A, B, C] and [A, B, D].

A period of past history is needed to baseline the cliques. This bootstrapping period can either be set to a fixed period or to just ignore clique behaviors until sufficient data can be gathered.

A *clique violation* is defined as a message sent to a group inconsistent with the user's past clique behavior. An email sent from a user is regarded as inconsistent with the user's cliques if its recipient list is not a subset of any user cliques belonging to that user.

The usefulness of this model depends not only on how quickly new groups of recipients form over time but also on how it is combined with other models. That is why this feature is combined with others as part of a profile (section 6.5). We would also like to know the frequency of new clique formation to put violations in context. Users with low frequency new cliques can be assured of a clique violation more than a user who frequently engages in new cliques. Other models would have to detect violation for such users.

We note that if a user ever sends a single broadcast email to everyone in their address book (recipient collection), there would be only one user clique remaining in the model for that user. This would render the model almost useless for virus detection task because no clique violation is possible as long as a user does not communicate with someone new. In practice, however, this scenario is highly unlikely to happen. It has been shown that most of the time a user will send a single email to less than 10% of the people in his address book (Stolfo et al., 2003c). For an account with a small address book, a single email could cover 20%, 30% or an even higher percentage of the address book. The probability of an email covering a given range of percentages of an address book decreases quickly as the percentage

range increases (Stolfo et al., 2003c).

3.2.4 VIP Communication Model

The VIP Communication model computes a behavior model describing which relationships are relatively more important than others to a specific user. These measurements are computed by comparing the average time it takes the user to respond to different correspondents, and which correspondent the user responds to fastest (i.e. the VIP). One may infer from this analysis the relative importance of individuals to a specific user based upon the user's response rate. Those to whom the user may respond to more quickly are intuitively more likely to be important to the user. This is one of many ways of measuring communication importance between users.

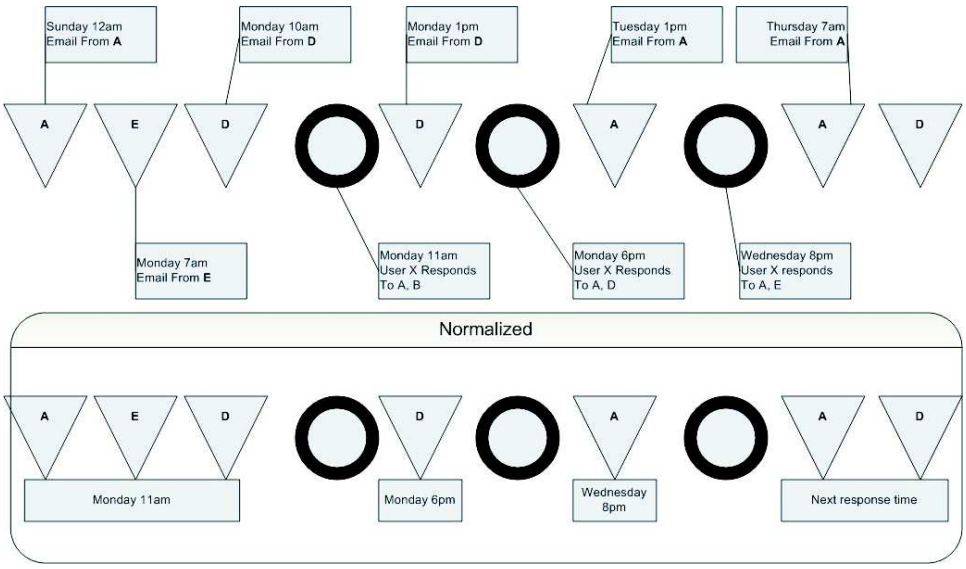

Figure 3.2: The VIP model is used to estimate communication lag time, by normalizing the time between communication bursts. The upper figure shows the actual communication flow, and the lower figure depicts the normalized flow.

We consider the average communication time of account A to account B as the average time elapsed between account A's response to an email sent by account B. However, in practice, it is difficult to compute this average time when we consider the typical interruptions of the workday, such as work-time/off-time, weekdays/weekends, meetings, vacations, etc.

Thus, we modify our definition of average communication time as a *relative average communication time*. This model naturally lends itself to a real time system. The algorithm works as follows; to calculate the VIP rank list for Account A, we batch all of A's incoming emails. As soon as A begins to respond, we reset the time stamp on all batched emails and move them to a wait queue. This is to compensate for off time by assuming all emails were received at the moment that the user starts to reply emails. We next examine the wait queue, and for any matching email recipients with whom A is now responding, we calculate the length of time (of the reset timestamp) to our current reply, and use this to average the communication time of account A to the specific account. This is graphically illustrated in Figure 3.2.

3.2.5 Organizational Level Clique Model

We have presented models for profiling individual accounts. Now we present a group behavior model. In order to study email flows between groups of users, EMT computes a set of cliques in an email archive over some period of time. We want to identify clusters or groups of related email accounts that participate with each other in common email communications, and then use this information to identify unusual email behavior that violates typical group behavior.

For example, intuitively it is unlikely that a user will send a distinct message to their spouse, their boss, their "drinking buddies" and their religious elders all appearing together as recipients of the same message (whether delivered in one email, or a series of emails) (Boyd et al.,). Of course this is possible, but it is rather unlikely. (In those unlikely situations the model would not be a reliable indicator for account behavior, discussed in chapter 4, in any case we would want to flag that type of behavior as interesting.) A virus attacking a user's address book at random would not know these social relationships and the typical communication pattern of the victim. Hence it would violate the users' group behavior profile if it propagated itself in violation of the user's social cliques. The same of a forged email: the sender would not know who else is typically cc'd in addition to the recipient user. Studying communication flows between groups allows us to build models of typical group interactions.

Clique violations may also indicate email security policy violations internal to a secured enclave. For example, members of the legal department of a company might be expected to exchange many Word attachments containing patent applications. It would be highly unusual, and probably unwise, if members of the marketing department and HR services would likewise receive these attachments. We can infer the composition of related groups by analyzing normal email flows to compute the naturally occurring cliques, and use the learned cliques to alert when emails violate that clique behavior.

Conceptually, two broad types of cliques can be extracted from user email archives: user cliques (presented in section 3.2.3) and enclave cliques. In simple terms, user cliques can be computed by analyzing the email history of only a single

user account, while enclave cliques are social groups that emerge as a result of analyzing traffic flows among a group of user accounts within an enclave.

For enclave cliques, we adopt the branch and bound algorithm described in (Bron and Kerbosch, 1973). Our hierarchical algorithm locates the largest cliques that are fully connected with a minimum number of emails per connection at least equal to a specified parameter.

We start by counting the number of emails exchanged between any two given users, regardless of the direction of the traffic flow. This absolute number is compared to a set threshold. If the count is above the threshold, then the link between the two underlying accounts is established. If the count is lower than the threshold, then we assume there is no link between the two users. At this point, we have a list of cliques of size 2. The 2 members in each clique are sorted lexicographically, and the clique counts (number of communications) are sorted in increasing order. We employ the hierarchical algorithm at this point and build lists of cliques of size n, with n increasing by 1 at a time. Throughout the algorithm, the list of cliques at level n is sorted, both among the sets of cliques, and within a set.

The hierarchical algorithm is a repetitive process of building cliques one level at a time. Given the current level, building the list on the next level is a two-step process; the first step is to generate a candidate set and the second step is to remove candidates that do not meet the clique definition.

The following example illustrates the building process. Assume that we are currently at level 2 and want to construct a level 3 candidate list. We have AB, AC, AD, BC, BD, CE on level 2.

First, we take each clique and combine it with each one of the cliques further down the list to form a candidate clique of size 3. This process has an outer loop and an inner loop. We can immediately see that two cliques can be combined only when they differ by one member. Furthermore, the member that is different has to be in the last position of the cliques. For example to avoid duplicates, AB and AC form a candidate clique, ABC. But AB and BC do not form ABC. The reason behind it is that suppose ABC is a legitimate clique, we would have had AB, AC, BC on level 2. Since the lists are sorted throughout, we would have had encountered AB and AC before AB and BC, thereby obviating the need of forming a candidate from them, which would result in a duplicate candidate anyway. Using a similar logic, we can see that we can terminate the inner loop of combining the current clique with another one further down the list as soon as we encounter a clique, which differs from the current clique by more than 1 position.

Once the candidate list for a given level is generated, we examine each candidate individually to see if the constituted cliques of one less in size are legitimate cliques. If so, the candidate is qualified. For example, to check if ABC is a qualified clique, we need to check if each member in AB, AC, BC is a clique on level 2.

At the end of the hierarchical process, we have cliques of all sizes. We then remove those that are subsets of another, leaving only maximal cliques at the end of the process. This process can be optimized by constructing hash tables at each level, with keys being the cliques. These hash tables also facilitate the process of checking candidate sets.

Overall, the speed of this algorithm is the same as that of the Bron & Kerbosch algorithm (Bron and Kerbosch, 1973).

3.2.6 URL Model

Another feature which can be extracted from email is a mathematical model of the typical URL link. The URL model allows us to profile a typical URL link found in a user's email. By modeling the typical URL link in a class of email we can differentiate between wanted and unwanted email links. The algorithm was developed to compute distances between groups of universal resource locators (URLs) found in sets of emails. For the spam detection domain, non-spam messages will typically contain embedded URLs that are likely to be similar to each other and different than those occurring in spam messages. We define a distance metric between 2 URLs (x,y) as follows:

$$D(URL_x, URL_y) = \text{scale} * \frac{(\sum_{i=0}^{y_{len}} URL_x[i] \neq URL_y[i]) + \frac{URL_x Length - URL_y Length}{2}}{URL_x Length} \tag{3.19}$$

where URL_x is the longer URL and $URL[i]$ is the i^{th} character of the URL. The distance is returned as a number between 0 and scale, with a smaller number between two URLs translating into a closer relationship between the two URLs. We also define 12 types of URLs: for example URLs can be found as image links or more common HTTP links (see table 3.2.6.) The type of URL is added to the final score as a base offset to differentiate between URL types.

For each email, we group the URLs into a single cluster. During training to recognize a specific class of labels the clusters are formed in the following manner. All URLs are extracted from a new training example to form a single cluster. We then test this cluster against all available clusters to see if the average difference

URL Types	Sample
WEB	www.????
ANCHOR	<a href ??? > ... < /a >
FORM	<form ??? < /form>
Email	mailto:???
FTP	ftp.???
JAVASCRIPT	<javascript ???
Area	<area ??? >
STYLESHEET	<stylesheet ???
RELATED LINK	<link ????>
IMAGE	<img ??? >
GOPHER	gopher://????
MISC	Anything else

Table 3.1: Classes of URLs, the string '???' in the table represents some URL pattern we are interested in matching.

is under some threshold. If it is, we just combine the two clusters. If it is not, we create a new cluster with this set. Combining sets of URLs, allows the data structure to keep counts of specific URLs, and a list of all URLs in a specific cluster.

During testing, we extract all the URLs from the test instance and treat them as a single cluster. We calculate the minimum distance from this cluster of URLs to any cluster from any target class observed during training. This is similar to a K-nearest neighbor algorithm (Fix and Hodges, 1952). The minimum cluster distance is then converted into a confidence score and outputted by the classifier.

The resulting score is a measure of how unusual or familiar a URL may be in an email message given the user's prior history of emails containing embedded URLs. We note that when an email contains no links, we cannot use this model to analyze it.

3.3 Attachment Models

Another aspect of modeling email behavior is to look at the behavior of the email attachments. The Malicious Email Filter (MEF) project is a malicious executable filter that can detect malicious email attachments (Schultz et al., 2001a). It first extracts byte sequence features from each labeled executable. These byte sequences (similar to N-Grams) are then used by the algorithm to generate detection models. This work was extended in the Malicious Email Tracking project to allow tracking the flow of email attachments to detect malicious behavior, without specifically knowing a priori that the attachments are malicious (Bhattacharyya et al., 2002). By keeping track of specific features of an unknown attachment, an email system can score the probability of a specific attachment being malicious (Schultz et al., 2001b).

For any attachment we can quantify the flow through a network and calculate global flows of the malicious attachments through the Internet, using a MET like framework adopted to work on an offline email collection. In general, while MET concentrated on malicious attachments, we also are interested in finding how specific attachments behave within the email collection. We compute the following metrics for each attachment of interest (others are possible):

Attachment Incident : the fraction of the total number of emails within the email collection related to a particular attachment, starting from a single point in time. Since each attachment is saved in the local repository with a Unique ID and malicious or benign classification, this value is simply the number of times each unique hash ID appears in the local repository.

Birth rate : the rate at which an attachment is copied from one account to anther. This value is calculated by determining the total number of email addresses an attachment is sent to per minute. If this value is set to a specific threshold, it can be used to determine whether or not an attachment is a self-replicating attachment. Obviously, any time quanta can be implemented, and is best determined by observing local email behavior. (We presume that a malicious payload will not have access to these statistics in order to make its spread behavior appear normal within the environment.)

Lifespan : the length of time an attachment is active. This value is calculated by subtracting the first time an attachment is seen from its last occurrence in the local repository. In malicious attachments this values reports the amount of time an attachment was free to cause damage to a network before it was detected.

Incident rate : the rate at which a specific attachment incidents occur in a given population per unit time, normalized to the number of emails in our data set.

Death rate : the rate at which an attachment is detected. This is calculated by taking the average lifespan of the attachment.

Prevalence : a measure of the total number of users which have been observed receiving or sending a particular attachment. This value is calculated by summing over the number users showing the same attachment.

Threat : the measure of how much of a possible danger an attachment may be. One straightforward way to measure threat is to calculate the incident rate

of an attachment added to the prevalence of the attachment divided by the total number of unique email users and the total number of attachments.

Spread : a measure of the global birth rate of an attachment. This is calculated by taking the average of the birth rates reported by the participating users or organizations.

These metrics are directly implemented by computing SQL aggregates over the databases (both local and organizational). Each time EMT determines that an attachment is malicious, it files a report in the reporting facility (see section 5.4.8) and Figure A.8.

3.4 Histogram Distance Metrics

As mentioned, EMT's rich set of models includes explicit statistical models of user behavior, some of which are represented by histograms. Histograms are compared to one another to find similar behavior or abnormal behavior between different accounts, and within the same account (between a long-term profile histogram, and a recent, short-term histogram). The histogram comparison functions also may be run "unanchored" or "aligned", meaning the histograms are shifted to find the best alignment with minimum distance, thus accounting for time zone changes.

Distance functions are used to measure histogram dissimilarity. For every pair of histograms h_1,h_2 there is a corresponding number $D(h_1, h_2)$ which indicates the distance between h_1 and h_2. The distance should satisfy the following requirement:

1. **Identity:** $D(h_1, h_1) = 0$

2. **Non-negativity:** $D(h_1, h_2) \geq 0$

3. **Symmetry:** $D(h_1, h_2) = D(h_2, h_1)$

4. **Triangle Inequality:** $D(h_1, h_2) \leq D(h_1, h_3) + D(h_3, h_2)$

Usually, the values in the histograms must be normalized before one can apply the distance functions. We utilized four distance functions: a simplified histogram intersection (L1-form), a histogram Euclidean distance (L2-form), a histogram quadratic distance, and a histogram Mahalanobis distance. In addition we present some statistical distance measurements.

3.4.0.1 L1-form (Simplified Histogram Intersection)

$$D_1(h_1, h_2) = \sum_{i=0}^{n-1} |h_1[i] - h_2[i]| \tag{3.20}$$

The histograms h_1, h_2, are compared to one another. n is the number of bins in the histogram. h_1, h_2 must first be normalized in order to make the F1-form satisfy the above distance function requirements. We normalize the sum of the histogram's bins to 1 before computing their distance.

3.4.0.2 L2-form (Euclidean Distance)

Another distance metric is the Euclidean distance function.

$$D_2(h_1, h_2) = \sum_{i=0}^{n-1} (h_1[i] - h_2[i])^2 \tag{3.21}$$

The L2-form is similar to the L1-form, but uses a second-degree function (Chakravarti et al., 1967). L1 and L2-forms are straightforward, but they have the following drawback. The individual components of the feature vectors, i.e., the

bins of the histograms, are assumed to be independent of each other, something not necessarily true in our environment, since behavior usually does not abruptly change per bin.

3.4.0.3 Quadratic

We now describe the Quadratic distance metric from (Chakravarti et al., 1967).

$$D_3(h_1, h_2) = (h_1[i] - h_2[i])^T \mathcal{A}(h_1[i] - h_2[i]) \tag{3.22}$$

This function considers the differences between different bins. $\mathcal{A}$ is a matrix, and a_{ij} denotes the similarity between bin i and j. We set $a_{ij} = a_{ji}$ (Symmetry), and $a_{ii} = 1$.

We set: $a_{ij} = |i - j| + 1$, in other words we expect the behavior of the email accounts to be more similar in adjacent hours. For example, the account's behavior between 9 a.m. and 10 a.m. should be more similar than that between 9 a.m. and 10 p.m. This is a very simple approximation but useful in smoothing out the profile. This simplification can be further refined to take in another side of the story, namely that the behavior between 11 a.m. and 2 p.m. (both work times) may be more similar than those between 1 p.m. (lunch time) and 2 p.m. (work time). One can also factor in work days and non work days (weekends) into the equation. We note these other details, but have chosen a simple smoothing by adjacent hours.

3.4.0.4 Mahalanobis Distance

The Mahalanobis distance (Chakravarti et al., 1967) is a special case of the quadratic distance equation mentioned in section 3.4.0.3. The formula is the same, but $\mathcal{A}$ is computed differently. Here the matrix A is given by the inverse of the covariance matrix obtained from a set of training histograms. We treat the elements in the histogram vectors as random variables, $H = [h_0, h_1, \ldots, h_{n-1}]$. Covariance matrix is B. $b_{ij} = Cov(h_i, h_j)$. $\mathcal{A} = B^{-1}$. When they are statistically independent but have unequal variance, matrix B is diagonal.

Let histogram h represent the user's normal behavior, which is determined from a training set, and let histogram h_1 be some specific period that we wish to test. In this distance metric, we make the assumption that the elements in the histogram vectors are random variables and statistically independent. Thus, we have the following formula:

$$D_4(h_1, h) = (h_1 - h)^T \times \mathcal{A}(h_1 - h) \tag{3.23}$$

$$\mathcal{A} = B^{-1}, b_{ii} = Cov(h[i], h[i]) = Var(h[i]) = \sigma_i^2 \tag{3.24}$$

Covariance Matrix

$$B = \begin{pmatrix} \sigma_0^2 & 0 & \cdots & 0 \\ 0 & \sigma_1^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{n-1}^2 \end{pmatrix} \tag{3.25}$$

Thus we get:

$$D_4(h_1, h) = \sum_{i=0}^{n-1} (\frac{(h_1[i] - h[i])^2}{\sigma_i^2}) \tag{3.26}$$

h_1 is the average number of a specific feature (emails or attachments etc.) sent out in hour i during the observed training period, and σ_i describes the variance around the arithmetic mean.

Starting with the basic Mahalanobis distance function, we modify it to a weighted one. We first reduce the equation from the second-degree function to the first-degree function, and then we give a weight to each bin so that the bins with larger counts will contribute more to the final result:

$$D_4(h_1, h) = \sum_{i=0}^{n-1} \frac{w_i(h_1[i] - h[i])}{\sigma_i} \tag{3.27}$$

$$\text{Weight } w_i = \frac{h_1[i]}{\sum_{j=0}^{n-1} h_1[j]} \tag{3.28}$$

3.4.0.5 Chi Square and KS

Methods from non-parametric statistics can be used to determine whether two arbitrarily shaped, empirical distributions (including histograms) represent the same distribution. We present two; chi-square (χ^2) and the Kolmogorov-Smirnov (KS) test.

The Chi Square test is designed, among other things, to compare two frequency tables. Its formula is:

$$Q = \sum_{i=1}^{k} \frac{\chi(i) - np(i)}{np(i)} \tag{3.29}$$

where $\chi(i)$ is the number of observations for each recipient (i) in the testing range, $p(i)$ is the true frequency calculated from the training range, n is the number

of observations in the testing range, and k is the number of recipients. There are $k-1$ degrees of freedom. The p-value of a Chi Square test is the probability that the frequencies from both training and testing windows originate from the same distribution. The number ranges from 1 to 0, where 1 indicates almost certainty that the distributions are the same and 0 indicates that they are different.

The KS test is designed to test the hypothesis that a given dataset could have been drawn from a given distribution (Chakravarti et al., 1967). It is based on the empirical distribution function. The result is equal to the maximum difference between the cumulative distribution function of data points and the normal distribution function.

$$D_{KS}(h_1, h_2) = \max_{x \subseteq X}(|F_{h_1}(x) - F_{h_2}(x)|) \tag{3.30}$$

$$F_h(x) = \frac{n(x)}{N} \tag{3.31}$$

Here we can treat h_1, h_2 as the normal distribution function. N is the total number of samples, and $n(x)$ signifies the number of points less than x. x is ordered from smallest to largest. The KS test is independent of arbitrary computational choice such as bin width. It does not depend on the underlying cumulative distribution function that is being tested. The KS test indicates how distant the two distributions are.

3.4.0.6 Hellinger Distance

In statistics, the Hellinger distance is used to compare differences in frequencies, in our case we can use this measurement to identify dynamics of user behavior over

time by comparing recipient frequency histograms. A recipient frequency histogram records the long term behavior of a user and the frequency with which they send emails to particular individual recipients. (Interestingly, the rank order frequency of recipients follows a Zipf-like distribution, i.e. a small population of users receive the bulk of emails from an individual user, while a large number of recipients receive very little email from that user.) We illustrate this graphically in figure A.5.

The Hellinger distance metric is defined as:

$$HD(f_1[], f_2[]) = \sum_{i=0}^{n-1} (\sqrt{f_1[i]} - \sqrt{f_2[i]})^2 \tag{3.32}$$

Where f_1 is the array of frequencies for the training set, f_2 represents the array of frequencies for the testing set. This distance metric can also be used to group similar frequencies as in (Lee and Shin, 1999).

3.4.1 Text Distances

For computing the distance between two textual features, we can leverage a variety of distance formulas which we present in the following sections.

3.4.1.1 K-Nearest Neighbor

When computing textual distance we can compute the closest distance to a set of precomputed clusters, or documents. The formula for weight adjusted nearest neighbor algorithm (Han et al., 2001):

$$\cos(X, Y, W) = \frac{\sum_{t \in T}((X_t \times W_t) \times (Y_t \times W_t))}{\sqrt{\sum_{t \in T} (X_t \times W_t)^2} \times \sqrt{\sum_{t \in T} (Y_t \times W_t)^2}} \tag{3.33}$$

given 2 documents X and Y, and a weight vector W. T is the set of tokens, X_t and Y_t are normalized token frequencies for token t in X and Y. W_t is the weight of token t. Details are given in (Han et al., 2001) on optimizations that make this model a useful tool for text categorization.

3.4.1.2 Centroid Cosine Distance

Given a set of training emails we extract the token frequencies and use the arithmetic average of the document vectors as the centroid for each set. For an unknown test message, we compute the cosine distance of this new example against each centroid created during training. If the cosine distance is 1, then the two documents are deemed identical. The smaller the value of the cosine distance, the more different the two documents are. The formula for the cosine distance is:

$$D(x,y) = \frac{\sum_{j=1}^{J} x_j y_j}{(\sum_{j=1}^{J} x_j^2 \sum_{k=1}^{K} y_k^2)^2} = \cos\theta_{xy} \tag{3.34}$$

J is the total number of possible tokens which appear in the training set and the test message. x is the document vector for the test email, and y is the centroid for the training set. x_j represents the frequency of the j^{th} token. Similarly y_k represents the frequency of the k^{th} token of the centroid.

Like the TF-IDF algorithm (section 3.1.7), the computed centroid is effectively a vector of the highest occurring tokens. Centroid cosine allows us to cluster a set of messages. To cluster we follow a simple algorithm:

1. Order all messages by some feature (size, date, etc). (Optional, but by grouping larger emails early on, we get a better running time to the algorithm).

2. Compute the first message's centroid.

3. Loop through all following messages; If a message is not part of any cluster, calculate the distance and remove it if it has a distance less than some threshold.

4. All messages removed on this loop form one group, and are marked with a cluster group number. The number serves to differentiate between cluster groups.

5. While unmarked messages left, GOTO 2.

Although the worst case behavior runs in n^2 in practice the algorithm converges much quicker.

3.4.1.3 Keyword Distance

In this algorithm, the distances are computed based on a keyword file. This keyword file containing word tokens of interest can either be provided by the user or can be computed based on a subset of messages, taking all the high/low frequency words and forming a keyword file (Figure A.4).

We start with a set of documents (emails) to score. For each keyword, we first compute the normalized document score, which normalizes the frequency to both document length and per document frequency. That way, one email with many occurrences of a specific keyword should be scored lower than an infrequent keyword across multiple emails. The normalized score is assigned as the weight of the specific keyword. We then compute a count per message, and sum the results.

For example if the word "apple" has a weight of 5, and is found 3 times, we would assign a score of 15 to the message.

Keyword distance measurements is one feature which can be used to cluster a set of messages, clustering them on keyword scores, or ranges of scores.

3.4.1.4 Frequency Distributions Edit Distances

Another clustering method can be calculated by studying the character distributions in a group of messages. We compute a one-character frequency distribution (1-Gram) reordering the 1-grams in ascending frequency. The ascending frequency profile forms a Zipf like distribution, the same distribution that models the naturally occurring frequency distribution of words or characters in natural language. We are in effect creating a unique code sequence to identify a text segment (message). Given two text segments, we can calculate the edit or Levenshtein distance(LD) (Levenshtein, 1966). This is a measure of the similarity between two strings, the distance is the number of deletions, insertions, or substitutions required to transform one into the other.

3.5 System Models

The individual classifiers described in section 3.1 and algorithms described in section 5.3 are used as a basis for the combination classifiers described in Chapter 4. Each classifier is used in supervised training to emit a class label and a confidence score. In EMT (Chapter 5) the user has the means of specifying arbitrary class labels, and choosing from a rich set of available features that are extracted from an email archive.

Chapter 4

Model Combination

In this chapter we describe a way of combining the disparate models described in Chapter 3. We describe the various combination schemes, and a way of measuring the benefit of using a combination scheme. Results of the different algorithms and their combinations are presented in Chapter 8. Some of this has been published in (Hershkop and Stolfo, 2005a).

4.1 Combining Knowledge

The goal of model combination is to leverage multiple learned experts over a given task to improve individual model performance. We may be interested in reducing errors, improving accuracy, or a combination of the two. In addition, by including the input of many types of classifiers we can protect ourselves from risk of any one classifier being compromised.

Many different studies have shown that combination classifiers either over raw features or over classifier outputs are better than any single individual classifier

in the group (Larkey and Croft, 1996; Kittler et al., 1998; Bilmes and Kirchhoff, 2000; Tax et al., 2000; Kittler and Alkoot, 2003; Tan and Jin, 2004; Zheng et al., 2004). We now present an overview of some combination algorithms and specifically illustrate them with examples in the spam detection domain.

For this discussion we will consider two different sets of assumptions.

- Binary Classification - We provide the discussion in the context of a binary classification problem. For example in the email domain, the classifiers are judging an unknown email on the probability that it is spam. In general we are classifying an unknown example between a label and a target label. In this case, "spam" is the target label.

- Output Score - In our work the models return a score in the range of 0-201 with larger numbers mapping to greater probability that an item being part of the target label. We generalize this in this chapter to be on a range of 0 - $\mathcal{SR}$, with $\mathcal{SR}$ as the highest possible score. In Section 8.1.1 we will explain how this is achieved, why 0 to 201, and how this can be generalized to n classifiers.

4.2 Maximum & Minimum

For many of our classifiers, the score returned by an individual classifier reflects a level of belief in the target label. In order to keep this mapping when combining classifier scores, we can use the output of a classifier as the best answer. The simple way to achieve this is when the combination algorithm calculates the probability that the sample is part of the target label, we can choose the maximum individual

score from any of the classifiers. The same for the reverse, when we calculate a low probability of the target label, we return the minimum score.

In the evaluation section we display the results of combining classifiers with and without min/max combination.

4.3 Simple Averaging (Equal Weights)

The simplest way of combining classifiers which requires no training knowledge, is to average the combination of returned scores by the individual classifiers. This has been shown to work well (Kittler and Alkoot, 2003; Kittler et al., 1998). We call this an "Equal Weight Combination" since it assigns an equal weight to each of the classifiers.

The main disadvantage of this algorithm is scaling. As the number of classifiers increase, the performance of average combination tends to peak, and can be improved with a non-average-weighted algorithm. This is because the sum of the errors of the individual classifiers has a cumulative effect of changing the correct responses (Kittler and Alkoot, 2003).

In addition the algorithm makes a strong assumption that all the classifiers are returning a smooth probability score. Smooth score is that for any individual classifier, the larger the score, the greater the probability of being part of the target label. Although classifier scores can be mapped to a smooth probability by learning over training data, some algorithms do not necessarily return smooth probabilities. For example, it might be the case, that a conservative classifier will return 115 (on scale 0-201) for many of the spam examples, although we expect most of the examples to map higher on the score range. There are also classifiers which return

a binary answer 0 or 201 which again does not map to a smooth probability curve.

4.4 Weighted Majority Algorithm

Another combination algorithm is a weighted majority algorithm adapted from (Littlestone and Warmuth, 1989). Each of the individual classifiers is initially assigned an equal weight vote.

During training, a threshold is learned for binary classification (correct or not) and a tally of scores is computed with the majority vote as the predicted classification. If the majority of the classifiers are correct no weights are updated. If it is incorrect, the algorithm deducts a cost β from each of the classifiers which contributed to the incorrect vote.

Modeled after the work in (Littlestone and Warmuth, 1989) we added a term α to each weight of the correctly voting classifiers. Unlike the original algorithm, we reward classifiers which had a correct vote when the overall majority were incorrect. α was set to .001 and $\beta = 4 * \alpha$ with the threshold set to 60. We found these values to work well with β during our tests over different sets of data.

For min/max smoothing, if the majority of weights are more confident that the example is spam, we return the maximum available raw score produced by one of the component classifiers. Conversely, if the weights are more confident that the example is normal, we return the minimum available score.

4.5 Naïve Bayes Combination

In the Naïve Bayes Combination algorithm we attempt to estimate the likelihood of an individual classifier being correct for a given score.

We can estimate this by studying a classifier's performance over a training sample. Since ground truth is known, we can measure the error rate of the classifier and its likelihood of being correct. This probability is estimated by mapping the scores computed for the training data of the classifier to pre-defined bins over the range of the raw scores. We use bins to allow us to cluster scores to achieve a high statistical sampling and reduce the amount of computation.

The number of bins, n, is a parameter. For each bin (score range), we count the number of true spam and number of true normal samples, while keeping a total count of each class label seen in the training set. Then, we estimate:

$$P(S|C_1C_2\ldots C_n) = \delta P(S)P(C_1|S)\ldots P(C_n|S) \tag{4.1}$$

where:

$$P(C_i, BIN_j|S) = \frac{\sharp S_j + 1}{\text{TOTAL\_SPAM}_j + \sharp BINS} \tag{4.2}$$

for the particular bin j, we use a Laplace smoothing factor of $\frac{1}{\sharp BINS}$, where $\sharp BINS$ is the total number of bins. C_i is the i_{th} classifier we are combining. The $P(C_i|NotSPAM)$ is calculated in a similar manner. The final score is returned in the range 0 - $\mathcal{SR}$ by normalizing the estimated probability $P(S)$, that the sample S is spam. The normalization is computed as follows:

$$Score(S) = \mathcal{SR} * \frac{P(S)}{P(S) + P(NotSPAM)} \tag{4.3}$$

4.6 Matrix Bayes

The Matrix Bayes algorithm attempts to closely model the probabilities of the combination of returned scores. We do not assume statistical independence among the classifiers, and thus sample the training examples using an $n \times n$ matrix (for $n = 2$). Intuitively, we expect that if some of the classifiers return a high probability of spam, and one does not, we can correct this particular combination by seeing what the real label was during training and learning the probabilities. In addition because we do not assume classifier independence we require a small number of bins or much more data to train upon.

For example, if we set the bins to size 50 the range [0-201] will map to 5 bins (0-49, 50-99, 100-149, 150-199, 200+). If we have 2 classifiers ($n = 2$), we compute a single 5×5 matrix. If we see a score of 40 from the first classifier, and 30 from the second, this will map to location $(0, 0)$. The probability can be calculated during testing by simply extracting values from the matrix seen during training in the following manner:

$$Score(S) = \mathcal{SR} * \frac{1 + S_{ij}}{S_{ij} + \sharp NotSPAM_{ij} + \sharp BINS} \tag{4.4}$$

where S_{ij} is the number of Spam observed and recorded in the matrix location (i, j). We use a Laplace smoothing factor of $\frac{1}{\sharp BINS}$, where $\sharp BINS$ is the total number of bins (n^2) we have chosen. We limit to $n = 2$ since as n increases the data will be overly sparse.

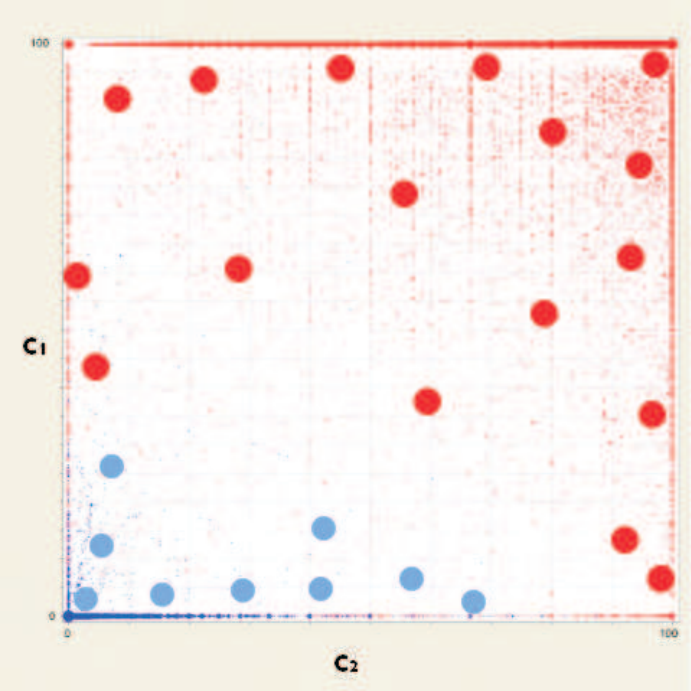

Figure 4.1: Score distribution for two classifiers, $C1$ and $C2$, and the centroids resulting from creating bins. The color of the centroids represents a high probability of spam (red) or low probability of spam (blue).

4.7 Centroid Based Models

Another way of estimating a probability distribution without direct binning is to use a centroid based model to combine classifiers. This adopts well in general when there is an issue of sparse data using direct binning. During the training phase, a centroid is computed for each bin (if we have a sufficient amount of data above some threshold). In cases where the centroid does not exist for a particular bin (because of the training data is sparse i.e. insufficient date) the closest centroid will act as a replacement for the void. Each centroid computed during training is associated with a probability of some target class (spam and non-spam in the binary label example). Figure 4.1 shows a score distribution for two classifiers $C1$

and $C2$, along with the cluster points showing high probability centroids.

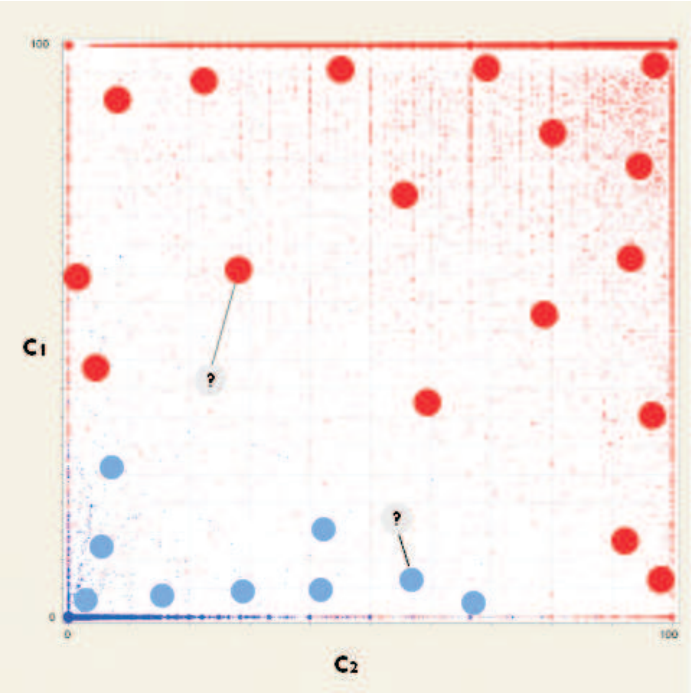

Figure 4.2: Using a nearest neighbor approach for calculating probabilities.

4.7.1 Nearest Neighbor

In the nearest neighbor algorithm, we use the closest centroid computed during training to classify a new unknown example. In figure 4.2 we show how this works. Notice that not every bin has a specific cluster and as expected from a smooth probability output the (min_x, min_y) and (max_x, max_y) have the largest amounts of data points.

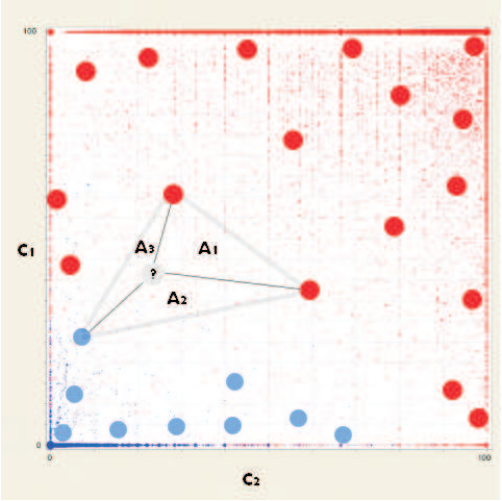

Figure 4.3: Triangulation of cluster points for calculating a probability score on the probability slope.

4.7.2 Triangle Combination

An alternative to using a single centroid is to use a group of centroids in a probabilistic framework. The triangle combination algorithm tries to triangulate an unknown point treating it as a point on a probability surface created by mapping n classifiers to an n dimensional probability space.

We illustrate this method using 2 classifiers. We map the two classifiers to a $n \times n$ grid using the raw scores as coordinates.

To optimize the algorithm we bin the scores and compute the closest three centroids (section 4.7) to form a triangle around the unknown point as in Figure 4.3.

The three centroids will be referred to as (x_1, y_1), (x_2, y_2), and (x_3, y_3) with the unknown point (score to label) as (x, y). They form triangles around the unknown point which we will refer to as A_1, A_2 and A_3. A probability can be associated with each of the centroids Z_1, Z_2 and Z_3.

We calculate the unknown probability in the following manner:

$$Z = \frac{Z_1 A_1}{A} + \frac{Z_2 A_2}{A} + \frac{Z_2 A_2}{A} \tag{4.5}$$

A is the total area of the triangle formed by the centroids. The areas can be calculated by taking the determinant of the matrix with regard to point (x,y) (Figure 4.3).

$$B = \begin{vmatrix} x & y & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \tag{4.6}$$

4.8 Measuring Gains from Model Correlation

There are several ways to measure the performance of the classifier combination. Zheng et al (Zheng et al., 2004) proposes a novel way based on the Data Envelope Analysis (DEA) method. This analysis produces a measurement of how accurate each classifier is in correctly classifying examples. This is different than ROC convex hull measurements proposed by Provost and Fawcett in (Provost and Fawcett, 2000).

Both methods make some strong assumptions about the performance of the underlying classifiers. For example they concentrate on combining the best classifiers, trying to measure what best means. We show how even weak classifiers can be combined in our context of computing a model correlation function.

We introduce a new way to measure the gain in the context of this two class problem, i.e. spam classifiers. If we were to calculate the maximum gain from any

classifier we could measure a relative gain in comparison with this maximum. We call this empirically measured maximum possible gain the Judge Combination.

4.8.1 Judge Combination

One of the current methods for measuring the gain of combining models is to have the training set split into many parts and measure the performance gains over each part and averaging the results. The theory is that one can generalize enough about the performance over the training set to accurately predict the performance gains for the testing set (Ho et al., 1994; Kittler et al., 1998).

We approach the problem from a different angle: we start with a simple question, how accurate is a combination of classifiers if we only combine the minimum or maximum available raw scores from the component classifiers in the correlation function (essentially ignoring lower scores otherwise) knowing the correct label?

Surprisingly, the answer is very accurate! We call the algorithm the Judge Combination. Given that we know the correct classification of an email used to train the classifiers, we return the maximum available score if it is spam and minimum if it is not spam. That is we aim to learn when to choose the maximum available score and when to choose the minimum score from any of the classifiers we are combining. This contrasts with the typical approach of trying to create a super classifier using the individual classifiers as inputs for a combination score.

The accuracy achieved by this combination method, is used as the theoretical limit of a possible combination algorithm and we scale all our gain results based on this accuracy estimate.

4.8.2 Gain formula

We define the gain as a measurement of how much accuracy potential a classifier has reached. This measurement can be used to decide which combination algorithm to use in a system.

$$\text{classifier\_gain} = \sum_{i=0}^{10000} \frac{(10000 - i)}{10000} \times FP_i \tag{4.7}$$

where FP_i is the measured false positive rate of the classifier over its training data. i is varied by 10,000 to allow two decimal precision for measuring the false positive rates from 0 to 100. This is simply the area under the ROC curve, biased towards lower false positive values, that is, we weigh lower false positive with greater weights. We can calculate the FP_i by moving a threshold over the data and calculating an ROC curve, and then averaging the results between points to interpolate the graph.

Since the Judge algorithm represents a maximum possible combination score achievable, we use it to scale the gain score for each individual classifier as follows:

$$gain = \frac{classifier\_gain}{Judge\_gain} \tag{4.8}$$

This provides an easy reference point of measuring different combinations in relation to the judge combination. We can now measure the performance of individual and combination classifiers in relation to each other. This allows us to decide between two combination algorithms that might visually be hard to tell apart, but measure differently in relation to each other's gain scores.

Chapter 5

Email Mining Toolkit

In this chapter we detail the Columbia University's Email Mining Toolkit (EMT). We have documented parts of it in various publications and now present it in its entirety in this chapter. It has been under development since June 2002 and is approximately 60,000 lines of source code implemented in Java. It was initially designed as a learning tool, to illustrate the power of behavior models. As such it contains much more than just the basic email analysis, but also database, reporting, and forensic tools all joined under a data mining framework.

EMT is architectured to be a standalone application for offline email analysis. We have implemented behavioral models to allow behavior analysis to take place in an organized and task-driven environment. Each window allows a user to study a certain aspect of the email behavior with the 'message' window,'flow' window, and 'forensic' window tying different models together.

5.1 Architecture

EMT's architecture is composed of four major components.

Parser - A parser which serves to import the email data. This tool is responsible for taking any email data format and importing it into the EMT database. The parser is described in section 5.1.1 and appendix B.

Database - An underlying database to store the email messages. In section 5.2.1 we describe the schema and rationale in detail. This component is where the actual email data resides for analysis by the models.

Models - A set of Information retrieval and behavioral models. The models which were described in chapter 3 form the basis for the mining toolkit. We describe the implementation and user interfaces in this chapter.

GUI - A front-end Graphical User Interface (GUI), which allows the data and models to be manipulated. It also allows the user to test a range of parameters for each model in the offline system, so they can accurately judge what are the ideal parameters for a specific set of email data.

5.1.1 Mail Parser

The EMT parser is designed to read data from an email collection and import it to the EMT database. We specify an API interface which can be implemented to any email data representation. This allows EMT to analyze any source of email and apply its behavior modeling over an EMT database. Appendix B has a more in depth detail of the parser and our implementation.

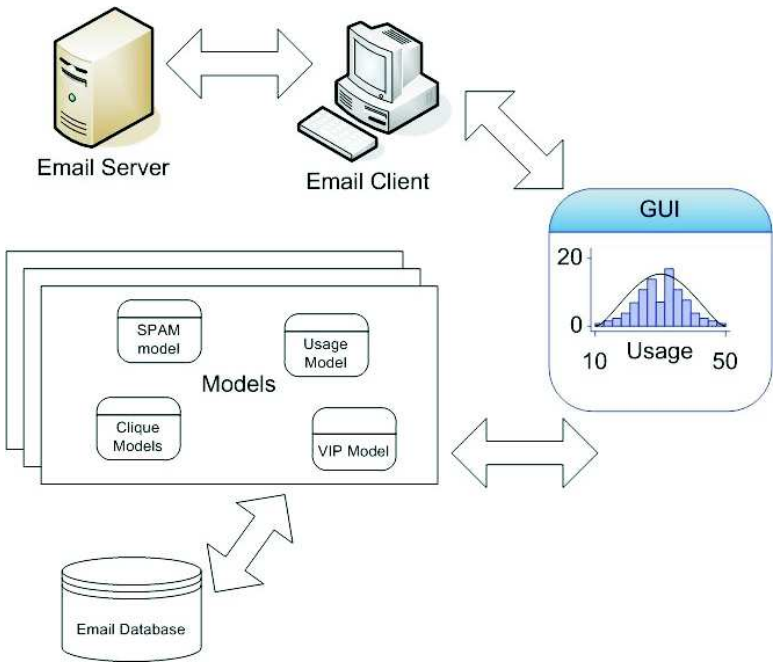

Figure 5.1: The EMT Architecture composed of a email parser, data base back-end, set of models, and a GUI front-end.

Note that the parser is a stand-alone application but incorporated into the EMT GUI framework for convenience. The GUI allows the parser to be used without invoking a separate command window.

5.2 Database

To improve on a simple flat file organization used by most email systems, we have implemented a database configuration to store the email messages. A database format allows us to quickly and efficiently store and retrieve individual and group messages. In addition it allows compact representation and quick statistical lookups to take place. It also builds upon current database technology for portability and rollback features.

We have made specific trade-offs between space and computation consideration in the design of the EMT database schema. We allowed data redundancy where computation would benefit, because space is currently far cheaper than computation time. Specifically many of the statistical features exist in more than one form in the database and index keys, but are part of the design of the system. In addition some of the data redundancy is to allow us to operate in "privatized" mode, allowing the actual data to be encoded for privacy while allowing meaningful statistics to be computed.

5.2.1 Schema

The Schema for storing email messages shown in Figure 5.2 in EMT consists of three main tables:

Email Table -

The email table is the primary collection of information about an email message. Its schema is presented in Figure 5.2. Information about the sender, recipient, date, and time are stored.

When a single email has many recipients, the table inserts a row for each recipient (as if the email were viewed by a distinct sender/recipient pair). The reason for this is that, conceptually, there is no difference between sending an email to many people at once and sending it to each person individually. The only difference is an administrative one, in the column titled "number of recipients" (this is the numrcpt field in database schema). It also allows many of the statistical computations to run faster, since we do not need

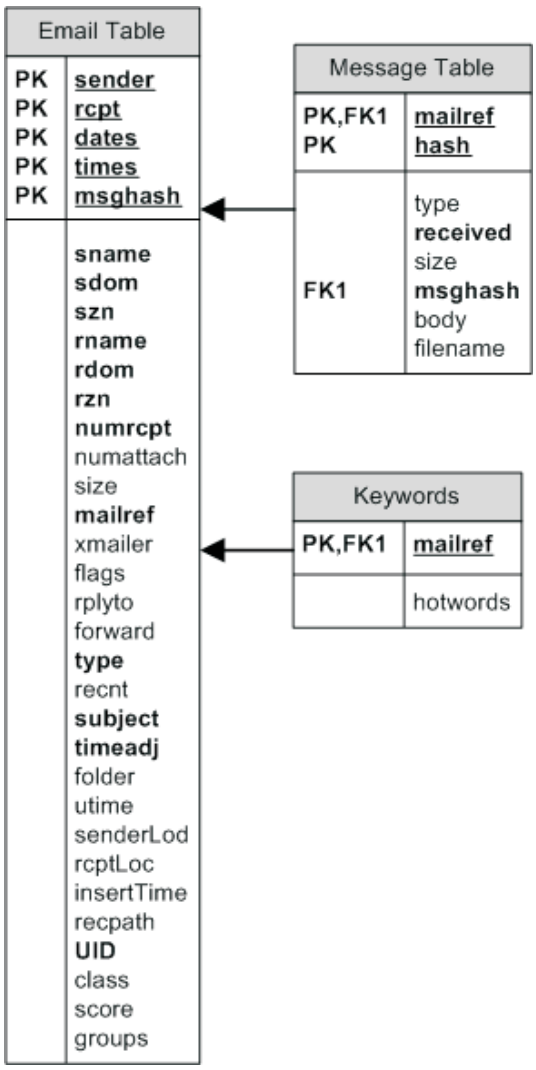

Figure 5.2: The EMT schema consisting of email, message, and keyword tables. **PK** is the primary key for a specific table, while **FK** is a foreign key in the database. Fields in bold represent indexed columns.

to recompute specific statistics when we are interested in specific users or communication patterns.

Each email has two unique identifiers in the schema. Each row in the table has a unique field that indicates the email's unique identification number (UID). In addition, each processed email has a unique mailref, although a few emails may share a single mailref if they were on the original email's "to" or "cc" list.

Notice that some information seems at first glance to be redundant. For example, why keep both the user full email and the email up until the "@" sign in two columns? The reason is that EMT is configured for privacy as described in Appendix B and when the data is scrambled in the database, this is the only way to maintain user statistics without revealing the actual user during analysis. This mode of privacy also allows the parser to gather email data in many locations (hashed), and then merge many small EMT databases into one large one without compromising the original email information‘.

We now describe each column in detail.

uid - A unique identification number for each row. This is mainly for database and insert statistics purposes.

sender - The full email name of the sender.

sname - The first part of the email name up until the '@' character.

sdom - The last part of the email name after the '@' character.

szn - The email zone of anything after the last period after the '@' character. For example .com, .edu etc

rcpt - The full email name of the recipient.

rname - The first part of the email name up until the '@' character.

rdom - The last part of the email name after the '@' character.

rzn - The email zone of anything after the last period after the '@' character. For example .com, .edu etc

numrcpt - The number of recipients.

numattach - The number of attachments

size - The size of the entire email.

mailref - A unique string to identify the email based on the hash, internal id. We append the folder name if not long enough.

xmailer - The mail client id string.

dates - An SQL formated date such as 0000-00-00

times - An SQL formated time such as 00:00:00

flags - A string representation of the flags in the email such as x-header information.

msghash - A md5sum hash of the entire email.

rplyto - The reply to email name if present.

forward - The forward flags.

type - MIME type of the message.

recnt - The number of 're', 're:' removed from subject line beginning.

subject - The cleaned up subject line.

timeadj - Time zone adjuster.

folder - File or folder name of the message.

utime - Unix time stamp of the date, a long type.

senderLoc - character representing sender source (Internal/External).

rcptLoc - character representing recipient source (Internal/External).

recpath - Full received path string.

class - One character representation of class label (Spam,Virus,Interesting,Not interesting,Unknown).

score - Either a user assigned score or learned score associated with the email, which we might want to store.

groups - Group id number.

ghour - The hour as an int.

gmonth - The month as an int.

gyear - The year as an int.

Message Table -

The message table contains information about the email's contents, including any attachments to the email. Attachments are represented by the md5 sum hash, which is based on the contents of the attachments. If more than one attachment is inserted into a single email, then each attachment is represented on a separate data row. Secondary information includes the path information and the first x bytes of the body. In addition to its unique hash, a message

part (body or attachment) is associated with a message mailref and message hash.

We now describe each column in detail.

mailref - A unique id for each email.

type - The mime type of the message part. This is not necessarily the same as the type found in the email table. For example, an email might be labeled "MIME multi-part" and have binary attachments and others.

hash - A unique hash of the message part.

received - The entire received line.

size - The size of the attachment part.

msghash - A hash of the entire message.

body - The full body of the attachment in binary format.

filename - The filename (if any) associated with the attachment.

Keyword Table -

The Kwords table collects keyword statistics about the content of an email. If a keyword file is used, then the table notes the frequency of keywords and updates the database.

We use this table as a fast updated cache when keyword functionality is enabled during EMT operations. By storing high or low frequency keywords associated with each email, we can compute group cluster more efficiently.

5.2.2 Implementation Details

EMT uses two types of databases. The first type is a Mysql database running either locally or remotely, and the second type is an open source Java based database technology called Derby. Because of the way both databases are wrapped, they look identical to the components of the system.

Mysql is an open source, platform dependent database (MYSQL, 2005). The main advantage to this option is that the database is written in machine native code which translates into quicker running time. The drawback of Mysql is that it requires user setup and configuration, and if not performed correctly hinders EMT from running because of permission errors. In addition to being machine portable, we need to download or provide a machine executable binary of Mysql for each operating system. When running large sets of data experiments, Mysql is a better choice.

Derby is an open source database technology that advertises itself to be pure Java, easy to use, small footprint, standards based, and secure (Derby, 2005). Derby is based on the former cloudscape database technology donated by IBM in 2004 to the Apache Foundation. It is a stand-alone platform-independent, Java application which we bundle with EMT. The advantage is that the user does not need to install any outside components. The drawback is that the current performance is a little slower than that which is provided by Mysql. We have found, when the number of email analyzed to be very large ($>$ 2 gig of data, 300,000 emails) the performance differences were noticeable but not considerably so.

5.2.3 Database Example

We illustrate the workings of the database by providing an example of how an email would be divided among the different database tables. If user A sends an email "TTT" to both user B and User C, the email table would have two entries: (A, B) and (A, C) both referencing msghash (message hash) of TTT. In addition each email is referenced by a unique mailref string.

The Message table would have one entry for the msghash of TTT, if TTT had multiple parts. Each part would generate one entry in the Message table. For example, if TTT is composed of parts 111 and 222, then each part would have a unique entry in the message table but still be associated with TTT's mailref and msghash (TTT,111) and (TTT,222).

When retrieving a message in EMT we can recompose the message by creating joins on the database table using the mailref as a key to link both sets of tables. The reason is to allow for example when one user repeatably sends a specific exact message multiple times, to reside only once in the database. This can happen when for example resending important documents, reminders, and advertisement messages.

5.3 Models

Each of the EMT models addresses a different part of a user's behavioral analysis. We described general models in Chapter 3. We will refer to specific models when describing specific GUI modules in section 5.4.

Because EMT was designed to analyze a user's email, we leverage specific

models for either incoming or outgoing analysis. For modeling outgoing behavior, we use the usage, similar, and clique models. For incoming behavior, we use the communication, attachment, and URL models. Because data might be richer in one direction over another when analyzing user-specific data, we make these distinctions over which model is for what email direction. For example, naturally we usually have more precise data to compare users when we look at their outgoing behavior (how they send emails) rather than incoming behavior (when they receive email). This of course is highly dependent on the email archive data retention, as the models can in reality be used in either direction. From our experience, some users will keep all mail sent and received, while others only keep either outgoing or incoming emails.

5.4 EMT GUI

The GUI provides a front-end to users to use the data mining system. Each of the different components is divided among data gathering, viewing and computing. The "Gathering" function includes the parser and database related views which are related to acquiring email data. The "Computing" functions are those sections of EMT which allow different types of machine learning and behavioral models to be computed and tested against the underlying data and to experiment with different parameter settings. The "Viewing" elements are where specific model parameters can be tested and results analyzed by an analyst.

5.4.1 Message Window

The Message window offers a particular view of the data in the database (Figure A.1). A view is composed of a set of constraints defined over the data. For example it can be all messages associated with a particular folder or user.

The following features can all be used alone or in combination to constrain the data view.

1. Date - All messages between a set of dates.

2. User, Direction - We can choose a specific user to view all their email, and also define which direction (inbound, outbound, or both) we would like to view.

3. Label - We can view specific emails, such as spam or virus.

4. SQL - An SQL statement can be defined to specifically choose a subset of the data. This allows users to extend the schema and use those extensions within the EMT framework.

Views exist to allow the system to scale to arbitrarily large amounts of messages without taking over all of the system resources. The message window also allows old views to be viewed by using a back and forth button near the top of the GUI.

In addition the message view allows machine learned model creation and evaluation. Individual or groups of messages can be labeled, and processed by any of the machine learning classifiers to create a model and saved for later use. Models can be loaded, and evaluated over a given view, to label or score a group of messages.

5.4.2 Attachment Statistics

Figure A.8 displays the attachment statistics window which allows analysis of attachment behavior. The Email attachments can be grouped by name, size, or content. MET Models (Bhattacharyya et al., 2002) can be run over all the data, and alerts issued for any attachment violating the MET measurements. These features and measurements are described in section 3.3.

Attachment statistics can be viewed, saved, or selected for further analysis. In addition the user can group similar attachments either by similar name or similar content (using N-gram distance).

5.4.3 Similar Users

The Similar users window (Figure A.3) is designed to find a group of email accounts that have similar usage behaviors with any selected account. It can also be used to view the usage histogram of different metrics of the email accounts.

A specific selected user can be compared to all others, or only those accounts which have issued emails. The difference is that for source emails, we have better statistics than those users which appear exclusively as recipients.

The histogram comparisons can be either normal or aligned (section 3.4). There are four features used to find similarity; the average number of emails sent out per hour; the average size of emails sent out per hour (in KB); the average number of attachments of emails sent out per hour; and the average number of recipients of emails sent out per hour. In addition there are four comparison functions available; L1-form, L2-form, Quadratic, and KS-test.

The window displays the results of the comparison as an ordered list of

similar users, and a set of histograms associated with each similar user. It allows an exploration of related accounts, when analyzing large email sets.

5.4.4 Usage Profile

The Usage Histogram window compares an email account's behavior over time (Figure A.2). A profile period or training period is selected (example: first 75%) and this profile period is compared to a test period known as recent period. The default behavior is to compare the last month as the profile and last week as the recent.

Either a specific account or all accounts can be checked in an automatic fashion. For those accounts whose recent period is anomalous (in a histogram sense) above a certain threshold, an alert is issued. The distance method used here is the Mahalanobis distance function, which is described in section 3.4.0.4. According to the distance and the alert level specified the system judges whether the recent behavior is normal, abnormal, or might be abnormal. These judgments are indicated by different colors: red for abnormal behavior alert, yellow for might be abnormal alert, and blue for normal behavior. If the behavior is abnormal or might be abnormal, the alert is sent to the whole system's alert log (section 5.4.8) for further analysis by a forensic expert.

5.4.5 Recipient Frequency

The Recipient Frequency window models email flows from an individual email account. It analyzes the frequency at which the selected user sends email message to each recipient.

We analyze either only emails sent by the user (out-bound), only the ones received by the user (in-bound), or only the ones sent with an attachment (out-bound with attachment).

The chart in the upper left side of figure A.5, called the Recipient Frequency Histogram, plots a bar graph indicating the frequency of each recipient to whom the selected user has ever sent an email. The table in the upper right side, called the Recipient List, displays the email address of each recipient along with the corresponding frequency of received emails from the selected user. This table is sorted is descending order, and each of its rows corresponds to a bar on the Recipient Frequency Histogram.

The chart on the lower left side of figure A.5 is the Total Recipient Address List Size over Time chart. By "address list", we mean all recipients who have received an email from the selected user between the two selected dates. Here, the chart plots the address list size, i.e. the number of recipients in the address list, over time. It starts at zero at the beginning of the dataset and grows as the selected user sends more emails to new recipients not already listed.

The chart in the lower right side of figure A.5 is shows the number of distinct rcpt and the number of attach per email blocks chart. It displays three variability metrics and the rolling averages of each of them.

We calculate and display the results of the Hellinger distance (section 3.4.0.6) over recipient frequency email activities. In addition there is a Chi Square window which compares the selected user's profile between two time frames (recent vs profile). The degrees of freedom and the p-value of the Chi-square test, as well as the Kolmogorov-Smirnov distance, are displayed at the bottom of the window.

Analysis of activity in relation to cliques can be done by first calculating the cliques and then running a Chi-square test in a sub-window. Cliques are groupings of users with large pair-wise email flows, thus representing tightly connected small groups of email users. The results are displayed in a table grouping users into cliques and displaying group statistical behaviors.

5.4.6 Cliques

To study email flows between groups of users, we compute a set of cliques in an email archive. We seek to identify clusters or groups of related email accounts that participate with each other in common email communications. Conceptually, two broad types of cliques can be extracted from user email archives: user cliques and enclave cliques. In simple terms, user cliques can be inferred by looking at email history of only a single user account, while enclave cliques are social groups that emerge as a result of analyzing traffic flows among a group of user accounts. We display enclave cliques in a tabular as well as a graphical form seen in Figures A.6 and A.7.

5.4.6.1 Enclave Cliques

A clique is a group of accounts that, pair-wise, have exchanged some minimal number of messages. To find cliques in the dataset, set the minimum message threshold on the top of the screen, and click Refresh Users. If you set the threshold to 30, for example, each pair of accounts in the clique will have exchanged at least 30 messages, perhaps 13 in one direction and 17 in the other. The results will also show the most common subject words for messages exchanged within each clique

group (Figure A.6).

5.4.6.2 User Cliques

We define a user clique as a group of individual mail accounts related by some common set of communication and involving at least k emails. For example, if User X sends emails to A, B, and C, at least k times, then we can say that A, B, C form a clique with respect to user X. In this definition, a clique consists of two or more users.

The threshold box specifies the k value. The training period refers to the time frame necessary to learn the user's cliques. The testing period refers to the time needed to test the learned cliques. These can be changed in order to study the behavior of an individual over time with respect to his or her cliques. The Clique Sets button shows the cliques generated from the training period. The Clique Violation button shows the cliques that were seen in the testing period but not seen in the training period.

When results are displayed, additional information is available on a per clique basis. The first column shows the number of times that the clique was observed. The weight column computes the weight of the users in the cliques. Then, the actual clique is shown.

A graphical representation of the users email over time is displayed for each user. This enables a visualization of how the training and testing sets are related (see figure A.9).

5.4.6.3 Graphic Cliques

The graphic cliques window displays a graphic representation of the clique groups. It allows the user to navigate and choose a specific clique to analyze in-depth by choosing specific constraints on the clique building algorithm. The graph uses polygons and lines to overview the results of the cliques relationships. Different shapes and colors are used to emphasize different aspects of cliques and their relationships.

Different parameters are user adjusted for selecting clique formation. EMT utilizes two different algorithms to calculate clique, With or Without Keyword (Common Subject Words). Minimum amount of connections is a threshold chosen by the user to define the minimum number of emails exchanged between cliques to be considered in the analysis.

The window display is divided among three panels. These panels, the Information Board, the Clique panel, and the User panel, are shown in Figure A.7 and are described below.

Clique panel :

This panel displays results for all of the cliques that have met the user specified constraints. Each dot on the graph represents a clique (not an email account) with an associated reference number. The different sizes and shapes of the dots indicates different types of cliques. Circles indicate 2-clique, triangles indicate 3-clique, squares indicate 4-clique, and so on.

The lines portray the connection (i.e. the relationship) between cliques. A line connecting two dots signifies that these two cliques share the same email account(s) (i.e. clique members). The color of the lines represents the per-

Color	Percentage
Orange	$< 10\%$
Yellow	$> 10\%$, $<= 20\%$
Red	$> 20\%$, $<= 30\%$
Cyan	$> 30\%$, $<= 40\%$
Blue	$> 40\%$, $<= 50\%$
Magenta	$> 50\%$, $<= 60\%$
Green	$> 60\%$, $<= 70\%$
Gray	$> 70\%$, $<= 80\%$
Pink	$> 80\%$, $<= 90\%$
Black	$> 90\%$

Table 5.1: Alert Color percentage values.

centage of members that the two cliques share. The table below defines which percentage is associated with which color. For example, a red line between two dots means that these two cliques share 30% of their clique members. This is presented in Table 5.1.

User Panel :

This graph displays all of the email accounts. The blue dots represent cliques; they are the same as the dots in the Clique panel, The dots use the same identifying reference number as in the Clique graph; each number indicates a particular clique.

Black dots represent all of the email accounts. The leftmost column of black dots indicates that each of these email accounts is involved with only one clique. The second left column indicates that each of these email accounts is involved with two cliques, etc. The rightmost dot(s) (i.e. email accounts) are involved with the most number of cliques. Thus, EMT displays which email

account(s) is involved with the greatest number of cliques on the far right.

The colored lines indicate connections between email accounts and cliques. A line connects an account with a clique thus showing that the account is in this particular clique. A clique may contain more than one account, and an account may be involved in more than one clique. Different columns use different colors.

Information Board :

The Information Board enables further analysis on the cliques. When a dot (clique) is chosen, the Information panel is updated with the email account(s) and "Common Subject Words" associated with this clique. It generates a small picture showing what a clique looks like. In addition, in the Clique panel the selected dot turns red (the others remain blue). In the User panel, the selected clique and its email accounts also turn red. To see which email account(s) that two cliques share, the user can choose the connecting line and the information board now displays information about the shared accounts, and the selected lines becomes black.

5.4.7 Email Flow

The Email Flow window (Figure A.10) portrays how a message permeates through an organization and shows how people relate to each other with a message flow containing the same content (via ngram analysis) as the original target message (chosen by the user). The flow starts at an individual email message, groups other emails either by subject or by content that have similar content, finds all the relative

or similar emails and shows details such as the relationship of senders and recipients, content and time.

We can visualize the flow of a message and easily find cliques between the email accounts by this graph. Thus, the flow pattern indicated by nodes (email accounts) and arcs (message exchanges between accounts) define interesting "content based cliques" within an organization. The graphical display shows a series of concentric rings. The inner most ring and node corresponds to the original target email. Time is depicted by stepping outward ring by ring. The nodes in the next closest ring are accounts that have exchanged email in the next time step. The spread and number of arcs and nodes provides a view of how an individual message affected communication over a broad set of email accounts and the time frame that this spread occurred. The lower left panel in Figure A.10, is a table form of the graphical representation on the lower right.

5.4.8 Forensics and Alerts

Email forensics have played an important role in various litigation and investigation proceedings. Many organizations by law are required to retain all email that passes through their systems. Large organizations might have many email servers, in many locations generating large amounts of data. In cases of litigation or investigation a specific subset of the email data is required. In many cases the organization is most interested in complying without releasing too much or too little information.

Figure A.11 displays the EMT forensic view. First a set of emails is chosen to analyze. The emails can be grouped by similarity to aid in the finding interesting emails. A specific email can be selected and their usage history and VIP list

generated. A specific recipient can be selected from the VIP list and the interaction between the main user and VIP user can be displayed as a message view, in the message window. In addition the main user can be studied in any of the other EMT windows.

This window also includes a reporting mechanism for organizing, displaying, archiving, and analyzing alerts from the various parts of EMT. This is highlighted in Figure A.12.

A forensic investigator working with an unknown set of emails, can use the EMT system to locate either accounts of set of messages to investigate. Interesting email accounts can be automatically selected based on a subset of behavioral models within EMT. For example, using the histogram models, one can locate email accounts which are behaving anomalous over a specific period of time based on past usages. Once an email account is located, its cliques and VIP accounts can be computed to aid in find other sets of email accounts to investigate. In addition, a subset of all the messages can be isolated by studying the communication flows and locating anomalous patterns.

Chapter 6

PET

EMT was initially designed as a learning tool, to illustrate the power of behavior models. We have packaged the integral portion of EMT into a package we call PET with the specific aim of making it easily integrated into any standard email environment through plug-ins.

6.1 PET

The Profiling Email Toolkit (PET) is designed to allow the user to view messages within context by automatically modeling past behavior information. For example when multiple messages arrive at the same time, the user is presented with the messages sorted by relevance. The relevance of a specific message can be computed by observing past behavior of the user with regards to equivalent or similar messages. User behavior models can be used in this context to provide message meta-data relevance scoring. In addition, different than the current available tools (Itskevitch, ; Pazzani, 2000; Segal and Kephart, 1999; Segal and Kephart, 2000; Winiwarter,

1999; Lewis and Knowles, 1997), it can not only suggest relevant folders but also offer additional information such as a score of message importance, suggested actions, past email flow (contextual), and related emails (references and follow-ups). Related emails would be either similar in structure or similar in behavior. Similar to credit card users, email users can be broadly classified into sets of similar users based on comparison functions or distance formulas applied to user profiles as discussed earlier.

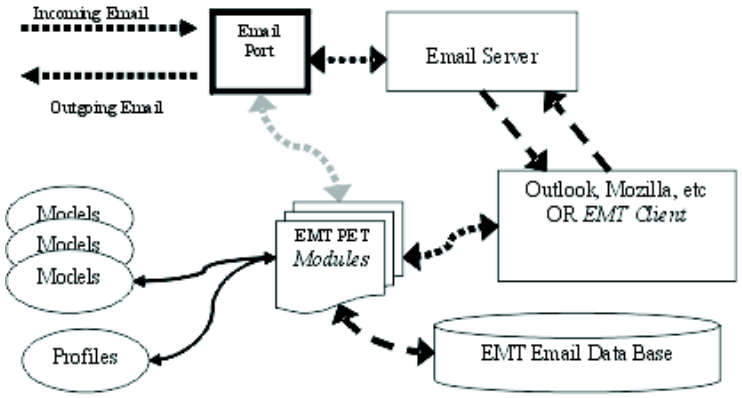

Figure 6.1: The PET Architecture.

Message view organization described in section 5.4.1 is one example of how to allow users to customize their email experience. Studies have shown that not all users treat email the same way (Ducheneaut and Bellotti, 2001; Whittaker and Sidner, 1996). For example, a doctor monitoring patients through email would appreciate an automatic system alerting to the presence of a message of utmost importance. Behavior modeling or simple similarity content test would be able to accomplish this task. Currently most users rely on hard coded rules based on a pattern in the subject line to flag messages. These are susceptible to false alerts by

spam messages.

No current client system which we are aware of offers an integrated set of behavior modeling features; in this chapter, we aim to demonstrate the power and utility of behavior-based modeling packaged in an easy to use deployable system. We now describe the PET system, a packaged toolkit for building and using behavior models.

6.2 Architecture

The email system consists of a set of servers and a user-side client reader, which performs the usual fetching, displaying, composing, and retrieval of email data. Examples such as Mozilla Thunderbird, Lotus Notes, Outlook, or Outlook Express are all typical email clients. The PET system integrates within the current email framework and consists of two components.

The first is the email storage component, which can replace or complement current client capabilities. EMT currently implements a database scheme for email stores. We use this implementation in our design. The actual data is stored in a relational database making relationship queries a natural way of organizing the data. By utilizing data base components, backups and verification of the data can be easily implemented and used within current email clients.

The second part of the system is the PET models section. This part takes as input an email message and updates the various models and computes feature measurements for each type of profile. In addition, it provides support of the enhanced search capabilities added to the client. This section also contains the logic to combine multiple profiles. The output of this section can either be logged

or is sent to a display widget on the email client.

We have interfaced to the Mozilla Thunderbird email client (Mozilla Foundation, 2005) but in theory any client side plug-in can be written to visually display the model output on the client side environment.

6.3 Requirements

We outline a number of requirements we impose on the PET system. The system should work within the current framework of protocols and should not alter what we perceive as underpinnings of the success of email; low computational cost, ease of use, trust, compatibility, and portability. We now list the requirements in detail.

6.3.1 Computation Cost

The PET application should not create heavy computational overhead. Any additions to the email client system should not be noticeable to the user for example with long pauses to update the display with behavior information. The models are multi-threaded in EMT and so the system updates each model in parallel, with low overhead.

6.3.2 Ease of Use

By ease of use, we refer to the intuitive look and feel of messages, folders, and addresses that have become standard in many clients should be in some way maintained. Although we aim to redefine what it means to use a folder and the entire email experience, we should still allow users to quickly grasp the underlying mech-

anisms by providing convenient and easy GUI displays, based on what they have been exposed to in the past.

The additions that we have added do not add another layer of complexity to the email system, since most of it will run in the background. In fact most of the components are invisible to the user with a very intuitive display of the underlying data.

The install package of PET allows an easy integration into the Thunderbird email client, which would allow the user to easily add these new features without having to install and learn new programming environments. The additional information help place messages within context, which we will elaborate later in the Thesis.

6.3.3 Security

In theory this framework of PET and its enhancements may be run as a client-side or a server-side plug-in. On the server, trust is of paramount importance, since it would be a central location for all the users models of behavior.

Although forging source email addresses is currently a trivial task, for the most part anecdotal evidence suggest that users take source addresses for what they appear unless they are obviously forged (Weisband and Reinig, 1995). Most users do not expect to receive an email from the President of South Africa for example. In most cases, user X who sees a message from user Y assumes it was generated by user Y. Virus and spam messages have taken advantage of this assumption and as a result have eroded the basic trustworthiness of an email message. By preventing misuse and detecting anomalies, users would likely find utility in our system of

restoring trust. Showing an anomalous score associated with a specific message, the recipient can be alerted to potential email misuse by unusual circumstances or hijacked account.

Privacy concerns need to be identified and addressed in any implementation. Server based solutions would need to compute history models, but encode them in such a way that would not compromise individual privacy. (Bloom filters would be ideal to store this type of information as they are both compact and one-way hash structures.)

6.3.4 Compatibility

Clearly, PET must be compatible with existing systems and protocols. As mentioned in section 2.3.4, some of the current propositions have proposed to fragment the email system into zones. This violates a fundamental requirement of the ease of use of email; that it be available to everyone equally. The decentralized nature of the Internet and email system is what lead in part to its phenomenal popularity.

There are many legacy systems in use which need to operate over email channels. If the solutions require a complete overhaul of the current scheme, many systems would be left out in the cold. This thesis allows the end user the power to view and manipulate data differently without destroying a system that works.

Because our solution integrates into an email client, this allows us to extend the current system without requiring overhauling it entirely. In a general sense, the technology behind the Internet and email has changed over time, and so the protocols will eventually have to reflect this reality and adopt. But this will take time, and solutions need to fit into the current framework without causing damage.

6.3.5 Portability

A popular feature of email is the ability of many email clients to access email content from anywhere using a web browser. We feel the same should apply to our system. User models should be portable to any email client through a pluggable architecture. Additionally system profiles should be portable in the sense that they should apply their statistics for a given account if the email server were to be replaced.

We achieve this by allowing the computed models to be stored in a separate data directory, allowing the user to port and synchronize them between systems. Although we have implemented our solutions for the Thunderbird email client, the toolkit can be extended to any client to create a PET information enhancement.

6.4 PET Features

We now describe the types of features collected by the PET environment. Each of these features provides information on either a message or user behavior.

6.4.1 User Behavior Features

Similar to EMT's base features described in Chapter 5 we make available features to model individual user behaviors including:

1. Average response times
   (a) First seen messages - identify starts of threads
   (b) Replies - identify replies (even if not quoted in message)

2. Cliques - groups of recipients
    (a) Long/short term cliques
        i. Average sizes - the average size clique for the user
        ii. Average creation - frequency of new cliques
    (b) Per clique behavior
        i. Number of messages exchanged
        ii. Frequency of correspondence - for each recipient
        iii. Type of communication one-many, many-many counts
3. Recipient Frequency
    (a) Incoming
    (b) Outgoing
    (c) Average attachment counts incoming
4. Histogram of usage based on hourly/day of the week behavior
    (a) Average number of emails sent
    (b) Average number of emails received
    (c) Average number of Attachments outgoing
    (d) Average number of recipients
5. User Email Information
    (a) Name before @ sign (user name)
    (b) Domain

(c) End of domain (zone)

(d) First time seen

These histograms are computed for both incoming and outgoing email on the main user account. In addition each email account seen, has individual histograms associated with it.

6.4.2 Message Behavior Features

Features to model individual and group messages include:

1. Content of Message

   (a) Entire content - bag of words

   (b) Subject line

   (c) Date

   (d) Time

   (e) Average size

   (f) Flags

   (g) Location of sender (internal/external)

   (h) Location of recipient (internal/external)

2. Level of Interest

   (a) Deleted only

   (b) Deleted after read short time

(c) Deleted after read long time

(d) Moved to new Folder

(e) Moved to old Folder X

3. Thread of Discussion

(a) Average type of discussion

(b) Average number in discussion

(c) Clique changes in discussion

(d) Attachment changes in discussion

4. Attachment Statistics

(a) Malicious Email Filter, MEF score (Schultz et al., 2001a)

(b) Type of attachment

(c) Average size

(d) Average number

(e) Frequencies of types

(f) Incoming/outgoing

(g) Keyword centroid

(h) N-Gram centroid

(i) Histogram of 1-gram sorted (distribution of characters in attachment).

5. Flow Behavior

(a) Within/outside of cliques

(b) Histogram of messages between replies

6.5 Profiles and Alert Reports

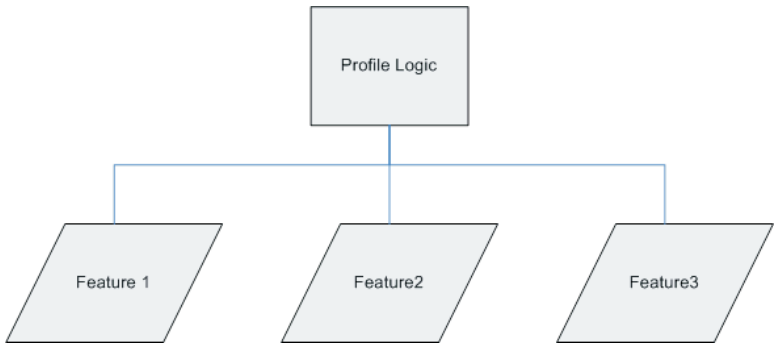

Figure 6.2: A profile is a data structure that represents a behavior model over a subset of features

A profile is a data structure which represents a behavior model over a subset of features. Each profile has a unique name, and the user can specify custom profiles. Additional features can be added to the system by a programmer and used for future profile enhancements. The different windows presently in EMT all represent unique profiles. We extend EMT in PET by creating portable profiles, and the ability of user-specified profiles.

For example the 'Usage' profile currently in the usage window is composed of user features 4a-d but is hard-coded in EMT to the specific window.

The profile logic also contains the rules for issuing alerts from a specific profile. Alerts can be issued either on a specific profile or a correlated set of profiles.

In addition messages can be associated with each other under some specific profile. In that case a group of messages are summarized under one profile that models a centroid of the messages.

Given a set of messages, we can profile them, and then for each message in the archive find the distance from this profile. All distances under a pre-computed threshold would indicate a similarity to be included in the message group cluster. In this case the profile structure would show related messages.

This is implemented and described in section 7.4. The alerting mechanism is described in section 5.4.8.

6.6 Sample Profile - User Similarity

We present a sample PET profile that is currently implemented in EMT's "similar user" window.

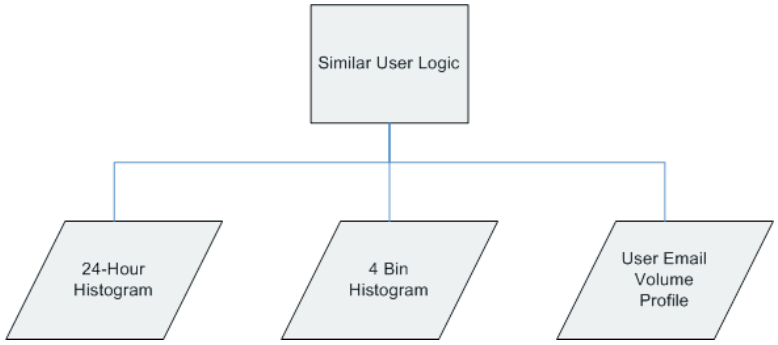

Figure 6.3: Similar User Data Structure Profile

Similarly behaving user accounts may be identified by computing the pair-wise distances of their histograms (e.g., a set of accounts may be inferred as similar to a given known or suspect account that serves as a model). We balance and weigh the information in the histogram representing hourly behavior with the information provided by the histogram representing behavior over different aggregate periods of a day. This is done since measures of hourly behavior may be too low a level of

resolution to find proper groupings of similar accounts. For example, an account that sends most of its email between 9am and 10am should be considered similar to another that sends emails between 10am and 11am, but perhaps not to an account that emails at 5pm. Given two histograms representing a heavy 9am user, and another for a heavy 10am user, a straightforward application of any of the histogram distance functions will produce erroneous results.

Thus, we divide a day into four periods: morning (7am-1pm), afternoon (1pm-7pm), night (7pm-1am), and late night (1am-7am). The final distance computed is the average of the distance of the 24-hour histogram and that of the 4-bin histogram, which is obtained by regrouping the bins in the 24-hour histogram.

Because some of the distance functions require normalizing the histograms before computing the distance function, we also take into account the volume of emails. Even with the exact distribution after normalization, a bin representing 20 emails per day should be considered quite different from an account exhibiting the emission of 200 emails per day. Figure 6.3 graphically displays the similar user profile structure. The type of histogram distance measurement and the weight assigned to each feature in the profile is what defines the specific profile. Notice that a profile can contain other profiles in a recursive fashion.

6.7 Plug-in Modules

The plug-in modules of PET (shown in Figure 6.1) are extensions to a standard email client. The modules allow the output of the profiles to be displayed in the email client window. In addition it allows the user to create profiles, view the internal representation of the profiles, and set system-wide preferences.

6.8 Summary

To summarize we have developed a plug-in system to allow the EMT behavioral models to be used by any email client in an environment we call PET. This allows a standard email client to access the behavioral model information and add context to the email experience. In the next chapter we illustrate other behavior-based extensions to email made possible by leveraging historic email behavior applied to new messages.

Chapter 7

Priority Mechanisms

The PET system described in Chapter 6 is a data mining extension to email clients designed for a variety of analysis tasks applied to email. In this chapter we describe how to apply several of PET's built-in analytical features to prioritize emails for user consumption based upon an analysis of prior user behavior. This application to prioritize emails extends the typical email experience provided by current generation email clients. The user's prior email behavior is used to extract contextual information to propose a priority ordering over new messages.

7.1 Background

Standard email clients for the most part, treat each message as an independent entity presenting messages to the user as some combination of sender, recipient, subject, and date (Figure 7.1). Additional information about the status of the current message such as unread, read, and answered is also usually provided.

Some email clients provide a means of threading messages based upon subject

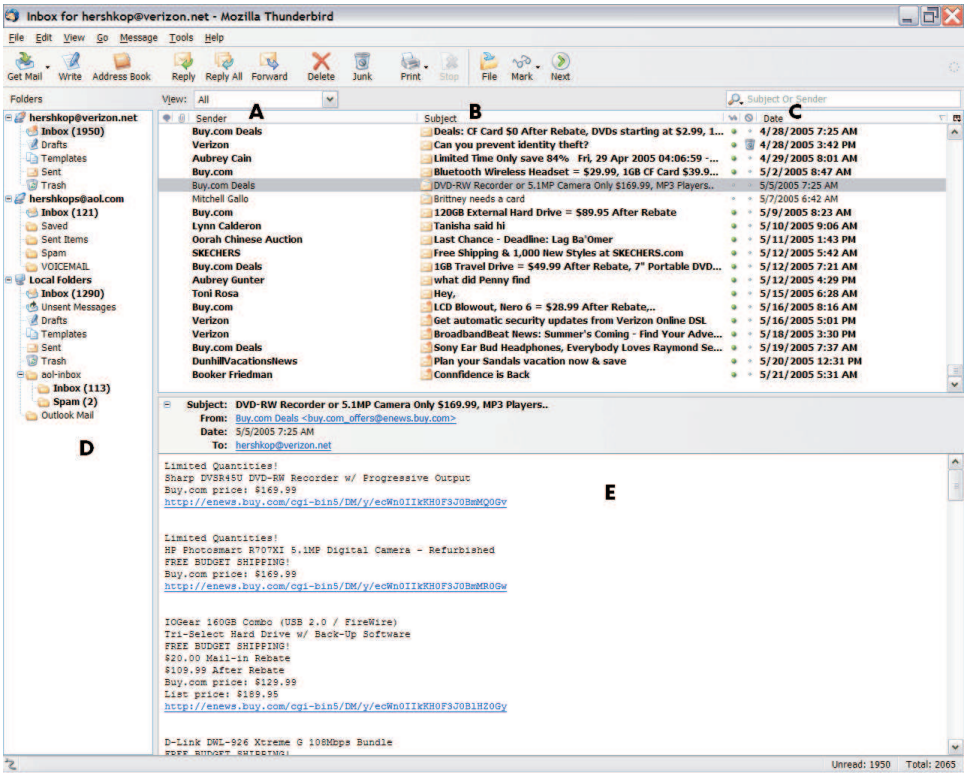

Figure 7.1: Standard email view in Thunderbird using sender (A), subject (B), and date (C). In addition standard email clients also include Folder lists (D) and message preview (E).

line analysis. For following discussions, clients can be set to embed prior email bodies in a reply message memorializing a long email discussion in an increasingly large and complex body.

Furthermore, most clients support "sender priority", a means of a sender marking messages with a user specified prioritization marked on the message when received in the recipient's inbox. This prioritization is sender initiated, not necessarily what the recipient would agree with. That is to say, to protect against abuse, the recipient ought to decide what is high priority and what is not, regardless of the sender indication.

Moving messages into folders is the standard method for helping a user manage their email communications. Some clients provide automatic suggestions to help the user decide on the target folder (Segal and Kephart, 1999) while others provide a rule-based system (Pollock, 1988) or manual methods.

The 'single message view' paradigm is lacking in the fact that it does not include contextual information and ignores the fact that each email message does not exist in a void, but rather forms a larger picture of the individual user's email behavior profile over time. Contextual information is essentially the responsibility of each user, who implicitly has the knowledge such as who is important and which message/task needs priority.

Unfortunately the most popular clients do not currently implement automatic prioritization of individual messages based on past usage. Even the act of grouping messages into individual folders by the user does not address the issue of information overload. Over time most users find that folder topics need to be split into sub-folder topics, consuming valuable user time and potentially allowing messages to 'slip between the cracks'.

7.2 Priority Models

The email inbox has become the 'all in one' unorganized pile for both pending and new tasks. By computing message priority automatically from user behavior we can contribute to making email usage enjoyable once again to the end user. In addition, by prioritizing filtered email, either at the good level or spam level, we can pick out outliers, or false positive examples. We now describe a behavioral model for prioritizing email.

7.2.1 VIP Communication

As mentioned in section 3.2.4 the VIP Communication model compares the average time it takes a user to respond to different correspondents and computes which correspondent the user responds to fastest (i.e. the VIP). One may infer from this analysis the relative importance of individuals based upon the user's response rate. Those to whom the user may respond more quickly are intuitively more likely to be important to the user. Although this might be hard to judge on a per message analysis, the long term behavior over many replies, converges to a very accurate model of importance. Note, that this model does not work well for every single type of email message, as some messages will typically refer to non-email related tasks. For example, an email reminder about a meeting (which will usually not illicit a reply email), those can be addressed with new task based models (Dredze et al., 2006).

We note that within some organization structures, certain individuals might mask their behavior by holding back replies to certain sets of individuals. This might be done to impress upon the recipients that the user is overworked, or to mask the fact that they are available. We premise that this behavior will be consistent over time, and thus directly translate into an ordered list of important users to the recipient. For the rest of the chapter we will make the simplifying assumption that most users reply to messages in the order of importance in which they are received.

7.2.1.1 VIP ranking

To determine the importance of a person that the user exchanges emails with, the average communication time (to send a reply) is not enough. We also need to

consider the number of the emails exchanged between users.

Since people often reply very quickly to some simple emails, yet often take more time to think and respond to more important, complex, and long emails we need to factor that into the calculations. If we only use the communication time to rank the VIP, we tend to see some accounts with very short communication time but also with very few emails exchanged. Usually, we consider the VIP to be the person(s) with whom we communicate both often and quickly. One can dissect the communication behavior by viewing all three numbers: number of emails exchanged, average communication time, and VIP rank.

Thus, we derive our formula to compute the VIP score with respect to some user A:

$$VIP = \frac{\#email}{avg\#email} * f + \frac{avgTime}{time + \frac{avgTime}{10} * (1 - f)} \tag{7.1}$$

Where $\#email$ is the amount of emails exchanged, $avg\#email$ is the average amount of emails over all accounts exchanged with user A. Similarly $time$ is the turn around time to a reply and $avgTime$ to any user, are calculated based on these emails. We use $\frac{avgTime}{10}$as a pseudo number to avoid a small denominator. f is a factor between 0 and 1 that can be adjusted, i.e. weighting the amount of emails versus communication time in determining VIP rank. Larger f results in more weight assigned to the amount of email, and vise-versa. The larger the VIP score, the more important this account is to the selected user. We scale the score from 0-$\mathcal{SR}$ with $\mathcal{SR}$ being the score of the most important user.

7.2.2 Usage Behavior

The Usage behavior model allows comparison between the behaviors of different periods of the same email account. Usually, the comparison is between the behaviors of the current period with that of a previous period which we assume to be normal behavior. Thus, we can calculate whether current behavior differs from the normal behavior and can use it to prioritize an email.

We use a histogram to represent an account's behavior and to make comparisons based on the distance between histograms. We use the weighted Mahalanobis distance function, which is a modified version of the basic Mahalanobis function mentioned in section 3.4.0.4.

7.3 Prioritizing: Combining Usage and VIP Scores

We combine the VIP and usage scores when prioritizing the emails in the user's inbox. Prioritizing is implemented in the message view of EMT. A message view consists of a specific folder or specific set of constraints on the email messages such as specific dates, users, groups of users, etc. EMT's Message view is designed to allow a view to be quickly and seamlessly configured (Figure A.1 screen-shot of message view). A PET profile would interact with an email client to achieve the same results.

Once a specific view is chosen, we can reorder the messages automatically using our prioritizing scheme. The VIP and usage models are kept up to date as new messages enter the email repository. We use those models to calculate the VIP and Usage scores and combine them using the following algorithm:

- Order the messages by time.
- Calculate the sum of VIP
- Reorder messages by VIP score.
- Calculate usage statistics
  - For High VIP, high usage score (unusual) push the priority score higher by α
  - For low VIP, high usage scores, push priority lower by β

The reasoning is that important users, who might send an unusual email (based on past behavior) would be more interesting than a non-important user who sends an unusual email.

The advantage to this scheme is that none of the parameters have to be set by the user; simply using the email client is enough to allow the model to be computed and deployed.

7.4 Message Grouping

An alternative method for reordering messages is clustering or grouping messages. Once groups are created, we can display them as either message summaries or groups of related messages.

For example, currently EMT displays clique groups and summarizes them by most frequent subject words as in Figure A.6.

In general clustering messages, will also include threads of discussion. Many clients which implement threading do so based on subject lines (Thunderbird) or

direct reply association (Gmail). So if a user retypes a subject in a reply, or creates a new message with a reply to a previous message, those client will not correctly associate the two messages. Clustering allows us to group similar messages and find threads of conversation.

7.4.1 Content-Based

In many cases it is useful to analyze individual message content, and use those contents to help group similar-looking messages. In many cases, messages not directly related to individual threads of discussion will appear in a group. This is because email sometimes refers to work which is performed outside of the specific time-frame of the communication exchange. For example, an email might ask a student to perform a certain experiment. The results of the experiment are not necessarily contained within the reply to the original message. In some instances, a period of time has passed since the original email, but we can relate the two emails (or thread) by grouping emails based on content similarity.

7.4.1.1 Subject

In subject grouping, we use an n-gram distance to group messages. Similar to what is described in section 3.4.1.2 we cluster all messages in a view, and present the groups to the user to examine.

7.4.1.2 Body

In body grouping, the cosine distance is calculated over the textual contents of the message. For computation efficiency, the first n bytes can be used without affecting

the message clusters. In general it has been shown that the first 800 bytes are sufficient to profile and group message contents (Hershkop and Stolfo, 2005b; Li et al., 2005). This of course is best used on emails where the new contents are at the beginning, as opposed to users or environments where comments are attached to the end of the email message.

7.4.1.3 Attachment

In attachment grouping, we can either group by attachment name, attachment size, attachment type, or a 1-gram frequency distribution.

7.4.2 Keyword

Keyword grouping allows messages to be grouped based on a set of keywords. Each keyword is weighted by a TF-IDF style weight, which assigns greater weight to less frequent keywords (see section 3.4.1.3). For each message, we can calculate the weight of the matching keywords, and assign it a score. We can then rank-order the scores and combine the scores into groups using a threshold, and present the groups to the user.

Keywords can either be specified by the user, or extracted from a set of messages using basic statistical properties of occurrences. A standard keyword list and the interface to add, remove, load, and save keywords is shown in Figure 7.2. Figure 7.3 shows the wizard display which allows a set of messages to be analyzed and have the keywords extracted. In this view, the common stop words have been extracted before running the analysis.

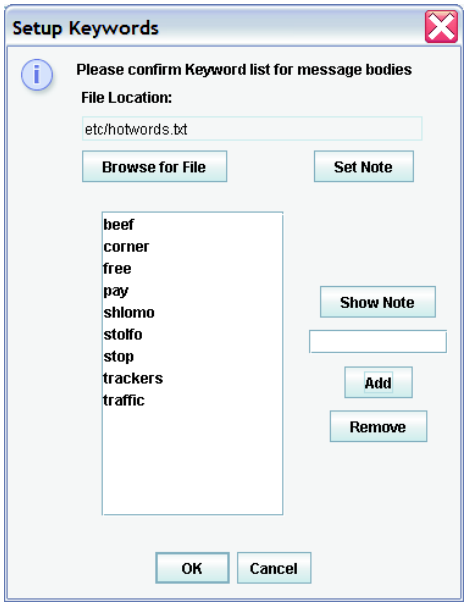

Figure 7.2: Standard keyword list.

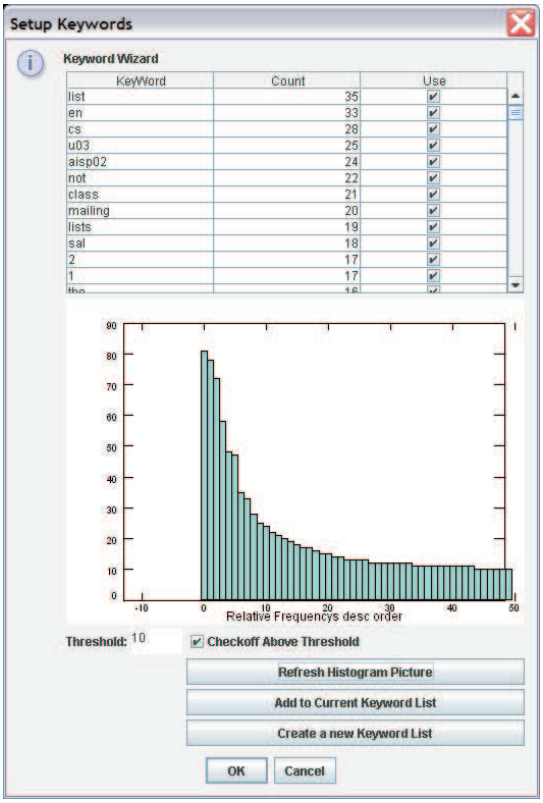

Figure 7.3: Wizard to help extract keywords based on frequencies.

Chapter 8

Evaluation

Throughout the thesis we have introduced and illustrated the data mining and modeling components of our email analysis engine. We now present an overview of some of the tests and results of the components of the system in regards to detecting unwanted emails and prioritizing messages.

- Evaluations of the individual models presented in Chapter 3.
- Evaluation of the combination algorithms presented in Chapter 4.
- Evaluation of SpamAssassin, an open source solution over the same set of data.
- Illustration of message reorganization using principles from Chapter 7.

8.1 Results

8.1.1 Setup

Each of the component classifiers presented in this thesis and embedded in EMT produces a classification output as a score in the range [0-201] with a high number indicating confidence in the prediction that an email is unwanted or spam. We refer to these outputs as the 'raw scores', which we combine through the various correlation functions described in chapter 4.

The training regime requires some explanation. A set of emails are first marked and labeled by the user indicating whether they are spam, or normal. This information can also be gleaned by observing user behavior (whether they delete a message prior to opening it, or move it to a "garbage" or "spam" folder). Although sometime the entire message will be contained in the subject line (example, Meeting canceled!), most users will on average also click to make sure if there is further details in the message body. For our experimental results, users provided their email files with those messages considered spam placed in a special folder. Those we labeled as spam, while all other messages we labeled as normal. These were all messages received, deleted emails were moved to a deleted folder, but not actually deleted.

This data set of real emails was also used to study the model combination methods. Our data set consists of emails collected from five users at Columbia University spanning from 1997 to 2005, a user with a Hotmail account, and a user with a Verizon.net email account. In total we collected 320,000 emails taking up about 2.5 gigabytes of space. Users indicated which emails where spam by moving

Year	Number of Emails
1997	85
1998	687
1999	2355
2000	9553
2001	25780
2002	56021
2003	145291
2004	81176

Table 8.1: The number of emails per calendar year for the main data set in the experiments.

them to specific folders. Table 8.1 indicates the spread of the data over time.

Because current spam levels on the Internet are estimated at 60%, we sampled the set of emails so that we would have a 60% ratio of spam to normal over all our emails. We were left with a corpus of 278,274 emails time-ordered as received by each user.

We tested the models using the 80/20 rule with 80% being the ratio of training to testing. Hence, the first 80% of the ordered email are used to train the component classifiers and the correlation functions, while the following 20% serve as the test data used to plot our results. This set up mimics how such an automatic classification system would be used in practice. As time marches on, emails received are training data used to upgrade classifiers applied to new incoming data. Those new data would be used as training for another round of learning to update the classifiers. Earlier tests used 5-fold cross validation without any statistical difference on the results (of the 1-fold), so we opted to keep it simple with the 1-fold tests.

The data used was pristine and unaltered. No preprocessing was done to

the bodies of the emails with the exception that all text was evaluated in lower case. Headers of the emails were ignored except for subject lines that are used in some of the non-content based classifiers. While adding header data would have improved individual classification, there is much variability in what is seen in the header, and we felt it might over-train and learn some subtle features of tokens only available in the header data present in the Columbia data set. For some of the individual classifiers: Ngram, TF-IDF, PGram, and Text Classifier, we truncated the email parts so that we only used the first 800 bytes of each part of the email attachment. This was used for both efficiency and computational considerations, as there were many large executable attachments in our dataset. In addition the increase in detection was about 10% with the same false positive rates over using full email bodies. The reason is because of noise in the number of tokens seen in very large spam messages.

8.1.2 Features

Traditional machine learning modeling of email has been based on the textual content of email messages. Typically tokens are extracted from the email body (and sometimes header data) and then processed by some machine-learning algorithm.

In prior work, we have proposed and demonstrated how non-content features can be used to profile and separate virus and spam from normal emails (Hershkop and Stolfo, 2005b). The non-content features are specific static features extracted from the email envelope that are not part of the actual message body.

In our behavior classifier using a naïve Bayes algorithm described in section 3.1.3 the set of features extracted is a set of static features including sender and

recipient email names, domain names, and zones (domain ending such as com, edu, etc). In addition, the size of the message, number of recipients, number of attachments, and the MIME-type of the email are used.

8.1.3 Evaluation Measurements

The evaluation of any spam filter has traditionally borrowed heavily from the information retrieval domain. We now define the measurements we have used:

Detection rate - is the total number of spam detected divided by the total number of spam. This number is an indication of the effectiveness of the model to weed out spam messages. The number alone is not a total indication of the effectiveness of a model, as labeling all messages as spam will result in 100% detection but 100% false positive rate.

$$Detection = \frac{tp}{tp + fn} \tag{8.1}$$

False positive rate - represents the number of non-spam emails flagged as spam divided by the total non-spam messages. This number measures the amount of normal emails flagged as spam. The ideal goal would be to have a 100% detection rate, and 0% false positive rate.

False negative rate - is the amount of spam flagged as normal. This serves as an indication of how many messages are slipping through the system.

Error rate - is the percentage of examples that the model has misclassified divided by the total number of examples:

$$\frac{fp + fn}{tp + fp + tn + fn} \tag{8.2}$$

In other words the amount of emails misclassified regardless of the mistakes made by the model.

Cost - was introduced in some of the literature (Hidalgo and Sanz, 2000; Androutsopoulos et al., 2000b) as an important but hard to evaluate factor in spam detection. The feeling is that deleting a good message should cost much higher than simply moving the message into a spam folder for user authorization to delete. In the literature there is a tendency to assign a cost of 1, 9, or 99 as a penalty of misclassifying spam.

Total Cost Ratio is given as:

$$\frac{N_s}{\lambda n_{L \to S} + n_{S \to L}} \tag{8.3}$$

where λ is the cost associated with mis-detection, N is the total emails, $n_{L \to S}$ is the number of misclassified spam and $n_{S \to L}$ is the misclassified normal.

The study of how cost interacts with different types of spam such as unwanted, offensive, security related in relation to different levels of wanted emails is outside the scope of this thesis.

8.1.4 Spam Detection

8.1.4.1 Individual Classifiers

The individual classifiers described in Chapter 3 were evaluated over our email corpus. The following individual classifiers were used:

NBayes : is a Non-Content Naïve Bayes Classifier, that takes behavior features and combines them using Bayes theory.

NGram : is a N-gram based classifier that uses an n length sequences from the body of the email to learn probabilities of individual tokens.

Limited NGram : is the same as NGram but ignores most tokens except those which hash to a specific primary number base. This has the effect of reducing the token space to about 10% of the total space, allowing larger N-gram sizes without the memory overhead associated with large N-grams.

Textclassifier : uses whole words as tokens and learns a probability associated with each token during training.

PGRAM : is a biased text classifier that is described by Paul Graham in his work on spam detection (Graham, 2002). We use the classifier because it has been promoted as a one size fits all solution in some literature and it does perform reasonably well over general text.

TF-IDF : is a classifier using whole words as tokens and calculating the probabilities based on term frequencies divided by document frequencies. This results in making a fair trade between low-occurrence words in shorter documents vs. higher occurring word tokens in larger documents.

URL : is a classifier that analyzes only the URL found in the email body and learns probabilities associated with parts of the URL. For emails without any URL information, a score of 0 (normal) is assigned.

Figure 8.1 shows the performance results of the individual classifiers over the email data set. Of particular interest is that the NBayes non-content, Text Classifier, and PGram classifiers are all very strong classifiers. Table 8.2 displays the detection rates highlighted at certain points to give a sense of how well they compare to each other. The gain is calculated as described in section 4.8. Ideally we would like an individual classifier with high detection and almost no false positive. As can be seen in the graph, the classifier peak at some point in the ROC curve.

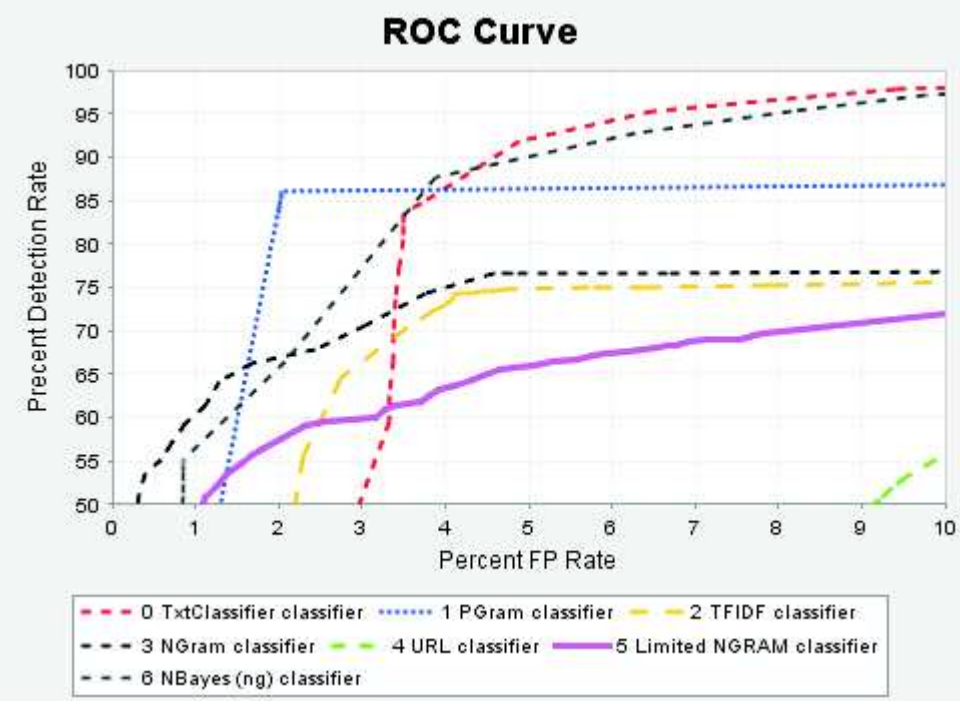

Figure 8.1: Results of individual Spam Classifiers using the full data set and 80/20 evaluation.

Classifier	Detection Rate	False Positive Rate	Gain
NBayes(non-content)	88%	3.8%	79.8%
Ngram	75%	4.0%	72.2%
TextClassifier	90%	5.0%	70.0%
Pgram	90%	5.5%	77.2%
TF-IDF	74%	4.2%	61.5%
Limited Ngram	66%	5.0%	61.4%
URL	55%	10%	32.0%

Table 8.2: Individual Classifier Performance Over Spam Experiments

8.1.5 SpamAssassin Results

In addition to our own spam classifiers, we compare the results of running SpamAssassin version 3.0.2. We ran it using the default rule set (April 2005). Performance is shown in Figure 8.2. Notice when it only sees the first 10% of the data it achieves very high initial accuracy but seems to reach a plateau afterward. This is probably because of the nature of its highly specific rule set that has been highly tuned for the older spam which we are testing over. The plateau indicates that as spam adapts over time, this level will sink unless the rule set is updated in a timely fashion. As can be seen from the full dataset, differentiating between spam and non-spam using only rules generated from outside datasets results in around 70% detection rates.

We should point out that although we tested SpamAssassin for comparison to a well-used solution, in practice it seems to run at higher accuracy with the addition of a sophisticated Bayes engine included with the program. We did not use its engine due to the overhead of recomputing the batch probabilities in our experiments and issues to fine tune the runtime of their Bayes component. We did not have any success in combining SpamAssassin with our other classifiers due to

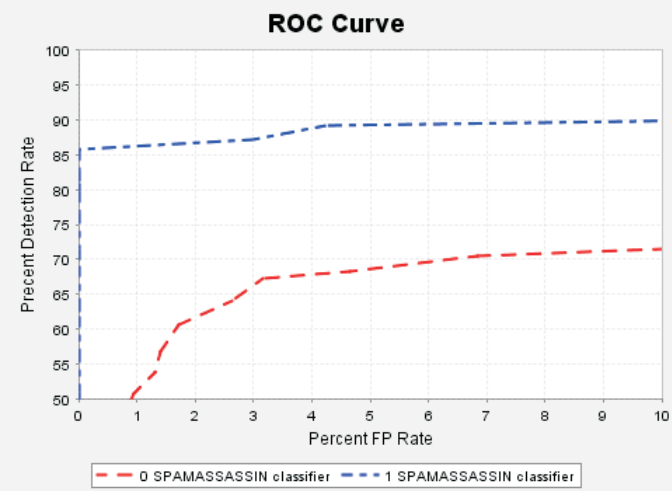

Figure 8.2: Results of SpamAssassin using 10% and the full data set and 80/20 evaluation. The lower accuracy scores (bottom curve) were on the full dataset.

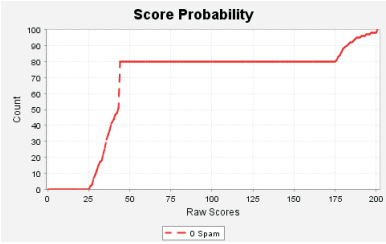

Figure 8.3: Probability curve of SpamAssassin classifier over entire dataset. The x-axis is the score range from 0-201, and the y-axis is the cumulative probability of being spam.

the nature of the scores generated. The score probability graph (Figure 8.3) shows that the scores generated by SpamAssassin do not necessarily fall into an easily divided space of low to high probabilities. For example, when looking at NBayes classifier score range (Figure 8.4), we see a nice curve between good and spam scores.

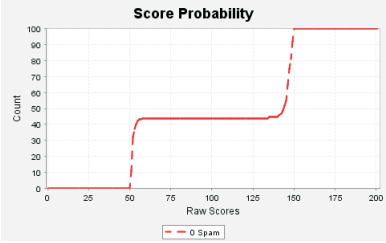

Figure 8.4: Probability curve of naïve Bayes behavior classifier over entire dataset. The x-axis is the score range from 0-201, and the y-axis is the cumulative probability of being spam.

Classifier	Detection Rate	False Positive Rate	Gain
Equal Weights	87%	2.3%	84%
Single Dimension NB	93%	3.6%	85.1%
Judge Combination	99%	0.025%	-
N Dimension	88.7%	2.3%	84.5%
Weighted Majority	85.5%	2.5%	79.9%

Table 8.3: Combination Classifier Performance Over Spam Experiments

8.1.6 Model Combination

Once the component classifiers are applied to the labeled email corpora, the set of model outputs (the classifier raw scores mapped to the range [0-201]) are combined by a specific correlation function. Some of these correlation functions require a training phase to learn the necessary parameters. The component classifiers are tested against their training data and these model outputs are used to train the correlation function.

Since some of the correlation functions require training data, we would like to train the combination algorithm concurrently with the training of the individual classifiers. Although we could have first given each classifier all the training examples and then extracted scores over those examples, we felt that would not

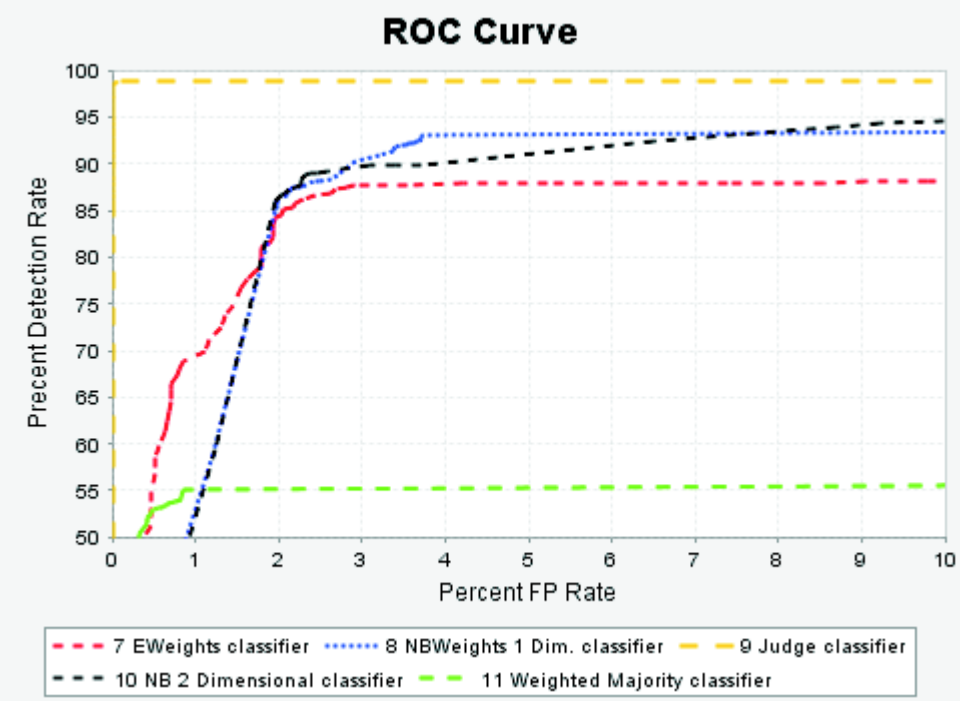

Figure 8.5: Results of combination algorithms using the full data set and 80/20 combining all the individual classifiers.

reflect the real world setting where only partial examples of spam would ordinarily be available, only those seen to date by the user.

To mimic this process, we batched the training data into groups of 1000 examples used to train the classifiers. After each batch, each classifier was executed to generate raw scores for each of the examples seen in the current batch. These scores were then used as input to the model combination algorithms to train the correlation function. We used a batch size of 1000 for efficiency purposes, although any size should be acceptable to achieve comparable results. We realize, however, that individual classifiers will shift individual scores depending upon the amount of training data used. For example, for some classifier, its scoring might be different if we train 5000 emails rather than 1000. However, we preferred to mimic a more

realistic training regimen reflecting how such a system may actually be used as a real application, and thus we believe this batch approach is not unreasonable.

For the experiments we evaluated the following combination algorithms:

EWeights - is a classifier that assigns equal weights to all the individual classifiers.

NBweights - is a classifier that builds a table of the probability that a classification by each classifier is correct. That is, it looks at the classification generated by individual classifiers and then calculates the probability of it being correct based on performance over the training data. The result is mapped to a choice between the maximum or minimum available score for each classifier.

NB 2 dimensional - is a classifier that learns that weighs to assign each classifier by binning the scores and learning probabilities associated with each bin interval.

Weighted Majority - is a classifier that adjusts the weight of each classifier based on the performance during training. For those classifiers which correctly identify an unknown example, the classifier assigns a larger weight to their vote, while subtracting weight from those classifiers which make mistakes.

Judge - is the classifier which chooses the maximum available score or minimum given the additional information of knowing which class the example should be in. This classifier is for testing purposes to evaluate how well the combination algorithms are performing.

Figure 8.5 shows the results of combining the classifiers. Table 8.3 has the combination algorithms and the gains achieved in each case. Notice that false

positives have been reduced by about 3% and detection improved by about 4% over the best individual classifiers. This is also reflected in about a 15% improvement in the gain measurement showing clear advantage of filtering with multiple classifiers.

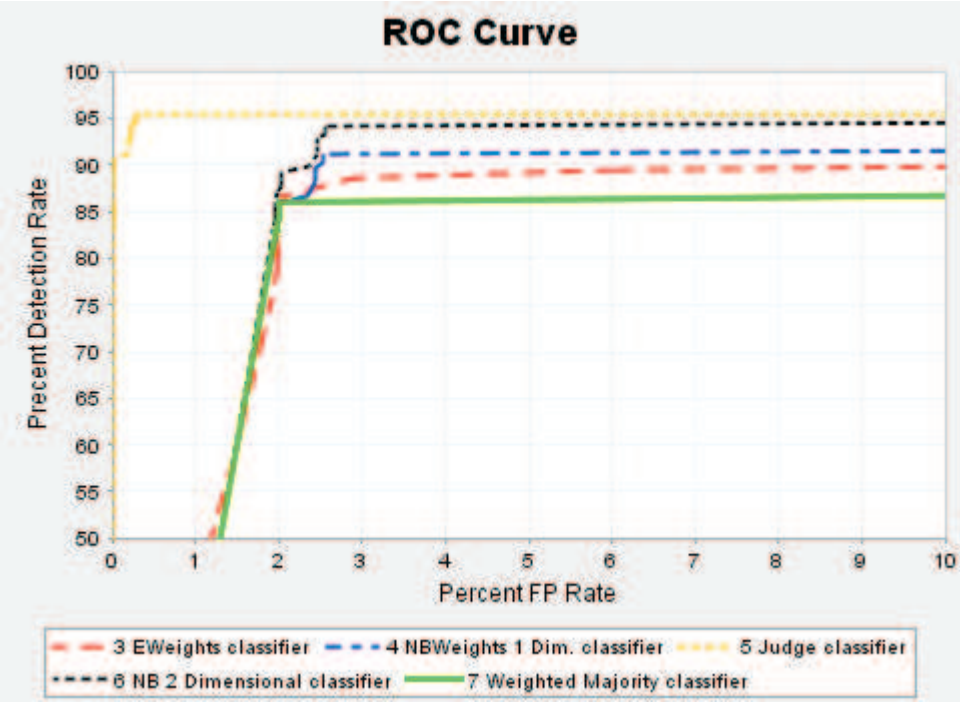

Figure 8.6: Results of combining 3 strongest algorithms.

We next compare the merit of only combining strong or weak classifiers. In Figure 8.6 we combined the three strongest algorithms, namely non-content, text classifier, and PGram. Notice that approximately a 2% false positive reduction is achieved over the strongest component classifier.

We compare the combination of Ngram, URL, Limited Ngram in Figure 8.7. Surprisingly the weighted majority and NB1 are almost the same here. Since TF-IDF on a full email body has a very low detection rate, we tried a combination of URL, TF-IDF(full), and Limited Ngram in Figure 8.8. Although there is a negligible improvement in false positive rate, there is a very strong detection improvement

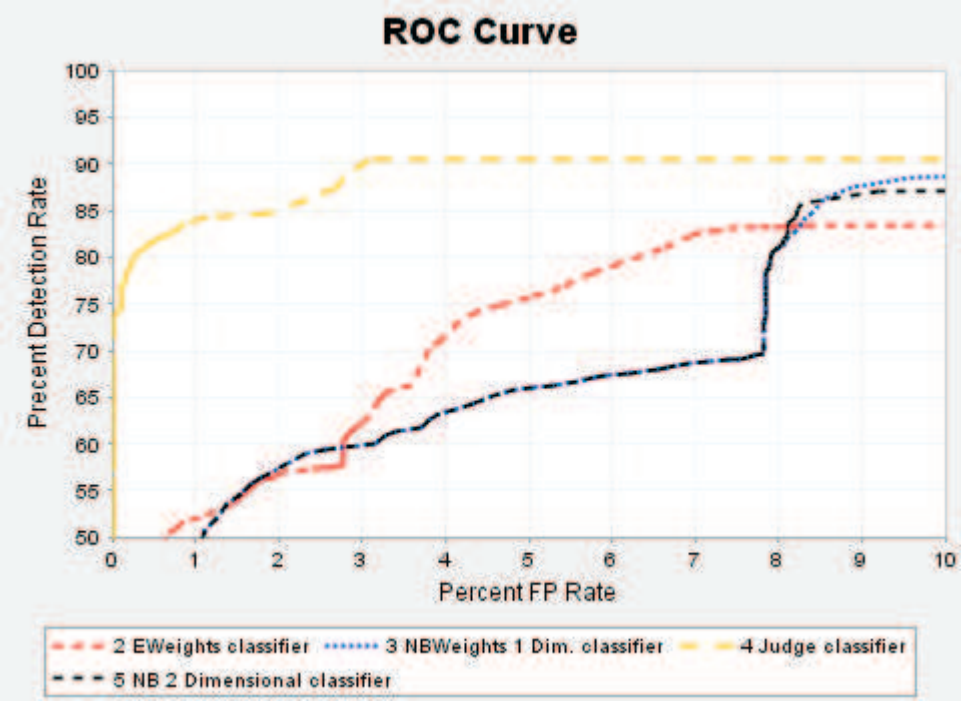

Figure 8.7: Results of combining Ngram and Ngram-Limited.

of about 10%.

In Figure 8.7 we highlight what happens when combining two similar classifiers. Notice that the performance of the Judge algorithm has been significantly reduced, as expected. This confirms that the individual classification errors strongly overlap, thus the maximum combination is also lower.

The TF-IDF (full body) classifier was a surprisingly poor performer and thus represents a weak classifier combination. We showcase what happens to the TF-IDF algorithm as it is exposed to less of the email parts. The improvements are going from full, to first 800, to first 400 bytes and compared to NBayes in Figure 8.9. We also show the combination when we combine the best and worst classifiers in Figure 8.10. Notice that the Judge is not returning the theoretical limit and the reason is that TF-IDF does not return a confidence score, but rather a distance

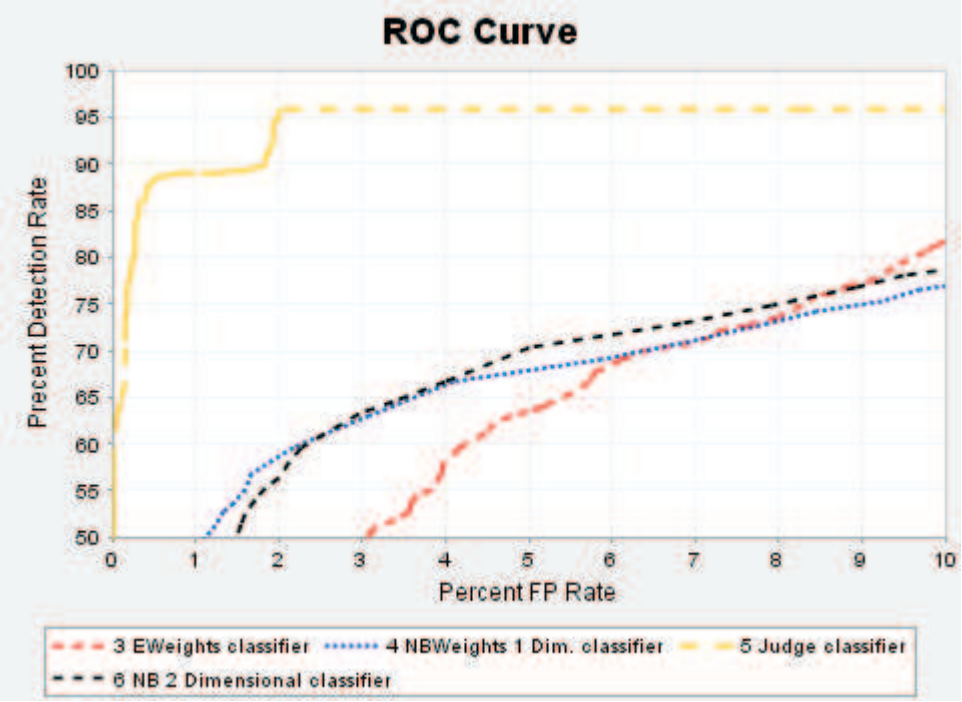

Figure 8.8: Results of combination of URL, TF-IDF(full), and Limited Ngram.

metric to some document centroid. In addition because it is being trained on the entire body, the algorithm is being overwhelmed by noisy token probabilities.

Because of this, the scores here are not easily combined by a simple combining scheme. On the other hand, the Naïve Bayes combinations are mapping the scores to a probability space, where they can be interpreted as a confidence value. This is also the reason that it has been found that combining confidence scores works better than combining binary classification (Kittler et al., 1998). There is inherently more information in such cases. The same is true with the SpamAssassin results. Figure 8.3 displays the probability curve for SpamAssassin vs. naïve Bayes probability curve shown in Figure 8.4.

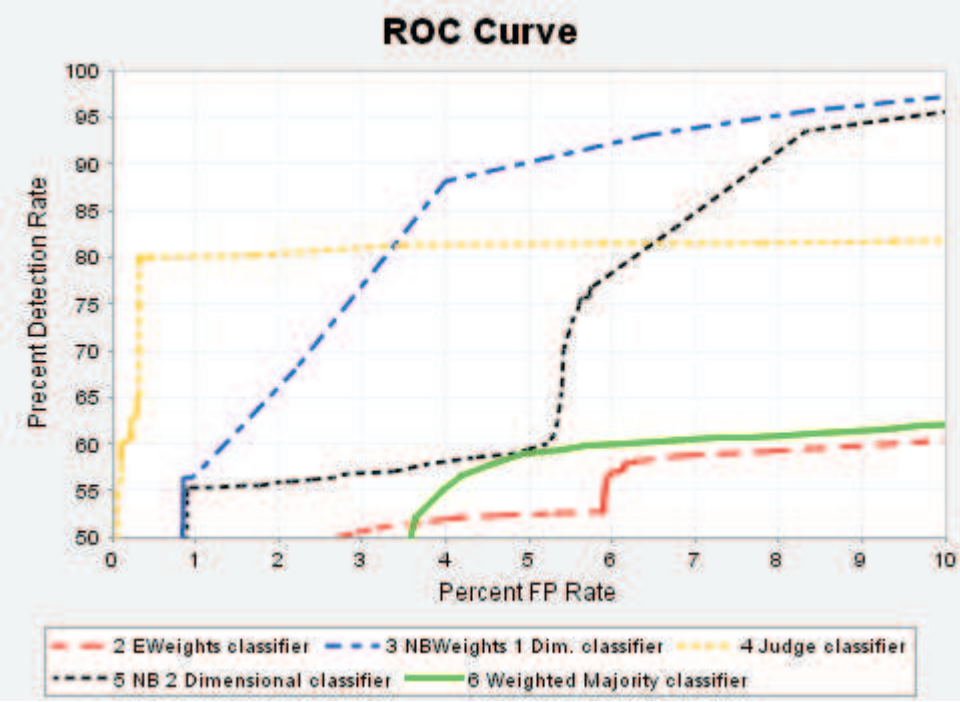

Figure 8.9: Results of exposing TF-IDF to different body lengths.

8.1.7 Prioritization

To test a practical application of the algorithm on a large data set, we applied the algorithm to our set of pre-labeled data from our email corpus. The goal was to detect any mislabeled spam messages using the priority algorithm to separate and cluster groups of related spam messages. We prioritized one spam folder, and found that there was a group of messages which had been mislabeled, and mistakenly put into the spam folder (they actually were messages from an anti-spam discussion group). Manually digging through all the messages would not have been practical due to the size of the dataset.

In the next section we analyze these results and draw some conclusion and thoughts for future directions.

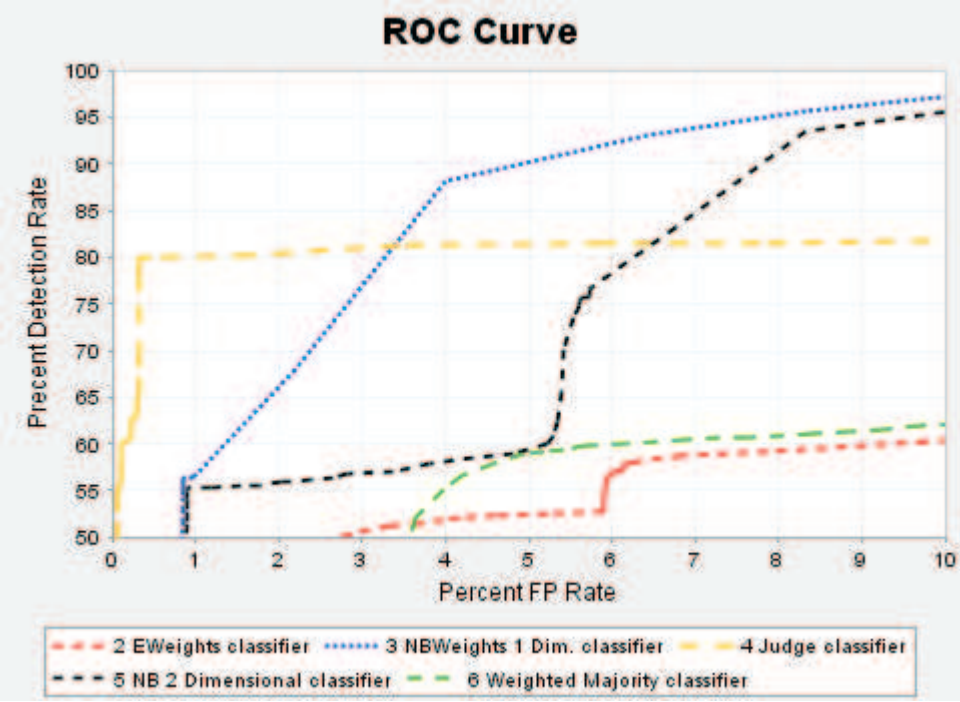

Figure 8.10: Results of combining Non-content Naïve Bayes and TF-IDF (full body).

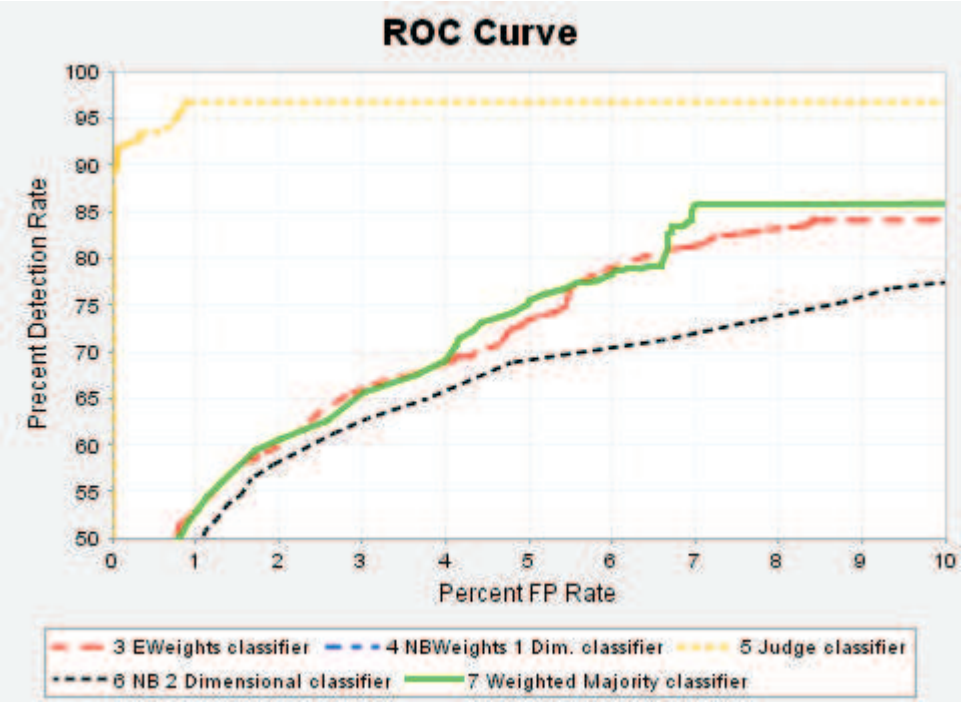

Figure 8.11: Results of combining weak classifiers.

Chapter 9

Conclusion

The research presented in this thesis identifies shortcomings of the current email systems and offers implemented solutions to help address these weaknesses. In the course of producing this thesis, we have implemented a data mining framework to mine many useful features that are present in the underlying email data.

In this chapter, we discuss the results of the previous chapter and summarize the main research contributions of this work. We then overview the state of the current implementation, list the main limitations of this approach, discuss future research directions and conclude the thesis with implications of this work on the field of data mining and email analysis as a whole.

9.1 Discussion of Results

For the task of spam detection our evaluation has shown that although individual spam classifiers perform very well, they each have different areas of expertise. This is not surprising as the features extracted were different in most of the classifiers. In

addition, the numbers reflect the real world email universe in a university setting, where the types and amount of emails vary widely from day to day and semester to semester.

Our research shows that unwanted messages can be detected by considering the behavior of the email user. We have investigated several particular model combination algorithms and plotted their performance using a number of supervised machine learning classifiers. We inspected several of the emails that were persistently misclassified by all the combinations. These peculiar emails comprised messages with only graphics attached, flaming debates on some news groups, and some empty emails that had no bodies (their message was essentially their subject line). We believe that much lower false positive rates can be achieved on this data set by including features extracted from the email header data, where patterns such as the type of email client, or Internet section of the IP class can shed more evidence on the spaminess of a message.

In summary the results of our experiments have shown how combination algorithms can be directly applied to spam classifiers, and used to improve both false positive and detection rates on either strong, weak, or a combination of these classifiers. We rely on the gain formula to help choose a specific algorithm. The improvements are most effective when the component classifiers have been trained on different feature sets, similar to results achieved in other fields (Tan and Jin, 2004; Tax and Duin, 2001). What is surprising is the robustness of the equal weight classifier for combining two classifiers in our experiments. We do achieve better results when combining more classifiers, but intuitively equal weights shows surprising robustness. We attribute it to the smoothness of the probability graphs

of the individual classifiers. That is, they somewhat approximate a confidence in the spam scores.

A very strong reason to work with combination algorithm is something learned from the computer security domain: resistance to attack. Concentrating on only a single classifier, to fine tune it to perfection, will only encourage spammers to try to, and eventually defeat, that specific mechanism. By using a combination algorithm, an email enclave can be protected against compromise of any single Spam classifier. This is highlighted in Figure 8.10 where the scores of the TF-IDF (over full body) can be thought of as being compromised in some fashion so that they do not reflect a true confidence. The combination algorithms that are based on remapping probabilities are remarkably resilient on the face of this kind of attack. In reality, a simple equal weights algorithm can be used, and a second probability algorithm run along side as a reality check on the performance of the classifier. Having automatic checks and balances is one way in which spam classification can gain user confidence by displaying a confidence metrics, without requiring the user to trudge through mounds of spam in the spam folder searching for misclassified examples.

The results achieved indicate that the implemented models in EMT and PET provide a fairly robust and accurate system for a variety of analysis tasks. These concepts are applicable to a far wider range of problems, including virus detection, security policy violations, and a host of other detection tasks (Stolfo et al., 2003b; Stolfo et al., 2003c). It is important to note that testing EMT in a laboratory environment only suggests what its performance may be on specific tasks and source material. The behavior models are naturally specific to a site or

particular account(s) and thus performance will vary depending upon the quality of data available for modeling, and the parameter settings and thresholds employed. Although the model combination functions work well in these test cases, there are still other analytical tasks that should be explored to determine how to fully automate a model combination and correlation function feature. In the case of the two-class spam detection problem, the methodology is straightforward. This may not be the case for multi-class learning problems.

9.2 Contributions

This research makes five main contributions to the state of the art in research in data mining and email analysis:

Email behavior models - We present new models to represent email usage. These models represent past usage patterns and calculate current pattern deviations using standard metrics. The types of models and pattern calculations are described in Chapter 3. Our research shows that unwanted messages can be detected by considering only the behavior of the email user.

The analysis of email from a user, enclave, and attachment view provide a rich set of information to improve the email experience.

Framework for mining email data - We present a database back-end to store and analyze email data. The advantage of the database file system is that it is both fast and scalable. It allows individual features for statistical analysis to be quickly and easily calculated without having to process all the data sequentially with a custom built application. In addition it allows an EMT

like system to be placed either on a per account basis or a system wide basis using the same underlying technology. A similar database back-end was also developed as an architecture component in analyzing IDS attack data in a real time system (Lee et al., 2001).

Behavior Profiles - We develop the concept of a behavior data structure that stores the user behavior in a compact representation. The representation not only allows behavior comparisons to be done efficiently, but also allows a user to port their behavior profiles between machines and accounts.

Users are used to being able to port their individual contact lists and email data between applications, it should not be different for their usage patterns. By encoding the data in a portable form, we allow the behavior to help protect the user data to any client side application. For example adding it as a secondary security layer to a typical user-name and password authentication.

Automatic message prioritization - We introduce a novel scheme to prioritize messages in an email collection based on past behavior in Chapter 7. The user's own behavior can be used as an indication for which messages and users are more important and hence PET reorders the list of emails based on this criteria.

Allowing the behavior model to adopt as the user changes habits is a powerful addition to the email experience, especially in defining what is important. Context evaluation by the user, is currently a manual subconscious process when communicating through email.

Spam classifier model combinations We have developed a novel scheme for

combining various spam classifiers to achieve higher detection and lower false positive rates. The advantage of combining classifiers is to allow a layer of security over the filters so that zero-day exploits can be mitigated through multiple classifiers.

9.3 Software System

Aside from the theoretical contributions, this thesis contributes to the state of the art by making several tools available to the research community. PET is a fully implemented system, allowing components to be developed for producing behavior models of an individual or group of email accounts. In addition, the functionality to compare behavior patterns over time and message clustering over non-content features were developed. We enumerate some of the software prototypes that this thesis has generated.

- `Java Email Parser` - The email parser developed for EMT is a very useful tool on its own. There are no open sourced, parsers written in Java to parse through email archive files in MBOX formats. Java-Mail is a Java package from Sun Microsystems that can interface to a specific type of email server and retrieve email messages that the user can store and manipulate locally. Unfortunately this package cannot read a mbox format file but must fetch the messages from some email server configured in a specific way. In many cases, users do not have access to an email server to host old messages. Usually they will have a collection of emails they would like access in some standard way.

The Java parser bundled with EMT can process mbox and outlook emails, and either move them to an EMTstyle database or can be extended to work with a user specified data formats or email collections.

- EMT - The email mining toolkit has been developed over time to allow users to analyze large amounts of offline emails. The behavioral models developed for EMT each illustrate an important behavior analysis of users, groups, and attachments in an email enclave. The combination of individual models allows us to leverage multiple views of the underlying behavior data to quickly highlight unusual or important email flow information. EMT has been tested by and released to a host of users and agencies who are collaborating on specific features and extensions for EMT for general use.

- PET - The PET toolkit is a bundled toolkit to allow a standard email client to be extended to use behavior models. It can interact with standard email clients (we implemented a proof of concept using Mozilla's Thunderbird) to allow the behavior patterns to be gathered in a real time environment and displayed to the user.

- EMT Forensics - We have extended EMT to allow a forensic analyst to quickly locate interesting email by tying together the different disparate behavior models into one window location. The usefulness of the system is not limited to forensic investigators, but also to an automatic system to detect account misuse or abuse.

9.4 Limitations and Future Work

The current email system is highly reliable in routing and delivering messages from point to point across the Internet. It still lacks basic authentication and security features such as sender verification, default message signing, and message delivery receipt. Some of these are being standardized by different organizations with many different agendas and goals driving their implementations. Many of the problems being faced by email users today were foreseen by Denning (Denning, 1982) in 1982. This thesis contributes to solving some of the issues related to ease of use, message prioritization, and information organization related to the email experience.

It has been observed that email has evolved to be many things to many people. Because of this, automatic tools to augment the email experience will need to be developed to allow users to enjoy using email without being overloaded by the experience. At the same time as with any other emerging technology base, it will require user education to keep ahead of the various fraudulent schemes (Balvanz et al., 2004).

One way to extend the priority algorithm presented in this thesis is by adding more features to allow it to fine tune itself to the user's personal preferences. For example, we currently do not take into account the length of emails when calculating the VIP score. In addition, the behavior features are not tuned so that very old data is moved out of the model. Allowing size and other features to play a role will undoubtedly make the algorithm more accurate.

In addition, research on task extraction (Dredze et al., 2006; Gwizdka, 2002; Huang et al., 2004) has shown promise with the possibility of trying to extract basic tasks from the email messages. Allowing task goals to be factored into the

priority scheme will allow pending items to be put ahead of already accomplished tasks. For example, studies have shown that many people email themselves notes with links, URLs, or reminders. Placing those at the top of the priority list would make more sense.

There has been research on how to generate automatic email responses based on past responses (Scheffer, 2004; Bickel and Scheffer, 2004). Although very promising the techniques have only been so far applied at trying to match previously manually answered emails with new incoming ones, and select the most appropriate reply. It would interesting to apply similarity measures presented in the thesis to the answering domain.

A problem we faced when trying to test out new ideas dealing with email systems, was an inherent limitation of the available data. Because we only have access to our own data, our results and experiments no doubt reflect some bias towards our university environment. Much of the work published in the spam detection domain also suffers from the fact that it tries to reach general conclusions using very small data sets collected on a local scale. Ultimately it will require either an organization with large amounts of email users and access to large amounts of email data to refine the ideas presented in the thesis or an effort by volunteer users to help push forward the ideas presented here using some type of shared data sets respecting individual privacy of the data. Some have begun to take notice, and study very large amounts of spam email traffic to extract other useful features (Gomes et al., 2004; Jung and Sit, 2004; Hulten et al., 2004; Klimt and Yang, 2004).

EMT has much potential for specific email related tasks such as forensic

studies and email search and the application of behavior modeling in other domains. For example, although we have touched the surface of virus detection, prior publications of using behavior models to catch virii have shown great promise (Stolfo et al., 2003c). In addition, the type of analysis of the behavior of the user can be directly applied to file types. Studying the behavior of file types might lead to better virus detection.

The behavior models presented by this thesis can also be used to analyze network behavior, when analyzing network information flows. Clique analysis has been recently used at Columbia to both help find compromised hosts and attempt to detect zero day worms (Sidiroglou and Keromytis, 2005). We hope to extend some of the newer work of PET to this domain too.

In addition a straightforward analysis of instant message traffic can be analyzed by the models presented in this thesis. We hope to be able to conduct a study on some type of electronic messaging traffic in the near future. Preliminary work was attempted but insufficient data has been a problem in the past.

The VIP ranking algorithm can be extended with other features, such as including the length of a message over time into the formula. The value of automatically ranking importance can be applied to many other areas related to electronic communication. With electronic devices playing multiple roles in every day situations, organizing the attention of the user, and knowing when something is important, is literally important. You do not want to bother the user during a critical time, to let them know their gas bill is due. New proposals on how to actually grab a user's attention has been studied with mixed results (Zhang et al., 2005) there is much to be done in this area.

Many companies have become involved in research that has started to extend the basic email experience. For example Microsoft's "Lifebits" tries to place email communication in the context of other forms of electronic activity. Google's "Gmail" has allowed end users to keep all their old emails, by being the leader in storage capacity and introducing new email features on a continuous basis. IBM has worked on task extraction, task management, and client synchronization, all deployed as features in their Lotus email client. Mozilla's "Thunderbird" is designed around allowing users to add multiple extensions easily into the system with minimum overhead. All these systems now incorporate one or more forms of email spam filtering technology, although with slightly different accuracy measurements and qualifications. Each of these products represents an email base with many thousands of users. Email's evolution is a reflection on how the base users have traditionally perceived its functionality. Email has moved from a novelty application or electronic memo, towards a personal communication medium. New forms of analysis and features will help email move beyond plain messages integrating itself into the users electronic experience.

Other data mining techniques to study email have started to emerge as researchers try to use email data as a seeding ground for knowledge discovery (Aery and Chakravarthy, 2004; Dredze et al., 2006). Some of the machine learning theory problems discussed in this thesis are part of a larger open problem set in the machine learning domain. Especially the "False Positive" problem, which in the context of spam detection we have used multiple models to effectively try to minimize the false positive rates. This issue is a general machine learning problem, and any solution will have wide reaching implications for furthering the state of the art. We hope to

pursue research on this problem in the near future.

Our experience on working on EMT has shown us the importance of combining both a clean GUI design along with the ability to perform sophisticated analysis, to create a useful tool. Finding the balance between information helpfulness and information overload is a critical task especially when tasked to help the user focus on what is the most important aspect of email: a personal communication tool.

References

Aery, M. and Chakravarthy, S. (2004). eMailSift: mining-based approaches to email classification. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 580–581, New York, NY, USA. ACM Press.

AHBL (2004). The abusive hosts blocking list. http://www.ahbl.org.

Ahmed, S. and Mithun, F. (2004). Word stemming to enhance spam filtering. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*. Available: http://www.ceas.cc/papers-2004/167.pdf.

Allman, E. (2003). Spam, spam, spam, spam, spam, the ftc, and spam. *Queue*, 1(6):62–69.

Androutsopoulos, I., Koutsias, J., Chandrinos, K., Paliouras, G., and Spyropoulos, C. (2000a). An evaluation of naive bayesian anti-spam filtering. pages 9–17.

Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., and Spyropoulos, C. D. (2000b). An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In Belkin, N. J., Ingwersen, P., and Leong, M.-K., editors, *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 160–167.

Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C., and Stamatopoulos, P. (2000c). Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In Zaragoza, H.,

Gallinari, P., , and Rajman, M., editors, *Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000)*, pages 1–13, Lyon, France. Available: http://arXiv.org/abs/cs/0009009.

Aris, A., Gemmell, J., and Lueder, R. (2004). Exploiting location and time for photo search and storytelling in mylifebits. Technical Report MSR-TR-2004-102, Microsoft Research.

Asaravala, A. (2004). New spam filters cut the noise. *Wired News*.

Asker, L. and Maclin, R. (1997). Ensembles as a sequence of classifiers. In *15th International Joint Conference on Artificial Intelligence*, pages 860–865, Nagoya, Japan.

Balvanz, J., Paulsen, D., and Struss, J. (2004). Spam software evaluation, training, and support: fighting back to reclaim the email inbox. In *SIGUCCS '04: Proceedings of the 32nd annual ACM SIGUCCS conference on User services*, pages 385–387, New York, NY, USA. ACM Press.

Bhattacharyya, M., Hershkop, S., Eskin, E., and Stolfo, S. J. (2002). MET: An experimental system for malicious email tracking. In *New Security Paradigms Workshop (NSPW-2002)*, Virginia Beach, VA.

Bickel, S. and Scheffer, T. (2004). Learning from message pairs for automatic email answering. In *ECML*, pages 87–98.

Bilmes, J. A. and Kirchhoff, K. (2000). Directed graphical models of classifier

combination: application to phone recognition. In *ICSLP-2000*, volume 3, page 921.

Boardman, R., Sasse, M. A., and Spence, B. (2002). Life beyond the mailbox: A cross-tool perspective on personal information management. In *CSCW 2002 Workshop: Redesigning Email for the 21st Century*.

Bondedsender.com (2005). Bonded sender program. http://www.bondedsender.com/.

Boyd, D., Potter, J., and Viegas, F. Fragmentation of identity through structural holes in email.

Bron, C. and Kerbosch, J. (1973). Algorithm 457: Finding all cliques of an undirected graph. *Comm ACM*, 16:575–577.

Cadiz, J., Dabbish, L., Gupta, A., and Venolia, G. D. (2001). Supporting email workflow. Technical Report MSR-TR-2001-88, Microsoft Research.

Cahill, M. H., Lambert, D., Pinheiro, J. C., and Sun, D. X. (2000). Detecting fraud in the real world. Technical report, Bell Labs, Lucent Technologies.

Carreras, X. and Márquez, L. (2001). Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, BG.

CDT (2003). Why am i getting all this spam. Technical report, Center for Democracy & Technology.

Chakravarti, I. M., Laha, R. G., and Roy, J. (1967). *Handbook of Methods of Applied Statistics*. John Wiley and Sons.

Chan, P. K. and Stolfo, S. J. (1998). Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Knowledge Discovery and Data Mining*, pages 164–168.

Clemen, R. T. (1989). Combining forecasts: A revew and annotated bibliography. *International Journal of Forecasting*, 5:559–583.

Cohen, A., Mantegna, R. N., and Havlin, S. (1996). Can zipf analyses and entropy distinguish between artificial and natural language texts? Technical report.

Cohen, W. (1996). Learning rules that classify e-mail. In *Machine Learning in Information Access: AAAI Spring Symposium (SS-96-05)*, pages 18–25.

Corporate Email Systems (2002). Spamcop. http://www.spamcop.net.

Cranor, L. F. and LaMacchia, B. A. (1998). Spam! *Communications of the ACM*, 41(8):74–83.

Crawford, E., Kay, J., and McCreath, E. (2001). Automatic induction of rules for e-mail classification. In *Sixth Australian Document Computing Symposium*, Coffs Harbour, Australia.

Crocker, D. H. (1982). Standard for the format of arpa internet text messages. Technical Report RFC 822, IETF.

Culotta, A., Bekkerman, R., and McCallum, A. (2004). Extracting social networks

and contact information from email and the web. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*.

Dai, R. and Li, K. (2004). Shall we stop all unsolicited email messages? In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*. Available: http://www.ceas.cc/papers-2004/189.pdf.

Damashek, M. (1995). Gauging similarity via n-grams: Language-independant sorting, categorization and retrieval of text. *Science*, 267:843–848.

Denning, P. J. (1982). Electronic junk. *Communication of the ACM*, 25(3):163–165.

Derby (2005). http://incubator.apache.org/derby/.

Diao, Y., Lu, H., and Wu, D. (2000). A comparative study of classification-based personal e-mail filtering. In *4th Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD'00)*, pages 408–419, Kyoto, JP.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15.

DNSRBL (2005). http://www.dnsrbl.org.

Dredze, M., Lau, T., and Kushmerick, N. (2006). Automatically classifying emails into activities. In *To appear: IUI06*.

Drucker, H., Vapnik, V., and Wu, D. (1999). Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054.

Ducheneaut, N. and Bellotti, V. (2001). E-mail as habitat: an exploration of embedded personal information management. *interactions*, 8(5):30–38.

Duda, R. and Hart, P. (1973). Pattern classification and scene analysis. *Journal of Documentation*, 35:285–295.

Dwork, C., Goldberg, A., and Naor, M. (2003). On memory-bound functions for fighting spam. In *Proceedings of the 23rd Annual International Cryptology Conference (CRYPTO 2003)*. Available: http://www.wisdom.weizmann.ac.il/ naor/PAPERS/mem.pdf.

Eide, K. (2003). Winning the war on spam: Comparison of bayesian spam filters. Available: http://home.dataparty.no/kristian/reviews/bayesian/.

Fahlman, S. (2002). Selling interrupt rights: A way to control unwanted email and telephone calls. *IBM Systems Journal*, 41(4):759–766. Available: http://www.research.ibm.com/journal/sj/414/forum.pdf.

Fallows, D. (2003). Spam: How it is hurting email and degrading life on the internet.

Fallows, D. (2005). Can-spam a year later.

Fawcett, T. (2003). 'in vivo' spam filtering: A challenge problem for data mining. *KDD Explorations*, 5(2).

Fawcett, T., Haimowitz, I., Provost, F., and Stolfo, S. (1998). Ai approaches to fraud detection and risk management. *AI Magazine*, 19.

Fawcett, T. and Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*.

Fix, E. and Hodges, J. L. (1952). Discriminatory analysis: nonparametric discrimination: small sample performance. Technical Report No. 11. Project No. 21-49-004, USAF School of Aviation Medicine.

Freeman, E. and Gelernter, D. (1996). Lifestreams: A storage model for personal data. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(1).

Fu, Y. (2003). discovery of working activities by email analysis. In *IEEE-ACM Joint Conference on Digital Libraries Workshop on Information Visualization Interface for Retrieval and Analysis (IVIRA)*.

Gabber, E., Jakobsson, M., Matias, Y., and Mayer, A. J. (1998). Curbing junk e-mail via secure classification. *Financial Cryptography*, pages 198–213.

Gaudin, S. (2004). Record broken: 82% of u.s. email is spam. *internetnews.com*.

Gburzynski, P. and Maitan, J. (2004). Fighting the spam wars: A remailer approach with restrictive aliasing. *ACM Trans. Inter. Tech.*, 4(1):1–30.

Gee, K. R. (2003). Using latent semantic indexing to filter spam. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 460–464, New York, NY, USA. ACM Press.

Gemmell, J., Lueder, R., and Bell, G. (2003). The mylifebits lifetime store. In *ACM SIGMM 2003 Workshop on Experiential Telepresence (ETP 2003)*.

Gemmell, J., Williams, L., Wood, K., Bell, G., and Lueder, R. (2004). Passive capture and ensuing issues for a personal lifetime store. In *First ACM Workshop*

on Continuous Archival and Retrieval of Personal Experiences (CARPE '04), pages 48–55.

Gomes, L. H., Cazita, C., Almeida, J. M., Almeida, V., and Wagner Meira, J. (2004). Characterizing a spam traffic. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 356–369, New York, NY, USA. ACM Press.

GoodMail.com (2003). Goodmail systems - restoring trust in email.

Goodman, J. T. and Rounthwaite, R. (2004). Stopping outgoing spam. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 30–39, New York, NY, USA. ACM Press.

Graham, P. (2002). A plan for spam. Technical report. Available: http://www.paulgraham.com/spam.html.

Gray, A. and Haahr, M. (2004). Personalised, collaborative spam filtering. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS).*

Gross, B. M. (2002). Personal digital libraries of email. In *RCDL 2002*, Dubna.

Gruen, D., Sidner, C., Boettner, C., and Rich, C. (1999). A collaborative assistant for email. In *CHI '99: CHI '99 extended abstracts on Human factors in computing systems*, pages 196–197, New York, NY, USA. ACM Press.

Gwizdka, J. (2002). Reinventing the inbox: supporting the management of pending tasks in email. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems*, pages 550–551, New York, NY, USA. ACM Press.

Hall, R. J. (1999). A countermeasure to duplicate-detecting anti-spam techniques. Technical report, ATT Lab.

Hallam-Baker, P. (2003). A plan for no spam. Technical report, Verisign.

Han, E.-H., Karypis, G., and Kumar, V. (2001). Text categorization using weight adjusted k-nearest neighbor classification. In *PAKDD*, pages 53–65.

HashCash.org (2002). "hashcash - a denial of service counter-measure".

Helfman, J. and Isbell, C. (1995). Ishmail: Immediate identification of important information. Technical report, AT&T Labs.

Hershkop, S. (2004). Behavior based spam detection. In *Proceedings of the Spam Conference*. Available: http://www1.cs.columbia.edu/ sh553/MITSPAM04/.

Hershkop, S. and Stolfo, S. J. (2005a). Combining email models for false positive reduction. In *Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*, Chicago, IL.

Hershkop, S. and Stolfo, S. J. (2005b). Identifying spam without peeking at the contents. *ACM Crossroads*, 11.

Hidalgo, J. M. G. and Sanz, E. P. (2000). Combining text and heuristics for cost-sensitive spam filtering. In *Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop*, Lisbon.

Ho, T. K., Hull, J. J., and Srihari, S. N. (1994). Decision combination in mutiple classifier systems. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, 16(1):66–75.

Hollmen, J. (1999). *Probabilistic Approaches to Fraud Detection*. PhD thesis, helsinki University of Technology.

Holt, J. D. and Chung, S. M. (2001). Miltipass algorithms for mining association rules in text databases. In *Knowledge and Information Systems*.

Horvitz, E., Jacobs, A., and Hovel, D. (1999). Attention-sensitive alerting. In *Proceedings of UAI '99, Conference on Uncertainty and Artificial Intelligence*, pages 305–313.

Huang, Y., Govindaraju, D., Mitchell, T., de Carvalho, V. R., and Cohen, W. (2004). Inferring ongoing activities of workstation users by clustering email. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*. Available: http://www.ceas.cc/papers-2004/149.pdf.

Hulten, G., Penta, A., Seshadrinathan, G., and Mishra, M. (2004). Trends in spam products and methods. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*. Available: http://www.ceas.cc/papers-2004/165.pdf.

IBM (2005). Lotus notes. http://www-306.ibm.com/software/lotus/.

IDG.net (2001). Email mailboxes to increase to 1.2 billion worldwide by 2005. Technical Report DC #W25335.

Ioannidis, J. (2003). Fighting spam by encapsulating policy in email addresses. In *Proceedings of the Symposium of Network and Distrib-*

uted Systems Security (NDSS), San Diego, California. Available: http://www.isoc.org/isoc/conferences/ndss/03/proceedings/papers/1.pdf.

Itskevitch, J. Automatic hierarchical e-mail classification using association rules. Master's thesis, Simon Fraser University.

Iwanaga, M., Tabata, T., and Sakurai, K. (2004). Some fitting of naive bayesian spam filtering for japanese environment. In *WISA*, pages 135–143.

Jeffrey O. Kephart, D. M. C. and White, S. R. (1993). Computers and epidemiology. *IEEE Spectrum*.

John, G. and Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345.

Johnson, S. (2003). Pass your e-mail through some new software and the answer will become obvious. *DISCOVER*, 24.

Jung, J. and Sit, E. (2004). An empirical study of spam traffic and the use of dns black lists. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 370–375, New York, NY, USA. ACM Press.

Keim, D. A., Mansmann, F., and Schreck, T. (2005). MailSOM - visual exploration of electronic mail archives using self-organizing maps. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*. Available: http://www.ceas.cc/papers-2005/135.pdf.

Kerr, B. (2003). Thread arcs: An email thread visualization. In *2003 IEEE Symposium on Information Visualization*, Seattle, Washington.

Kerr, B. and Wilcox, E. (2004). Designing remail: Reinventing the email client through innovation and integration. Technical Report RC23127, IBM.

Kittler, J. and Alkoot, F. M. (2003). Sum versus vote fusion in multiple classifier systems. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, 25(1).

Kittler, J., Hatef, M., Duin, R. P., and Matas, J. (1998). On combining classifiers. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, 20(3).

Kleinberg, J. and Sandler, M. (2004). Using mixture models for collaborative filtering. In *Proc. 36th ACM Symposium on Theory of Computing*.

Klensin, J. (2001). Simple mail transfer protocol. Technical Report RFC 2821, IETF.

Klimt, B. and Yang, Y. (2004). Introducing the enron corpus. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*. Available: http://www.ceas.cc/papers-2004/168.pdf.

Knight, W. (2005). Spain's national ISP blocked by anti-spammers. *NewScientist.com news service*.

Kolcz, A. and Alspector, J. (2001). Svm-based filtering of e-mail spam with content-specific misclassification costs. In *Workshop on Text Mining (TextDM'2001)*, San Jose, California.

Kolter, J. Z. and Maloof, M. A. (2004). Learning to detect malicious executables in the wild. In *KDD '04: Proceedings of the tenth ACM SIGKDD international*

conference on Knowledge discovery and data mining, pages 470–478, New York, NY, USA. ACM Press.

Krim, J. (2003). Spam's cost to business escalates. *Washington Post*, page A01.

Lane, T. and Brodley, C. E. (1998). Approaches to online learning and concept drift for user identification in computer security. In *Fourth International Conference on Knowledge Discovery and Data Mining*, pages 259–263.

Larkey, L. S. and Croft, W. B. (1996). Combining classifiers in text categorization. In *SIGIR-96: 19th ACM International Conference on Research and Development in Information Retrieval*, pages 289–297, Zurich. ACM Press, NY, US.

Lashkari, Y., Metral, M., and Maes, P. (1994). Collaborative Interface Agents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 1, Seattle, WA. AAAI Press.

Lee, B. and Park:, Y. (2003). An e-mail monitoring system for detecting outflow of confidential documents. In *1st NSF/NIJ Symposium on Intelligence & Security Informatics(ISI 2003)*, pages 371–374, Tucson, AZ, USA.

Lee, C.-H. and Shin, D.-G. (1999). Using hellinger distance in a nearest neighbor classifier for relational data bases. 12:363–370.

Lee, W., Stolfo, S. J., Chan, P. K., Eskin, E., Fan, W., Miller, M., Hershkop, S., and Zhang, J. (2001). Real time data mining-based intrusion detection. In *Proceedings of DARPA Information Survivability Conference & Exposition II- DISCEX '01*, volume 1, pages 89–100.

Lee, Y. (2005). The can-spam act: a silver bullet solution? *Commun. ACM*, 48(6):131–132.

Lentczner, M. and Wong, M. W. (2004). Sender Policy Framework: authorizing use of domains in MAIL FROM.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. Technical report.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In Nédellec, C. and Rouveirol, C., editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15, Chemnitz, DE. Springer Verlag, Heidelberg, DE.

Lewis, D. D. and Knowles, K. A. (1997). Threading electronic mail: A preliminary study. In *Information Processing and Management*, pages 209–217.

Leyden, J. (2003). Zombie pcs spew out 80 percent of spam. *The Register*.

Li, W.-J., Herzog, B., Wang, K., and Stolfo, S. (2005). Fileprints: identifying file types by n-gram analysis. In *IEEE Information Assurance Workshop*.

Littlestone, N. and Warmuth, M. K. (1989). The weighted majority algorithm. *IEEE Symposium on Foundations of Computer Science*.

Mackey, W. E. (1988). Diversity in the use of electronic mail: A preliminary inquiry. In *ACM Transactions on Office Information Systems*, volume 6.

Macskassy, S. A., Dayanik, A. A., and Hirsh, H. (1999). Emailvalet: Learning user preferences for wireless email. In *Learning about Users Workshop, IJCAI'99*.

Maes, P. (1994). Agents that reduce work and information overload. *Commun. ACM*, 37(7):30–40.

Manaco, G., Masciari, E., Ruffolo, M., and Tagarelli, A. (2002). Towards an adaptive mail classifier. Technical report.

Manber, U. (1994). Finding similar files in a large file system. In *Usenix Winter*, pages 1–10, San Fransisco, CA.

Mander, R., Salomon, G., and Wong, Y. Y. (1992). A pile metaphor for supporting casual organisation of information. In *CHI92 - ACM Conference on Human Factors in Computing Systems*, Monterey, California. ACM Press.

Mandic, M. and Kerne, A. (2005). Using intimacy, chronology and zooming to visualize rhythms in email experience. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1617–1620, New York, NY, USA. ACM Press.

Mangalindan, M. (2002). For bulk e-mailer, pestering millions offers path to profit. *Wall Street journal*.

Massey, B., Thomure, M., Budrevich, R., and Long, S. (2003). Learning spam: Simple techniques for freely-available software. In *USENIX Annual Technical Conference, FREENIX Track*, pages 63–76.

Mertz, D. (2002). Six approaches to eliminating unwanted e-mail. Technical report, IBM.

Metz, C. (2003). Corporate antispam tools. *PC Magazine*.

Mitchel, T. (1997). *Machine Learning*. McGraw-Hill.

Mock, K. (1999). Dynamic email organization via relevance categories. In *International Conference on Tools with Artificial Intelligence (ICTAI'99)*, Chicago, IL.

Mozilla Foundation (2005). Thunderbird. http://www.mozilla.com/thunderbird/.

Murakoshi, H., Shimazu, A., and Ochimizu, K. (2000). Construction of deliberation structure in e-mail communication. *Computational Intelligence*.

MYSQL (2005). http://www.mysql.com.

Nardi, B. A., Whittaker, S., Isaacs, E., Creech, M., Johnson, J., and Hainsworth, J. (2002). Integrating communication and information through contactmap. *Communications of the ACM*, 45(4):89–95.

Nwana, H. S. (1995). Software agents: An overview. *Knowledge Engineering Review*, 11(2):205–244.

Pantel, P. and Lin, D. (1998). Spamcop: A spam classification & organization program. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin.

Pazzani, M. J. (2000). Representation of electronic mail filtering profiles: a user study. In *Intelligent User Interfaces*.

Peng, F. and Schuurmans, D. (2003). Combining naive bayes and n-gram language models for text classication. In *25th European Conference on Information Retrieval Research (ECIR)*.

Pollock, S. (1988). A rule-based message filtering system. *ACM Trans. Office Automation Systems*, 6:232–254.

Postel, J. B. (1982). Simple mail transfer protocol. Technical Report RFC 821, IETF.

Postini.com (2004). Postini - preemptive email protection.

Prakash, V. V. (2005). Vipul's razor. http://razor.sourceforge.net/.

Provost, F. and Fawcett, T. (2000). Robust classification for imprecise environments. *Machine Learning*, 42:203–231.

Provost, J. (1999). Naive-bayes vs. rule-learning in classification of email. Technical report.

QMAIL (2005). Mbox format. http://qmail.org/qmail-manual-html/man5/mbox.html.

Rhyolite Software (2001). Distributed checksum clearinghouse. http://www.rhyolite.com/anti-spam/dcc/.

Rigoutsos, I. and Huynh, T. (2004). Chung-kwei: a pattern-discovery-based system for the automatic identification of unsolicited e-mail messages (spam). In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*.

Ringel, M. and Hirschberg, J. (2002). Automated message prioritization: Making voicemail retrieval more efficient. In *Conference on Human Factors in Computer Science*, Minneapolis, Minnesota USA.

Rios, G. and Zha, H. (2004). Exploring support vector machines and random forests for spam detection. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*. Available: http://www.ceas.cc/papers-2004/174.pdf.

Rohall, S. L., Gruen, D., Moody, P., Wattenberg, M., Stern, M., Kerr, B., Stachel, B., Dave, K., Armes, R., and Wilcox, E. (2003). ReMail: A reinvented email prototype. Technical Report RC22949, IBM.

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

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. D., and Stamatopoulos, P. (2001). Stacking classifiers for anti-spam filtering of e-mail. In *Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing*.

Salton, G. and McGill, M. J. (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Scheffer, T. (2004). Email answering assistance by semi-supervised text classification. *Intell. Data Analysis*, 8(5):481–493.

Schneider, K. M. (2003). A comparison of event models for naïve bayes anti-spam e-mail filtering. In *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*, Budapest, Hungary.

Schneider, K.-M. (2004). Learning to filter junk e-mail from positive and unlabeled examples. In *Proceedings of the 1st International Joint*

Conference on Natural Language Processing (IJCNLP-04), Sanya City, Hainan Island, China, pages 602–607. Available: http://www.phil.uni-passau.de/linguistik/mitarbeiter/schneider/pub/ijcnlp2004.pdf.

Schultz, M. G., Eskin, E., and Stolfo, S. J. (2001a). Malicious email filter - a unix mail filter that detects malicious windows executables.'. In *USENIX Annual Technical Conference - FREENIX Track*, Boston, MA.

Schultz, M. G., Eskin, E., Zadok, E., and Stolfo, S. J. (2001b). Data mining methods for detection of new malicious executables. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA.

Segal, R., Crawford, J., Kephart, J., and Leiba, B. (2004). Spamguru: An enterprise anti-spam filtering system. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*. Available: http://www.ceas.cc/papers-2004/126.pdf.

Segal, R. B. and Kephart, J. O. (1999). Mailcat: An intelligent assistant for organizing e-mail. In *3rd International Conference on Autonomous Agents*.

Segal, R. B. and Kephart, J. O. (2000). Incremental learning in swiftfile. In *17th International Conf. on Machine Learning*, pages 863–870, San Francisco, CA. Morgan Kaufmann.

Sidiroglou, S. and Keromytis, A. (2005). countering network worms through automatic patch generation. volume 3, pages 41 – 49.

Siefkes, C., Assis, F., Chhabra, S., and Yerazunis, W. S. (2004). Combining Winnow and orthogonal sparse bigrams for incremental spam filtering. In

Boulicaut, J.-F., Esposito, F., Giannotti, F., and Pedreschi, D., editors, *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2004)*, volume 3202 of *Lecture Notes in Artificial Intelligence*, pages 410–421. Springer. Available: http://www.siefkes.net/papers/winnow-spam.pdf.

Sipior, J. C., Ward, B. T., and Bonner, P. G. (2004). Should spam be on the menu? *Commun. ACM*, 47(6):59–63.

Snyder, J. (2004). What is a false positive? *Network World*.

SpamAssassin (2003). Spamassassin.com.

Spamhaus (2005). http://www.spamhaus.org/.

Spertus, E. (1997). Smokey: Automatic recognition of hostile messages. In *AAAI/IAAI*, pages 1058–1065.

Spews (2005). Spam prevention early warning system. http://www.spews.org.

Spring, T. (2005). Slaying spam spewing zombie pcs. *PC World*.

Stanfill, C. and Waltz, D. (1986). Toward memory-based reasoning. *Commun. ACM*, 29(12):1213–1228.

Stolfo, S. J., Chan, P. K., Fan, D., Lee, W., and Prodromidis, A. (1996). Meta-learning agents for fraud and intrusion detection in financial information systems.

Stolfo, S. J., Hershkop, S., Wang, K., Nimeskern, O., and Hu, C. (2003a). A behavior-based approach to securing email systems. *Mathematical Methods, Models and Architectures for Computer Networks Security*.

Stolfo, S. J., Hershkop, S., Wang, K., Nimeskern, O., and Hu, C. (2003b). Behavior profiling of email. In *1st NSF/NIJ Symposium on Intelligence & Security Informatics(ISI 2003)*, Tucson, Arizona.

Stolfo, S. J., Lee, W., Hershkop, S., Wang, K., wei Hu, C., and Nimeskern, O. (2003c). Detecting viral propagations using email behavior profiles. *ACM Transactions on Internet Technology*.

Sullivan, B. (2003). Who profits from spam? surprise. *MSNBC*.

S.Vidyaraman, Shah, N., Pramanik, S., Vuliyaragoli, P., A.Sankaralingam, and Upadhyaya, S. (2002). PRUDENT: profiling users and documents for insider threat mitigation. Technical report, University at Buffalo.

Tan, P.-N. and Jin, R. (2004). Ordering patterns by combining opinions from multiple sources. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 695–700, New York, NY, USA. ACM Press.

Tax, D. and Duin, R. (2001). Combining one-class classifiers. In Kittler, J. and Roli, F., editors, *Second International Workshop MCS 2001*, pages 299–308, Cambridge, UK.

Tax, D., van Breukelen, M., Duin, R., and Kittler, J. (2000). Combining multiple

classifiers by averaging or multiplying? *Pattern Recognition*, 33(9):1475–1485.

Tesauro, G., Kephart, J. O., and Sorkin, G. B. (1996). Neural networks for computer virus recognition. *IEEE Expert*, 11:5–6.

TrimMail (2005). Cat fight! spf claws sender-id. http://www.emailbattles.com/archive/battles/spam_aabeeighag_ef/.

Twining, R., Williamson, M., Mowbray, M., and Rahmouni, M. (2004). E-mail prioritization: reducing delays on legitimate mail caused by junk mail. In *Proceedings of USENIX 2004 Annual Technical Conference, General Track*, pages 45–58.

Tyler, J. R., Wilkinson, D. M., and Huberman, B. A. (2003). Email as spectroscopy: automated discovery of community structure within organizations. *Communities and technologies*, pages 81–96.

U.S. Senate and House of Representatives (2004). Controlling the assault of non-solicited pornography and marketing act of 2003. S. 877. Full text of the CAN-SPAM act available from the US Library of Congress at http://thomas.loc.gov/.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

Vel, O. d., Anderson, A., Corney, M., and Mohay, G. (2001). Mining e-mail content for author identifcation forensics. In *SIGMOD Record*, volume 30, pages 55–64.

Vel, O. D., Corney, M., Anderson, A., and Mahay, G. (2002). Language and gender author cohort analysis of e-mail for computer forensics. In *DFRWS02*, Syracuse, NY.

Venolia, G. D. and Neustaedter, C. (2003). Understanding sequence and reply relatioship with email conversations: A mixed-model visualization. In *CHI 2003*, volume 5, pages 361–368, Ft. Lauderdale, FL. ACM.

Viegas, F., Boyd, D., Nguyen, D. H., Potter, J., and Donath, J. (2004). Digital artifacts for remembering and storytelling: Posthistory and social network fragments. In *HICSS-37*, Hawaii, HI.

Wagner, D. and Soto, P. (2002). Mimicry attacks on host based intrusion detection systems. In *Ninth ACM Conference on Computer and Communications Security*.

Weisband, S. P. and Reinig, B. A. (1995). Managing user perceptions of email privacy. *Commun. ACM*, 38(12):40–47.

Weiss, A. (2003). Ending spam's free ride. *netWorker*, 7(2):18–24.

Whittaker, S., Bellotti, V., and Moody, P. (2005). Revisiting and reinventing email. *Special Issue of Human-Computer Interaction*, 20.

Whittaker, S. and Sidner, C. (1996). 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. ACM Press.

Winiwarter, W. (1999). PEA - a personal email assistant with evolutionary adaption. *International Journal of Information Technology*.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152.

Yerazunis, B. (2003). Sparse binary polynomial hash message filtering and the crm114 discriminator. In *Proceedings of the Spam Conference*. Available: http://crm114.sourceforge.net/CRM114_paper.html.

Yerazunis, B. (2004). The plateau at 99.9% accuracy, and how to get past it. In *Proceedings of the Spam Conference*. Available: http://crm114.sourceforge.net/Plateau_Paper.pdf.

Yoshida, K., Adachi, F., Washio, T., Motoda, H., Homma, T., Nakashima, A., Fujikawa, H., and Yamazaki, K. (2004). Density-based spam detector. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 486–493, New York, NY, USA. ACM Press.

ZDNet UK (2005). Experts clash over merits of anti-spam authentication.

Zhang, L., Tu, N., and Vronay, D. (2005). Info-lotus: a peripheral visualization for email notification. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1901–1904, New York, NY, USA. ACM Press.

Zhang, L., Zhu, J., and Yao, T. (2004). An evaluation of statistical spam filtering

techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):243–269.

Zheng, Z., Padmanabhan, B., and Zheng, H. (2004). A DEA approach for model combination. In *KDD2004*, Seattle, WA.

Zuger, A. (2005). For a retainer, lavish care by 'boutique doctors'. *NY Times, Oct 2005*.

Appendix A

Screenshots

In this section we present screen shots of the implemented EMT and PET systems.

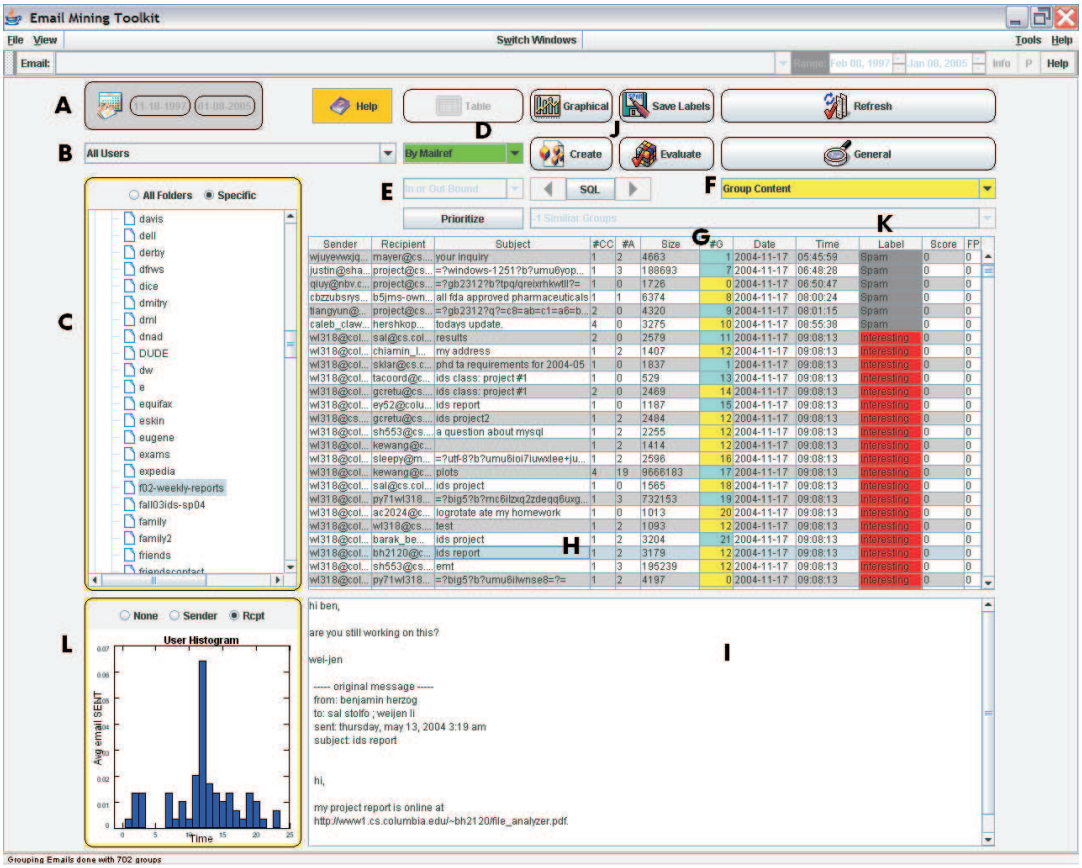

Figure A.1: Main EMT window. Message views can be constrained by time (A), users (B), folder source (C), types (D), and direction (E). Once viewed (G), they can be clustered over some feature (F). Machine learning algorithms can be built and evaluated (J) and labels manually or automatically adjusted (K). When a message is selected (H) the preview is shown (I) and some usage history is also displayed (L). Notice that either the inbound or outbound usage can be viewed here. Clicking on any column re-sorts the messages in ascending or descending order.

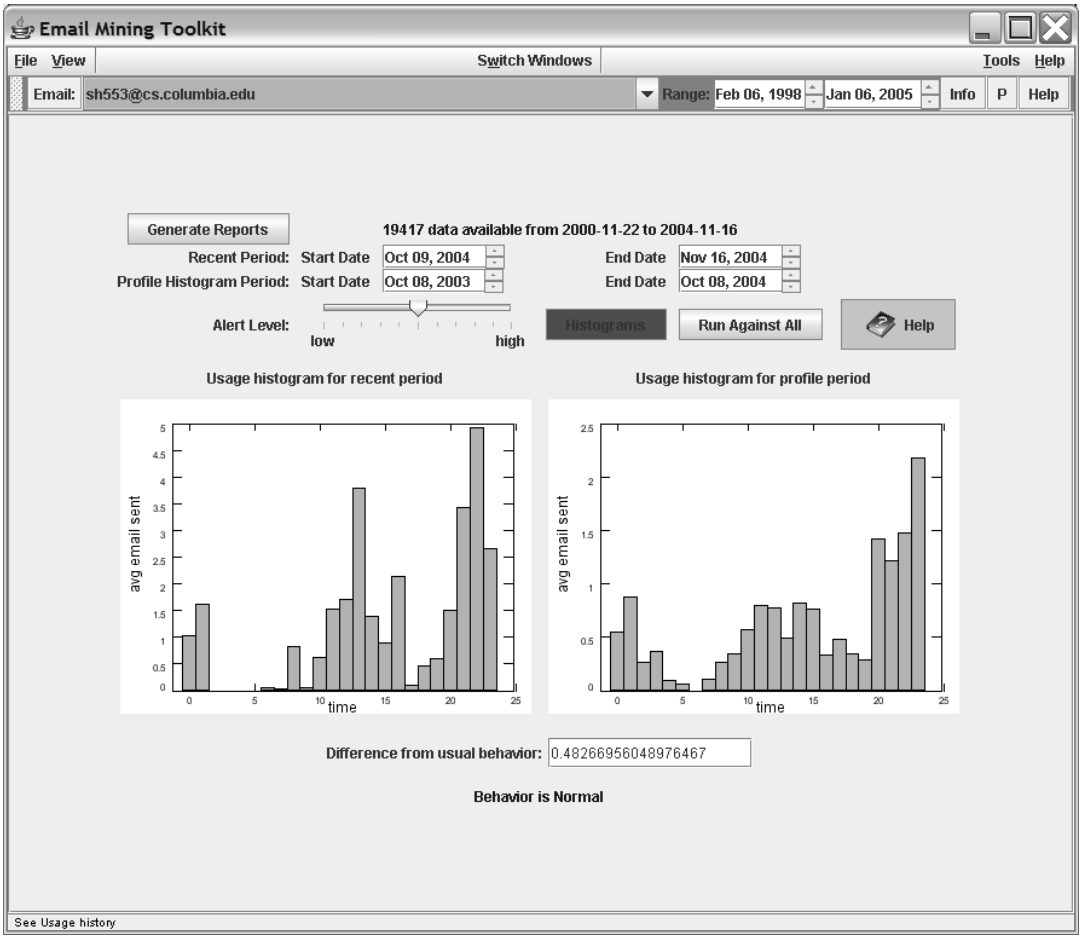

Figure A.2: Usage profile of an individual user. The user 'sh553@cs.columbia.edu' past year of use is compared to the last 3 weeks of usage. The difference between the two periods is calculated on the bottom, and judged to be 'normal'.

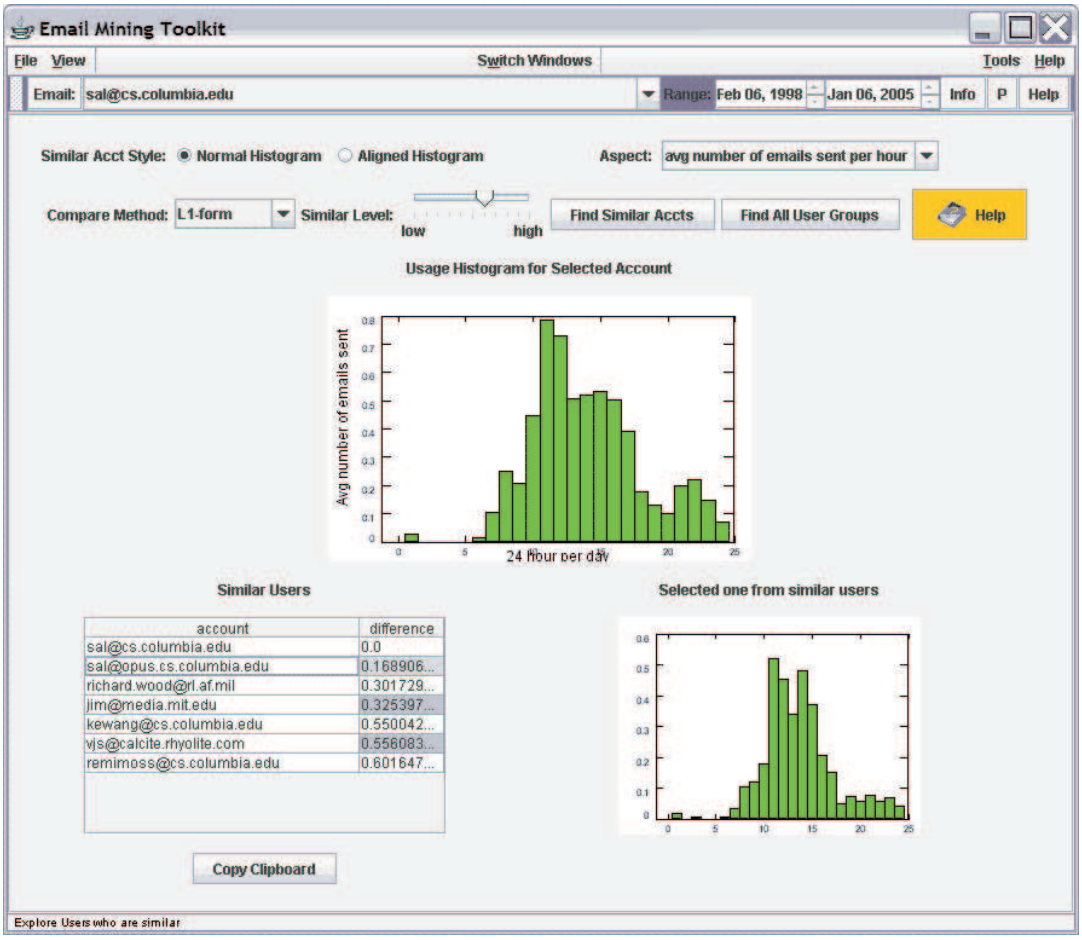

Figure A.3: Given a specific user, we can find similar behaving users based on a specific set of behaviors. In this case, we are comparing average emails using normal histogram alignment, and L1-form comparison.

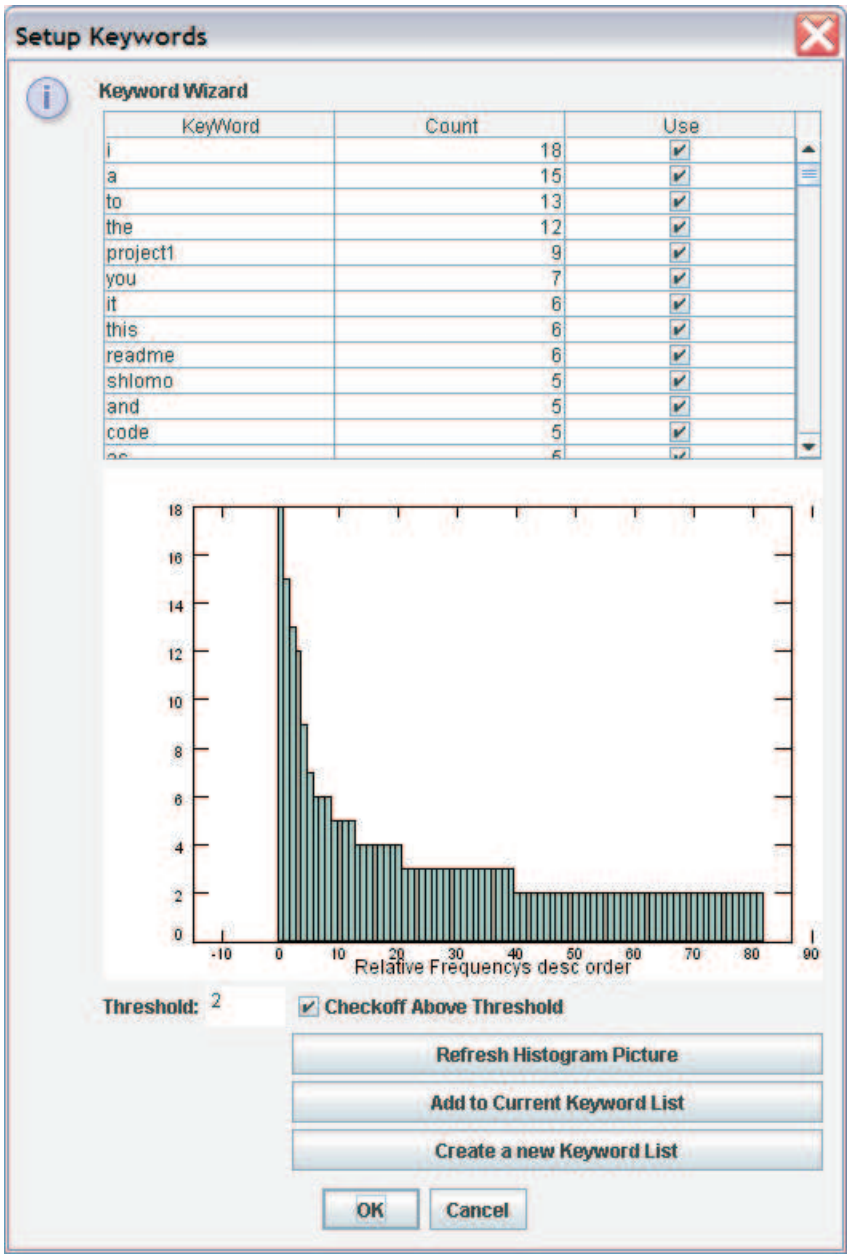

Figure A.4: Keywords automatically extracted from a set of emails.

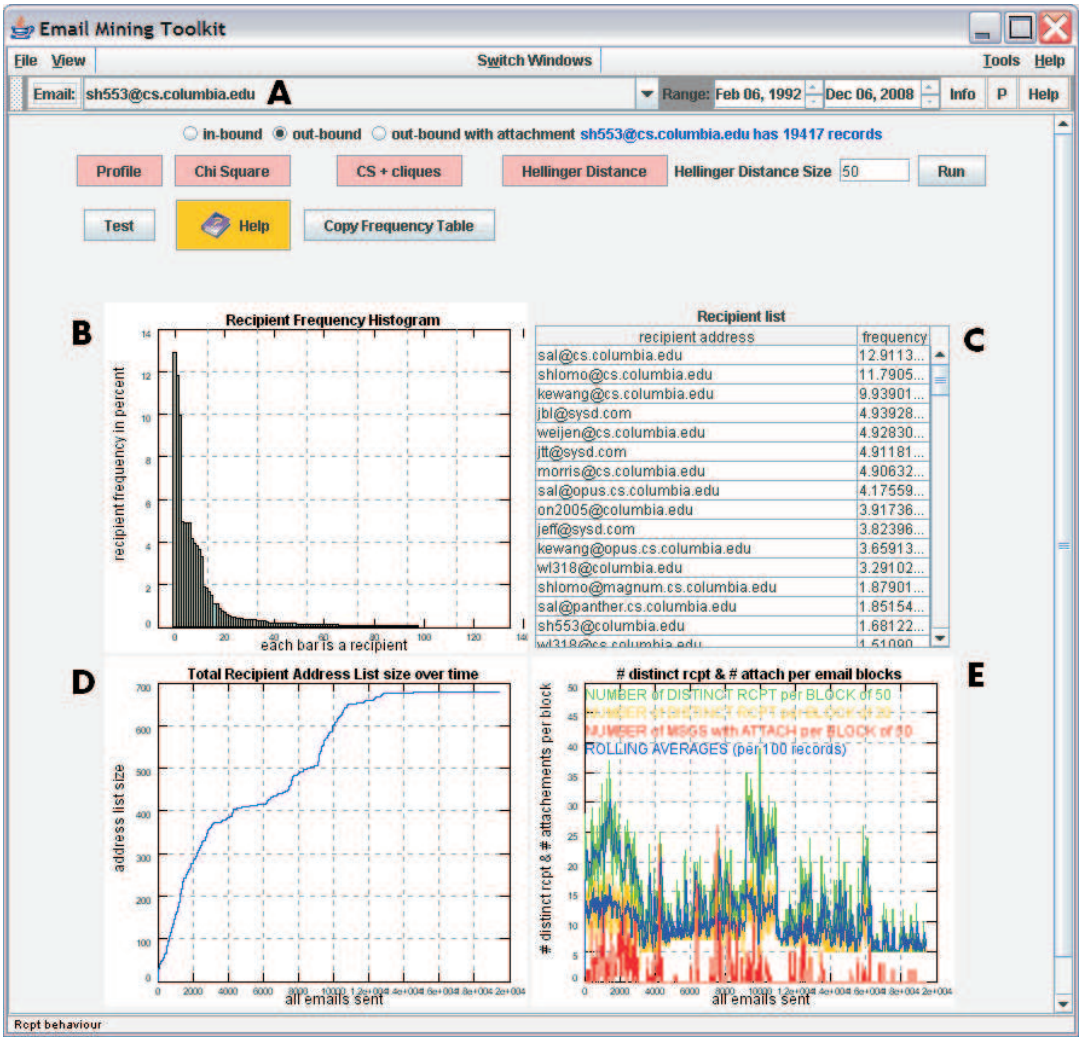

Figure A.5: GUI window to analyze recipient frequencies. A specific user is chosen (A) and we display an outbound profile. (B) A graphical ordering of most frequent recipients, with the same data in tabular form (C). (D) displays the number of new recipients over time, for normal users. One expects this to climb slowly over time. (E) displays attachment analysis using Hellinger distance between different blocks of data over time.

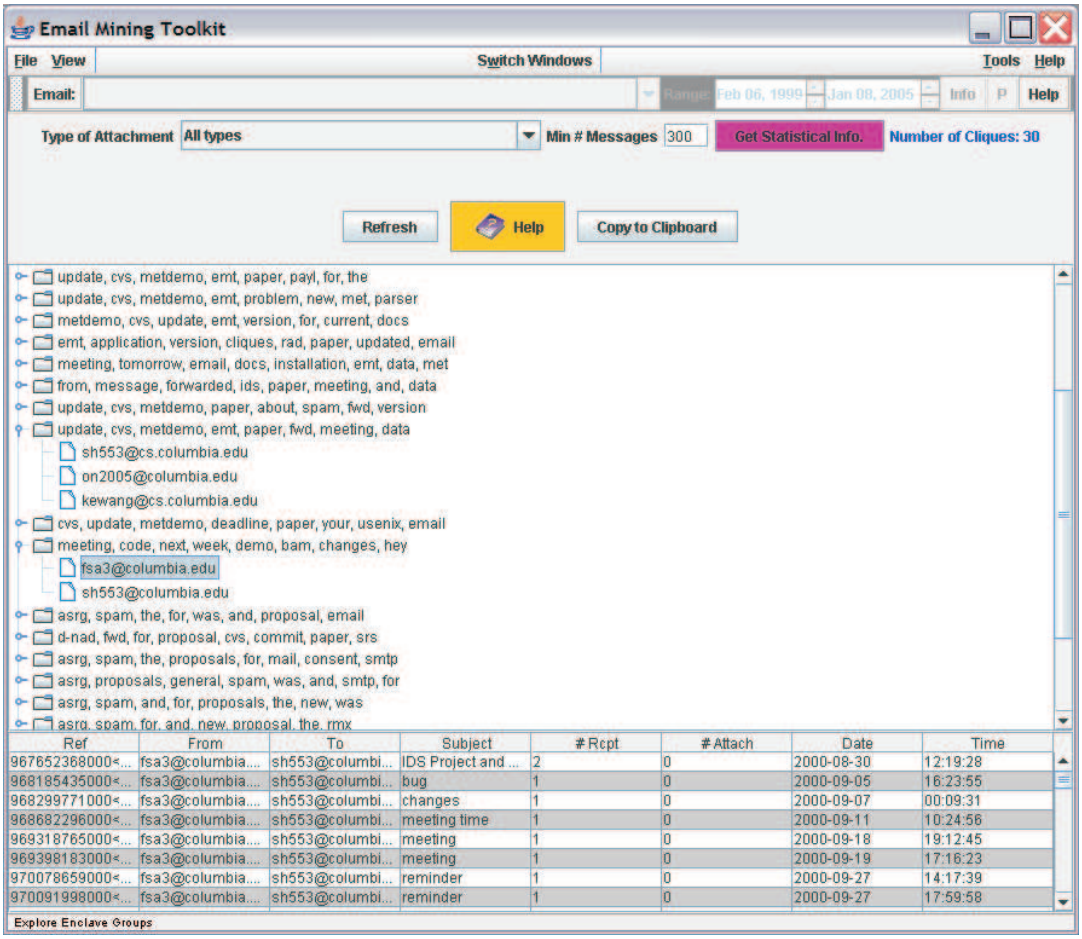

Figure A.6: GUI window to view group cliques.

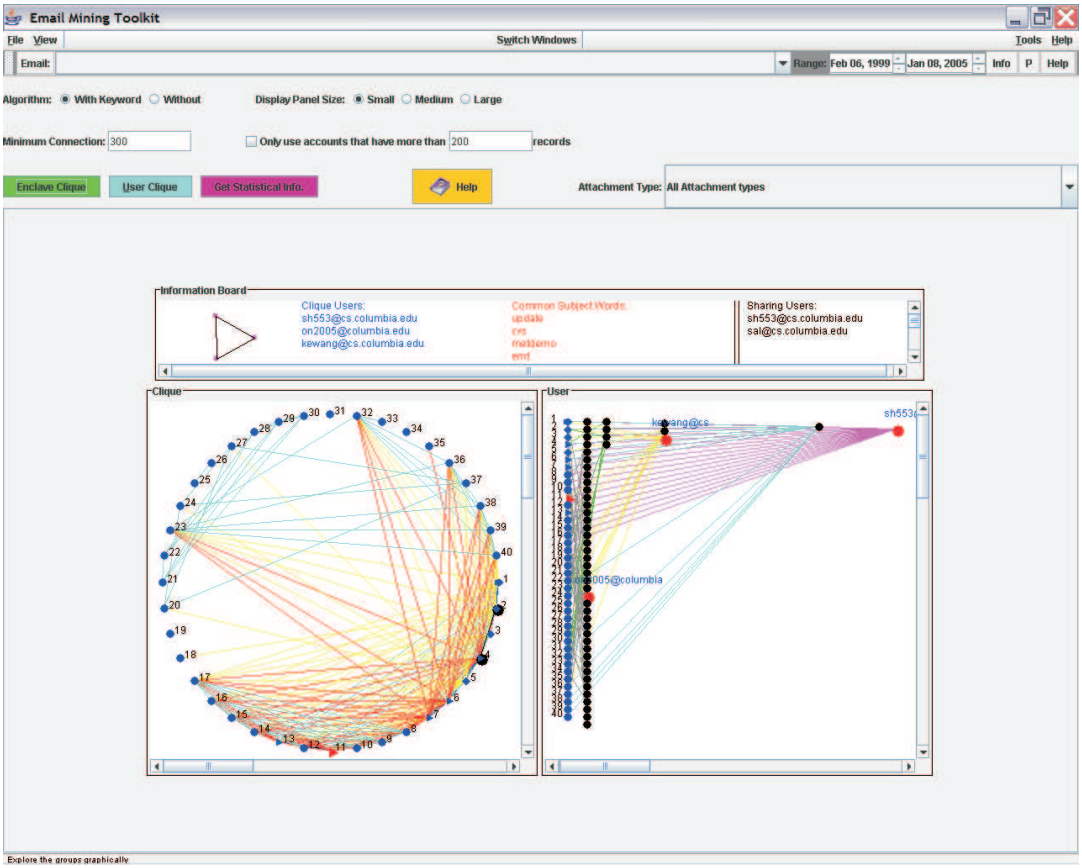

Figure A.7: GUI window to analyze cliques in a graphical fashion.

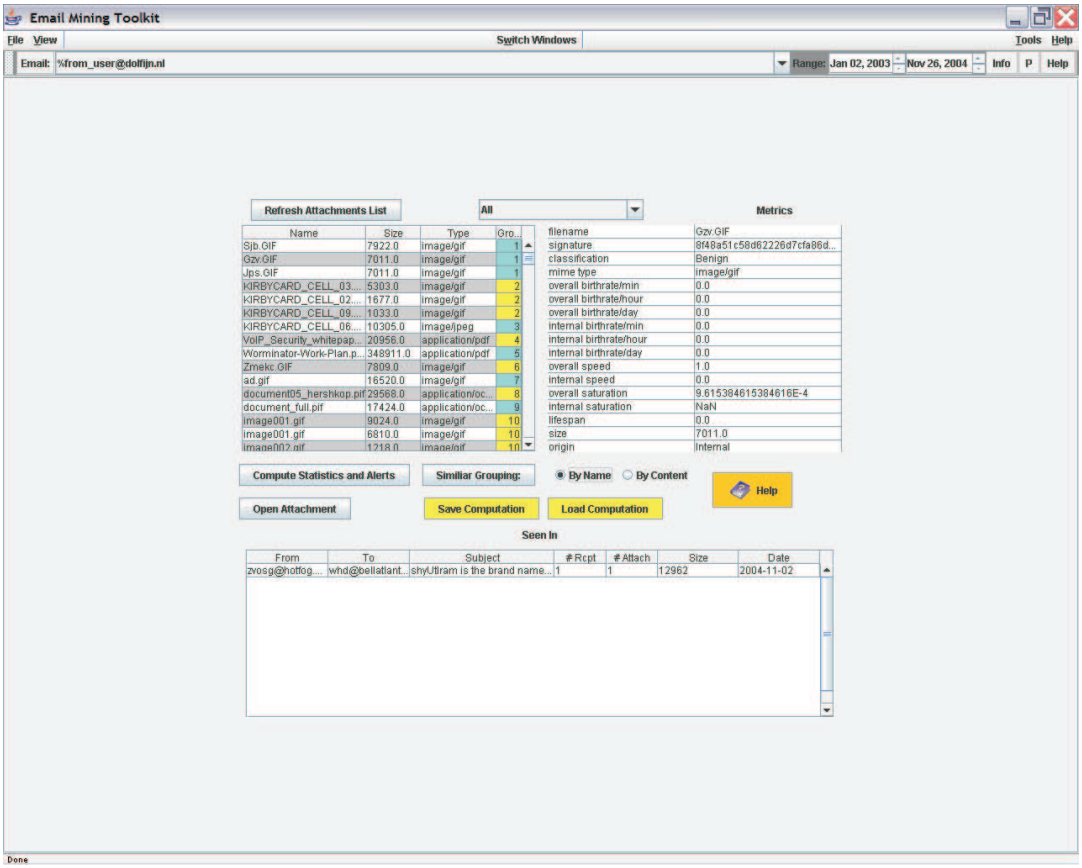

Figure A.8: GUI window to analyze attachments.

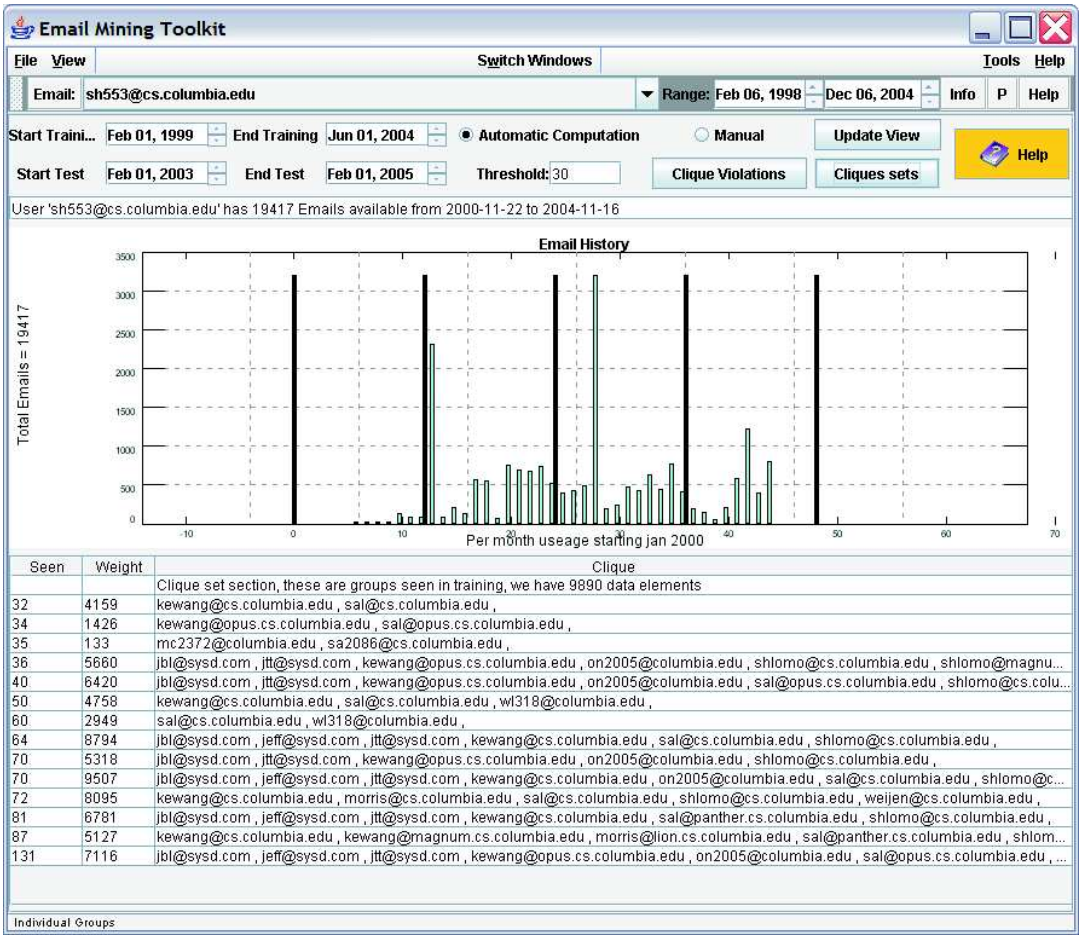

Figure A.9: GUI window to analyze user cliques.

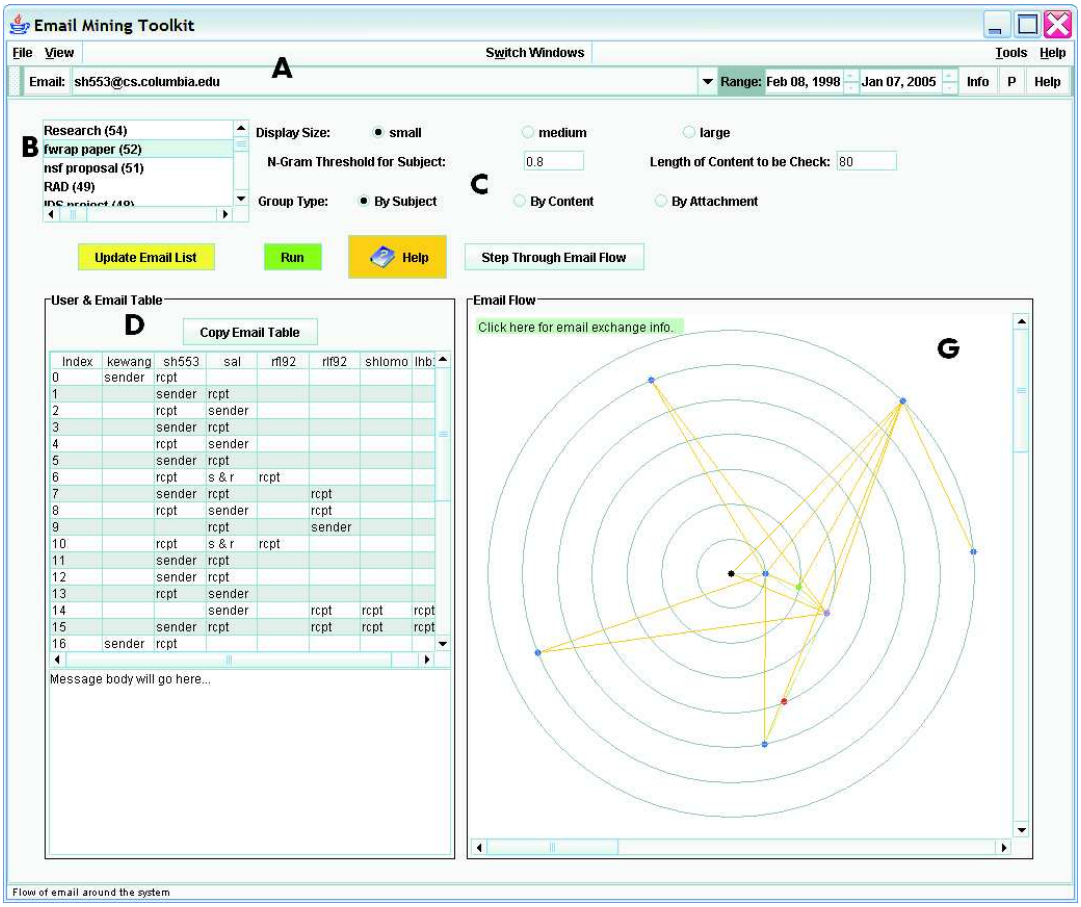

Figure A.10: Email flow among users in the system. A specific user (A) is chosen, and a specific subject is highlighted (B). Analysis can be done either by subject, content, or attachment similarity (C). The results are displayed as a table (D) and graphic (G). Specific message exchanges can be displayed by choosing the links graphically or highlighting a specific table entry.

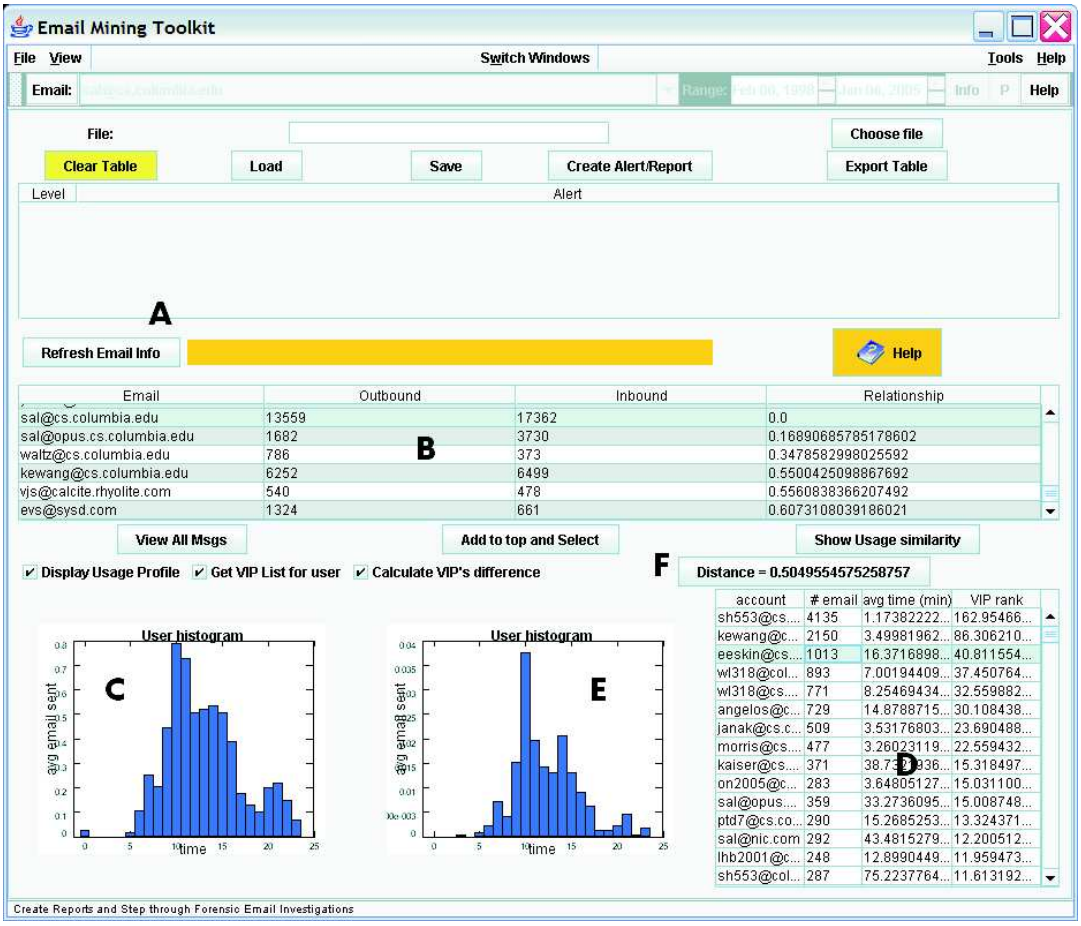

Figure A.11: Email forensics window. Users can be specified by some level of activity (A). Basic statistics can be viewed (B) and a main user can be selected (here sal@cs.columbia.edu). The chosen user's histogram is displayed (C) and a VIP list can be displayed (D). Any VIP user can be compared visually (E) and numerically (F). In addition all messages can be viewed in the message window for a specific user or between a user and a specific VIP.

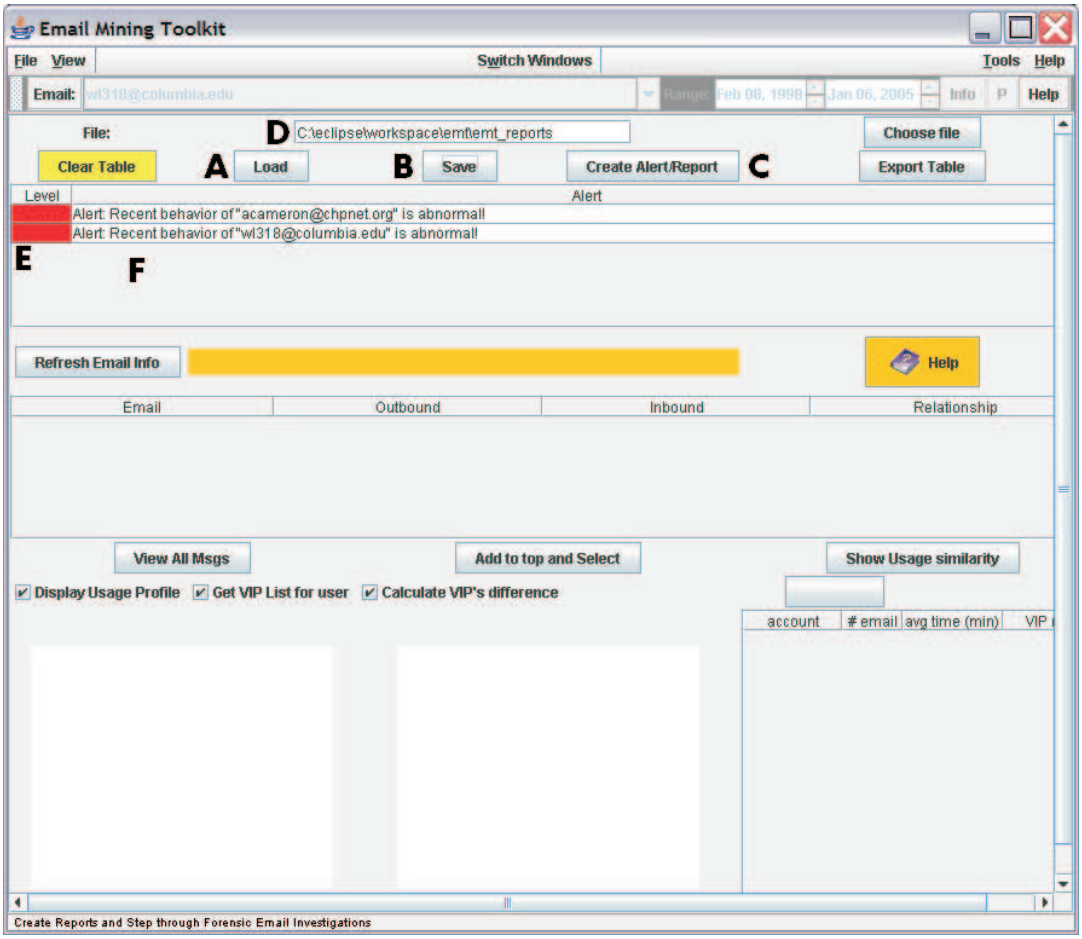

Figure A.12: Email forensic window, report section. Reports can be loaded from a file (A), saved to file (B), and manually created (C) using a specific name (D). Each report item is associated with a report alert level red, yellow, green (high-low) (E). The reports can be viewed and sorted in the main report section(F).

Appendix B

Mail Parser

B.1 EMT Mail Parser

The EMT parser is written in Java, and can be used as a stand-alone application. It can read MBOX, .dbx (Outlook Express), and .pst (Outlook) file formats.

The parser is called "EMTParser.java" and it converts email data to EMT database format.

B.2 Data Format

Traditionally there have been many different formats to store email data. One of the most popular has been the mbox format, which stores email in a flat text file. The mbox format has been a popular choice on many UNIX and Linux type environments. Many email servers store their email in this format.

On the Microsoft Windows platform the default mail client program called Outlook Express, stores email in a binary encoded format with the file extension

".dbx". The upgraded email client known simply as Outlook (with a year version, e.g. Outlook2003), stores its files in a binary version with the file extension ".pst".

B.3 Implementation

In the mbox format, email is stored as a 7-bit ASCII-encoded flat file. Each message begins with a line starting with the word 'From ' ('From' + space) followed by optional source address and optional date. If the word 'from' appears in the beginning of a line in the body, it is the client's responsibility to append some character so that the email data will not be corrupted. A message is supposed to end with a blank line.

A message has two main parts. The header or envelope, contains tokens and values separated by a colon (':'). For example the date is encoded as follows: "Date: Tue, 3 May 2005 09:05:01 -0400". These header fields usually contain a minimum of some standard set of values, such as subject, date, recipient, and sender but can also include non required fields such as email client, priority flags, etc. There is not any specific enforcement of what can be included in the header fields, and many servers and clients typically will include maintenance flags as header fields.

The body of the message is either plain text or composed of multiple parts with any binary data encoded as uuencoded format. Uuencoded converts binary data into ASCII characters, by sliding a 3-byte window and adding 32, and outputting the ASCII values on the range of 32 to 95. Every group of 60 characters are line-separated with a control code saying how many bytes on the particular line. Additional control codes regarding the file name and file permission are appended to the text (QMAIL, 2005).

Many different email clients seem to implement the mbox rules more as a very loose recommendation than a strict standard. This is because many clients will try to recover partial email information and attempt to deliver malformed data. In addition, clients do not replace the body 'from' lines in a consistent fashion and messages do not necessarily end with a blank line. Some clients implement their own version of the first "from" line separator etc. Because of this, when comparing the performance of the parser to any email client, we have attempted to follow the standards.

B.3.1 Logic

The parser has been implemented with the following simple logic.

1. assume start of header.
2. while header items i.e. token *colon space* tokens or token *colon space* tokens *newline tab tokens*
   process specific header features.
3. if newline or non-header line assume the start of the email body.
4. while not start of a new header
   append to the current body text.
5. else we are done with body, process email taking care to decode any mime encoded or multi-part messages in a controlled recursive fashion.
6. GOTO 1.

B.3.2 Arguments

- -n - This option tells the parser not to privatize any of the data.
- -folder - This option specifies that the folder name where the email data has been collected should be hashed for privacy concerns.
- -i *name* - Specifies a specific input file to parse. If this is a directory the parser will perform a recursive step through all the files in the directory and try to parse them.
- -db *dbname machine user password* - Specifies the database to use, machine name (example: localhost), user to access the database, and password for the user.
- -log - The log argument creates a log file, showing each subject line and email info of each record processed.
- -flush - The flush argument erases all current entries from the database before adding any new ones.
- -s *or -rs* - These allow processing of files with one email per file. This is not strictly MBOX format where each folder has its one file. The first flag allows a single directory of emails to be processed, and the "-rs" is a recursive call to process multiple directories with multiple files in each.