

PROJECT NUMBER: CS-CEW-0203

An Analysis of Spam Filters

A Major Qualifying Project Report

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Joseph Chiarella

and

Jason O'Brien

Date: April 23, 2003

Professor Craig Wills, Advisor

Professor Mark Claypool, Advisor

Abstract

Spam is a serious problem that has been increasingly plaguing users of the Internet. Programs known as *spam filters* are employed to assist the user in deciding if an email is worth reading or not. This paper focuses on comparing the filters known as SpamAssassin and Bogofilter on mailboxes from a variety of users, while providing a shorter look into the effectiveness of whitelists and blacklists as spam prevention tools. The performance of these filters was not consistent with neither filter being more effective at identifying spam than the other. Filtering spam is a difficult task and mainstream filters still have room for improvement to work for a variety of users.

Contents

1	Introduction	6
2	Literature Review	8
2.1	Defining Spam	8
2.2	Spam Filters - Conceptual	11
2.2.1	Rule-based Filtering	11
2.2.2	Blacklisting	11
2.2.3	Whitelisting	12
2.2.4	Paul Graham's Bayesian Filtering	13
2.2.5	Sending-side Filters	14
2.3	Spam Filters - Practice	15
2.3.1	Bogofilter	17
2.3.2	SpamAssassin	17
2.3.3	Vipul's Razor (aka SpamNet)	19
2.3.4	MAPS mailabuse.org	20
2.4	Tests of Other Products	21
2.5	Summary	22
3	Data Collection	23
3.1	Students	23
3.2	Pure Spam	23
3.3	Professors	24
3.4	Small Company	26
3.5	Summary	28
4	Methodology	30
4.1	Bogofilter versus SpamAssassin	30
4.1.1	Headers or No Headers?	33
4.2	Whitelists and Blacklists	34
4.3	SpamAssassin Scores	35
4.4	Data Analysis Script	35
4.5	Comparing Corpora	36
4.6	Summary	37
5	Analysis	39
5.1	Spamassassin vs. Bogofilter	39
5.2	Whitelists and Blacklists	47
5.3	Spamassassin Alone	53
5.3.1	Finding the Right Threshold	53
5.4	Bogofilter Alone	58
5.4.1	Corpus Analysis	58
5.4.2	Student 1 vs Free Email 1	58
5.4.3	Comparison of Company Person 1 and Company Person 2	61

5.4.4	Small Company Test Results	64
5.5	Summary	65
6	Discussion	66
7	Conclusion	69
7.1	Bayesian or Bayesish?	69
7.2	Improvements	69
7.2.1	Script Improvements	70
7.2.2	Corpora Comparisons Improvements	71
7.3	Whitelists and Blacklists	71
7.4	Site-Wide Implementations of Bogofilter	72
7.5	Future Work	72
A	Historical Spam	74
A.1	Monty Python : "Flying Circus", Spam	74
A.2	The first Spam email	76
A.3	The first Spam USENET post	77
A.4	The first time Spam was called "Spam"	78
	References	80

List of Figures

2.1	Mail Transport Process	16
4.1	Bogofilter Analysis Algorithm	32
4.2	SpamAssassin Analysis Algorithm	32
4.3	New Email Addresses Over Time Algorithm	35
5.1	Company Person 2's Ham Over Time	41
5.2	Company Person 2's Spam Over Time	42
5.3	SpamAssassin Vs. Bogofilter on Ham Summary	45
5.4	SpamAssassin Vs. Bogofilter on Spam Summary	46
5.5	SpamAssassin Vs. Bogofilter Overall Summary	47
5.6	New Email Addresses in Ham	49
5.7	New Email Addresses in Spam	50
5.8	Rate of New Email Addresses in Ham	51
5.9	Rate of New Email Addresses in Spam	52
5.10	CDF - Professor 1's SpamAssassin Scores	53
5.11	CDF - Professor 2's SpamAssassin Scores	55
5.12	CDF - Company Person 1's SpamAssassin Scores	55
5.13	CDF - Company Person 2's SpamAssassin Scores	56
5.14	CDF - ST1, ST2, FE1, & FE2's SpamAssassin Scores	57
5.15	CDF: Corpus Comparisons	62

List of Tables

3.1	Summary of Email Data	28
5.1	Summary of Bogofilter and SpamAssassin Performance on Ham	43
5.2	Summary of Bogofilter and SpamAssassin Performance on Spam	43
5.3	Summary of Bogofilter and SpamAssassin Performance on Ham	44
5.4	Summary of Bogofilter and SpamAssassin Performance on Ham	44
5.5	Corpus of Student 1 vs Free Email 1	59
5.6	Student 1 vs Hotmail - Top Ham Words	60
5.7	Student 1 vs Hotmail - Top Spam Words	60
5.8	Company Person 1 vs Company Person 2	63
5.9	Corpora Comparison Chart	64

1 Introduction

As the Internet started to gain popularity in the early 1990's, it was quickly recognized as an excellent advertising tool. At practically no cost, a person can use the Internet to send an email message to thousands of people. When this message contains an unsolicited advertisement, it is commonly known as spam. Whether or not spam actually benefits the advertiser is not really known, but to email users that receive over one hundred unwanted emails a day, the problem is serious.

Email users fight back against the spammers by trying to avoid having to read spam. If a user knows ahead of time which emails are spam and which are not, they can waste less time reading and deleting advertisements. Programs known as *spam filters* are employed to assist the user in deciding if an email is worth reading or not.

This paper presents an analysis of spam filters, including a look into the history of spam, to help understand how this problem was started. In this paper, spam filters are discussed on a conceptual level and various common algorithms are described. This discussion leads into descriptions of actual spam filtering software available and how they work.

The meat of this paper is the analysis. Our main goal was to provide a comparison between the widely-used rule-based filter, SpamAssassin, and the relatively new learning filter, Bogofilter. Bogofilter is based on a new method of filtering spam where the filter learns over time. We wanted to see which filter worked better under different conditions on email from several users. The email we tested came from a wide variety of users, including computer science students and employees at a small company.

Our other analysis includes a brief look into the effectiveness of whitelists and blacklists.

These methods involve keeping a list email addresses and either accepting (for a whitelist) or blocking (for a blacklist) email from addresses on that list. We also determined how much email differs between users to see how possible it is for one filter to work for all users.

It is important to keep in mind that the field of filtering spam, like most computer-related fields, is a rapidly changing one. The filters analyzed were the best ones available at the time this project was started. Improvements to both filters used here were made during the time the analysis was being performed. While the filters have improved, the concepts behind them remain the same.

In the remainder of this report, we first give the history of spam and spam filtering in Chapter 2. In Chapter 3 we describe the sources of the email we used for testing. In Chapter 4 the methods that we used to perform our tests are detailed. The results of our analysis are presented in Chapter 5 with Chapter 6 containing our conclusions. Future work is discussed in Chapter 7.

2 Literature Review

There are many efforts underway to to stop the increase of spam that plagues almost every user on the Internet. In this section we show how to define and detect spam, the history of spam, and the various filters that are currently available. Additionally, the results of other spam filter analyses are mentioned.

2.1 Defining Spam

Before one can really begin to filter spam emails, an accurate definition must first be devised. Simply claiming it is any unwanted email is too opinionated, and calling it any unasked for email for is unrealistic. Mail-abuse.org defines spam as follows[8]:

“An electronic message is “spam” IF: (1) the recipient’s personal identity and context are irrelevant because the message is equally applicable to many other potential recipients; AND (2) the recipient has not verifiably granted deliberate, explicit, and still-revocable permission for it to be sent; AND (3) the transmission and reception of the message appears to the recipient to give a disproportionate benefit to the sender.”

Others describe spam as simply “unsolicited commercial email sent in bulk” [3].

The history of the usage of the word “spam” being associated with unsolicited commercial emails is not entirely clear. SPAM was created by Hormel in 1937 as the world’s first canned meat that didn’t need to be refrigerated. It was originally named “Hormel Spiced Ham,” but was eventually changed to the catchier name, “SPAM.”

Its connection to email is, according to Hormel and many other sources, due to a sketch on the British comedy TV show, Monty Python’s Flying Circus. In the skit, a group of Vikings sing “SPAM, SPAM, SPAM” repeatedly, drowning out all other conversation in the restaurant [4]. The text of the SPAM sketch is included in Appendix A.1.

Brad Templeton, founder of ClariNet Communication Corp., the first business on the Internet, has done the most in-depth research into the history of spam on the Internet [17]. According to him, the first email spam was from 1978, and was sent out to all users on ARPANET (several hundred users). It was an ad for a presentation by Digital Equipment Corp., and its full text can be seen in Appendix A.2.

Templeton notes that the origin of spam as we know it started on Usenet and migrated to email. In an attempt to raise money for college, a college student, Rob Noha, posted to as many newsgroups as he could find on May 24, 1988 using his account JJ@cup.portal.com. The post did receive several annoyed responses, but the term “spam” was not used to describe it until 1996. The full text of this posting is available in Appendix A.3.

It was not until 1993 that a USENET posting was called “spam.” In an attempt to implement a retro-moderation system that allowed posts to be deleted after they had been posted, Richard Depew accidentally created a monster. His software, ARMM, had a bug in it which caused it to post 200 messages to news.admin.policy. Readers of this group were making jokes about the accident, one person referring to the incident as “spamming.” The posting and response are available in Appendix A.4 on 78.

USENET seemed like a natural place to spam. Anyone could see all of the available groups and post to all of them relatively easily. On January 18, 1994, the first large-scale USENET spam occurred. A message with the subject “Global Alert for All: Jesus is Coming Soon” was cross-posted to every available newsgroup. Its controversial message sparked many debates all across USENET, and according to Templeton, the student who posted this message did get in trouble.

In April of 1994, spamming first became a business practice. Two lawyers from Phoenix,

Canter and Siegel, hired a rogue programmer to post their “Green Card Lottery- Final One?” message to as many newsgroups as possible. What made them different was that they did not hide the fact that they were spammers. They were proud of it, and thought it was great advertising. They even went on to write the book *How to Make a Fortune on the Information Superhighway : Everyone’s Guerrilla Guide to Marketing on the Internet and Other On-Line Services*[2]. They planned on opening a consulting company to help other people post similar advertisements, but it never took off.

As the spam problem grew larger, the interest in spam filters grew accordingly. As with any new technology, new terms were introduced that are used frequently in this paper that need to be explained. We refer to email that is not spam as ham. We first saw this naming convention in the program Bogofilter. The newer versions of Bogofilter do not use this naming convention anymore, however we liked it, and continued using it. The job of spam filters is to mark emails as spam, and thus, an email marked as spam is a “positive.” We also use the term “false positive” to refer to a ham message that was incorrectly marked as spam, and “false negative” as a spam message incorrectly marked as ham. False positives are the worst outcome of a spam filter. When a user gets a false positive, an email that was intended for them does not get through. In the case of a false negative, the user merely ends up seeing an email that they did not want to see. We also use the term “corpus” (plural “corpora”) to refer to a collection of email from a user.

2.2 Spam Filters - Conceptual

There are many ways to go about solving the problem of filtering spam. This section describes several methods of filtering spam that are used. Section 2.3 lists several specific filters and the methods they use.

2.2.1 Rule-based Filtering

Perhaps the most straight-forward method of filtering spam is a rule-based algorithm. Rules are defined to classify emails as spam or ham based on different characteristics. An example rule could be that all emails with magenta-colored text are spam. Another example would be that all emails that contain the text “order confirmation” are ham. A good rule-based filter would note which rules match, and make a decision based on all of the rules combined.

A rule-based filter is probably the easiest to write, but making one that works requires determining a set of rules that makes sense. Version 2.43 of SpamAssassin has 410 different rules, with weights assigned to each one [16]. The problem with rule-based filters is that the rules are written by people looking for the obvious characteristics of spam.

The people who actually send out the spam are doing their best to make their spam *not* look like spam. It is not uncommon to get spam with a subject like “order confirmation,” “re: your inquiry,” or a similar phrase. The end result is that it is hard to make simple rules that will work well in all cases.

2.2.2 Blacklisting

Blacklisting is a form of rule-based filtering that uses one rule to decide which emails are spam. A blacklist is a list of traits that spam emails have, and if the email being tested

contains any of those traits, it is marked as spam. It is possible to organize a blacklist based on “From: ” fields, originating IP addresses, the subject or body of the message, or any other part of the message that makes sense. Blacklists can be used on both large and small scales. A large-scale blacklist would usually be provided by a third party. The user typically does not contribute to a large list like this. On a smaller scale, the user could simply tell their email client to not allow email from certain addresses. A small-scale blacklist works fine if the user gets spam from one particular address. On a larger scale, where the user does not have any control over the blacklist, there must be a mechanism in place for dealing with accidental blacklisting of other users.

2.2.3 Whitelisting

While blacklisting is a way of deciding which emails are spam, whitelisting decides which emails are ham and assumes all other email is spam. Users would presumably whitelist everyone that they would expect to receive email from, and as long as their friends never send them spam, all spam will be filtered out. The obvious problem is that it is impossible to predict who is going to send email, and anyone previously unknown to the user will be filtered out. One way to avoid this problem is to read through the filtered email regularly[11], but there is no point in filtering if the user must view all of the email anyways.

Another method is to let the senders of all emails marked as spam know that their email was marked as spam while providing them with a method of getting added to the whitelist [9]. Presumably, no spammer is going to go through the trouble of getting added to a user’s whitelist, and hopefully friends will. It probably still blocks all spam, but there is still the problem of dealing with automated order confirmations and mailing lists. When a user joins

a mailing list, they can easily add the address of that list to their whitelist, but it has to be done manually. For order confirmations, for example, the user can not always know what address the confirmation will be coming from, so this approach is not flawless either.

2.2.4 Paul Graham's Bayesian Filtering

Paul Graham was one of the primary developer's of LISP and has been working on a derivative called Arc. In August of 2002 he wrote *A Plan for Spam* in which he discussed why rule-based filtering fails to effectively filter spam [5]. The paper proposed a new method of spam filtering using Bayes' rule.

Bayesian filters use probabilistic reasoning to decide whether or not a message is spam. These filters base their choices on Bayes' rule, which is useful for calculating the probability of one event when one knows another event is true. In the case of email, the rule is used to determine the probability that an email is spam given that it contains certain words.

What makes Bayesian filters different from other filters is that they learn. To decide the probability that an email is spam based on the words it contains, the filter needs to know about the emails that a user receives. Since the interest is solely in the words and their frequencies (and not their ordering in this implementation), the solution is to keep a hash table to record how often each word appears. Spam and ham are kept in separate hash tables, so that probabilities can be calculated later. When an email is declared spam, the spam table is updated by incrementing the frequency counts for each word contained in that email. Ham counts are incremented similarly. Over time, these hash tables begin to characterize a person's ham and spam well enough that some basic math can be used to guess which category a new email fits into. The number of spam and ham emails is also

recorded for use in later calculations.

Graham suggests using a modified Bayes' rule to calculate probabilities. Bayes' rule, when combining multiple probabilities, is:

$$P(A|B \wedge C) = \frac{P(A|C)P(B|A \wedge C)}{P(B|C)}$$

Graham's modification of Bayes rule is:

$$P(A|B \wedge C) = \frac{P(A|B)P(A|C)}{P(A|B)P(A|C) + (1 - P(A|B))(1 - P(A|C))}$$

$P(A|B \wedge C)$ would be read is "the probability that event A is true, given that events B and C are true." In the case of spam filtering, event A would represent an email being spam, while B and C correspond to certain words being in the email. That is, we want to know the probability that an email is spam (A) if it contains words B and C. This equation would be expanded to include more words (Graham proposes using the most interesting 15 words). Graham's Bayes' rule assumes one already knows the probability that an email is spam for each individual word. This is calculated with the following formula:

$$P(\text{Spam}|\text{word}) = \frac{\frac{b}{nbad}}{\frac{g}{ngood} + \frac{b}{nbad}}$$

where b and g are the number of times a word appears in spam and ham emails, respectively, and $nbad$ and $ngood$ are the total numbers of spam and ham emails received, respectively.

2.2.5 Sending-side Filters

All of the previous spam filtering methods are devised to work on the receiving end. Spammers have to get their Internet connection through some Internet service provider (ISP) and cutting it off on the sending side would be the job of the ISP. Whether the spammer uses the ISP's email server or their own, it should not be too hard to detect when a user sends out

thousands of emails. Merely detecting a user sending out email after email and terminating their access would probably be sufficient to block spammers.

The problem does not lie in detecting the spam, however. The problem is that some ISPs are willing to let spammers use their service to send out thousands of emails. Convincing all ISPs to aggressively monitor for and terminate spammers is not an easy task, and is not in the scope of this project.

2.3 Spam Filters - Practice

Figure 2.1 shows a diagram of the mail transportation process. In theory, spam filters can be implemented at virtually any location in this process. Also, multiple stages of the process can occur on the same machine. There are devices one can purchase that will be a firewall, router, and spam filter all in one. Filtering can also take place on the email server of either the sender or the receiver, or in the client's personal email software. All of the following implementations use some method of filtering that has been described already. In this study we attempt to analyze the effectiveness of filters on their conceptual level by testing implementations of each.

For simplicity, all of our tests involved running spam filters using Procmail[6]. Procmail is a Mail Delivery Agent (MDA) that, in its simplest form, puts emails in users' mailboxes. We used Procmail primarily because it is free, widely used, and easy to configure. It is capable of running any kind of filter, including combinations of filters. It is also capable of working with several email programs. Above all else, we were familiar with its functionality, and most spam filters in Linux support the use of Procmail, making it a comfortable solution.

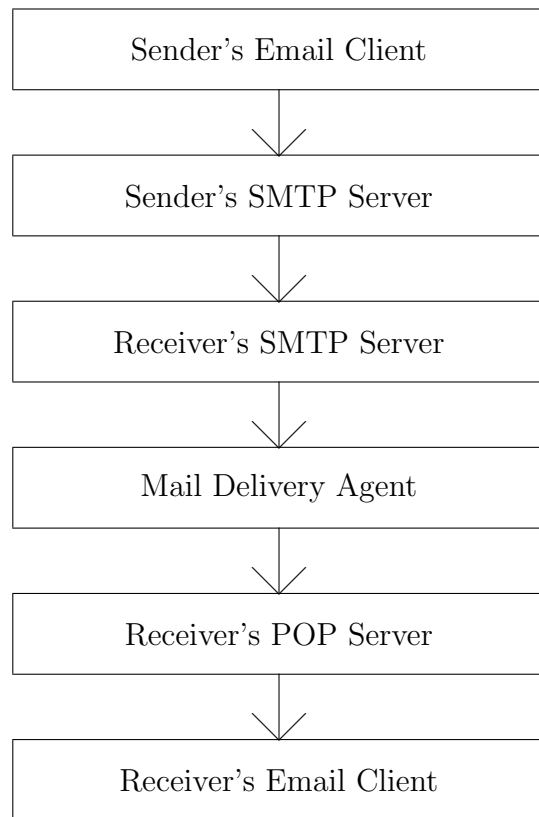


Figure 2.1: Mail Transport Process

2.3.1 Bogofilter

Bogofilter[14] is an implementation of Paul Graham's Bayesian filter. It was written by Eric S. Raymond in August, 2002. He follows Graham's algorithm with a few small modifications.

From the Bogofilter README:

“This version substantially improves on Paul's proposal by doing smarter lexical analysis. In particular, hostnames and IP addresses are retained as recognition features rather than broken up. Various kinds of MTA cruft such as dates and message-IDs are discarded so as not to bloat the word lists.”

We chose to test Bogofilter over other Bayesian filters, like Bayesian Mail Filter and Bayesbam, for several reasons. It was written in C which means it is significantly faster than versions written in other languages. Speed was an important consideration to us because we would be processing thousands of emails through two different filters. If the computer used to run the filters is used by more than one user, this test could lock up the machine for hours. Bogofilter appeared to be the least obtrusive Bayesian filter available.

It takes an email on stdin (the standard input to the program) and it returns 0 or 1 depending on whether or not it thought it was spam. With different command line switches, it can be told to register a word as spam or ham. It also has the ability to undo a previous addition to the database if it was erroneous.

2.3.2 SpamAssassin

SpamAssassin[16] is a widely used rule-based spam filter. From the web site:

“Using its rule base, it uses a wide range of heuristic tests on mail headers and body text to identify “spam”, also known as unsolicited commercial email.

header analysis: spammers use a number of tricks to mask their identities, fool you into thinking they've sent a valid mail, or fool you into thinking you must have subscribed at some stage. SpamAssassin tries to spot these.

text analysis: again, spam mails often have a characteristic style (to put it politely), and some characteristic disclaimers and CYA text. SpamAssassin can spot these, too.”

Based on comments from many sources, we felt that SpamAssassin was the best rule-based filter available at the time we started our project, which is why we chose to test it.

SpamAssassin uses a point system when analyzing an email. Every email is scanned for instances of each characteristic from its list. If a characteristic is found, that email gains the number of points associated with that characteristic. Points can be both negative and positive. For instance, if an email contains the word “GUARANTEED” (in all caps) in the subject line, that email’s score will gain several points. Ham-like features subtract points from an email’s score. If the final score of an email is over a certain threshold, configurable by the user, the email gets flagged as being spam.

The list of email characteristics is extensive, including specific words to the color of text and beyond. It is also configurable by the user. The job of assigning scores to rules was done using a genetic algorithm to minimize false positives and false negatives. The SpamAssassin creators maintain a database of emails, the rules that fit them, and whether or not those emails were spam. It is not necessary to store the contents of the emails after the matching rules are calculated, so the actual emails are not stored. The SpamAssassin team encourages users to contribute to this database to improve SpamAssassin’s accuracy.

Our primary concern with SpamAssassin was its static nature. Everyone using SpamAssassin has the same settings and a spammer could easily test their new message on the latest version of the filter to determine if it would be blocked. There are even services available online to save the perpetrator the expense of maintaining the latest versions of all spam pre-

vention software. This static nature is, however, an advantage in regards to implementing the filter. Since it does not learn, its effectiveness is not based on user input and as a result requires no maintenance.

2.3.3 Vipul's Razor (aka SpamNet)

Vipul's Razor[12] is a distributed blacklisting system. From the web site:

“What is Vipul's Razor?

For nearly two years, Razor has been successfully fighting spam with the help of the Unix community and is the technology that has enabled us to build its windows counterpart, SpamNet, currently in use by more than 100,000 users. Razor, or SpamNet, is a distributed, collaborative, spam detection and filtering network. It establishes a distributed and constantly updating catalogue of spam in propagation. This catalogue is used by clients to filter out known spam. Upon receiving a spam, a Reporting Agent (run by an end-user or a troll box) calculates and submits a 20-character unique identification of the spam (a SHA Digest) to its closest Catalogue Server. The Catalogue Server echoes this signature to other trusted servers after storing it in its database. Prior to manual processing or transport-level reception, Filtering Agents (end-users and MTAs) check their incoming mail against a Catalogue Server and filter out or deny transport in case of a signature match. Catalogued spam, once identified and reported by a Reporting Agent, can be blocked out by the rest of the Filtering Agents on the network.”

Because it is a blacklist, Vipul's Razor should have no false positives at all. However, due to the time required for a new email to enter the central database as a certified spam, we expect that it will have many false negatives. We would expect the accuracy of such a filter to increase proportional to the number of recipients of each email. As more people receive a spam email, the chance of it getting reported to Vipul's Razor increases.

Vipul's Razor was not formally tested in project for several reasons. The central server was unresponsive during the day due to high load, limiting the amount of testing we could perform. A realistic test of Vipul's Razor would also have to be done as emails arrive, as opposed to on a mailbox that was collected months earlier. One can assume that the chance

of an email being blocked by Vipul's Razor is much higher for an email that was received several months ago, as it is more likely that someone else had received that same message and reported it. While testing Vipul's Razor is not impossible, we did not have time in this project to construct a test that would have given valid results, and thus, we left it untested.

2.3.4 MAPS mailabuse.org

MAPS[8] stands for Mail Abuse Prevention System. From their website:

“We are a not-for-profit California organization whose mission is to defend the Internet's e-mail system from abuse by spammers. Our principal means of accomplishing this mission is by educating and encouraging ISP's to enforce strong terms and conditions prohibiting their customers from engaging in abusive e-mail practices.”

They are frequently involved in litigation concerning companies that support spam as an acceptable means of advertising.

MAPS maintains several lists that can be used for blacklisting purposes. Their lists are available for a small fee in order to keep their quality high. They provide lists of IP addresses known to spam, IP addresses that belong to dial-up users, IP addresses of mail servers that allow spam, and several others. What a user does with these lists is up to them.

The issues surrounding this method of blacklisting can be rather serious. When a user's IP is on one of these lists, there is little they can do to send an email to someone who filters based on such a list. Sometimes a user is not even told that their email has been dropped by the receiver's email server.

2.4 Tests of Other Products

PC Magazine[10] has performed an elaborate test on many different spam filters. They included both personal and corporate filters and evaluated them on much more than just accuracy, including ease of installation, ease of use, ability to be customized, and many other factors.

The main focus of PC Magazine's tests is the same as our's, namely how effective the filter is at detecting spam. Eleven personal spam filtering products (Including Microsoft Outlook's and Mac OS X Mail's built-in filtering) and 6 corporate spam solutions (3 that could be installed inside the company and 3 that used other companies' mail servers). None of the products tested worked perfectly and some were clearly better than others. It is important to note that the email used in this test is only from a single account set up by the reviewer.

The Editor's Choice product, SpamAssassin Pro, was not the best performer at detecting ham or detecting spam, "but it had the best balance for catching spam while minimizing false positives." In their tests on about 1000 emails, SpamAssassin Pro incorrectly classified 4.3% of ham as spam (false positives) and 11.1% of spam as ham (false negatives).

The corporate spam prevention products generally performed better than their personal counterparts. The most effective was CipherTrust IronMail 210, essentially a replacement email server that is installed on site. It was also the most expensive at \$27,000. In PC Magazine's test, it had 0.24% false positives and 5.92% false negatives. From the article:

"To combat spam, the IronMail uses an arsenal of techniques. The most significant for our testing was its statistical lookup service, which lets the device check a database of spam signatures in real time at the CipherTrust home office. Additional options include reverse DNS, real-time black holes, header analysis, and dictionary-based filtering."

There are many more corporate spam prevention solutions than were tested in this article (it lists 17 untested ones) and it is possible that there are more effective tools that exist.

The PC Magazine study shows that while no product perfectly identifies spam, it is possible for users to get some relief from the constant barrage of unwanted email. All the personal products that were reviewed needed more refinement to improve their marks.

2.5 Summary

Strictly defining spam is not as easy as it may seem. This definition varies from user to user and building a spam filter that works for everyone is extremely difficult. Spam filtering is currently an active field of research and filters should continue to improve. There may be a limit as to how well filters can work, but we believe this limit is not near yet.

There are many spam prevention products available, using several different algorithms to detect spam. The following techniques are used:

Rule Based Filtering - Uses a pre-defined set of rules to determine if a particular email is spam.

Blacklisting - Blocking all email from a specific list of senders.

Whitelisting - Only allowing email from a specific list of senders.

Bayesian - Uses probabilities to determine if a particular email is spam. Must be trained to be effective.

In this project we analyze the effectiveness of two products in particular: SpamAssassin (rule based) and Bogofilter (Bayesian). We also investigate the possibilities of using a whitelist or a blacklist to filter spam. This analysis and the results are presented in the remaining chapters.

3 Data Collection

To analyze any spam filter, one needs large amounts of email to test it on. These emails need to be sorted into two mailboxes; One of spam, one of ham. It is important that these mailboxes are pure. By pure, we mean that the ham mailbox should contain absolutely no spam, and the spam mailbox should contain absolutely no ham.

To do a proper analysis, this email should come from many different sources. We were fortunate to have several people allow our tests to be run on their email. The sections below describe the sources of the email and how it was collected, along with the names we associate with each user's mailboxes. We also describe the steps taken to ensure the purity of the mailboxes.

3.1 Students

Student 1 and Student 2 are college students in computer science programs. Student 1 does not receive any spam at all, but has a large collection of ham. The ham mailbox from Student 1 (ST1) contains 1727 emails from November 2001 through September 2002. This was the personal inbox of that student, after most mailing lists had been filtered out. Student 2's mailbox (ST2) was similar to Student 1's. It has 495 emails, all ham, from March 2002 through November 2002.

3.2 Pure Spam

Because the above mailboxes were purely ham, we needed some spam to which we could compare them. The implications of using email from different sources is described later in Section

4.1.1. To get pure spam, we used free email accounts from sites like Hotmail and Yahoo. These accounts will be known as Free Email 1 (FE1) and Free Email 2 (FE2). To retrieve these emails, we used the freely available Linux programs *gotmail*[7] and *fetchyahoo.pl*[13].

The Free Email 1 account that we used was several years old and rarely received ham anymore. This account gathered 20-30 spam messages every day. From September 8, 2002 to September 23, 2002, 939 messages were collected, all spam. They were checked by hand to ensure there were no ham emails in the mailbox.

The Free Email 2 account represents a different kind of spam. The biggest difference between this account and the Free Email 1 account was that the email here was already marked as “bulk mail” by the account’s provider. They use a system where emails that are sent in bulk are filtered into a folder named “Bulk Mail.” The exact methods used by this filter are unknown and not available. The contents of this folder were downloaded nightly between August 10, 2002, and September 9, 2002. 867 spam emails were collected. These emails were also checked to make sure that there were no ham emails.

3.3 Professors

We were also able to enlist the help of two computer science professors in our research. To ensure their privacy, we were never given copies of their mailboxes. We provided them with scripts to run, and the results were emailed back to us. The purity of their spam and ham mailboxes was in their hands. To ensure that no spam emails were in the ham mailbox and no ham emails in the spam mailbox, we encouraged the professors to check them by hand. The scripts that we provided to them recorded which emails the filters got wrong. If an email was

accidentally placed in the wrong mailbox, there is a good chance that one of the filters would get it wrong. For example, suppose we are testing a filter that classifies emails correctly 95% of the time. If there was a ham email in the spam mailbox, there is a 95% chance that it would be marked as ham. Because it was identified as ham and was in the spam mailbox, this would be marked as a false negative and would be saved to a mailbox with any other false negatives. The user can then look through the mailbox of false negatives and if they see any ham emails, they can move them from their spam mailbox to their ham mailbox. This process is much less time-consuming than having to check several thousand emails. After checking the false positives and false negatives mailboxes and handling any misplaced emails appropriately, the scripts are re-run and these mailboxes are checked again for errors. While such a method does not guarantee the purity of the ham and spam mailboxes, it is a good way to double-check.

From Professor 1 (PR1), we received results from 777 ham emails and 777 spam emails. He filtered his spam out by hand over the past few months by saving it to a separate folder. This email was from July 2002 through February 2003.

Professor 2 filtered his email using SpamAssassin into a spam folder. He checked this folder periodically to see if there were mistakes and he corrected them by moving them to the correct folder. He gave us results from 1064 ham emails and 1064 spam emails. His ham was from August 1997 through February 2003 and his spam was from September 2002 through February 2003.

3.4 Small Company

Merely testing spam filters on computer science students and professors would not yield results that could be applied to everyone. Another source of data is needed, so we enlisted the help of a small business. This particular company is a small, family owned farm market that has sold locally grown fruits and vegetables for over 100 years. They have been selling gift packs through the Internet for the past few years and as a result, spam has become quite a problem. On most days, more spam is received than ham, and the owner has been searching for a solution.

We had several goals in mind with this part of the project. Primarily, we wanted to further expand our data collection. This small business offered us a different email environment than the students and professors could. Their email would test the filters in ways we could not. In addition to business messages, the employees also use their email accounts for personal email, which we hoped would provide interesting insight into how well the filters could distinguish between the two. Due to the popularity of their web site and the visible email addresses all over it, everyone at the company receives a large volume of spam.

The company provided us with a computer and access to it. We installed Red Hat 8.0 and configured it as an email server. This email server would act as middle-ware to their current email system. Fetchmail was used to retrieve messages from their current email server and messages would then be filtered and delivered into the local server's user mailboxes. It was then be possible to retrieve the messages from our server via a POP email client.

When new emails arrived, Bogofilter would analyze and flag them accordingly. We added a new header field to every email and inserted "*****SPAM*****" into the subject of all

potential spam emails. Users would then retrieve their email from our server and check the spam filter's accuracy.

When messages came in, they were added to whatever corpus Bogofilter thought they belonged in. If any mistakes were made, it could be harmful to the filter's accuracy, as it would be learning based on incorrect information. To remedy these mistakes, corrections were sent by Company Person 1 and Company Person 2 to a special address where all incoming email was automatically refiled into the opposite category and deleted from the original corpus.

We debated over the implementation for correction quite a bit. The most accurate method would have been for the users to forward all mail to special accounts set up to receive spam and ham. We decided against this method because of the amount of effort required from the user. Already receiving hundreds of spams a week, this approach was an unacceptable solution. We decided to use a system where the users would send in only those emails that were improperly classified by the filter. This system of correction required the least amount of effort on the part of the users, especially since, if the filter worked, the number of errors made by Bogofilter would go down as time went on. Also, since the filter was completely dependent on learning, there was plenty of incentive to be faithful to our project and send in all corrections.

The biggest threat to our data collection was not faulty software. Uncooperative users would have made this whole section of the project a waste of time. Luckily, the volunteers had been plagued by spam for quite some time, and were eager to find a solution themselves. Still, we had some concerns about the integrity of our data. Neglecting to send any corrections will degrade the quality of the filter. On a busy day, it would be quite simple to forget, or

just not have time, to send in the spam corrections.

The email server processed 955 hams and 1005 spams for Company Person 1, and 308 hams and 358 spams for Company Person 2 from December 2002 through February 2003. After the Bogofilter trial period was complete, we copied the databases off of the email server for analysis in their final state. We also manually scanned through the email, separating it into spam and ham. Using the correction account's email, this was a simple and quick task. We found several emails that appeared to have never been corrected. The reason for this manual sorting was to be able to use the email again for additional testing. After the analysis was complete, we informally interviewed the employees using the filter to find out how well the filter worked at their site.

Table 3.1: Summary of Email Data

Person	# Ham	# Spam	Total	Start Date	End Date
Student 1	1727	0	1727	11/2001	9/2002
Student 2	495	0	1727	3/2002	11/2002
Free Email 1	0	939	939	9/8/2002	9/23/2002
Free Email 2	0	867	867	8/10/2002	9/9/2002
Professor 1	777	777	1554	7/2002	2/2003
Professor 2	1064	1064	2128	8/1997 (ham), 9/2003 (spam)	2/2003
Company Person 1	955	1005	1960	12/2002	2/2003
Company Person 2	308	358	666	12/2002	2/2003
Totals	4626	5010	9636		

3.5 Summary

From all these sources, we collected 4626 ham emails and 5010 spam emails. The messages analyzed come from two sources from each category. In addition to providing more emails for our primary analysis, the small business allowed the additional testing of Bogofilter. All

emails were manually sorted by their owner to insure the integrity of the data. Table 3.1 summarizes all the emails involved in the analysis and their origin.

4 Methodology

There are many ways one can go about testing spam filters. When we started this project in September of 2002, Bayesian filters were brand new and untested, other than Paul Graham's own tests. Rather than spend our time doing brief looks at many different spam filters, we decided it would be better to do an in-depth look at only two filters: The Bayesian filter, Bogofilter, and the widely-used rule-based filter, SpamAssassin.

Our analysis includes a side-by-side comparison of Bogofilter and SpamAssassin on all of our available mailboxes. We also wanted to look at how quickly Bogofilter learns and at what point it works better than SpamAssassin (if at all). Additionally, SpamAssassin can be analyzed by itself in great detail to determine the effectiveness of the filter at its default threshold and the effects of changing this threshold.

In addition, we look into how email differs from user to user. We examine how effective whitelists and blacklists are when based solely on who an email is from. The results from this test can be used to show how many different addresses each of our test mailboxes receives email from. In addition, using Bogofilter's word frequency databases, we can calculate how similar the email one user gets is to that of another user, overall.

The sections below describe how we went about performing the tests described above.

4.1 Bogofilter versus SpamAssassin

Since Bogofilter is a learning filter, it makes sense to test it over time. The exact algorithm we used is described in Figure 4.1. Basically, there are two mailboxes as inputs; one of pure spam and one of pure ham. It is important that these mailboxes are pure as any emails

placed incorrectly will skew the results. The basic idea is to test Bogofilter on a single email and record whether or not it is correct. We know if the email is spam or not by knowing which mailbox it came from. For example, if the email we are testing is from the ham mailbox, and Bogofilter marks it as spam, we record a miss for Bogofilter. Then, to teach Bogofilter, we would add the email to Bogofilter's ham corpus (even though it was marked as spam). Next, a spam email would be tested using Bogofilter, its result recorded, and its contents added to Bogofilter's spam corpus. We alternate between testing spam and ham and use the same number of emails from each mailbox. Some emails never get tested because the user could have one mailbox larger than the other. For instance, if a user has 500 ham emails and 700 spam emails, we only test 500 from each mailbox.

Initially we checked consecutive emails starting at the beginning of each mailbox. This method meant that for the larger of the two mailboxes, emails towards the end were never checked. We corrected our tests to use the concept of "stride." Stride means that for the larger mailbox, the emails we test are equally spaced throughout the whole mailbox. We feel this is a better simulation because if the two mailboxes we are testing cover the same time period, using stride keeps the dates for the emails being tested closer to each other. All of results presented later in this paper use the idea of stride.

The algorithm that we used for testing SpamAssassin was almost identical to the one we used with Bogofilter. This algorithm can be seen in Figure 4.2. Because SpamAssassin is not a learning filter, there is no step to train the filter. Instead, we simply test and record results.

Testing SpamAssassin over time on its own does not really make sense. One would expect the accuracy to be pretty constant. If this is true, there are much better ways to analyze

```

hamcount  $\Leftarrow$  the total number of ham emails
spamcount  $\Leftarrow$  the total number of spam emails
limit  $\Leftarrow$   $\min(\textit{hamcount}, \textit{spamcount})$ 
for  $i = 1$  to limit do
  hamindex =  $\lfloor (i * \textit{hamcount}) / \textit{limit} \rfloor$ 
  spamindex =  $\lfloor (i * \textit{spamcount}) / \textit{limit} \rfloor$ 
  if Bogofilter thinks ham[hamindex] is spam then
    Record false positive
  else
    Record correct ham identification
  end if
  Add ham[hamindex] to Bogofilter's ham corpus
  if Bogofilter thinks spam[spamindex] is spam then
    Record correct spam identification
  else
    Record false negative
  end if
  Add spam[spamindex] to Bogofilter's spam corpus
end for

```

Figure 4.1: Bogofilter Analysis Algorithm

```

hamcount  $\Leftarrow$  the total number of ham emails
spamcount  $\Leftarrow$  the total number of spam emails
limit  $\Leftarrow$   $\min(\textit{hamcount}, \textit{spamcount})$ 
for  $i = 1$  to limit do
  hamindex =  $\lfloor (i * \textit{hamcount}) / \textit{limit} \rfloor$ 
  spamindex =  $\lfloor (i * \textit{spamcount}) / \textit{limit} \rfloor$ 
  if SpamAssassin thinks ham[hamindex] is spam then
    Record false positive
  else
    Record correct ham identification
  end if
  if SpamAssassin thinks spam[spamindex] is spam then
    Record correct spam identification
  else
    Record false negative
  end if
end for

```

Figure 4.2: SpamAssassin Analysis Algorithm

it than plotting it over time, however we are interested in comparing it to Bogofilter. We wanted to be able to pinpoint the number of emails required for Bogofilter to be at least as accurate as SpamAssassin.

The above algorithms were implemented using a combination of the scripting languages Bash and Perl. The majority of the code was done in Bash for its simplicity and ease of running other programs. The actual running of the filters on emails was done inside Procmail. Both filters were designed to be done in Procmail, and Procmail provides easy methods for saving emails and stripping headers. Perl was used for post-processing the results outputted from Procmail. Bogofilter version 0.9.1 and SpamAssassin version 2.41 was used for the mailboxes from the students, small company employees, and free email accounts. The accounts from the professors were tested on a different computer that had version 0.9.1.2 of Bogofilter and version 2.31 of SpamAssassin.

4.1.1 Headers or No Headers?

In our initial tests of Bogofilter, after only a handful of emails it had 100% accuracy. While this certainly looked good for Bogofilter, it was unlikely that any learning filter could learn in only two or three emails. The problem was that the mailboxes we were testing on were from two different sources. The ham was from Student 1's mailbox, and the spam from the Free Email 2 account. Bogofilter quickly learned that the address of Student 1's email server meant ham, and the address of Free Email 2's mail server meant spam every time! In fact, even for emails from the same source, sometimes Bogofilter was associating names of months with ham and spam if the mailboxes were from different time periods.

Procmail supports only sending the body of emails to the filters, but we opted to pipe

the emails through a script that would remove all lines of the header except “From:” and “Subject:”. These two lines should be safe to include, regardless of to whom the email was addressed. We only removed headers when absolutely necessary, like when using mailboxes from different sources. For email from the same source and time period we tried to use headers whenever possible. In our analysis section we will clearly state for each test if we used headers or not.

4.2 Whitelists and Blacklists

In Sections 2.2.2 and 2.2.3 we talked about using only whitelists or blacklists to filter emails. In a whitelist, all email from a new sender would be blocked until the receiver gets an email from them and somehow approves their email address. This concept only works if the number of different email addresses the receiver gets email from is low. The assumption is that after running this kind of filter for a while, almost everyone the receiver would ever get email is already on the whitelist. The same theory can be applied to blacklists, however they are typically not as critical as blacklists should rarely block a ham message, but it is bad if it they do.

We constructed a test to see if whitelists are a reasonable method of filtering. Figure 4.3 shows the method we used of doing this test. The output of this algorithm would be a list of numbers, one for each email in the mailbox. These numbers represent the number of different email addresses seen at that point. For example, if the 100th line of output read “30,” it would mean that the first 100 emails in the mailbox were from only 30 different people. We would expect this to increase most rapidly at first and eventually plateau.

```
count  $\leftarrow$  0
mailcount  $\leftarrow$  the total number of emails in the mailbox
for i = 1 to mailcount do
  addr = the From: address of mailbox[i]
  if  $\neg$ exists(hash[addr]) then
    count  $\leftarrow$  count + 1
    hash[addr]  $\leftarrow$  1
  end if
  Output count
end for
```

Figure 4.3: New Email Addresses Over Time Algorithm

4.3 SpamAssassin Scores

SpamAssassin assigns a score to each email which is the sum of the weights of the rules that it matches. SpamAssassin comes with a default threshold of 5, but it is adjustable by the user. Since the scores assigned to each characteristic that SpamAssassin checks for seem rather arbitrary, it is important to test for SpamAssassin's best threshold setting. This could, of course, be different for every user.

In order to determine each user's appropriate threshold, it was necessary to analyze all the scores assigned by SpamAssassin for that user's email. Using the standard Unix "grep" command, we simply extracted all the scores from each spam and ham file and then used Microsoft Excel to create a cumulative distribution function (CDF) plot of these scores.

4.4 Data Analysis Script

All of the above methods of analysis were combined into one script to make it easy to use and more distributable. The script was initially written for our own PC running Linux, but we wanted to be able to distribute it to other people so that they could run it and send the results back to us. Given two mailboxes (one of spam, one of ham), the script would

run a series of tests on them. It first ran the Bogofilter vs. SpamAssassin tests described in Section 4.1. During these tests, the output of SpamAssassin was saved so that the assigned scores could be filtered out at a later time. Bogofilter's verbose output was also saved, which detailed the words it used to decide if an email was spam or not, along with the probabilities assigned to each word. This data from Bogofilter was never used other than to make sure it was working correctly. Following the above tests, the data to analyze new email addresses over time was collected. All of the data collected was packaged up in a file to email back to us.

In order to protect the user's privacy, the script did not make copies of any parts of any emails of the user. Many users consider their email to be confidential and private, and much care was taken when developing the script so that the result would protect the user's privacy and still provide us with the maximum amount of data.

4.5 Comparing Corpora

Making a universal spam filter is hard because of the differences between users' emails. We wanted to see how similar email is from user to another. If the email is similar, it should be easier to build a filter that will work well for everybody. Learning filters, like Bogofilter, work regardless of the differences between one user and another because they adapt to the individual's email. If using one Bogofilter database for multiple users, it is important to make sure that these users receive similar email. This test should give some idea of how email differs between users.

Comparing the email of different users presented difficult problems in several ways. First,

it was necessary to protect the user's privacy which made viewing their email directly out of the question. Second, we had no method of drawing a comparison between two sets of email.

To solve the first problem, we decided to use the databases created by Bogofilter after the user had run the data analysis script. From the database a list of words with their spam probabilities could be obtained using Graham's Algorithm. To compare two corpora, we used the following procedure with Microsoft Access:

1. Create a separate table of words with their respective spam probabilities for each user.
2. Append all words found only in one table to the other table with a neutral probability of 0.5.
3. Create a combined table with each word and the respective difference in probability between the tables. Use this table to create a histogram of the differences.
4. Take the sum of the probabilities and divide it by the number of words in the combined table.

This procedure gives a number that we use as the percentage difference between the first and second tables.

4.6 Summary

The series of tests we constructed look at several different areas of spam filtering. We do a side-by-side comparison of the effectiveness of SpamAssassin and Bogofilter. This provides a rough look at how well the filters work, and can be used to show how long it takes a Bayesian filter to reach the effectiveness of a static, rule-based filter. In addition to this, we outlined our method of examining the effects of changing SpamAssassin's threshold. The results from this test can be used to see if the default threshold of 5 is a good choice, and how much the optimal choice varies from user to user.

Our tests also include a brief look into whitelists and blacklists using only the “From:” field of an email. The results of this test will not only show the effectiveness of a simple whitelist/blacklist, but will also characterize a user’s email by showing how many distinct email addresses the user gets email from. In addition, we also describe a method of comparing Bogofilter’s databases between users, to get an idea of how similar one user’s email is to another. This last comparison is important because it shows how hard spam filtering can be. If two users get different email, it is unlikely that there will be one universal filter that will work well for both of them. The next chapter shows the results of these tests.

5 Analysis

In this chapter we describe the results of testing both filters extensively. Starting with the comparison of Bogofilter to SpamAssassin and analyzing their performance on each set of data collected. Next are the results from the whitelist and blacklist tests that were run on each data set. SpamAssassin is then tested by itself to determine how to find the best threshold for each user in our project. Bogofilter is then analyzed in detail with the comparison of the corpus from each user. Lastly, we look into how the small company users felt Bogofilter worked at their business.

5.1 Spamassassin vs. Bogofilter

In this section we analyze the performance of SpamAssassin and Bogofilter. We ran the tests described in Section 4.1 on all of our mailboxes. We had mailboxes of both spam and ham from four users: Company Persons 1 and 2, and Professors 1 and 2. We also had the two ham mailboxes from Students 1 and 2 along with two spam mailboxes from Free Emails 1 and 2. From these mailboxes we were able to make four combinations and we tested all four. Because Bogofilter would have had an easy time distinguishing emails from two different mailboxes (see Section 4.1.1), we ran Bogofilter using only the “From” and “Subject” headers for the four mailbox combinations. SpamAssassin, however, was always run with full headers. Had we tried to run SpamAssassin on emails with missing headers, it would have likely flagged them as spam because SpamAssassin has rules regarding missing headers. Keep in mind that in the graphs below, Bogofilter was “crippled” in a sense because it had limited headers on the combinations of Student and Free Email mailboxes.

To make an estimation of the effects of not using most headers with Bogofilter, we ran our Bogofilter test script against the same mailbox, both with and without headers. We chose to use the mailbox of Company Person 2 because we had full access to it (as opposed to the Professors' mailboxes), and its results were typical of Bogofilter, in our opinion. We tested on 358 email messages each of ham and spam. With headers, there were 38 false positives (10.6%) and 35 false negatives (9.8%). With only the "From" and "Subject" headers, these changed to 50 false positives (14%) and 29 false negatives (8.1%). By removing most headers, there were more ham emails mistakenly identified as spam, and less spam emails mistakenly identified as ham. The overall accuracy decreased, from 89.8% to 88.9%. In this single example, the change was slight. We expect the effect of not using most headers to be small for all mailboxes, as we believe the most important header information is in the "Subject" and "From" fields.

Figure 5.1 shows how well Bogofilter and SpamAssassin performed over time on Company Person 2's ham, as an example. This figure started as raw data with a 1 representing a correct classification and a 0 representing a mistake. To smooth out the results, a moving average of 10 was used so each point on the graph represents the average of it and the previous 9 points. Finally, this data was plotted using gnuplot with the "smooth bezier" option. We feel this method provides the best visual interpretation of the results of our tests. Using a shorter moving average makes the dips much more extreme and a longer one just smooths the whole plot out too much.

This graph does not clearly show which filter is better. The performance of Bogofilter and SpamAssassin on each mailbox is later summarized in several tables and graphs to make it more clear how accurate they were overall. Figure 5.1 is included to give a sense of how

Bogofilter’s performance changes over time for ham. It starts off low, but increases quickly to reach the accuracy of SpamAssassin. This graph that Bogofilter does not require much learning before it is accurate. Figure 5.2 shows a similar plot for Company Person 2’s spam. In this case, Bogofilter actually outperforms SpamAssassin through most of the mailbox. It is expected that Bogofilter will get the first few ham emails correct and the first few spam emails incorrect in each mailbox. In other words, Bogofilter will mark the first few emails as ham, regardless of what they are. To avoid false positives, Bogofilter is weighted so that it will mark emails as ham if it is unsure.

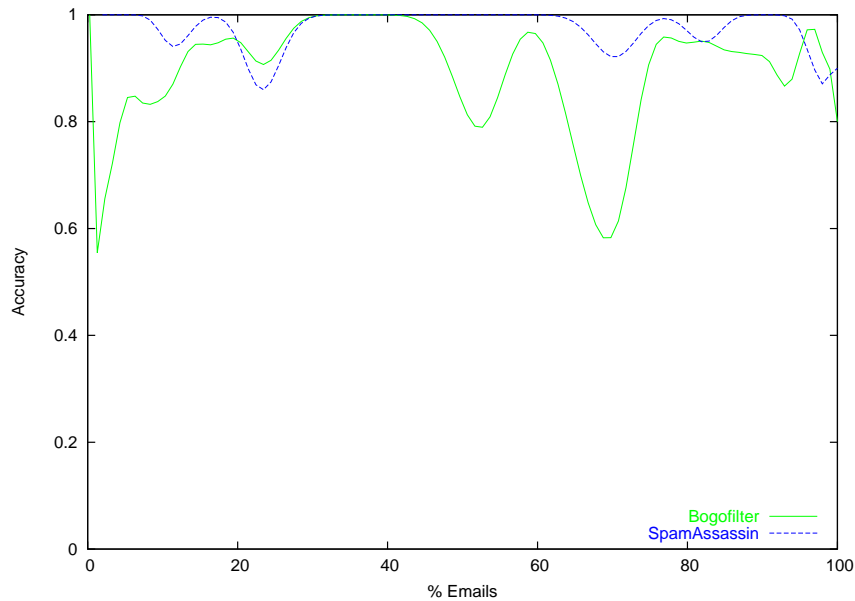


Figure 5.1: Company Person 2’s Ham Over Time

Tables 5.1 and 5.2 show the results of every single email tested. These numbers are unrepresentative of Bogofilter, however, because Bogofilter needs a learning period before it can be relied on. Tables 5.3 and 5.4 show the results of the same tests with the results of the first 50 emails removed. In our experience, 50 emails is more than enough for Bogofilter to start performing well. This can be seen in Figures 5.1 and 5.2, where Bogofilter performs

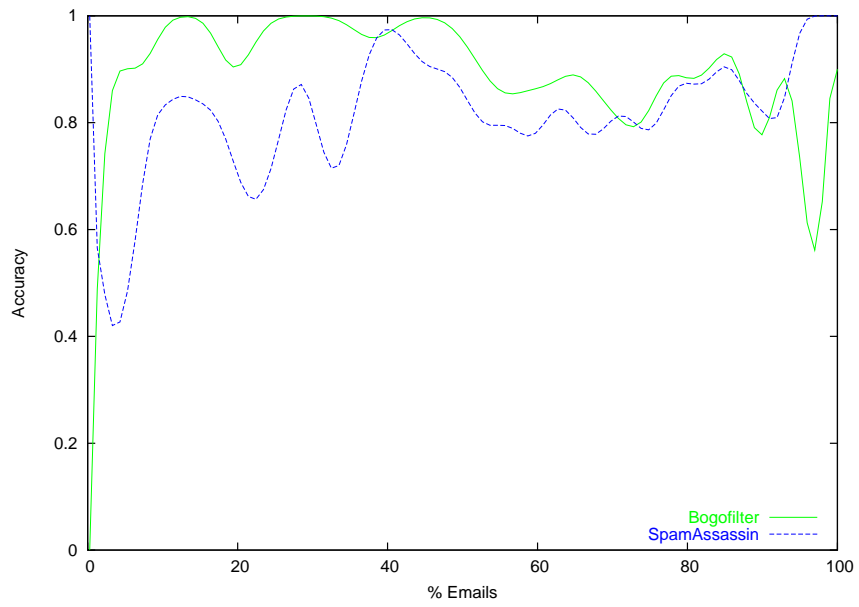


Figure 5.2: Company Person 2's Spam Over Time

well almost immediately.

The column and row headers are abbreviated to make them fit better. The columns in Table 5.1 represent the number and percentage of false positives (FP) for Bogofilter (BF) and SpamAssassin (SA). Table 5.2 shows the number and percentages of false negatives (FN) for Bogofilter (BF) and SpamAssassin (SA). The mailbox abbreviations were discussed in Section 3. The last four mailboxes are combinations of the two Student ham mailboxes with the two Free Email spam boxes. Remember that Bogofilter was only using limited headers on the last four mailboxes in each table.

Table 5.1: Summary of Bogofilter and SpamAssassin Performance on Ham

Mailbox	# Emails	BF # FP	BF % FP	SA # FP	SA % FP
PR1	775	20	2.58%	11	1.42%
PR2	1064	36	3.38%	19	1.79%
CP1	1005	37	3.68%	128	12.74%
CP2	358	38	10.61%	9	2.51%
ST1 - FE1	940	15	1.60%	81	8.62%
ST1 - FE2	878	22	2.51%	57	6.49%
ST2 - FE1	495	8	1.61%	12	2.42%
ST2 - FE2	495	22	4.44%	12	4.44%
All	6010	198	3.29%	329	5.47%

Table 5.2: Summary of Bogofilter and SpamAssassin Performance on Spam

Mailbox	# Emails	BF # FN	BF % FN	SA # FN	SA % FN
PR1	775	61	7.87%	118	15.22%
PR2	1064	11	1.03%	67	6.30%
CP1	1005	450	44.78%	135	13.43%
CP2	358	35	9.78%	65	9.88%
ST1 - FE1	940	158	16.80%	64	6.81%
ST1 - FE2	878	73	8.31%	59	6.72%
ST2 - FE1	495	56	11.3%	24	4.85%
ST2 - FE2	495	16	3.23%	34	6.87%
All	6010	860	14.3%	566	9.4%

Table 5.3: Summary of Bogofilter and SpamAssassin Performance on Ham
(Excluding the first 50 emails)

Mailbox	# Emails	BF # FP	BF % FP	SA # FP	SA % FP
PR1	725	18	2.48%	11	1.52%
PR2	1014	28	2.76%	19	1.87%
CP1	955	32	3.22%	128	13.40%
CP2	308	30	9.74%	8	2.60%
ST1 - FE1	890	10	1.12%	77	8.65%
ST1 - FE2	828	10	1.21%	51	6.16%
ST2 - FE1	445	7	1.58%	12	2.70%
ST2 - FE2	445	15	3.37%	12	2.70%
All	5610	150	2.67%	318	5.67%

Table 5.4: Summary of Bogofilter and SpamAssassin Performance on Ham
(Excluding the first 50 emails)

Mailbox	# Emails	BF # FP	BF % FP	SA # FP	SA % FP
PR1	725	55	7.59%	105	14.48%
PR2	1014	9	0.89%	67	6.61%
CP1	955	433	45.34%	124	12.98%
CP2	308	31	10.06%	49	15.91%
ST1 - FE1	890	135	15.17%	63	7.08%
ST1 - FE2	828	66	7.97%	54	6.52%
ST2 - FE1	445	43	9.66%	23	5.17%
ST2 - FE2	445	10	2.24%	33	7.42%
All	5610	813	14.49%	518	9.23%

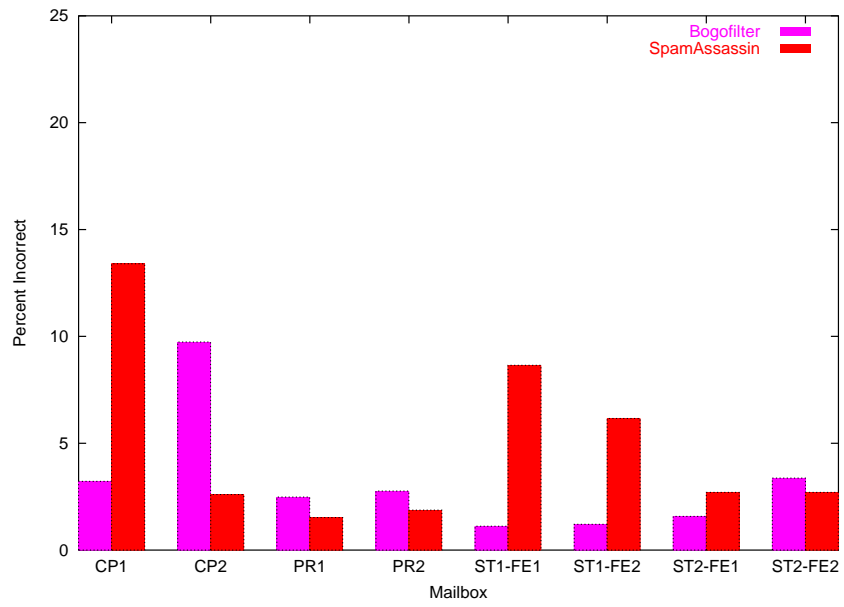


Figure 5.3: SpamAssassin Vs. Bogofilter on Ham Summary

Figure 5.1 shows the percentage of ham emails misclassified by Bogofilter and SpamAssassin. This figure excludes the first 50 emails, for a fair comparison. On four of the eight mailboxes, Bogofilter performed better by having less false positives than SpamAssassin. It is interesting to note that three of these four were from the Student and Free Email combinations, where Bogofilter was run with only limited headers. It is also interesting that for Company Person 1, Bogofilter significantly outperformed SpamAssassin, while for Company Person 2 the results were just the opposite.

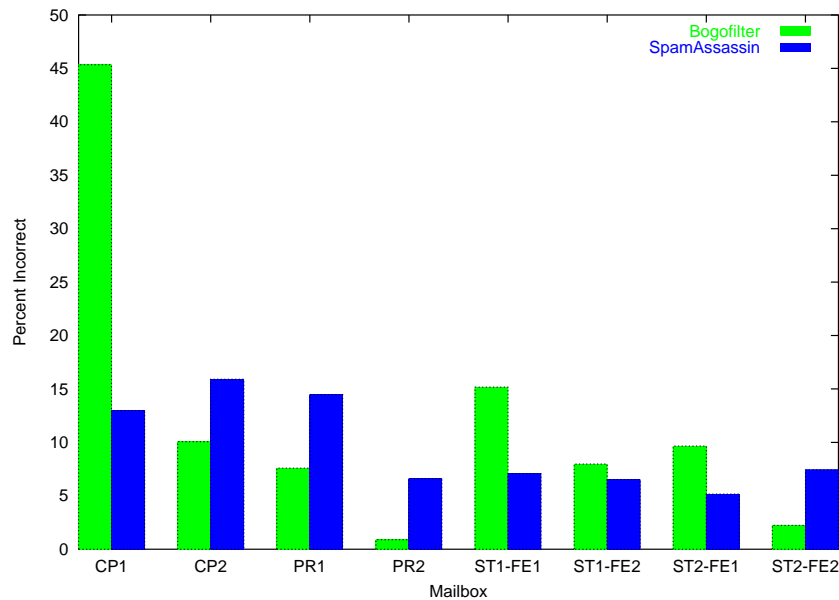


Figure 5.4: SpamAssassin Vs. Bogofilter on Spam Summary

Figure 5.1 shows how well Bogofilter and SpamAssassin performed at classifying spam. This figure also excludes the results of the first 50 emails. Bogofilter outperformed SpamAssassin on four of the eight mailboxes. In fact, Bogofilter outperformed SpamAssassin on the same four mailboxes where SpamAssassin outperformed Bogofilter on the ham. This result means that neither filter completely outperformed the other filter on any one person's mailbox. The most notable occurrence in Figure 5.1 is that Bogofilter misclassified around 45%

of Company Person 1’s spam. Considering how well Bogofilter worked for Company Person 1’s ham, one could conclude that Bogofilter was marking most emails as ham, regardless of what they were. In the case of ham, it worked well, however for spam it performed horribly.

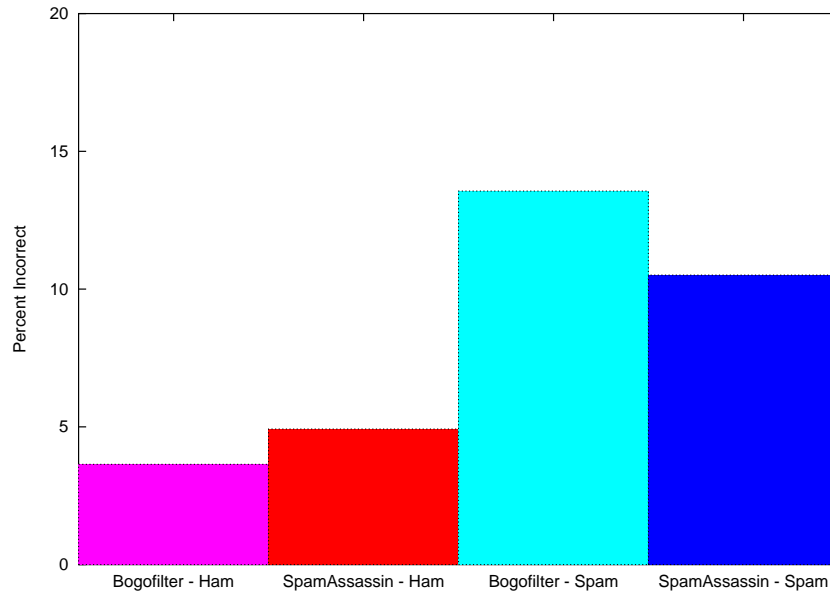


Figure 5.5: SpamAssassin Vs. Bogofilter Overall Summary

Figure 5.1 shows the overall summary of Bogofilter and SpamAssassin. In short, Bogofilter did better for ham, while SpamAssassin did better for spam. This summary is an average of the percentages for each mailbox from the last two figures, with the four Student and Free Email combinations weighted half as much as the Professors’ and Company Persons’ mailboxes. This was done because the four combinations really only represent two ham mailboxes and two spam mailboxes and the different combinations obviously share many emails.

5.2 Whitelists and Blacklists

Whitelists and blacklists are only effective if the number of addresses a user gets email from is limited. We looked at how frequently new addresses appear for all of our datasets. It is

important for these tests that the emails are ordered somewhat chronologically or else the results will be misleading. These tests only look at the email address in the “From:” header.

Figure 5.2 shows the number of different email addresses a user receives ham email from proportional to the total number of email they get. If every email received was from a different address, it would be represented by a diagonal line from the lower left corner to the upper right. If every email was from the same address, then there would be a line along the X-axis. The values on the Y-axis are the number of addresses seen divided by the total number of emails. Both of the professors get email from roughly the same number of email addresses, as do both of the company persons. The lines for Student 1 and Student 2, however, are not all that close to each other. Observe that none of the lines level off and are steadily rising throughout the entire mailbox.

Whitelists should, in theory, block all spam. The variable is how much ham gets blocked, in this case, because it is from a previously unseen email address. For Company Persons 1 and 2, a simple whitelist would block about 20% of their ham. For Student 2, a simple whitelist would block around 40% of his ham.

Notice that for Professor 2, there is a large flat spot from around 25% of his mailbox through 50%. We later discovered that his mailbox was not ordered chronologically and had subfolders that contained mail from only one or a few users. When testing that section of his mailbox, no new email addresses were seen, and hence, a flat section appears on the graph.

Figure 5.2 shows the same graph only for spam. Note that the Y-axis is scaled differently from Figure 5.2. This graph is strikingly different from the previous one. Remember that if every email was from a different user, it would be shown as a straight line from one corner to the other. The lines in Figure 5.2 are close to this maximum. One could interpret this

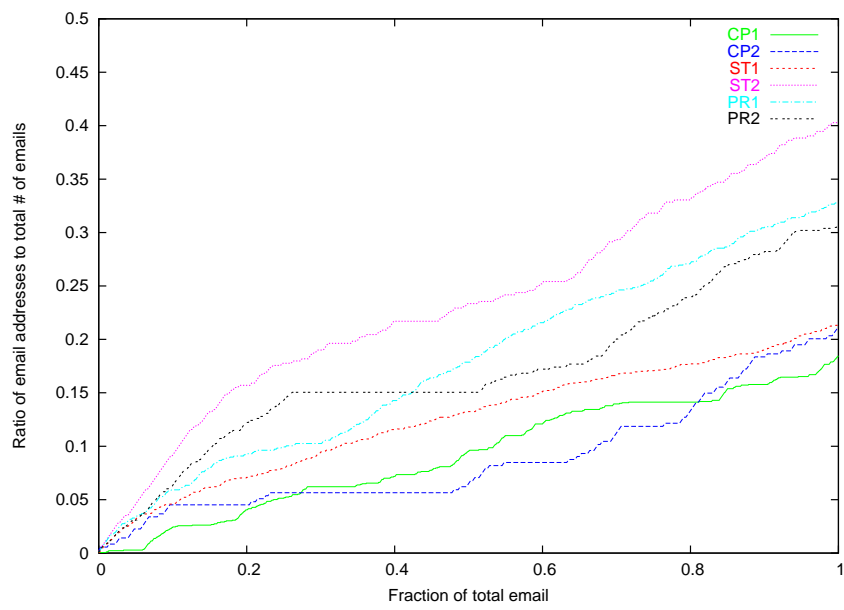


Figure 5.6: New Email Addresses in Ham

graph as showing that 60% to 80% of all spam comes from different users and that a blacklist would only stop the remaining 20% to 40%.

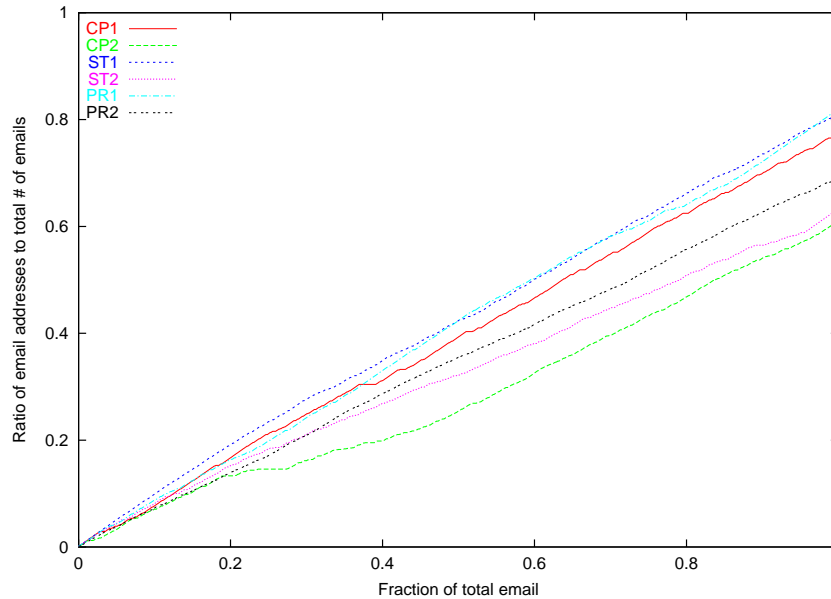


Figure 5.7: New Email Addresses in Spam

While the above graphs showed the number of new email addresses over time, the graphs below show the rate at which new email addresses appeared. The expected curve starts out high (as at first, all email addresses are new) and decreases over time, and levels off at some point. As the curves get closer to 0, it means that the number of new email addresses being seen is decreasing.

These graphs are essentially a numerical derivative of the previous plots. Every 50 emails, the number of new addresses seen in the previous 50 emails was calculated. This data was then plotted using Gnuplot’s “smooth bezier” setting. This fits a bezier curve to the data to show its general trend. The actual data is not nearly as smooth as these graphs show, but we feel the bezier curve shows the trends more clearly.

Figure 5.2 shows the rate of new email addresses for ham. Four of the five mailboxes tested

started high and decreased quickly, as we expected. After the initial drop, the percentage of new email addresses is in the 10% to 40% range. Regarding whitelists, this means that for a user like Company Person 1, a whitelist would block about 20% of their ham from being received. For Student 2, a whitelist could block as much as 40% of their ham from being received. Note how the line representing Professor 2's ham starts high, dips low, and then rises again. As discussed previously, Professor 2's ham was not ordered chronologically and the curve is not representative of how this professor's actual email would perform.

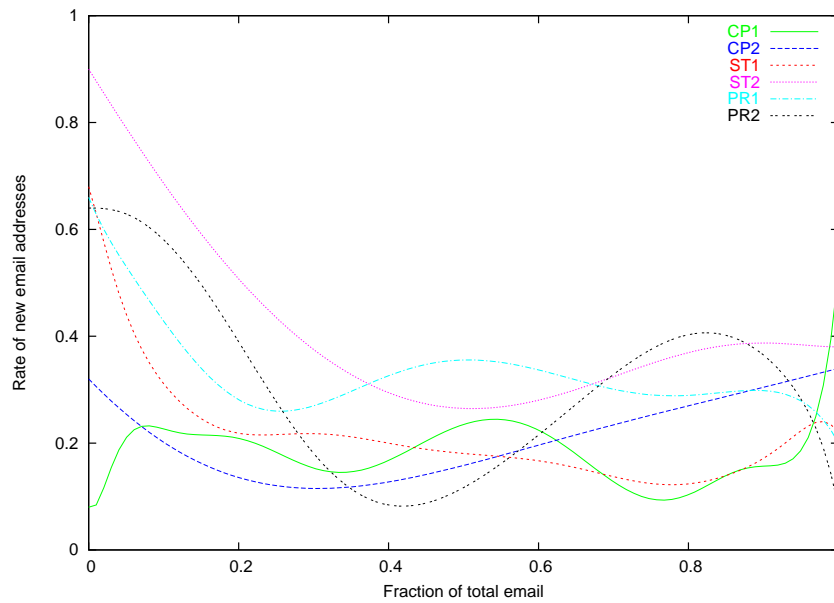


Figure 5.8: Rate of New Email Addresses in Ham

Figure 5.2 shows the rates of new email addresses for spam. The rates at which new email addresses are seen is much higher for spam. Company Person 2 has a short dip at one point, but otherwise the rates stay above 60% the entire time. This means that for these users, a blacklist could block at most 40% of their spam. For the Free Email 1, a blacklist would stop about 20% of its spam.

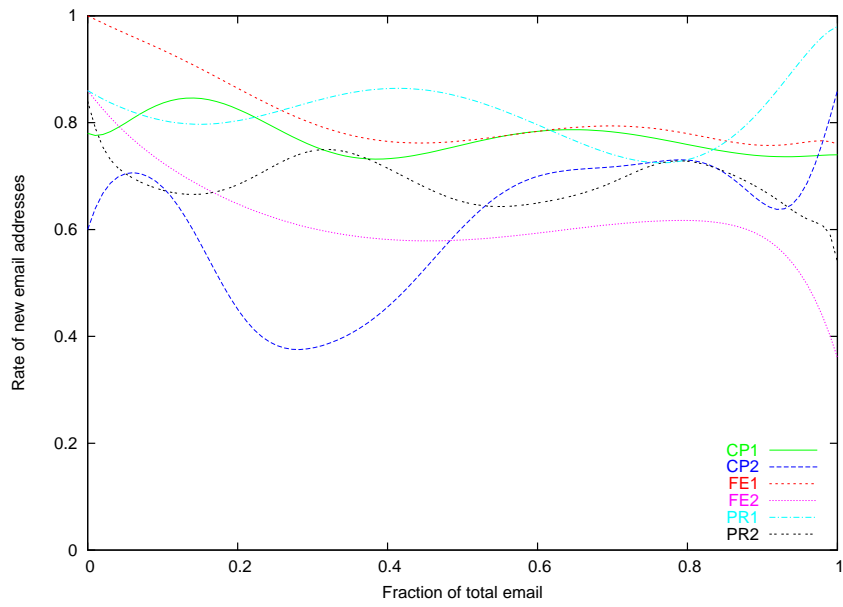


Figure 5.9: Rate of New Email Addresses in Spam

5.3 Spamassassin Alone

In this section we investigate the characteristics of SpamAssassin by itself. The intent is to determine how to adjust SpamAssassin's threshold to optimally filter spam for each user. It is also interesting to look at how different each user's threshold should be.

5.3.1 Finding the Right Threshold

Because SpamAssassin outputs the final score of an email in the body of that email, it is not necessary to run the filter repeatedly to determine how well it performs due to changing its threshold. By plotting a cumulative distribution function of the scores for a particular user, it is possible to see the effects of setting the threshold at any point.

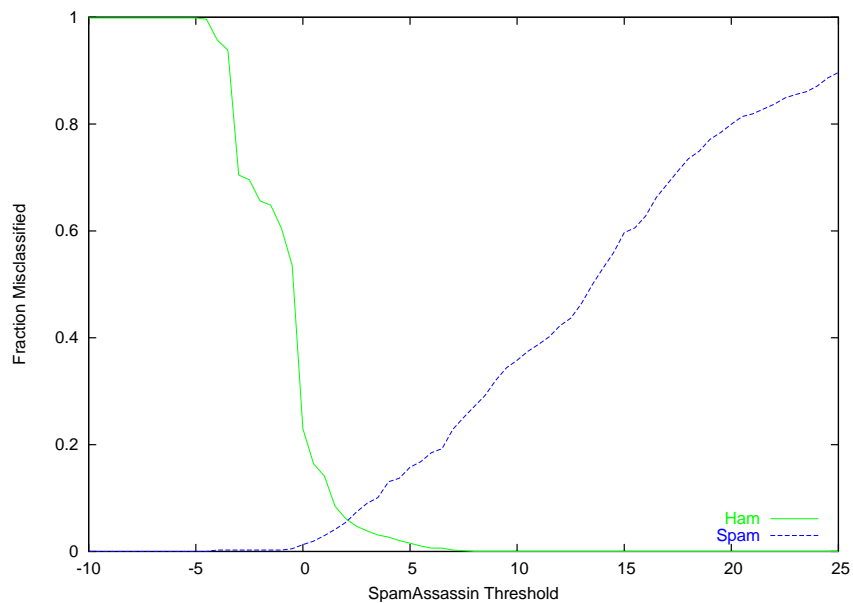


Figure 5.10: CDF - Professor 1's SpamAssassin Scores

Figure 5.3.1 shows the plot of Professor 1's email. The X-axis is the SpamAssassin score and the Y-axis is the percentage of email classified incorrectly. The ham line is actually the inverse of the CDF (1 minus the normal CDF value at that point). This adjustment allows

the user to view both the ham line and the spam line in the same fashion. To determine the effectiveness of SpamAssassin at a particular threshold, pick the point on the X-axis for that threshold and note the height of the ham and spam lines. The height of the ham line represents the number of ham emails that would be misclassified as spam. The height of the spam line represents the number of spam emails that would be misclassified as ham. If the user were strictly interested in minimizing both ham and spam, they would set their threshold to wherever the ham line crossed the spam line.

If Professor 1 was interested in minimizing false positives, the professor would set SpamAssassin threshold to approximately 6, where the ham line is nearly 0. At 6 however, SpamAssassin will let about 20% of spam through. Ideally, there would be a point on the graph where there are 0 misclassifications for both ham and spam. Unfortunately, since that is not the case, Professor 1 must make a compromise on how much ham to risk being blocked and how much spam to let through.

Of particular interest on these graphs is the slope of the ham line and the slope of the spam line. On Professor 1's graph, the ham line has a steep downward slope, which means that most of the professor's ham email falls within a small range of SpamAssassin scores. The spam line, on the other hand, has a much more gradual slope, meaning the spam messages have a broad range of scores they can receive. Near the x-axis, both lines appear to stretch as if they were approaching an asymptote. The graph shows that, although most ham is easily identified, Professor 1 still receives some ham that looks spam-like.

Professor 2's graph is shown in Figure 5.3.1, and is similar to Professor 1's. Professor 2 has a steeper slope on his spam line, which means the received spam is generally farther from the received ham than Professor 1's.

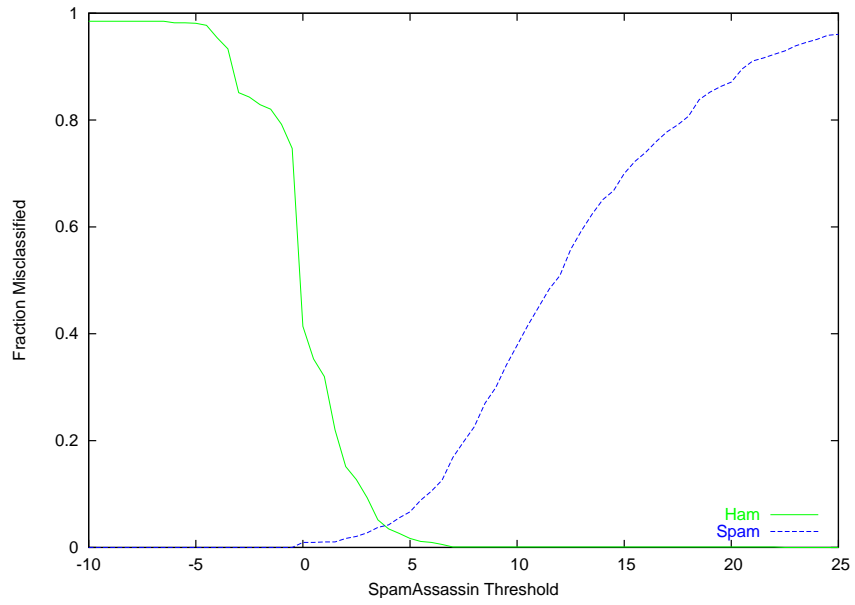


Figure 5.11: CDF - Professor 2's SpamAssassin Scores

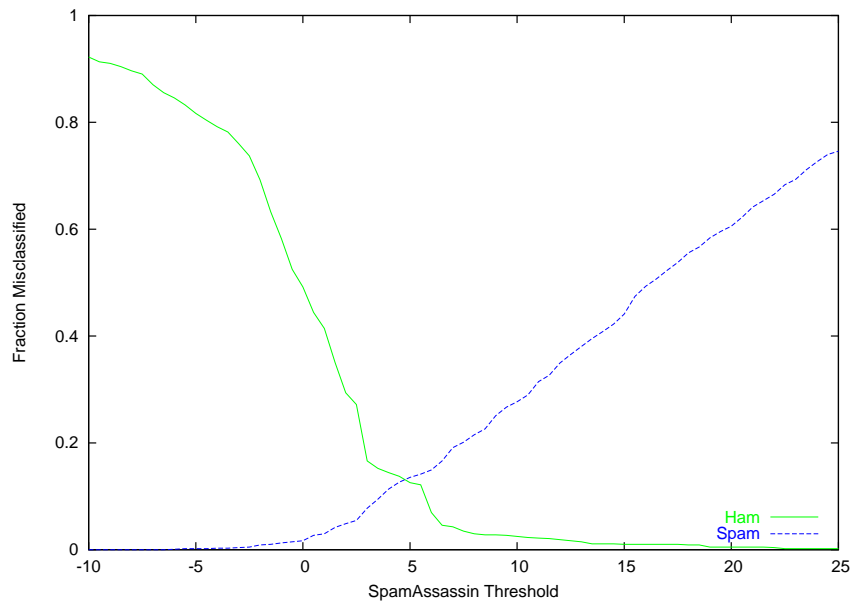


Figure 5.12: CDF - Company Person 1's SpamAssassin Scores

Figure 5.3.1 clearly shows that SpamAssassin did not do well with Company Person 1's email. Minimizing false positives would mean setting the threshold at about 13.5 and letting almost 40% of spam through! The height of the intersection of the spam line and the ham line shows how difficult it was for SpamAssassin to classify email correctly. 13% of Company Person 1's spam had a score of less than 5, while almost 12.5% of the ham had a score of more than 5.

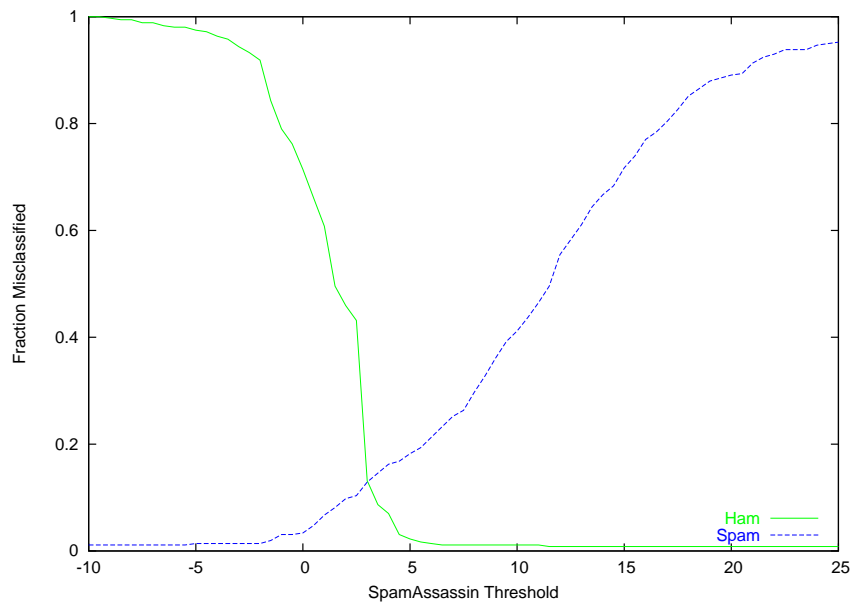


Figure 5.13: CDF - Company Person 2's SpamAssassin Scores

SpamAssassin had a much easier time classifying Company Person 2's email than Company Person 1, as shown in figure 5.3.1. The intersection of the spam line and the ham line is at about the same height, but the ham line continues to plunge to the right of the intersection and is not extended like on Company Person 1's graph. This means that Company Person 2 could set their threshold to 6.5 and this would minimize false positives without a gross amount of spam still penetrating the filter.

Figure 5.3.1 shows both students and both free email accounts. Student 1's ham line

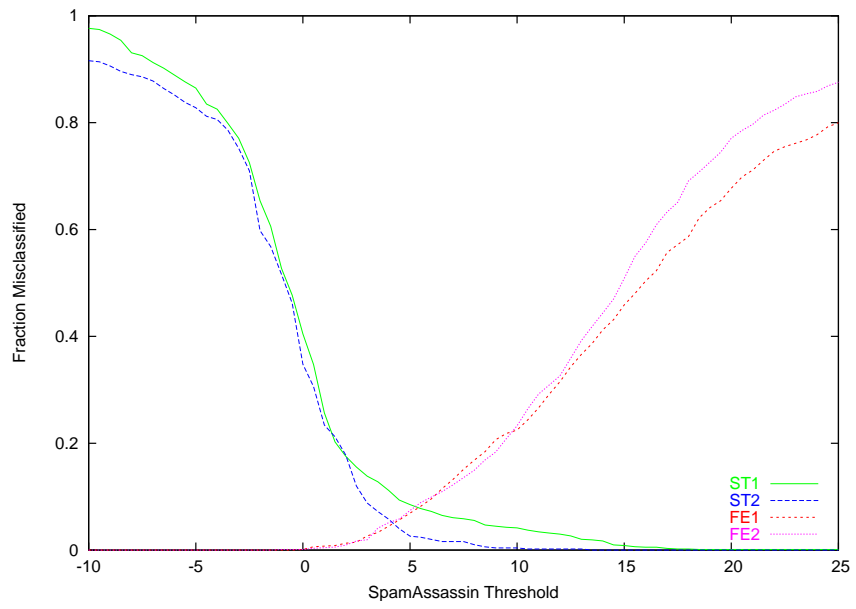


Figure 5.14: CDF - ST1, ST2, FE1, & FE2's SpamAssassin Scores

shows the difficulty that SpamAssassin had with the student's email. It is interesting to note that both free email accounts had similar lines. Although the spam in each account is different, they both had the same distribution of SpamAssassin scores.

5.4 Bogofilter Alone

In this section we dig deeper into how bogofilter works and try to extract characteristics from the corpora it creates. These characteristics provide for a method to compare one user's corpus to another user's corpus. From here, we can estimate the effectiveness of using a shared corpus across multiple users.

5.4.1 Corpus Analysis

Merely analyzing the effectiveness of Bogofilter did not seem sufficient. Rather than looking at how Bogofilter analyzed each email, looking at the spam and ham databases themselves should provide interesting information. This is not a simple task, however, because each corpus contains many thousands of words. We also wanted to find a way to compare two corpora to see how alike they are. The method for comparing one Bogofilter corpus to another is described in Section 4.5.

5.4.2 Student 1 vs Free Email 1

Table 5.5 shows the most commonly found words in the corpus of 940 ham emails from Student 1's mailbox and 940 spam emails from Free Email 1. The most common words in this corpus are clearly not words that will be used often to determine whether or not an email is spam. Most of them seem to be close enough to 50% that they would not normally

Table 5.5: Corpus of Student 1 vs Free Email 1

Word	Spam Probability	Total Occurrences
the	43.47%	4957
you	53.16%	4244
and	48.48%	3909
http	73.29%	3871
from	51.34%	3798
for	46.03%	3252
this	52.37%	2994
your	60.96%	2792
html	79.45%	2632
subject	48.05%	2329
are	48.76%	2016
arial	79.00%	1957
that	39.73%	1943
here	71.41%	1938
click	79.64%	1906
with	48.39%	1899
email	57.54%	1863
have	41.70%	1760
will	40.34%	1626
our	57.28%	1615
not	52.67%	1443
text	60.39%	1386
nbsp	73.51%	1374

be used at all. There are a few that could be significant. The word “http” was found more than 3871 times, approximately twice per email. From the spam probability, its easy to see that spammers use the word “http” quite often, whereas legitimate email does not usually contain “http” (but sometimes does).

The words “arial” and “click” had the highest spam probability with 79%. While not as many occurrences as “http”, these are clearly indicative of a spam.

Tables 5.6 and 5.7 list the most commonly found ham and spam words, respectively, with their counts in Student 1’s email. There is no Spam Probability in these lists because all

Table 5.6: Student 1 vs Hotmail - Top Ham Words

Word	Total
wpi.edu	911
worcester	271
wpi	271
tuesday	238
guitar	198
bass	171
april	168
cs.wpi.edu	165
musician's	147
march	147
j.pl	139
dave	134
article	127
musical	125
campus	121
www.wpi.edu	115
bands	115
musicians	114
guests	111
december	110

Table 5.7: Student 1 vs Hotmail - Top Spam Words

Word	Total
collapse	291
border-collapse	274
mortgage	262
loan	131
busycorp.net	120
t.pl	117
irewardstech	106
penis	106
to.pl	99
unsub.cgi	90
obligation	89
www.dagny.com.biz	84
link2buy.com	84
remove.html	82
opt-in	74
www.virtumundo.com	73
cs.jsp	71
opted	70
pill	69
text-align	69

of the top ham words have probability 0% and all of the top spam words have probability 100%.

From the ham words, it is easy to see that this student gets email from a musical mailing list and from school.

The spam list shows some of the topics of rather common spam and http tags used in many spam emails. Also in the spam list are companies which frequently send out spam, namely “busycorp.net”, “www.dagny.com.biz”, “link2buy.com”, and “www.virtumundo.com”. It is rather ironic that the word “opted” is in the list. After browsing through the spam messages, many have a line claiming that the reason the owner of the Free Email 1 account received this email is because he/she opted to, which is not true.

Including the analysis of all the other corpora would be redundant since they all convey essentially the same meaning. Looking at the top ham and spam words for each individual data set does not show information of interest as far as spam filter effectiveness goes. Each one had the expected results of the most common ham words being associated with the individual user and the most common spam words being words commonly found in spam.

5.4.3 Comparison of Company Person 1 and Company Person 2

It is more interesting to look at how the corpora compare to one another. Specifically, we wanted to know whether many users share the same ham words and whether or not all spam corpora were similar. We first looked at comparing the corpora from Company Person 1 and Company Person 2.

The comparison was made by creating a table of the words that were in both corpora and calculating the difference between the two probabilities. Company Person 1 and Company Person 2 get individual copies of some of the same email so we expected their corpora to closely match.

The corpus of Company Person 1 had 6,713 words that appeared in emails more than 10 times while the corpus of Company Person 2 had 7,971. There were 6,349 of these words that occurred in both corpora. Table 5.8 lists the top 20 words with the largest difference in probability between the two corpora. The count columns refer to the number of times that particular word appeared in the respective user's corpus. This table by itself does not show results of particular interest until one analyzes the probability differences using a continuous distribution function.

Figure 5.15 shows a continuous distribution function of several corpora comparisons.

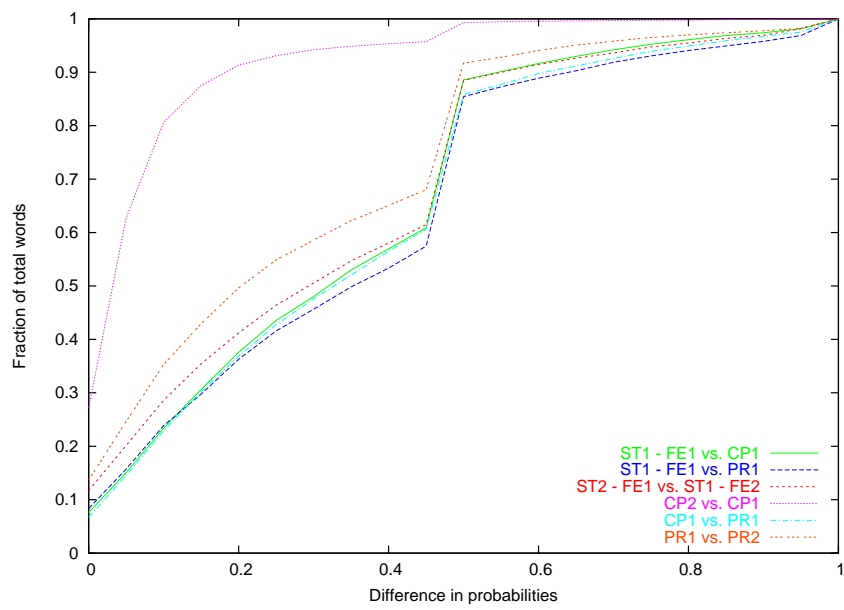


Figure 5.15: CDF: Corpus Comparisons

Table 5.8: Company Person 1 vs Company Person 2

Word	CP1 Prob	CP2 Prob	CP1 Count	CP2 Count	Difference
dvds	100.00%	42.37%	23	118	57.63%
absmiddle	100.00%	45.16%	21	62	54.84%
unsub	26.67%	80.52%	15	77	53.85%
movie	87.50%	33.75%	24	80	53.75%
x-accept-language	0.00%	50.00%	30	62	50.00%
implied	26.67%	76.60%	15	47	49.93%
reproductions	0.00%	46.15%	14	26	46.15%
reliance	91.67%	47.83%	12	23	43.84%
additionally	45.45%	88.68%	11	53	43.22%
decisions	0.00%	42.11%	18	38	42.11%
beach	6.25%	47.37%	16	38	41.12%
usa.net	100.00%	60.00%	18	30	40.00%
cholesterol	100.00%	60.00%	11	30	40.00%
https	10.53%	50.00%	19	54	39.47%
stimulus	0.00%	39.13%	14	23	39.13%
warranties	13.33%	52.38%	30	63	39.05%
upgrade	60.87%	22.06%	23	68	38.81%
cats	10.53%	48.48%	19	33	37.96%
cat	35.71%	72.97%	14	37	37.26%
abuse	20.00%	56.92%	35	65	36.92%

Looking at just one line on the graph shows how different the two corpora are. The steeper the slope of the line is the closer the two corpora are to each other. The sharp, linear increase on all lines at the 50% mark (except CP1 vs. CP2) is the result of those words listed in one corpus but not the other. These words tended to be either entirely ham (0%) or entirely spam (100%) words in one mailbox, and would be neutral (50%) in the mailbox that had not seen them before, so there was a group of words with a difference of exactly 50%.

Table 5.9 shows the differences between each corpus we tested. This table shows some rather interesting results. Company Person 1 and Company Person 2 had very similar corpora with only 7% difference. This similarity was expected since they both receive copies of many of the same exact emails. Professor 1 and Professor 2 was the next closest comparison

Table 5.9: Corpora Comparison Chart

	ST1	ST2	CP1	CP2	PR1	PR2
ST1	0%	31.18%	32.53%	32.93%	34.48%	34.78%
ST2	x	0%	34.29%	33.53%	32.05%	33.16%
CP1	x	x	0%	7.07%	33.49%	36.02%
CP2	x	x	x	0%	33.31%	34.84%
PR1	x	x	x	x	0%	26.89%
PR2	x	x	x	x	x	0%

with a 27% difference. Every other comparison was between 31% and 36%.

5.4.4 Small Company Test Results

Installing Bogofilter at the small company provided us with insight into the possibilities of implementing a learning filter for use by multiple people. Even though only two people were using the filter, Bogofilter had a difficult time identifying their spam and ham.

Actual statistics regarding Bogofilter’s effectiveness at the small company would be impossible to compute because on several occasions, Bogofilter’s ham and spam databases needed to be rebuilt from pre-sorted email and the process started over. On one occasion, one of the databases became corrupt and every email was being flagged as spam. On another occasion, Bogofilter was misclassifying more emails than it was correctly classifying.

After talking with both users of the filter, it became clear how sensitive a learning filter can be. Bogofilter’s accuracy was poor throughout the test period. Company Person 1 estimated that only 60% of all incoming email was being classified correctly. It was obvious the filter was not being trained as rigorously as it needed to be. This lack of training created a situation where the ham and spam databases were incorrect and did not apply specifically to either user.

5.5 Summary

The results from comparing SpamAssassin to Bogofilter are rather surprising. Neither was more effective than the other at sorting ham emails from spam emails. In addition, adjustments to the SpamAssassin threshold can make the filter behave more appropriately depending on the email a particular user receives.

Whitelisting and blacklisting, while effective at reducing false negatives and false positives, are still undesirable solutions when used on their own. For all the users we tested, the rate at which email from new addresses arrived for both ham and spam were just too great to make these solutions acceptable.

Comparing the corpora created by Bogofilter provided some interesting information. It would appear that each user's email isn't that different from any other user's email. But it also appears that the amount by which each user's corpus differs is consistent across all corpora. This implies that each individual user has a somewhat different set of email with many emails similar, but still having some that are different from everyone else.

The trial at the small company revealed that implementing Bogofilter for use by multiple users who share a corpus is not simple task. Maintaining integrity of the database relies on all users properly using the filter and making all necessary corrections in a timely manner.

6 Discussion

In this chapter we describe the meaning of the results from our analyses. Starting with the comparison of Bogofilter to SpamAssassin, followed by whitelists and blacklists. Additionally, the separate investigations into SpamAssassin and Bogofilter are summarized. Finally, we review the results from the small company experiment.

Both SpamAssassin and Bogofilter had inconsistent results in the analysis. In the comparisons of both ham emails and spam emails, each filter was better at exactly half of the data sets being tested. When Bogofilter performed better for a user's ham, SpamAssassin performed better on their spam. The reverse of this is also true, that when SpamAssassin performed better on a user's ham, Bogofilter performed better on that user's spam. Based on these results, despite Bogofilter and SpamAssassin using different methods of analyzing an email, their effectiveness is about the same.

Whitelists and blacklists are the most basic form of rule-based email filtering. For many years email programs have allowed users to sort their email based on who it is to (in the case of mailing lists) or who it is from. Spammers know that even the most inexperienced user will learn quickly how to block email from a particular address. Our tests confirmed that most spam email (60% to 80%) comes from an email address that the user has never seen before. This means that blacklists become lists of email addresses that a user has received spam from, but only 20% to 40% of the user's future spam email will come from addresses on this list. In other words, blacklists can reduce a user's spam, but the majority of it will still get past the blacklist.

Whitelists, on the other hand, can be useful if the user tends to only receive ham email

from a small, established group. Our tests showed that whitelists could effectively mark 60% to 80% of a user's email as ham correctly. The remaining email could either be marked as spam, or perhaps run through another filter (like SpamAssassin).

Whitelists and blacklists did not show promising results when used alone, but in combination with other filters, they can only improve the results of those filters, assuming that there are no addresses mistakenly listed.

The analysis of SpamAssassin's threshold revealed that the default setting of 5 was close to the best threshold for every data set. Slight adjustments of this threshold could be made by the user to conform to their personal preference, but any major change, either positive or negative, would result in a dramatic increase in misclassifications. This result is not surprising, as SpamAssassin's authors chose weights for rules using a genetic algorithm to minimize mistakes using the default threshold of 5.

Analyzing each user's corpus allowed a unique view of that user's email. Bogofilter provided us with databases listing how frequently different words appeared in a user's ham and spam. Using these databases, we were able to make comparisons between users on a word-by-word basis. The analysis showed that there was a consistent difference between virtually all the data sets we analyzed. Table 5.9 shows that almost every data set is between 25% and 35% different. This means that the average difference of the probabilities of words between any two given corpora is about 0.3. The graphs show that while there are similarities, each user's email is unique.

The Bogofilter trial at the small business was ultimately inconclusive. Quite a few problems cropped up while it was running and left the users rather unmotivated to participate fully for the length of the trial. When it was later discovered how much trouble Bogofilter

had in our primary analysis in identifying Company Person 1's spam, the frustration of the users is understandable.

When we first started the project we had hopes of finding one filter that would be much better than others. After performing these analyses and looking at many spam emails, we have concluded that filtering spam accurately across different users is difficult. While the majority of spam is easy to spot, there are some messages that are designed to get around filters. These messages try to masquerade as legitimate email by sounding conversational, often linking to a web site rather than advertising directly. For a filter to be accurate, it needs to understand the intention of the email and be able to decide if the user wants to see that email.

7 Conclusion

In this chapter, we discuss how our results can be used in future projects, along with what we could have done differently.

7.1 Bayesian or Bayesish?

Paul Graham's Bayesian filter algorithm has come under scrutiny from a programmer and mathematician named Gary Robinson [15]. Robinson argues that Graham made too many false assumptions when he was creating his Bayesian algorithm, and that the math he uses does not make as much sense to use. In fact, he believes that Graham's algorithm can only be called Bayesian when some obscure assumptions hold true. This aside, he offers several improvements to Graham's algorithm. These improvements include a new formula for calculating probabilities for single words that has more desirable results when a word has been used very infrequently. His other large change is to the formula to combine probabilities. His method uses all of the words in the email, instead of using just 15 words. Graham's method ignores most of the words in large emails because it only chooses 15 words.

Robinson's algorithm is included in newer versions of Bogofilter as the default algorithm. We did no formal testing of Robinson's algorithm, but for the small set of emails which we observed it on, it outperformed Graham's algorithm.

7.2 Improvements

There are several parts of this project that could have been done differently, if we had to do them again. Some of these changes would have had no effect on the analysis, while others

could have improved results. We offer these changes here to benefit anyone who may work on a similar project in the future.

7.2.1 Script Improvements

There are several changes to the scripts we wrote that would have made them run an order of magnitude faster, in addition to making it possible to reduce the number of scripts. These changes would have had no effect on the outcome of the scripts, but would have made them more portable to different systems for other users to run, in addition to making them faster.

Procmail could have been avoided altogether. For each email tested, a new instance of Procmail was run. The functionality of Procmail could have been simulated using Bash or Perl. Procmail is a good tool for testing filters on emails, however it is hard to work with for complex projects and has limited output abilities. We worked around Procmail's inability to make useful output by using a Perl scripts for post-processing, but these additional scripts could have been removed entirely by not using Procmail.

We also made the decision to use the mbox format for storing mailboxes. This format uses a flat file to store the emails. As a result, accessing the last email in a mailbox requires reading through all of the emails before it. In fact, in our scripts, for each email tested, the mailbox was opened, all emails up to the one we wanted was read, and then the mailbox was closed. This means that the execution time of our scripts scaled quadratically with the size of the mailbox. Mailboxes with 50 emails would be tested in under a minute, but ones with with 500 emails would take an hour.

To get around this problem, a format like MH[1] should have been used. MH format stores each email in a separate file, allowing any email to be accessed instantly. We did rewrite

some of the scripts to work with MH format, but it was not heavily tested or distributed.

7.2.2 Corpora Comparisons Improvements

Since, as far as we know, no one has ever tried to compare two bogofilter corpora before, we had make some decisions regarding how to calculate the difference between two lists of words with associated probabilities. In this report, we used the spam word list and ham word list combined into one list for each user and compared those. It would most likely be more accurate to compare them separately in the same fashion we used and discuss how one user's ham corpus compares to another user's ham corpus.

7.3 Whitelists and Blacklists

Our tests showed that whitelists and blacklists are not effective when used alone. Combining them with other filters, however, should yield different results. Assuming that no addresses are whitelisted or blacklisted incorrectly, these lists can only improve the accuracy of a spam filter. For example, a user has a filter setup where any email from a user on their whitelist always gets placed in their inbox. Similarly, any email from a user on their blacklist is delivered to a spam mailbox. Any email which is not on either list is sent on to a filter like SpamAssassin. Based on our data, 20% to 40% of a user's ham and 60% to 80% of a user's spam would not be on either list, and would be sent to this filter. This technique reduces the number of emails that the filter has to look at. Assuming that every email found on the whitelist or blacklist is placed correctly, the accuracy of the overall system is improved over using just the filter alone (with no lists).

7.4 Site-Wide Implementations of Bogofilter

Bogofilter was designed to be used by individuals. The results of sharing wordlists for multiple users are not been widely known. For our tests at the small company, we shared wordlists between the two users as an experiment. The trial at the small company showed how even just two people sharing a common Bogofilter database can cause problems for the filter. It was not clear from this test if the poor results were due to the filter being trained incorrectly or simply the filter not performing well.

One additional method we came up with but were not able to test involved all users sharing a spam corpus and simultaneously maintaining their own individual ham corpus. The assumption behind this method is that all users receive similar spam, but different ham. This technique ensures that all users benefit from one user receiving a new spam, but still accepts that different users receive different email. We believe this method could improve the results of using a Bayesian filter with multiple users.

7.5 Future Work

There are a lot of future projects that could be done involved spam filtering. Many of the ideas presented earlier in this section would be excellent topics to research.

Robinson Algorithm - Test the effectiveness of Gary Robinson's improvements to Paul Graham's bayesian algorithm.

Whitelists and Blacklists - A more thorough look into whitelists and blacklists would involve looking at more than just who an email was from. Integrating whitelists and blacklists into other filters to improve their performance should also be looked at. Also, it may be worth looking into the usefulness of the lists provided by mailabuse.org.

Combining Bogofilter and SpamAssassin - Bogofilter can be used as a rule in SpamAssassin. Does this actually improve the results of SpamAssassin?

Collaborative Filters - Filters like Vipul's Razor, discussed in the literature review, were not looked at in this paper, but could be useful tools. Vipul's Razor can also be used as a rule in SpamAssassin.

Site-wide Filters - Determine the effects of having a site-wide spam corpus, along with individual ham corpora for each user.

Spam filtering is a rapidly growing area of research, and we believe that any future project should attempt to analyze the state of the art in filters. For this project, we tested bayesian filters because they were new and mostly untested. At the time of this writing, Robinson's algorithm is becoming more popular than Graham's, but there may be still more improvements in the future. One thing still for sure, however; this war is far from over!

A Historical Spam

A.1 Monty Python : "Flying Circus", Spam

Scene A cafe. One table is occupied by a group of Vikings with horned helmets on. A man and his wife enter.

Man (Eric Idle) You sit here, dear.

Wife (Graham Chapman in drag) All right.

Man (to Waitress) Morning!

Waitress (Terry Jones, in drag as a bit of a rat-bag) Morning!

Man Well, what've you got?

Waitress Well, there's egg and bacon; egg sausage and bacon; egg and spam; egg bacon and spam; egg bacon sausage and spam; spam bacon sausage and spam; spam egg spam spam bacon and spam; spam sausage spam spam bacon spam tomato and spam;

Vikings (starting to chant) Spam spam spam spam...

Waitress ...spam spam spam egg and spam; spam spam spam spam spam spam baked beans spam spam spam...

Vikings (singing) Spam! Lovely spam! Lovely spam!

Waitress ...or Lobster Thermidor au Crevette with a Mornay sauce served in a Provencale manner with shallots and aubergines garnished with truffle pate, brandy and with a fried egg on top and spam.

Wife Have you got anything without spam?

Waitress Well, there's spam egg sausage and spam, that's not got much spam in it.

Wife I don't want ANY spam!

Man Why can't she have egg bacon spam and sausage?

Wife THAT'S got spam in it!

Man Hasn't got as much spam in it as spam egg sausage and spam, has it?

Vikings Spam spam spam spam (crescendo through next few lines)

Wife Could you do the egg bacon spam and sausage without the spam then?

Waitress Urrghh!

Wife What do you mean 'Urgghh'? I don't like spam!

Vikings Lovely spam! Wonderful spam!

Waitress Shut up!

Vikings Lovely spam! Wonderful spam!

Waitress Shut up! (Vikings stop) Bloody Vikings! You can't have egg bacon spam and sausage without the spam.

Wife (shrieks) I don't like spam!

Man Sshh, dear, don't cause a fuss. I'll have your spam. I love it. I'm having spam spam spam spam spam spam spam beaked beans spam spam spam and spam!

Vikings (singing) Spam spam spam spam. Lovely spam! Wonderful spam!

Waitress Shut up!! Baked beans are off.

Man Well could I have her spam instead of the baked beans then?

Waitress You mean spam spam spam spam spam spam... (but it is too late and the Vikings drown her words)

Vikings (singing elaborately) Spam spam spam spam. Lovely spam! Wonderful spam! Spam spa-a-a-a-am spam spa-a-a-a-am spam. Lovely spam! Lovely spam! Lovely spam! Lovely spam! Lovely spam! Spam spam spam spam!

A.2 The first Spam email

This email was sent out to every single user of ARPANET in 1978. The "To:" field was 400 lines long, and omitted from this report.[17]

Mail-from: DEC-MARLBORO rcvd at 3-May-78 0955-PDT

Date: 1 May 1978 1233-EDT

From: THUERK at DEC-MARLBORO

Subject: ADRIAN@SRI-KL

DIGITAL WILL BE GIVING A PRODUCT PRESENTATION OF THE NEWEST MEMBERS OF THE DECSYSTEM-20 FAMILY; THE DECSYSTEM-2020, 2020T, 2060, AND 2060T. THE DECSYSTEM-20 FAMILY OF COMPUTERS HAS EVOLVED FROM THE TENEX OPERATING SYSTEM AND THE DECSYSTEM-10 <PDP-10> COMPUTER ARCHITECTURE. BOTH THE DECSYSTEM-2060T AND 2020T OFFER FULL ARPANET SUPPORT UNDER THE TOPS-20 OPERATING SYSTEM. THE DECSYSTEM-2060 IS AN UPWARD EXTENSION OF THE CURRENT DECSYSTEM 2040 AND 2050 FAMILY. THE DECSYSTEM-2020 IS A NEW LOW END MEMBER OF THE DECSYSTEM-20 FAMILY AND FULLY SOFTWARE COMPATIBLE WITH ALL OF THE OTHER DECSYSTEM-20 MODELS.

WE INVITE YOU TO COME SEE THE 2020 AND HEAR ABOUT THE DECSYSTEM-20 FAMILY AT THE TWO PRODUCT PRESENTATIONS WE WILL BE GIVING IN CALIFORNIA THIS MONTH. THE LOCATIONS WILL BE:

TUESDAY, MAY 9, 1978 - 2 PM
HYATT HOUSE (NEAR THE L.A. AIRPORT)
LOS ANGELES, CA

THURSDAY, MAY 11, 1978 - 2 PM
DUNFEY'S ROYAL COACH
SAN MATEO, CA
(4 MILES SOUTH OF S.F. AIRPORT AT BAYSHORE, RT 101 AND RT 92)

A 2020 WILL BE THERE FOR YOU TO VIEW. ALSO TERMINALS ON-LINE TO OTHER DECSYSTEM-20 SYSTEMS THROUGH THE ARPANET. IF YOU ARE UNABLE TO ATTEND, PLEASE FEEL FREE TO CONTACT THE NEAREST DEC OFFICE FOR MORE INFORMATION ABOUT THE EXCITING DECSYSTEM-20 FAMILY.

A.3 The first Spam USENET post

This is the first known instance of USENET Spam.

From: JJ@cup.portal.com (JJ@cup.portal.com)

Subject: HELP ME!

Newsgroups: comp.graphics, comp.ivideodisc, comp.lang.ada, comp.lang.apl

Date: 1988-05-23 17:00:08 PST

Poor College Student needs Your Help!! :-(

Hi. I just finished my junior year in college, and now I'm faced with a major problem. I can't afford to pay for my senior year. I've tried everything. I can't get any more student loans, I don't qualify for any more scholarships, and my parents are as broke as am I. So as you can see, I've got a major problem. But as far as I can see, there is only one solution, to go forward. I've come along way, and there is no chance in hell that I'm going to drop out now! I'm not a quitter, and I'm not going to give up.

But here is why I'm telling you all this. I want to ask a favor of every one out here on the net. If each of you would just send me a one dollar bill, I will be able to finish college and go on with my life. I'm sure a dollar is not much to any of you, but just think how it could change a person's life. I'd really like to encourage all of you to help me out, I've no other place to go, no other doors to knock on. I'm counting on all of you to help me! (PLEASE!) If you would like to help a poor boy out, please send \$1 (you can of course send more if you want!! :-)

Jay-Jay's College Fund
PO BOX 5631
Lincoln, NE 68505

PS. Please don't flame me for posting this to so many newsgroups, I really am in dire need of help, and if any of you were as desperate as I am, you just might resort to the same thing I am. Also, please don't tell me to get a job! I already have one and work over 25 hrs a week, plus get in all my classes, plus find time to study! So hey, please consider it! It would really mean a lot to me. Thank you!

NOTE: Any extra money I receive will go to a scholarship fund to help others in the same situation. :-)

A.4 The first time Spam was called “Spam”

What follows is an example of one of the 200 posts that were made to news.admin.policy when Richard Depew’s retro-moderation program failed. Below it is a response in which the term “spam” is used to describe an email of this sort for the first time.

```
Newsgroups: news.admin.policy,news.software.b
From: red@redpoll.neoucom.edu (Richard E. Depew)
Subject: ARMM: ARMM: ARMM: ARMM: ARMM: ARMM: ARMM: Supersedes or Also-Control?
Message-ID: <AAAAAAC4qG8t.B8y@redpoll.neoucom.edu>
Supersedes: <AAAAAAC4qG8t.B8y@redpoll.neoucom.edu>
Supersedes: <AAAAAC4qG8t.B8y@redpoll.neoucom.edu>
Supersedes: <AAAAC4qG8t.B8y@redpoll.neoucom.edu>
Supersedes: <AAAC4qG8t.B8y@redpoll.neoucom.edu>
Supersedes: <AAC4qG8t.B8y@redpoll.neoucom.edu>
Supersedes: <AC4qG8t.B8y@redpoll.neoucom.edu>
Supersedes: <C4qG8t.B8y@redpoll.neoucom.edu>
Followup-To: news.admin.policy
Keywords: preference?
Organization: Home, in Munroe Falls, OH
Date: Wed, 31 Mar 1993 02:58:04 GMT
Lines: 30
```

Automated Retroactive Minimal Moderation (tm) by ARMM5. Press 'n' to skip.

Automated Retroactive Minimal Moderation (tm) by ARMM5. Press 'n' to skip.

Automated Retroactive Minimal Moderation (tm) by ARMM5. Press 'n' to skip.

Automated Retroactive Minimal Moderation (tm) by ARMM5. Press 'n' to skip.

Automated Retroactive Minimal Moderation (tm) by ARMM5. Press 'n' to skip.

Automated Retroactive Minimal Moderation (tm) by ARMM5. Press 'n' to skip.

Automated Retroactive Minimal Moderation (tm) by ARMM5. Press 'n' to skip.

I’ve had a few complaints that my earlier ARMM5 moderated posts with "Also-Control: cancel" headers were not being handled correctly, and it has been suggested that I use "Supersedes:" in place of "Also-Control: cancel".

Is there any reason to prefer one over the other if the intent is to replace one article with another?

This post carries the Supersedes header. It behaves just like the Also-Control header with C-news, at least so far as I can tell.

Dick

--

Richard E. Depew, Munroe Falls, OH red@redpoll.neoucom.edu (home)
"...plug the RS-232 connector on the back side of the Mini Modem 2400 into
the RS-232 connector on your computer, then screw up." - modem instructions

Joel Furr posted a response containing this suggested change to the Jargon File.

Usenet History, I tell you. This needs its own listing in the Jargon File:

:ARMM: n. A USENET posting robot created by Dick Depew of Munroe Falls, Ohio.
Originally intended to serve as a means of controlling posts through
anon servers (see also {anon servers}). Transformed by programming
ineptitude into a monster of Frankenstein proportions, it broke loose
on the night of March 31, 1993 and proceeded to spam news.admin.policy
with something on the order of 200 messages in which it attempted, and
failed, to cancel its own messages.

References

- [1] Borden, Bruce S., Marshall T. Rose and John L. Romine. *RAND MH Message Handling System*. <http://www.ics.uci.edu/~mh/>
- [2] Canter, Laurence A. and Martha S. Siegel. *How to Make a Fortune on the Information Superhighway : Everyone's Guerrilla Guide to Marketing on the Internet and Other On-Line Services*. (HarperCollins) January, 1995
- [3] Coalition Against Unsolicited Commercial Email. *Quick FAQ*. <http://www.cauce.org/about/faq.shtml>
- [4] Glasner, Joanna. *A Brief History of SPAM, and Spam*. <http://www.wired.com/news/business/0,1367,44111,00.html>
- [5] Graham, Paul. *A Plan for Spam*. <http://www.paulgraham.com/spam.html>
- [6] Guenther, Philip and Stephen R. van den Berg. *Procmail*. <http://www.procmail.org/>
- [7] Hawkins, Peter. *Gotmail*. <http://www.nongnu.org/gotmail/>
- [8] Mail Abuse Prevention System. *Definition of "spam"*. <http://mail-abuse.org/standard.html>
- [9] MailCircuit.com. *MailCircuit's Email HandShake VerificationTM and Spam Filter Process*. <http://www.mailcircuit.com/filter.htm>
- [10] Metz, Cade. *Slam the Spam*. PC Magazine Vol. 22, No. 3. pp. 74-97, Feb 2003.
- [11] Oskoboiny, Gerald. *Whitelist-based spam filtering*. <http://impressive.net/people/gerald/2000/12/spam-filtering.html>
- [12] Prakash, Vipul Ved. *Vipul's Razor*. <http://razor.sourceforge.net/>
- [13] Ramkissoon, Ravi. *Fetchyahoo*. <http://fetchyahoo.twizzler.org/>
- [14] Raymond, Eric S. *Bogofilter*. <http://bogofilter.sourceforge.net/>
- [15] Robinson, Gary. *Gary Robinson's Rants*. September 16, 2002 <http://radio.weblogs.com/0101454/stories/2002/09/16/spamDetection.html>
- [16] SpamAssassin. *Tests Performed*. <http://spamassassin.org/tests.html>
- [17] Templeton, Brad. *Origin of the term "spam" to mean net abuse*. <http://www.templetons.com/brad/spamterm.html>