A Spam-Detecting Artificial Immune System

by

Terri Oda

A thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Canada

January, 2005

Your file Votre référence
ISBN: 0-494-00764-8
Our file Notre référence
ISBN: 0-494-00764-8

Abstract

Electronic junk mail, colloquially known as spam, has increasingly become a part of life for users of email. Although people dislike the constant barrage, it is difficult to design an anti-spam solution that can adapt appropriately to the constantly changing messages.

Inspired by models of the human immune system, the spam immune system is a spam classifier that protects users from spam. The metaphor of the immune system is a compelling one because of the immune system's ability to protect the body by distinguishing self from non-self, even as the non-self changes.

The spam immune system creates digital antibodies which are used as email classifiers. As the system is exposed to messages, it learns about the user's classification of spam and non-spam. Once trained, the resulting system can achieve classification rates comparable to those of existing commercial anti-spam products.

Acknowledgements

Many people have helped me bring the spam immune system from an idea to an entire thesis. Foremost among them is my supervisor, Tony White. His swarm intelligence class introduced me to the idea of artificial immune systems, and his encouragement, support and enthusiasm for this project has made it not only possible, but also a great experience.

My undergraduate supervisor, Irwin Pressman, was the first to encourage me to consider graduate school, and has provided a wealth of useful advice. He has been a great inspiration, and he was right when he said that graduate work was more fun than undergraduate work.

I wish to thank my thesis committee, who have helped make this possible even before they agreed to be part of the committee. Stan Matwin's course in data mining helped me understand many of the other approaches to text classification, and Anil Somayaji's "Computer Immunology" paper was among the first immune systems papers I read, and it is one that made me think this idea might just be worth pursuing.

This work would not have been possible without the support of my family and friends. They cheered me on, provided advice and background information (especially about biology), and often listened to me talk about my system, asking questions that helped me make my explanations more clear. In particular, I would like to thank Ken O'Byrne for providing not only invaluable support, but also providing extra computing power when I needed it most.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Several years ago, the word "spam" usually denoted a particular brand of luncheon meat, but in recent times, the word is used colloquially to represent a variety of junk, especially unwanted junk email. Many people dislike constant bombardment with advertisements, but at least with traditional advertisements the cost is borne by the sender; with spam, the recipient pays. Add to that the sheer volume of spam being sent – A study of Canadian Internet users in March 2004 found that 68% of their email could be classified as spam [IF04] – and spam is quickly seen as a problem that needs solving.

While there are already many good solutions available, spam is a co-evolutionary problem: the users try to find new ways to avoid spam, and the spam senders (spammers) try to find ways to force users to see their messages. As such, new solutions are created constantly to compensate for the ever-changing nature of the problem. And while fairly effective anti-spam products exist, many don't meet the expectations of reviewers or users [Smi04].

This thesis details a new anti-spam solution based upon the artificial immune

system model. The core ability of the immune system is its ability to differentiate the organism it protects from all invaders. The human immune system is remarkably effective at handling viruses, bacteria and other things which might harm the body. Viruses evolve and mutate, making them hard to detect, yet despite the fact that most humans are constantly exposed to these dangerous invaders, immune systems keep their hosts from getting sick with every new exposure.

Artificial immune system models have been used for a variety of applications, including computer security [FHS97] and document classification [GC03]. This self/non-self classification ability can be extended to spam/non-spam, so the immune system model can easily be adapted as a spam classifier. The adaptive nature of the immune system also makes it an attractive choice, as it would be nice if the systems themselves could adapt to help with the co-evolutionary balance. Viruses and spam have some things in common: they're both undesirable, and they both evolve to avoid detection. As such, it makes sense to use this biological metaphor to deal with this similar problem.

1.2 Problem Statement

People don't like spam [Pro04], so the less they have to deal with, the better. The aim of spam solutions is to pre-sort messages into two categories, spam and non-spam, and to do this with high accuracy so that the user doesn't need to second guess the sorting constantly. These two features are the most important ones for any spam detector, and Section 2.1.4 describes other desirable qualities.

My more specific goal was to produce a spam detector inspired by knowledge of the mammalian immune system. Section 2.2.3 explains how the artificial immune system model has qualities which are desirable in a spam detector.

1.3 Contributions

This thesis represents the most comprehensive work on the application of an artificial immune system to spam detection. My earlier paper, [OW03a], was the first publication detailing a spam-detecting artificial immune system.

While this solution is interesting because of the theoretical basis used, it is more than a theoretical whim: it is also a good spam classifier. Its peak accuracy of 93.6% with 1.1% false positives is comparable with results from commercial and non-commercial anti-spam products [And04]. This is particularly interesting given that this spam immune system represents only a single approach, whereas more complete anti-spam products typically use several different methods.

In order to produce this system, it was necessary to create and test a number of algorithms:

1. An algorithm for creation of classifiers,
2. An algorithm for classification of messages based upon these classifiers, and
3. Algorithms for evolution of classifiers over time with respect to the changing nature of the emails being classified.

All of these algorithms are described in Chapter 3. The algorithm for creation is described in Section 3.2.1, the algorithms tested for classification are described in Section 3.2.3 (and the varied results for each is described in Section 4.2.2), and the complete algorithm for the whole lifecycle, including evolution, is described in Section 3.2 with more information in Sections 3.2.2, 3.2.3, and 3.2.4, with results described in Sections 4.2.6, 4.2.7 and 4.2.8.

In addition, this thesis explores several different libraries used in the creation of classifiers. These are described in Section 3.1.3: the English library in Section 3.1.3.2, the Bayesian token library in Section 3.1.3.3 and the heuristic library in Section 3.1.3.4. The results from testing these libraries are described in Section 4.2.4.

1.4 Thesis Organization

This chapter (Chapter 1) gives an overview of this thesis.

Chapter 2 is divided into two major parts. Section 2.1 describes some background information about spam, including what it is and why it is a significant problem. This section also contains an overview of several of the anti-spam solutions currently available, particularly those which inspired this work. The second part of Chapter 2, Section 2.2, details information about biological immune systems and the artificial immune system model used to create the spam immune system.

The spam immune system is described in further detail in Chapter 3. In this chapter, the algorithms used for the entire lifecycle are described. Section 3.1 describes the components of the immune system and Section 3.2 describes the lifecycle of the system, including all the algorithms used.

The experimental setup and results from testing the spam immune system can be found in Chapter 4. Setup is covered in Section 4.1, then the results and discussion can be found in Section 4.2 and 4.3 respectively.

Finally, Chapter 5 gives my conclusions and some suggestions of future work in this area.

Chapter 2

Background

2.1 Spam

In the past few years, the amount of electronic junk mail (spam) received by email users has increased considerably. This chapter gives an overview of information about spam and about the solutions currently available to deal with the increased number of messages sent and received daily.

The definition of spam is discussed in Section 2.1.1.1. Though most users are quite sure what spam is to them, the definitions are fairly varied.

Sections 2.1.1.2 and 2.1.1.3 discuss the severity of the problem and the reasons for this. As one might guess from the amount sent daily, obtaining email addresses is not overly difficult or expensive. The next section, 2.1.1.4, explains how spam senders obtain email addresses.

In order to help explain later sections, Section 2.1.1.5 gives a brief description of the components of an email message. Then Section 2.1.1.6 explains why spam classification is fairly difficult. However, it's not impossible, so Section 2.1.2 describes the types of solutions available, and Section 2.1.3 describes some of these solutions in greater detail.

2.1.1 About Spam

2.1.1.1 What is Spam?

Within an email context, the term "spam" refers to the electronic equivalent of junk mail: a range of unsolicited and/or undesired junk messages. The American Heritage Dictionary defines spam as "Unsolicited e-mail, often of a commercial nature, sent indiscriminately to multiple mailing lists, individuals, or newsgroups; junk e-mail." [spa00]

Although many Internet users know immediately what is meant by the term spam, the exact definition tends to vary from person to person or from organization to organization. Most definitions limit spam to messages sent to many people without the consent of the recipients.

Although I choose to use "non-spam" or "legitimate email" to refer to messages which are not spam, a popular slang term for non-spam is ***ham***. See Section 2.1.1.1 for more on the origin of this term and the word spam.

Legal Definitions of Spam When it comes to making a legal definition of spam, it is not sufficient to limit the definition to unsolicited messages. Many legitimate personal or business communications are unsolicited except in the vague sense that contact information was made available. But if it is assumed that consent is given based on the fact that contact information was made public, then spam would count as solicited mail. A job seeker may receive many legitimate queries about her resume, which is available online, but also receive junk mail related to job searches – by making her contact information available, does that mean all job-related things sent to this person are legitimate? To narrow the definition closer to what people instinctively believe is junk, several classes of email are often used [CAU04]:

Unsolicited Commercial Email (UCE) consists of advertisements which have been sent out without request or consent of the recipients.

Examples of UCE include advertisements for medications, websites, some illegal items, and less potentially offensive advertisements such as ones for children's' toys, assuming that these messages have been sent without the recipient's consent.

Unsolicited Bulk Email (UBE) consists of messages which have been sent to many parties without their request or consent.

Examples of UBE may include political or religious notices, hate mail against certain groups, or scams. A common spam scam involves fake emails which appear to come from a legitimate agency such as PayPal, a bank, or credit card company, requesting verification of credit card number (and other personal details) but instead allow thieves to steal identities.

UBE also includes UCE, so commercial messages sent to many people would still be considered UBE as well as UCE.

Unsolicited Automated Email is a related definition used by Graham as another attempt to define what people really see as spam [Gra02]. He points out that if a neighbour sent a for sale advertisement for something he wanted, he'd be delighted, despite it being both unsolicited and commercial. The use of the word "automated" instead of "bulk" illustrates that a low-volume message may still be spam.

Any of these classes, sometimes combinations thereof, may be used as a definition for spam. But although these classes seem reasonably straightforward on the surface, it can be a challenge to define each in sufficiently precise terms for legal pursuit of offenders.

An interesting thing to note is that for some business dealings, consent actually is implicit. When a user buys something from an online store, he doesn't mind when they send a receipt, but may not be impressed if they then send a newsletter every

week. Yet some legal definitions exclude mail sent by companies with an existing relationship with the recipient.

This leads to quite a number of questions that need to be answered:

- What does consent entail?
- What implicit consent is given in a business transaction?
- Can an individual be said to have "solicited" advertisements from a company if they signed up for an unrelated service and gave the company permission to send messages related to that?
- What if a clause was buried deep within an agreement before the user could sign up for anything?
- Does there need to be an easy "opt-out" checkbox somewhere on the form?
- Does that checkbox need to be unchecked by default?

These are all interesting questions that are debated in the courts, but as we shall see in Section 2.1.1.1, they need not be investigated to produce a useful spam filter, since a more pragmatic definition of spam can be used for the purpose of spam filtering.

A Definition for Spam Filters While all these definitions of spam are interesting, and it is important to realize that spam definitions are not the same for everyone, the definition that really matters for a learning filter is the one that's being taught. A recent prediction for 2004 suggests that because email filtering is becoming increasingly common and easy to use, people will use it even for things which are not typically viewed as spam:

> Consumer and office-worker definitions of spam will shift, thanks to the capabilities found on desktop spam-control products. Spam, once the

> domain of unsolicited junk e-mail, will become plain unwanted e-mail. Mail I requested last week is spam this week. A worker who subscribes to a mailing list in January will no longer want it in April. It will be easier to mark the message as spam than it will be to unsubscribe to the list. The messages will keep on flowing – at the user's request – but will be blocked before the user sees it. [Gre04]

Especially with mailing lists, what one person considers junk may be something another person wants. There is also the issue that users will unintentionally report mailing lists as spam [Ros03b]. This isn't a problem if their spam report affects only their mail, but in the case of the spam report going to a large company such as AOL, the complaint may result in an innocent site and sender being blacklisted. Although it is convenient for email servers to be able to use global spam filters for many users, doing so can be very dangerous.

Thus, the definition of spam is dependent upon the individual, and even then, the definition may continue to evolve over time.

A fairly practical definition can be used when discussing filtering methods such as the one being proposed here: Spam is what the recipient considers to be junk mail and does not wish to receive.

This definition does not hold up well in a legal sense since it would be nearly impossible for companies or even individuals to avoid sending something which could be claimed to be spam, but the definition works well for cases where no retributive action is taken against senders of spam, and it more accurately represents how filters are likely to be used. It is this definition that will be used for the purposes of this paper.

A note on the history of the word "spam" The word spam comes from the name of a product made by Hormel Foods Corporation. SPAM Luncheon Meat,

within the food context, is the trademarked name of Hormel's Spiced Ham. Hormel Foods requests that capital letters be used when referring to their trademark and small case letters be used when used as a slang term for junk, particularly junk email [Hor04b] [Hor04a].

The use of the word spam to represent junk is often said to come from a comedy routine by Monty Python's Flying Circus, in which the word "spam" is repeated over and over, drowning out other conversation. Since spam is often seen to drown out other communication on the Internet, the word was adopted to refer to junk messages online [spa03a].

This also explains the origin of the term "ham" to refer to non-spam messages. The idea is that canned meat is not "real" meat, thus the opposite of the canned SPAM would be real ham. I choose not to use this term because I find it confuses people, but you may see it in other references related to junk email classification.

2.1.1.2 How pervasive is the spam problem?

A study in March 2004 found that 68% of email received by Canadian internet users can be classified as spam [IF04]. The numbers for the United States and the rest of the world are similar [Bri04]. Canadians find all this spam highly annoying: in a survey of marketing techniques, spam rated just below the top offender, telemarketing (Spam rated 4.5 and telemarketing rated 4.6 on a scale of one to five with five representing extremely annoying) [Pro04].

Spam is a problem that has attracted the attention of many large companies, organizations, and governments in the past year. Despite many very good efforts, the amount of spam sent and received daily continues to rise.

When I started work on the spam immune system, I was getting approximately 25 spam messages per day. There are now days in which I have received over 200. (Thankfully, these are mostly quarantined by my automated solutions so I rarely have

to deal with more than 25 in a day.) While I get more than many people I talk to, my percentages are not out of line with that seen by large filtering company Brightmail, which notes an increase from 48% in May 2003 to 64% in April of 2004 – 16% in the past year. [Bri04]

All this spam costs money. One legal firm in Los Angeles reported their costs for spam as roughly half a million dollars after email devices were distributed to their firm's 220 lawyers. The costs were mostly in lost productivity, although bandwidth and storage paid a role in the final numbers. Spam had cost them more than $1000 per year per user. With the help of a spam filtering company, the law firm saved more than $400,000 per year [Fon03].

SpamCon Foundation estimated a cost of $8,900,000,000 per year to corporations in the United States. Each spam is worth $1-2 in lost productivity, and those numbers add up quickly [Atk03]. Estimates from 2003 suggest that spam may take around 2% of an employee's productive time [Han03]. It quickly becomes worthwhile to have a spam filter to save humans from doing all the work manually.

While many promising solutions have been proposed and implemented, the current feeling is that spam will continue to be a problem in the future. Thankfully, the many filtering solutions available can be layered to provide users with fairly good solutions for a time. As email exploitation evolves, however, new solutions must be invented and existing solutions adapted to deal with new messages.

2.1.1.3 Why is the problem so severe?

Simply put, spam makes money for those who send it. According to a study of American email users conducted in 2003 [Fal03], 7% of email users have purchased something offered in an unsolicited mail (although not all unsolicited mail is considered to be spam). The cost of spending spam is so low that this means many companies could make significant profits from spam. Many spammers are known to

be fairly rich [spa03c], and Wired News reported that one company was grossing over half a million dollars in a four week period during summer 2003 [McW03].

The most effective way to stop spam would probably be to make it unprofitable. All of the spam solutions do this in one way or another. Legislation attempts to do this by imposing severe penalties on those who are caught. Spam filters provide an economic deterrent as well: by making it so that fewer people receive the spam, fewer people can respond. Thus, the costs of sending spam go up, since more messages must be sent in order to reach enough buyers. Some, including software giant Microsoft, have suggested that adding a small cost per email sent (like putting postage on a regular letter) would go a long way towards limiting spam by putting the cost burden on the sender rather than largely on the recipient [Jes04].

2.1.1.4 How do spam senders get email addresses?

In order to reach many people, spammers must first gather email addresses to use. This can be accomplished in many ways:

1. Addresses can be harvested from many public locations. A program can be written which looks for anything that "looks like" an email address. One might, for example, look for any string containing the @ sign, or use a more detailed pattern such as one that requires there to be characters before and after the sign. The program can then just search through publicly available information and find valid email addresses.

   Some examples of publicly available places to find email addresses include:

   (a) Posts to Usenet newsgroups

   (b) Web pages

   (c) Mailing lists

   (d) Web browsers

(e) IRC and chat rooms

(f) Ident daemons

(g) Finger daemons

(h) AOL profiles

(i) Domain contact points

(j) White & yellow pages

(k) Web and paper forms

Many sites that make email addresses publicly available also try to obfuscate the addresses so that they cannot be easily harvested. See Section 2.1.3.4 for more on how this is done.

2. Guessing and cleaning

   Many email addresses can be guessed: dictionary words are fairly common, and first-initial last-name combinations are guessable for those with common last names. Some Internet Service Providers also use guessable combinations. For example, the National Capital Freenet in Ottawa has addresses which always start with two characters and end with three numbers, such as as648. Some spam programs may also try every possible address: a, b, c... aa, ab, ac... and so on [Tec03].

3. Having access to the same server

   In many systems, it is possible to determine all the valid email addresses on a server. In UNIX systems this can be done by checking the "passwd" file. In many work environments, company directories are made available.

4. From the previous owner of the email address

Common addresses may wind up being re-used when one customer leaves a service provider, and this address may already be receiving spam even if no one has used it in some time.

5. Using social engineering

   Many people are willing to give up their address to enter a contest, and while many contests are legitimate, there are offers out there where email addresses are being harvested and sold.

6. Buying lists from others

   Anyone who can make a list can also sell it. Addresses sold may have come from public harvesting, from organizations that went out of business, or any sort of organization that might have collected email addresses at some time. This includes email addresses that might have been collected for other, even legitimate, purposes.

7. Hacking in to sites

For more details on any of this list see Raz's summary [Raz04]

While spammers may use all of these methods to gain addresses, one method is currently preferred. According to a study completed in March 2003 [Tec03], 97% of the spam they received was delivered to addresses that had been posted on the public web. Usenet newsgroup postings followed at under 2%.

2.1.1.5 Anatomy of Email

In order to understand how spam detection works, it helps to have a basic understanding of the email format. Email can be divided into two parts: the headers and the body of the message.

A simplified email message might look like Figure 2.1.1.5. The first three lines are the headers, the last three lines are the body.

```
To: michael@example.com
From: susan@example.com
Subject: Next Thursday.
We're meeting in front of City Hall at noon.
See you then!
 Michael
```

Figure 2.1: A simplified email message

The ***headers*** are a set of meta-information found at the top of the email. They take the form of `<Name>: <value>` and while some of them are standard, more optional headers can be set. They include information about when a message was sent, to whom the message is being sent, from whom it was sent, what the subject of the message is, along with optional information such as the name of the program used to send the mail.

The ***body*** is the actual message content. This is usually in plain text or HTML format, and can include other attachments. Plain text and HTML can be parsed fairly easily by a program looking for spam terms, but identifying a picture or an image containing text is much more computationally intensive.

2.1.1.6 Why is spam classification hard?

One of the barriers to legislation against spam is the fact that not everyone uses exactly the same definition, as we saw in Section 2.1.1.1. It doesn't help that laws may be made at different levels even within the same country, let alone laws in different countries. With so many different and sometimes conflicting laws, prosecution can be very difficult.

Another barrier both to legislation and practical filtering is that email is not designed in such a way that the sender can always be traced easily. There is no authentication of the sender built in to the protocol used by email, leaving it possible for people to forge sender information. This makes it hard to trace back and prosecute the sender, or to avoid receiving messages from a known spammer in the future. There

are several proposals to adapt this protocol, and these are described in Section 2.1.3.6.

Spam changes over time as new products become available and seasons change. For example, Christmas-themed spam is not usually sent in June. But beyond that, there are targeted changes happening in spam. Perhaps the largest problem of spam filtering is that spammers have intelligent beings working to ensure that "direct email marketing" (the marketing term for spam) is seen by as many potential customers as possible. Many anti-spam tools are freely available online, which means that spammers have access to them too, and can learn how to get through them. This makes spam detection a co-evolutionary process, much like virus detection: both sides change to gain an advantage, however temporarily. Although it does change, spam is not completely volatile. Terry Sullivan found that while spam does undergo periods of rapid changes, it also has a core set of features which are stable for long periods of time [Sul04].

Spam varies from person to person. This is partially due to targeting on the part of the address harvesters, who try to guess the interests of the recipients so that the response rate will be higher. But more importantly, legitimate mail also varies from person to person. In theory it should be possible to discover spam without much attention to the legitimate mail. However, the great success of classifiers which use both, such as Graham's Bayesian classifier (See Section 2.1.3.11) and the CRM114 discriminator [Yer04], implies that use of data from both legitimate and spam email is very beneficial.

One final thing to note in the difficulty of spam classification is that all mistakes in classification are not equal. ***False negatives***, messages that have accidentally been tagged as non-spam, are usually seen by the user. They may be annoying, but are usually easy to deal with. However, ***false positives***, messages that have been accidentally tagged as spam, tend to be more problematic. When a single legitimate message is in a pile of spam, it is much easier to miss seeing it. (A typical user

will not read all spam, but instead scans subject and from lines quickly to see if anything legitimate stands out.) While there is relatively little impact if a person receives a single spam, missing a real message which might be important is much more dangerous. One research firm suggests that companies lose $3 billion dealing with false positives [Han03].

2.1.2 Classifying Current Solutions

Spam solutions can be divided loosely into several classes:

Social solutions try to solve the solution at its root by discouraging spam senders (spammers) through social methods such as legal defences.

Typically, social solutions aim at the root of the problem, making it difficult for spammers to make a living sending spam. This provides a fairly strong economic motivation for people to stop sending spam.

There are also some interesting ideas about trust and social networking which come in to play here. Similar to the PGP web of trust [Fei02], it may be possible to use social networking to determine whether a given person or group can be trusted not to send spam. In social networking systems, people typically indicate those who they feel to be friends. On the assumption that people rarely get spam from their friends or perhaps on the assumption that no one will be friends with those who send spam regularly, people can often trust friends of friends and so on.

Technological solutions use technical means to make it more difficult for spammers to get their messages through to recipients.

In a loose analogy, technological solutions usually try to fix the symptoms rather than the disease. That is, they typically make life better for the end user of a solution rather than worrying about getting rid of those who are causing the

problem by sending spam. However, the side effect of users seeing less spam is that fewer people will buy what's being sold, making it harder for spammers to make money. This economic deterrent makes even the most purely technical solution have an effect on the root of the problem.

Within the technological solutions, there are several ways in which spam is classified:

Content - Messages are classified based upon their contents (for example, do they contain given words or phrases?). The contents may include plain text, HTML or other enhanced text formats, images, documents, or other attachments. Most of the systems mentioned here focus upon the plain text and HTML parts of the message. Other attachments may also be analyzed, and the mere existence of these contents can also be used to help make a classification.

Source - Messages are classified based upon their source. Blacklists (Section 2.1.3.7) and whitelists (Section 2.1.3.8) are examples of classifiers based upon the message source.

It must be noted that one of the problems with current electronic mail protocols is that it can be very difficult to ascertain that the stated source is in fact the actual source. Some methods of filtering include ways to verify the source of a message by checking if all parts of the address are valid, or more complex methods such as cryptographic identification. These methods are not directly relevant to my work, so they have not been covered in detail in this document.

Conglomerations / Combinations While it's nice to describe each of these techniques as if they are separate, this is not the case. Most complete spam solutions use multiple techniques in order to improve accuracy. Some techniques work to-

gether naturally, such as Challenge-Response, a method whereby senders must take action before the recipient will accept their messages (See Section 2.1.3.9), and Whitelisting, a method where only messages from trusted senders are accepted (See Section 2.1.3.8).

One well-known product, SpamAssassin, uses a combination of heuristics [sa-04d], automatic whitelisting [sa-04a], Bayesian rules [sa-04b], DNS blacklists [sa-04c], and a number of other methods [sa-04d]. Each detection rule is given a weight and the sum is used for the final score of a message. These weights are determined by a Genetic Algorithm once per release, which means that the weights are changed and optimized for best detection rates before each new release of the software.

Social defences can be combined with technical ones, too. A common tactic for sites using a blacklist (see Section 2.1.3.7) is to send out a mail letting anyone whose mail is being blocked know why it is being blocked. This encourages users to complain to their service providers, making it harder for spammers to find a service provider, and harder for service providers who run poorly configured mail servers that allow easy sending of spam.

2.1.2.1 When do these solutions work?

Not all of these solutions fit in the same point in the process of handling email. Figure 2.2 shows where some of the solutions described in the next section take place in this process.

Legislation, copywrite, and address obfuscation help to stop a spam before it is sent by placing penalties upon the act of sending spam messages. If these penalties act as a deterrent, the spam is never even composed.

Since these do not always act as sufficient deterrent, the next step is at the server level: blacklisting and heuristics can be applied to all the mail sent to a server, and

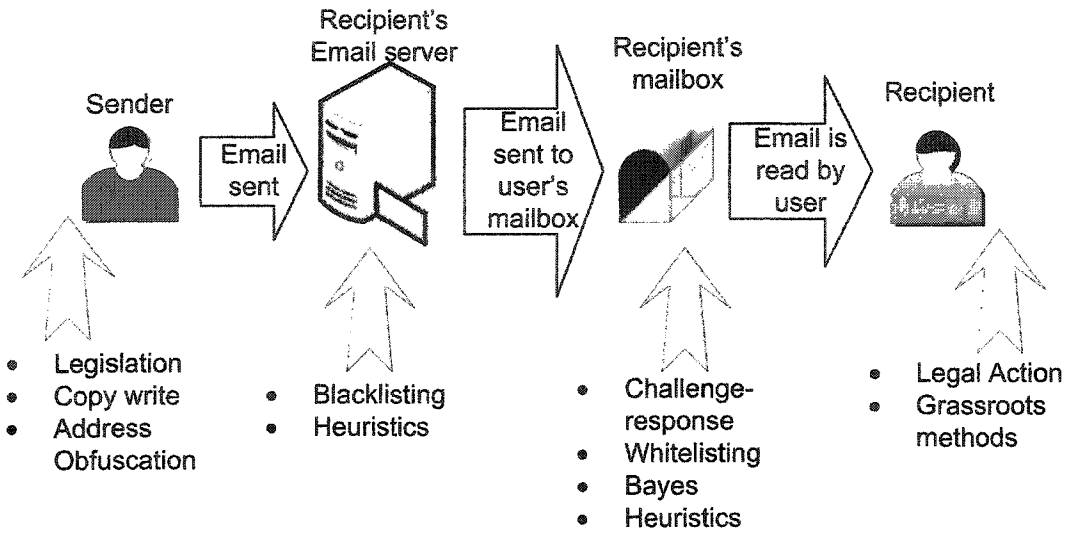

Figure 2.2: Where spam solutions work in the process of handling email

those messages which do not pass the tests can be discarded.

The next level occurs when the message is sorted into the individual user's mailbox. At that point, challenge-response, whitelisting, Bayes classifiers and more heuristics can take effect, taking any email which doesn't pass out of the inbox. These methods require more information about the specific user whose mail is being filtered, and thus may potentially be more accurate than the server-level methods. Note that we may have two-level solutions here, where filtering occurs immediately when the mail is sorted to the recipient's mailbox, and then it may be filtered again (with more information) right before the user sees the mail. None of the solutions described in this section use this sort of method, so it is not pictured in the diagram, but it can be done.

At the recipient level, once a person has received a spam message, he or she can take the option of legal action to impose penalties upon the spammer, or can employ more grassroots methods to penalize the spam sender. This is the final level, but these methods all tie back to the first level since the thread of retribution is what provides the deterrent to sending spam in the first place.

Some methods make take effect in more than one place in this process, but they

tend to have one place where they take most effect. For example, blacklisting also works as a preventative measure (before the spam is sent) because the threat of being blacklisted will discourage service providers from allowing spammers to use their networks, but its primary effect is at the server level when messages on the blacklist are denied.

2.1.3 Current Solutions

As the amount of spam sent daily has grown, so has the number of solutions available for dealing with the deluge. Many of these solutions have inspired my work or help to explain why certain choices were made in the design of the spam immune system, so these ones are described in more detail here.

My work has been most influenced by the solutions currently known to be used by popular anti-spam solutions. However, there is a wealth of academic work, particularly in the area of natural language processing, that could be drawn upon for future work. Beyond the Bayesian approaches described in Section 2.1.3.11, there have been good results with Support Vector Machines, boosting of trees, rule-learning algorithms and other approaches [DWV99] [CM01]. Many of these published and tested methods may be in use, but are simply not advertised since many anti-spam solution providers prefer to keep their methods confidential.

2.1.3.1 Spam Legislation and Legal Action

Solution type
Social

Many people hope to make the sending of spam email illegal, so that punishments can be brought to bear against those who send spam. Unfortunately, this is a difficult thing to do for several reasons:

- Laws will vary from place to place, be it countries, regions within countries, or groups of countries.

- The definition of spam is not universally agreed upon.
- There is significant pressure (from spam senders, who are frequently rich and influential [spa03c]) not to enact these laws.

Individual countries have enacted spam laws. The CAN-SPAM Act of 2003 [can03] in the US specifically makes it illegal to do a number of things commonly done by spam senders. This includes using a computer without authorization in order to send commercial email messages, doing a number of things intended to mislead recipients as to the origin of the message, the identity of the sender, etc. Penalties include jail terms of up to 5 years for repeat offenders. The first lawsuit under this new law was filed two months after it was passed [Asa04a]. Virginia is apparently the first U.S. state to have a felony conviction for spam, thanks to the laws in Virginia [Ras04]. One of thew two spammers convicted in Virginia has been sentenced to nine years in prison for the offence.

Critics of the CAN-SPAM Act of 2003 point out that it actually legalizes spam as long as the sender avoids doing any of the things specifically designated as illegal. Worse, this preempts state legislations which were sometimes stronger. Some even allowed citizens to sue spammers directly [Asa04b]. So even when spam laws are enacted, they can be ineffective or potentially make things worse for the average person.

Further, many people point out that making something illegal doesn't necessarily make it stop happening:

> MessageLabs' Sunner thinks CAN-SPAM promoters need a reality check. "There will never be a magic bullet for spam... ," Sunner says. "It's like a car. Stealing a car is illegal, but we still need car alarms. It's the same for spam. We'll always need a technical solution." [Ula04]

2.1.3.2 Copyright Law

Solution type
Social

A company called Habeas has a novel method of using the legal system to combat spam. They have written a haiku (a form of Japanese poetry that consists of three lines containing five, seven and five syllables respectively) that makes up part of a set of special headers that can be included with a message. This haiku is copyrighted, but the company licenses its use freely to individuals who do not send commercial mail and for a fee to businesses who comply with their standard of legitimate mail. The idea is that anyone who sends spam using this mark will be in violation of copyright law, and will be pursued accordingly using existing copyright law, rather than relying on new laws. [Hab03]

Those using filtering methods can let through anything with the Habeas haiku, knowing it not to be spam (thanks to Habeas' work in protecting their copywrite), and senders using the haiku can be more sure that their messages will make it through to their intended recipient. At time of this writing, the Habeas haiku has been used in popular solutions such as SpamAssassin and Spamcop, as well as popular service providers such as American Online, Juno and Prodigy.

2.1.3.3 Grassroots Methods

Solution type
Social

Many users would like to take matters into their own hands and tell spammers exactly what they think of the messages. For the most part, these are idle thoughts of revenge, not to be carried out, but people have been known to work en-masse to disrupt the lives of spammers.

After an article ("Spam king lives large off others' e-mail troubles" [Wen02b]) gave frustrated users a clue to how to find well-known spammer Alan Ralsky, a group of people organized a campaign to give him a taste of his own medicine. Reporter Mike

Wendland had agreed not to publish Ralsky's address, but did tell readers how he had found it, and a number of people signed Ralsky up for mailing lists and every advertising campaign they could find, keeping the US Postal service busy delivering catalogues, advertisements and brochures to his home[Wen02a].

While this is an amusing and not overly harmful reaction to spam, users have also been known to do more harmful things. Frustrated users have been known to take down the websites of spam senders [Art04], and spam was even cited as a possible motive in the murder of two known spam-senders [sho99b] [sho99a].

However these actions may have made their perpetrators feel, resorting to vigilante action has not helped stop the flow of spam.

2.1.3.4 Address Obfuscation

Solution type
Technological

The easiest and simplest way to avoid getting spam is to keep your email address private. While this is quite effective, it can be hard to strike a balance between hiding and making it easy for people to get in touch with you. Ideally, it should be possible to make an address accessible to human readers, yet incomprehensible to automated email address harvesters. People have come up with many different solutions such as using special encodings (for example, `name&#64;example.com`. @ is the HTML entity for the @ symbol.), inserting extra letters into their email addresses (for example, `name@NOSPAM.example.com`), a variety of human readable variants (for example, `name at example dot com`), and clever methods involving Java script.

A study by the Center for Democracy and Technology suggested that address obfuscation is very effective. Their tests found that none of the obfuscated addresses they posted on the web and elsewhere received spam [Tec03]. It is not very difficult for a spam harvester to reverse-engineer many types of obfuscation, and other studies suggest that these methods have varying degrees of success [Ros03a]. However, even if this technique only reduces the amount of spam sent to an email address, not avoid

spam entirely, it can still be a useful technique.

Others have also tried to "poison" email harvesters by creating web pages with many non-existent email addresses on them. An automated program for doing this, Sugarplum, attempts "to feed realistic and enticing, but totally useless or hazardous data to wandering address harvesters" [Car03]. The idea is that this will clog up the databases of the harvester, forcing the spammer to discard all information from your site, including any real addresses which may have been collected.

2.1.3.5 Heuristics

Solution type
Technological
Content and Source

The oldest and still common way of dealing with spam is through heuristic methods. People recognize patterns in spam that are not apparent in their regular mail and set up rules accordingly. For example, the phrase "discount Canadian drugs" appears rarely in most people's regular mail, but fairly frequently in the spam I receive. Most of the techniques described below are technically heuristics, in that they are not perfectly accurate rules, but the term is commonly applied to text-based rules developed by human intuition. SpamAssassin contains hundreds of these rules, which can be fairly simple or complex. For example, one might tag mail which starts with "Dear Sir" or make a more complex rule which handles "Dear sir" "Dear Madam" "Dear Ma'am" "Dear friend" and other variations upon the salutation.

Some spam-detecting heuristics found in SpamAssassin include:

- Does a link in the message use a numeric IP address?
- Is there a link to "unsubscribe" or "remove" in the mail?
- Is some variation on the phrase "barely legal" found in the message?
- Is some variation on the phrase "must be at least 18" found in the message?
- Does the message contain at least three dollar signs ($) in a row?

- Does the message contain at least three exclamation points (!) in a row?

Some heuristics also detect non-spam:

- Was this email sent by mutt, a text-based email client rarely used by spammers
- Was this email genuinely sent from eBay
- Is this message a Bugzilla bug status report
- Was the message sent by known mailing list software
- Does the email contain a PGP (pretty good privacy) signed message?

Paul Graham, whose work with Bayesian-inspired filters (see Section 2.1.3.11) brought statistical filtering to the foreground of anti-spam work, mentions that finding heuristics can be a fun challenge for many people. He claims the reason it took him so long to try the statistical approach "was because [he] got addicted to trying to identify spam features [himself], as if [he] were playing some kind of competitive game with the spammers." [Gra02] Even with the plethora of automated spam solutions available, some people prefer to set their own rules.

2.1.3.6 Sender Authentication

Solution type
Technological
Source

The basic idea of sender authentication methods is that somehow, there should be a way of checking whether a given mail server was allowed to send mail for a given domain. This can be done by signing messages in some way, by publishing in advance a list of servers allowed to send mail for a given address, etc. There are several different proposals available, including the following:

- SMTP-AUTH was described in RFC 2554 [Mye99] in March 1999. It adds a logging step where the email client must log in to the server before it can send

mail, and allows one mail server to indicate to another that the client had been authenticated.

- Sender Policy Framework emerged in late 2003 [spf04] as a combination of "Reverse MX" by Hadmut Danisch [Dan04] and "Designated Mailers Protocol" by Gordon Fecyk [Fec04]. It requires domains to publish information about the mail servers that are allowed to send outgoing mail from the domain. (Current records only keep the information about the incoming mail servers for a domain.)

- Yahoo's "Domain Keys" authentication software [Yah04] was submitted to the Internet Engineering Task Force (The body which handles many Internet standards) in August 2004 [Del04]. Using DomainKeys involves signing email (as in encryption techniques) and verifying the signature against a public key provided in the DNS record for each domain.

- Microsoft's "Caller ID for email" was announced in February 2004 [sen04a] and has merged with AOL's "Sender Policy Framework" to be come "Sender ID" [sen04b].

The way these protocols help to combat spam is that they limit spammers to using only valid servers to send spam from a given address. Thus, they must use their own addresses (or somehow compromise other people's servers), their own servers, and those who do not wish to receive email could simply block those servers. If all (or even most) legitimate email senders performed some sort of authentication, it would be possible for users to safely discard or quarantine any email that was not authenticated, so there would be no way for spammers to get their messages through to users who had chosen not to receive their messages. For SMTP-AUTH, the authentication is done between the sender and their email server, and the receiving server can check the authentication with the sending server. Figure 2.3 illustrates how the other sender authentication schemes work, with the authentication step taking

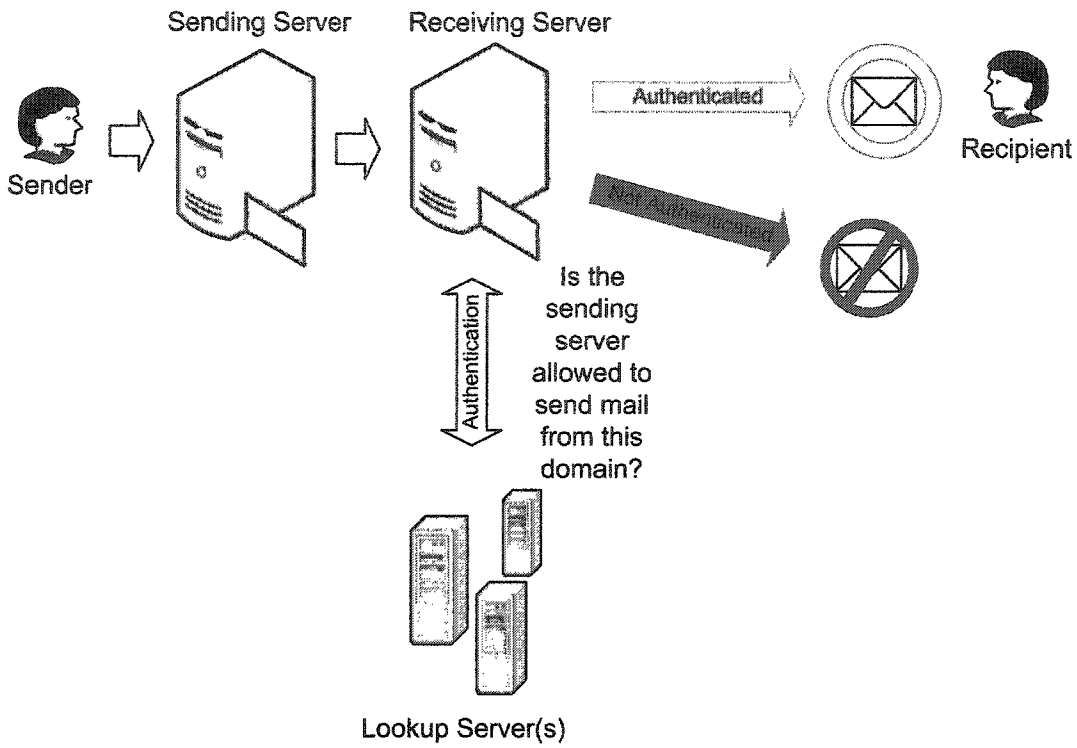

Figure 2.3: How sender authentication works

place at the receiving server. For many of the schemes, the lookup servers are the same servers used for domain name lookups.

At this time, no single proposal has emerged as the best, and without widespread use the impact of these protocols is limited, since individual users will still want email that has not been authenticated. However, with time, these authentication processes could be built in to the infrastructure in such a way that the average user would not even notice. Although there are ways to limit the effectiveness of this type of protocol, reliable knowledge of the email sender would be a valuable tool for spam detection and could aid in developing more accurate solutions. And even if the implementation is not widespread, spam filters can take advantage of the extra information found in authenticated messages.

2.1.3.7 Blacklisting

Since many spam messages are sent from the same machines, it is possible to block

those machines in order to greatly reduce the amount of spam received by a given mail server.

Solution type
Technological
Source

Many organizations provide lists of Internet addresses which in the opinion of the list maintainer, should be blocked. These may include known spam-senders [dns04], compromised or poorly-configured machines [ORD04], dial up addresses [DNS02], and even addresses of networks which may have policies that are "friendly" to spammers [Vix00]. The way in which these lists are determined can be quite varied: The popular Mail Abuse Prevention System's "Realtime Blackhole List" (RBL) is known to be quite harsh in listing even networks who are "neutral" to spammers [Vix00], while other networks are more moderate.

The practise of blocking addresses not only reduces spam, but provides an additional incentive for Internet service providers to avoid having customers who send spam. If customers find that their mail has been blocked because their service provider has been hosting spam, they complain and may even take their business elsewhere. Thus blacklisting is both a social and a technical defence.

The problem with simple blacklisting is that legitimate messages will be blocked along with the spam, and some claim that this is even necessary for the blacklists to be effective [Vix00]. Often, blacklists are used in conjunction with other tests so that they provide only part of the reason an individual message may be blocked.

2.1.3.8 Whitelisting

Solution type
Technological
Source

Whitelisting, as one might expect, is an opposite to blacklisting. A list of known/trusted senders is provided, and messages from senders not on the list will be treated as potential spam.

This is not a feasible solution for everyone: As a teaching assistant, I need to be able to receive mail from students whose email addresses and online names I may not know in advance. But many people communicate by

email with only with a small number of colleagues, friends and family. By limiting it so only their mail program only accepts mail from those who are pre-authorized (and providing some mechanism for unauthorized people to become authorized), the amount of spam is reduced, since spammers rarely know enough about the user to avoid being shunted to the "unauthorized" pile.

2.1.3.9 Challenge-Response

Solution type
Technological
Source

In order to avoid ignoring uncertain messages entirely, the challenge-response (CR) scheme can be used to improve a whitelist over time. Suppose Alice wants to send a message to someone she's never emailed before, Bob. When Alice sends a message to Bob, Bob's mail client recognizes that he does not know this person (or does not recognize her email address) and sends a message back to Alice asking her to verify herself. Once Alice has verified herself, her message is sent through to Bob, and any further messages from her email address will be automatically let through. This scheme is visually represented in Figure 2.4.

The idea is that while real users are quite capable of answering such queries, spammers would not have the time to do this (assuming they get the message at all, since reply-to addresses may not be real). Thus, the CR scheme ensures that it is a real person and not an automated machine trying to get in touch with you.

To increase the likelihood that only humans will successfully answer the challenge, some systems use a "Reverse Turing Test" (RTT). An RTT is a test which is difficult for computers to answer, but easy for humans. In one example of an RTT, the user is presented with images of distorted words (see Figure 2.5) and asked to type the word shown. More examples of RTTs and their applications can be found in [vABL04].

Unfortunately, there are issues with RTTs. One of the most commonly cited ones is accessibility: An RTT such as the one in Figure 2.5 requires the user to have sufficiently good vision. For a blind user, another test would have to be used, perhaps

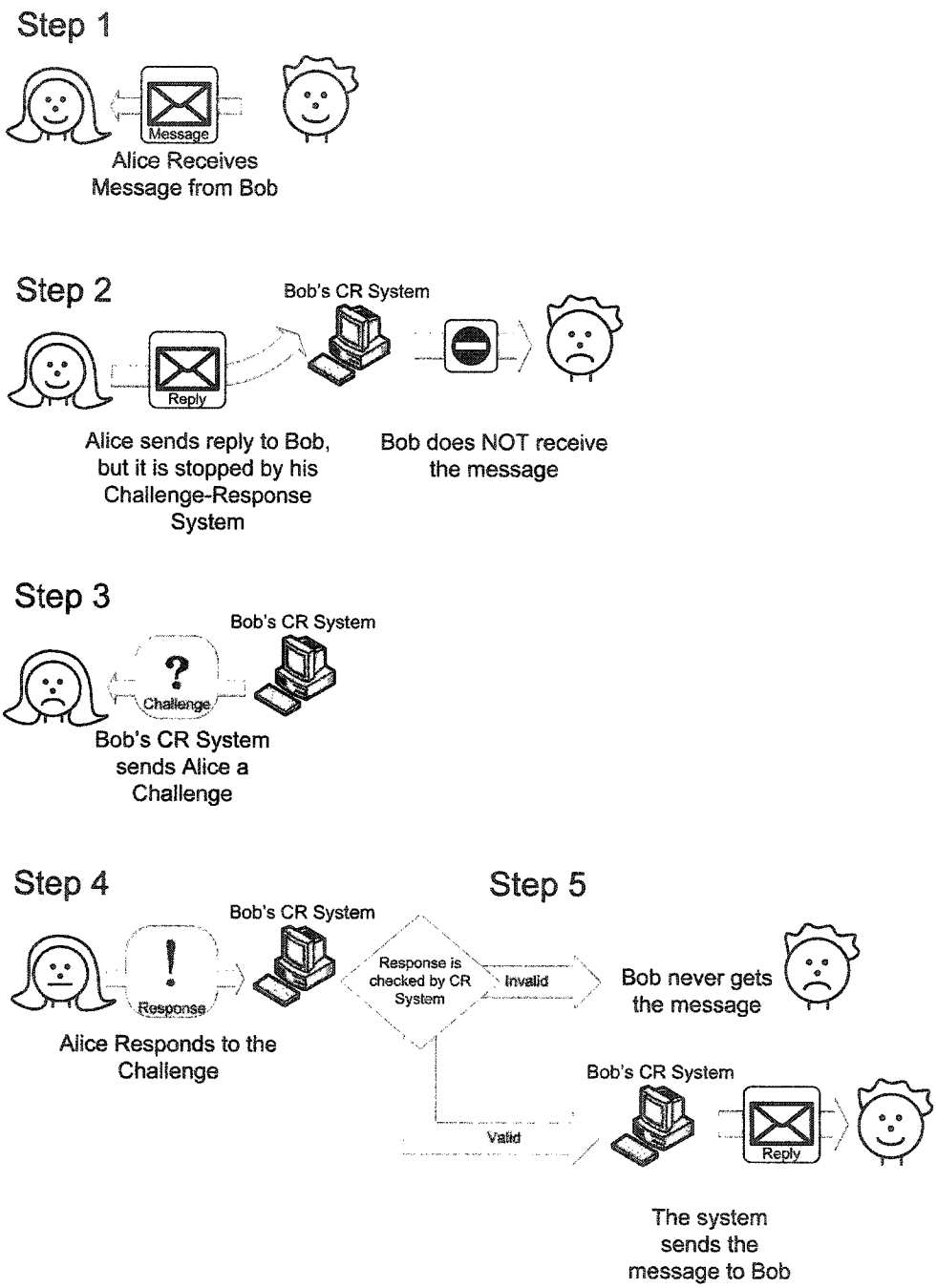

Figure 2.4: Typical Challenge-Response for Email

Figure 2.5: Reverse Turing Tests: "chipmunk" and "apple"

using audio if the user is not also hard of hearing. But with both pictures an audio, the bandwith required goes up considerably. This extra usage makes little difference to servers and users in places where bandwith is cheap, but for others the cost can be prohibitive, or the device being used may not even be able to handle the format used. Plain text questions can be used (for example, "What is the sum of three and five?" or "Who is the president of the United States of America?"), but they can be easier to beat with an algorithm, and "common knowledge" for the test maker may not be common knowledge for all the people who might encounter the RTT.

It is possible to beat RTTs with algorithms [MM03] [Rob02], and it is also possible to beat them by simply hiring humans to do them. This costs time and often money, which may be sufficient deterrent, but if the eventual profit is high enough, then RTTs can not be relied upon.

Beyond the issues related to using RTTs, Challenge-Response schemes have had their own technical problems: for example, they have been known to mistakenly reply to every message posted to a mailing list, becoming an annoyance to all subscribers. Even when those problems are eliminated, the system is still more obtrusive to senders, while most systems are the receiver's responsibility to maintain. This works well when the sender is willing to adhere to this protocol and send the necessary confirmation, but this cannot always be assumed.

Suppose Alice, an expert, has gotten a message from Bob, a stranger, asking for help. Alice may be quite happy to send off a quick email, but if Bob's system then sends her a challenge, she may simply decide that this is too much trouble or that Bob is being rude (after all, he contacted her first asking for help, then imposes more on her time by challenging her). These problems can be reduced by clever use of whitelisting

(for example, Bob's mail client automatically whitelists Alice's email address when he sends her a mail) but this cannot solve all problems (for example, Alice uses multiple email addresses and responds from one not automatically whitelisted, or Bob emailed a technical support address but gets a response from an individual support tech who uses his or her personal address, which would not have been available to Bob earlier).

So even though in theory, with everyone cooperating, the CR systems should be very effective, the reality is that with such a system, you risk frustrating people who send you mail and missing legitimate messages as a result.

2.1.3.10 Spamtrap email addresses

Solution type
Technological
Content and Source

If you know that a given email address is never used for real mail, it can be used to find known-spam without detecting legitimate messages. This allows people to build up corpora of spam without requiring a human to do the initial sorting, and thus can be a great tool for finding more messages.

Brightmail uses, among other methods, a "probe network" which is very good at detecting UBE. They set up a number of email addresses which are made available to spammers. These "spamtraps" are similar to "honeypots" in the security context: they attract and "trap" spammers who believe them to be regular email addresses. In fact, these addresses are then used to find and filter out mails which are sent to more than one of the addresses. Because no legitimate mail should be sent to these addresses, Brightmail can be quite sure that no legitimate mail will be caught. [Bri03] It does rely on the fact that most spam is sent in bulk, but used in conjunction with other methods, it helps Brightmail reach high accuracy levels.

This sort of idea is also employed by Cloudmark's SpamNet service [spa04c], only it uses real email addresses rather than ones which receive only spam. When a user of the SpamNet service reports a message as spam, his or her vote is used to decide whether to block that message from the rest of the community.

While Cloudmark and Brightmail's services are available to paying customers, this sort of information is also available for free thanks to community effots such as Vipul's razor [Pra04].

2.1.3.11 Bayesian-inspired Filters

Solution type
Technological
Mainly Content

The idea of using Bayes rule to sort spam was introduced in 1998 by [PL98] and [SDHH98], but the idea became much more popular after Paul Graham self-published his paper, "A Plan For Spam" [Gra02] which boasted of accuracies over 99% without accidentally marking any legitimate messages as spam. This accuracy was on his personal email using a system which was proprietary, so his results cannot be directly duplicated. His results can thus not be considered scientifically relevant, although they have made an impact upon research in this area. There have been challenges which show that Bayesian filters are not always so accurate, performing well below his stated results and not as well as other spam filtering products [CL04].

First, the message is **tokenized**, that is, it is divided up into a set of smaller, simpler features known as **tokens**. The actual definition of a token changes depending upon the specific algorithm used. For Graham's original algorithm, tokens were defined as sets of alphanumeric characters, dashes, appostrophes and dolar signs, with anything else considered token separators [Gra02]. (Note that Graham specifically excludes HTML comments from consideration, to avoid a specific attack where spam senders include HTML comments within words, so that a program sees `fr<!-- chipmunk -->ee` but the recipient sees the word `free`.) Later works have used more elaborate definitions including special handling of dollar ranges (for example, $20-25 becomes $20 and $25) or multi-word creations (See [Gra03] and [Lou03] as well as the CRM114 Discriminator [Yer04] and DSPAM [Zdz04]).

As an example, suppose we were trying to tokenize the phrase "It's 100% natural!" There are several common ways to tokenize this phrase. With the most basic

algorithm, which separates based on whitespace or punctuation, the resulting tokens would be as follows:

- It
- s
- 100
- natural

With an algorithm that includes punctuation, we would have the following tokens:

- It's
- 100%
- natural!

These might be in addition to the first set of tokens, depending upon the algorithm.

The multi-word tokenizers simply string together adjacent tokens. With a two-word tokenizer that doesn't include punctuation, we would have the following tokens:

- It s
- s 100
- 100 natural

And with a two-word tokenizer that includes punctuation, we would also have the following tokens:

- It's 100%
- 100% natural!

Some of the original work in Bayesian filtering included "stemming" words so that, for example, words such as *mailing*, *mailed* and *mail* were all handled as one [PL98]. It has not been shown that this ontology improves the results (quite to the contrary – Graham cites it as a possible explanation for poor results [Gra03]) so this technique does not appear to be widely used any longer.

Spam filtering using Bayes rule use two simplifying assumptions:

1. Tokens are independent.

   This is a common assumption in pattern recognition and the resulting Bayes rule is known as the **naive** Bayes rule. Although it is a common assumption, it is not actually true in many classification tasks. It is rarely true in email classification tasks: if an email to contains the word "lichen" it would also be more likely to contain nature-related words like "tree" or "field" than it would be to contain words about computer programming such as "algorithm."

   Despite the lack of validity of this assumpion, it significantly reduces the amount of calculation necessary, and even with this simplifying assumption, Bayes' rule can be fairly accurate.

2. The prior probabilities are identical (equal to 0.5).

   This estimation may actually be accurate for some people, but even when it isn't, it turns out that this estimation is good enough for most purposes. Using this estimation also simplifies the calculations. Graham suggests that using the actual prior probabilities could improve filter performance [Gra03]. These numbers would be derived by determining the ratio of spam to non-spam in the email already seen, and could updated constantly or on some sort of schedule.

With these assumptions, the simplified Bayes rule used for email filtering is as follows:

$$P(spam|token) = \frac{P(token|spam)}{P(token|spam) + P(token|nonspam)}$$

And each of the parts of that equation are defined as follows:

$P(spam|token)$ = the probability that the message is spam given that the current message contains a given token

$P(token|spam)$ = the probability of a token occuring in the message given that the message is spam

$$= \frac{\text{number of times the token is found in spam}}{\text{number of spam messages in database}}$$

$P(token|nonspam)$ = the probability of a token occuring in the message given that the message is nonspam

$$= \frac{\text{number of times the token is found in nonspam}}{\text{number of nonspam messages in database}}$$

Thus, for each token, the filter needs to store the number of times it has appeared in spam messages and the number of time it has appeared in non-spam messages. (The total can be found through summing these two numbers.) In this way, $P(token|spam)$ and $P(token|nonspam)$ can be recreated easily.

To determine the probability that a given message is spam, the filter looks at all the probabilities for each token found in the message and chooses the n most "interesting" ones. The most "interesting" tokens are those with probabilities furthest from the neutral 0.5, since those tokens indicate a strong tendency for the message to be either spam or non-spam.

$$\frac{\prod_{i=0}^{n} P(spam|token_i)}{\prod_{i=0}^{n} P(spam|token_i) + \prod_{i=0}^{n} (1 - P(spam|token_i))}$$

where n represents the total number of "interesting" tokens to use. Graham's algorithm uses $n = 15$ [Gra02].

The final score indicates the probability that a given message is spam, and users can then define a threshold so that the scores can be used to give a binary spam

or non-spam indication. Graham says that his threshold is at 0.9, but the exact threshold isn't very important, since few messages end up in the middle range of probabilities [Gra02].

2.1.4 Summary

Spam is a big problem which is escalating. Even though users are becoming increasingly frustrated with spam, significant numbers of people are purchasing the products and services offered. As spammers continue to profit, it seems unlikely that the problem will lessen in the near future.

The definitions of spam vary from person to person, but when it comes to spam filtering, "the customer is always right." That is, it is the deviation of the user whose mail is being filtered that should matter most to those producing the filter.

Although spam classification is a difficult task, there are already a variety of filters available. In studying these, we can pick out some features that seem to mark those which work best:

Dual Classification Ideally, we want to know if a message is spam or it isn't. Although it can be handy to know the likelihood of the message being spam and have a whole range of information available about the message, the core problem of spam classification is a simple yes/no classification. Supervised learning, where the user can order the system to change if it misclassifies messages, can help considerably in making this classification good for each user's needs.

Accuracy There is little point in having a system which is wildly inaccurate. The biggest danger in spam classification is that real messages might get lost. This is most likely to occur when the system incorrectly labels a non-spam message as a spam. This is called a false positive, and it is a problem because it is much easier to miss one legitimate message among many spams, since users typically don't

read all of their spams carefully, however they usually do read their regular mail. False negatives, where spam messages are incorrectly classified as non-spam, are much less problematic, although still undesireable.

Adaptability Because spam can be adapted as new versions of filters become available, it makes sense to have an adaptive system which changes even as spam changes. Current systems such as blacklists rely on constant updating, often done by humans. Automated updates could save a lot of human time and make the whole process faster.

Usability The cost of spam is in more than the hardware required to store and deliver the extra mails. Users waste significant amounts of time dealing with spam right now. If a system is easy for the senders and recipients of email, then it is not necessarily helping them very much. So while an ideal system would be adaptive, we want to limit the required human participation to the smallest possible levels. People may be quite interested in spending time tweaking their filters, but they shouldn't have to do so for the filter to achieve reasonable results.

Diversity In order to bring down the profits spammers are making, it would be ideal if many people were using many different systems. That way, one carefully tailored message could not make it through many systems.

An ideal system would work differently for every user, making it hard for a spammer to craft emails which can can beat every instance of that system. If the spammers have to circumvent a unique spam system for each user they want to reach, this brings up their costs considerably.

Generalizability While some systems have been successfully built based upon the idea of recognizing identical spam, some person or machine has to receive the spam first, report it, and then everyone else must be informed so that they

know to reject it. This is how spam trap email addresses work and this system also works fairly well for anti-virus tools. However, it would be even more convenient if an ideal system could immediately recognize similar messages, rather than requiring each new variation to be seen before it can be blocked. By having an associative memory which can recognize items similar to those seen previously, it should be possible to detect more spam with less work.

Section 2.2 explains how the immune system has some of the qualities we want to have in a spam detector.

2.2 Immune Systems

An immune system's main goal is to distinguish between self and potentially dangerous non-self elements. In a spam immune system, we want to distinguish legitimate messages from spam. Like biological pathogens, spam comes in a variety of forms and some pathogens will only be slight variations (mutations) of others. Self and non-self can be translated to non-spam and spam.

The immune system also possesses other qualities that make it attractive as a spam filter model. The systems produced are diverse, meaning that it would be difficult for a single email to be crafted to penetrate multiple filters. The biological immune system deals appropriately with pathogens except in the case of exceptional diseases. (If the immune system were not accurate, the lifespan of the average human would be much shorter as the system mistakenly attacked important cells or failed to attack viruses and other dangerous non-self.) Finally, as well as handling known infections, the immune system can adapt to new things reasonably quickly, and remembers what it has seen before.

Theories of how the biological immune system works can serve as starting point for creating computer systems. This section describes some of the components of the

immune system which have inspired my artificial immune system.

The description here is a simplification of the complex knowledge we have about immune systems. I have derived knowledge from a number of sources, some of which relate to immune systems as computer models ([SC01], [Das98], [TBH03]) and some of which describe the immune system from a more purely biological standpoint ([WC02], [Som03], [BSL96], [Bra04], [Pla84], [RBM98]). I have used a reduced immune system model as a starting point for my spam immune system, so it is only this reduced model which is described here.

2.2.1 What does the immune system do?

The immune system exists to protect organisms from potentially harmful agents such as bacteria, viruses, and other foreign life forms and substances. These dangerous non-self agents are often referred to as ***pathogens***. The immune system accomplishes this goal by carefully distinguishing the ***self*** (parts of the organism protected by this immune system) from ***non-self*** (anything else).

It is this classification of self and non-self that makes the immune system an appealing model for spam detection, which also requires a classification between the legitimate messages (the self) and spam (non-self).

As we have seen in the discussion of spam, most commercial-grade solutions use multiple techniques to achieve higher accuracy. Similarly, the immune system is not reliant upon a single technique, but instead consists of many layers which all help to protect the body:

The Skin and Mucous Membranes form the outer layer of defence, a physical barrier against attack.

Physiological Defences such as high pH or temperature make it harder for pathogens to survive in the body.

The Innate Immune System is a non-adaptive system available at birth. Its quick reaction to potential infections can stop many attacks before they can get underway, and it also serves to activate the adaptive immune system.

The Adaptive or Acquired Immune System can handle invaders which have been missed by the innate immune system. It takes longer to mount a response the first time something is encountered, but after that can detect the same thing or similar things very quickly.

All of these layers are quite interesting and may be useful as models for computer programs, but it is the adaptive immune system that inspired this particular work.

2.2.2 The Adaptive Immune System

The central component to the adaptive immune system is specialized white blood cells called ***lymphocytes***. These serve to identify anything in your body, and act upon those things which are not part of self. Rather than attacking everything, the body uses a system called the Major Histocompatibility Complex (MHC) which marks the cells of the body as self. Anything not carrying these markings may be attacked by the immune system [Bra04].

When a cell is infected, the MHC molecules present fragments of surface proteins on the surface of the cell. These fragments are called ***antigens***. The immune system checks the antigens using a specialized detector called an ***antibody***. Each lymphocyte actually has many copies of the same antibody on its surface, and it detects pathogens when antibodies on the lymphocyte cell surface bind to antigens. This binding does not have to be perfect: if the antibody and antigen are "close enough" to a perfect match, binding will still occur, although not as strongly as it would for a perfect match.

To apply these ideas to spam, we treat spam as a pathogen, and the complete

message is used as an antigen. The lymphocytes are digital bits of information, and each one includes a pattern which is used as an antibody. This is described in more detail in Section 3.1.

2.2.2.1 Creation of Antibodies

Antibodies are constantly being created by the body and each associated lymphocyte can live for as little as days or as long as years. The body has a library of gene fragments which represent all the necessary information to create detectors for all the possible pathogen types. In order to create the ***repertoire*** or population of lymphocytes in the body at any given time, elements from this library are randomly recombined to produce a diverse population of receptors. As you can see in Figure 2.6 a small set of gene fragments can be used to produce many different results.

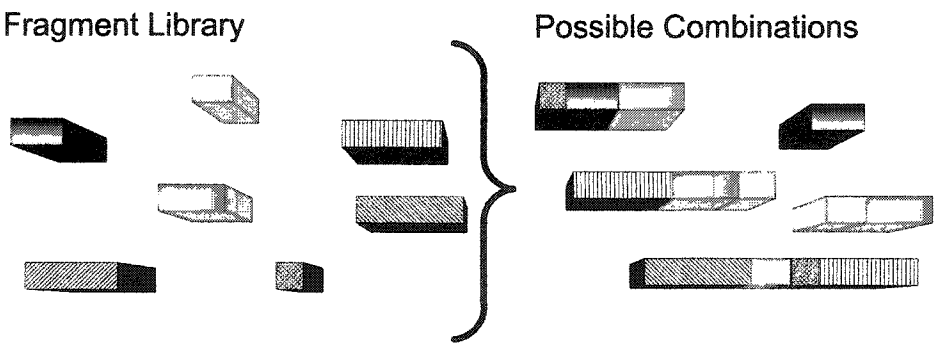

Figure 2.6: Random recombination of fragments

The creation of digital antibodies for the spam AIS is also done with a library, only instead of gene fragments, this library contains fragments of text-matching patterns. These fragments may not be combined completely randomly, since it may be desirable to bias the production of combinations based upon utility of the fragments during execution of the algorithm. More information about the creation of digital antibodies can be found in Section 3.2.1.

2.2.2.2 Autoimmune reactions

If the antibodies are created through random recombination, how can we be sure that these antibodies won't detect self? When antibodies detect self and a reaction occurs, this is called an ***autoimmune reaction***. These are avoided through a fairly complex process. There are actually classes of lymphocyte, the B-cells and the T-cells.

B-cells are named for the "Bursa of Fabricius" a lymphoid organ found in birds, where B-cells are produced. Humans do not have this organ, but it is thought that the equivalent is found in the bone marrow. [WC02] The B-cells originate and mature largely in the bone marrow of humans.

T-cells are named for the thymus, which is where they mature (although they also originate in the bone marrow). The thymus contains many different types of self-proteins, and as the T-cells mature, those which match self are killed off or are not selected to reproduce.

Before the body actually mounts any attack, B-cells must detect a cell as being foreign, and a T-cell must ***activate*** the B-cell before any further action is taken. Since T-cells cannot detect self, this confirms that the B-cell is not mistakenly detecting self. In this way, the body avoids autoimmune reactions.

2.2.2.3 Detection/Binding

Detection is done through binding, but it should be noted that this binding is not an exact process. One lymphocyte's antibodies may bind to many different antigens, although some will bind more closely than others. The antigens to which a given antibody will bind must be similar in shape, but do not need to be exactly the same.

Antigens are usually large proteins with fairly complex structures and different three-dimensional shapes. The antibodies are also three-dimensional shapes which fit together with the antigens, somewhat like a puzzle. The strength of the binding depends on how closely the two shapes can match. This is determined not only by

the actual physical shapes, but also by the charges involved which may attract or repel different parts of the detector and detectee. A given detector will bind to many targets, and a given target might have multiple detectors which can match it.

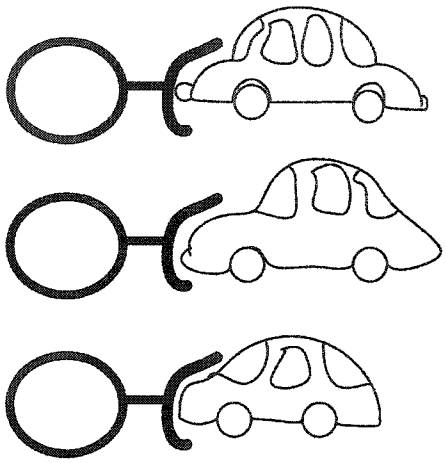

Figure 2.7: Inexact matching of antibodies

This inexact matching can been seen in Figure 2.7, where the same antibody matches three shapes. The first matching is not very strong, but the next two come much closer to matching the smooth curve of the antibody.

Although these are only two-dimensional representations, it may help to think of the antigen as having a shape like a much larger and more familiar item, such as that of a car. A car is a good analogy because there are many different shapes of cars, and it's hard to find truly identical cars, since you have to worry about manufacturing differences, dirt, dents, and other modifications to the car after it's been sold. So rather than matching exactly, we match them in groups: if the car manufacturer and model are the same, we claim that they're the same car. More broadly, we look at classes of car such as "family sedan" or "sports car" and consider those roughly equivalent for many purposes.

If we were building antibodies for cars, they would probably match at that last level. One antibody might match many types of compact hatchback cars, for example, by having a shape similar to that of the "hatch" trunk. This antibody might also

match other vehicles with similar shape (for example, some small vans might have similarly-shaped rear ends) and might be unable to detect some cars with alterations in this area (for example, a large spoiler attached to a hatchback car could stop the antibody from binding to it). The strength of the binding is determined by how close the receptor and the thing it's detecting can get (including attraction of charges as well as simple geometry), and a threshold level must be reached before the two are said to bind.

There are approximately 10^{16} different foreign proteins which the immune system must recognize, yet the repertoire of the immune system contains a much smaller number of actual receptor types, closer to 10^8 [SC01]. By doing this approximate binding, the immune system can use a smaller number of antibodies to detect a large number of potential pathogens, as long as pathogens have similar shapes.

2.2.2.4 Memory

Since lymphocytes with new antibodies are being created constantly, it seems that the immune system would be constantly re-learning the same pathogens over and over again. However, we know that for some diseases, such as chicken pox, most people are immune for life.

One theory as to how this occurs suggests that there may be special ***memory cells*** which are created once a pathogen has been successfully detected. These are a special type of long-lived lymphocyte which stays in the body forever, meaning that these lymphocytes are always available to react to an infection.

Another theory suggests that some pathogens stay in the body as low-level infections, so new lymphocytes are always being made which detect these pathogens.

Both of these theories are useful when applied to the idea of lymphocytes for spam. Email viruses and some spam messages contain substantial amounts of identical material, and you want to be able to recognize identical or similar messages for

long periods. These periods may not be forever – there's little benefit in keeping lymphocytes relating to spams that you won't see again, such as those for products related to current events. In spam, there is definitely a level of constant infection, so by combining the idea of long lived lymphocytes and constant stimulation, we can develop a system which balances the need to remember with the need to forget. This is described in greater detail in Section 3.2.4.

2.2.3 Summary

The immune system exists to protect organisms from potentially harmful agents such as bacteria, viruses, and other foreign life forms and substances. These dangerous non-self agents are often referred to as pathogens. The immune system accomplishes this goal by carefully distinguishing the self (parts of the organism protected by this immune system) from non-self (anything else).

This is done using special detectors called lymphocytes. Although they are created randomly, they are trained and remember infections so that the organism is protected from future intrusions as well as past ones.

It is the classification of self and non-self that initially makes the immune system an appealing model for spam detection, which also requires a classification between the legitimate messages (the self) and spam (non-self). Section 2.1 described some of the qualities desired in a good spam filter. The immune system model has many of those listed:

Dual Classification The immune system distinguishes between self and non-self.

Adaptability The system learns new pathogens as they arrive, and remembers them so re-infection does not always result in illness. In fact, the organism may never realize that it has been re-infected.

Usability Organisms don't generally have to do anything to maintain their immune

systems beyond things they would do anyhow (rest and eat, for example). A spam immune system likely cannot have this kind of transparent operation, since users are not "born" with a clear set of "self" that the system can learn. However, little input should be required to help the system along if the self and non-self do not change to rapidly.

Diversity A plague rarely kills all organisms. A cold or other disease rarely strikes all of those who are exposed to it. Why? Because individuals' immune systems are all different: each one has learned different pathogens over time, and each one has started with slightly different gene fragments from which to build antibodies. Immune systems are inherently diverse, like the organisms they protect.

Generalizability Because each antibody does not need to match exactly in order to bind to an antigen, the immune system already has the capacity to generalize. Each antibody has a whole shape space of similar antigens it can detect. With this type of associative memory, the immune system recognizes not only exact matches but also patterns similar to those already seen.

With these qualities in mind, Section 3 explains how a spam immune system can be designed.

Chapter 3

The Spam AIS

Solution type
Technological
Mainly Content

The human immune system distinguishes between self and non-self, so the spam immune system distinguishes between a self of legitimate email (non-spam) and a non-self of spam. Right from this seed idea, the biological immune system has an advantage over the digital. The biological self does not change in ways that matter to the immune system: the surface proteins used to distinguish the self remain the same over the lifetime of the organism. Unfortunately, the spam immune system has the same problem found in computer security immune systems [FHS97]: the self changes over time. As a person meets new friends and business contacts, discusses current issues, develops new interests, and even learns new languages, the content and meta-information for the messages that person receives will change.

This does not mean that the immune system model cannot be used for spam detection, only that the model must be used with some caution. The system must be able to forget as well as learn things. A healthy person might have no interest in pharmaceuticals, so any mail containing information about drugs could be dropped. But if that same person is diagnosed with a severe disease, he or she may start to have discussions which include words and phrases which formerly indicated spam.

The system must be able to adapt accordingly.

In addition to the changing self, an email classifier has a changing non-self. The mammalian immune system is built to handle a changing non-self, but by knowing more about the way in which spam changes, it should be possible to incorporate that knowledge to train detectors more effectively. It is known that spam is not very volatile for long periods of time, but can change very rapidly [Sul04], so we need to be prepared to keep lymphocytes for long periods, but be able to adapt more quickly when necessary.

This section describes the components of the spam immune system and how they work together to learn and un-learn messages in order to classify them accurately. Section 3.1 describes the parts necessary for the system to work: The input messages (Section 3.1.1), the lymphocytes (Section 3.1.2), and the library used to create the antibodies (Section 3.1.3). The lifecycle of a digital lymphocyte is described in Section 3.2, which explains how the antibodies are created, used, and how they eventually die.

3.1 Parts of the Spam AIS

The central part of the spam immune system is its detectors. In order to understand how they work, we must first look at what they detect (see Section 3.1.1), what comprises a detector (see Section 3.1.2), and what is used to create them (see Section 3.1.3).

3.1.1 Antigen and protein signatures: Email

For the purposes of this system, the entire message is used as the "antigen" or protein signature for the system to match. This means we use the headers and the body of the message, including any attachments. The system can thus be trained based on the source of the message as determined by the headers, or by the contents of the

message. Spam sites move around to avoid being continually blocked by sites using blacklisting, so it is expected that the source-related detectors will be shorter-lived than the content-related detectors. For this reason, the spam AIS is considered to be a mainly content-based technological filter.

It is not necessary to use the entire message: Results in email classification have been found using only header information (see Section 2.1.3.7 on Blacklisting for an example), and other solutions have used reduced versions of messages with reasonable success [PL98] [SDHH98]. However, we risk losing important information by discarding parts of the message [Gra02]. It is typically better to use whole messages with minimal alterations when compiling a corpus of messages [O'B03].

In the biological system, the matching is done upon the pieces presented by MHC (See Section 2.2.2). Some potential methods for finding a useful subset of the message data include:

- Using a text summarization tool to produce a shortened version of the email. This could be a list of keywords or a small abstract. One such tool is described in [Tur03].
- Converting all HTML to plain-text
- Looking only at a subset of the headers (for example, trying to classify based upon the subject and the sender data).
- Discarding information in the headers (since this is likely to be less useful in the long term since spam senders tend to change servers relatively frequently to avoid blacklisting) and using only data in the body.
- Discarding non-meaningful random words that do not form complete sentences. These random words are intended to confuse Bayesian filters [Gra03], and rarely have anything to do with the aim of the message.

- Removing common words (such as "the")
- Reducing words to their roots. Something along these lines was used in [SDHH98]. Some sort of semantic analysis would have to be done to ensure that the roots were related to the correct meanings.

If there was an established safe way to reduce the message to a useful subset, this would probably be a good approach. Reducing the size of the message could decrease the processing time required for such a system. Unfortunate, since experts do not seem to agree on what minimal information is necessary to detect spam with high accuracy, it seems safer to use the entire message at this time.

Although it would be possible to interpret attachments such as images in a more meaningful way, currently the entire message is treated as a text string. This is still useful, since text-encoded attachments can still contain patterns that might be recognized by the system. However, even more information can be gleaned by using a tool which understands a particular attachment. For example, future work might include doing text recognition on the images and incorporating that additional text into the antigen, or integrating the immune system with systems that could perform analysis on other attachments. For example, an anti-virus system could be used to recognize viral attachments.

While all these ideas are interesting and may represent future work, the current implementation of the spam immune system uses a very simple representation: raw, unaltered messages. No interpretation is done before the message is seen by the system, the whole things is treated as a simple text, and the system does not do anything special with images or other attachments.

3.1.1.1 The Non-self: Spam

The non-self in the spam immune system is the spam. The definition of spam used is the practical one described in Section 2.1.1.1, which says that spam is what the user

defines it to be. Because the system is learning from human input, it simply learns based upon that human's choices.

3.1.1.2 The Self: Non-spam

Unlike in a biological immune system, where the lymphocytes themselves are part of the "self" that needs to be protected, the self and the immune system are separate in the spam immune system. The self constitutes only non-spam messages sent to the user, as defined by the user.

3.1.2 The Lymphocytes and their Antibodies: Digital Detectors

It should be noted here that when I refer to a ***digital antibody***, I refer to the particular detector. I use the term ***digital lymphocyte*** to refer to a set of things: the digital antibody and associated weight information.

Each digital lymphocyte stores the following:

- The digital antibody. (See Section 3.1.2.1)
- The weighting information learned for this antibody. (See Section 3.1.2.2)

3.1.2.1 Digital Antibodies

Antibodies in the human immune system do approximate matching, and the spam immune system uses string pattern matching to simulate this. Inspired by the heuristics used by SpamAssassin, the system can build up moderately complex patterns.

SpamAssassin uses heuristics defined directly by humans. These heuristics describe common patterns in email. Some of these were described in Section 2.1.3.5.

Pattern matching is used so that a given antibody can be used against more than one "infection" of spam, just like the biological immune system reuses antibodies not

only for re-infection but also for similar infections. Using a memory that is associative, so that the specific emails and the regular expression patterns are associated, allows for a one-to-many sort of memory. Doing this allows the system to use less memory than it might otherwise. Even if memory is not an issue, there's very little point in matching an entire message exactly with spam, since so many spammers pad their messages with random text, a practise that has become increasingly common with the rise of Bayesian spam filters [Gra03]. This padding involves adding unrelated text to a message so that there will be more tokens available for the Bayesian filter. Since some Bayesian spam filters use only a certain number of the tokens whose scores are furthest from 0.5, it would be possible for a well-chosen text to entirely negate the effects of words that are common to spam. Although random words are a common tactic, some spammers use actual English texts. Some spams seen by the author contained a series of quotes from the popular television show The Simpsons, presumably on the assumption that many people would quote this show to their friends, and thus the tokens generated from that text would have a low spam score.

In a simple example, a spam message could use the string "enl4rge" instead of "enlarge" to avoid filters checking for that word but not for variants upon it. By using regular expression antibodies, the system can assign an identical weight to many possible strings. (These weights are used later to determine a spam score for each message.) "Enlarge" and "enl4rge" and "enlarg3" are all read the same way by the human recipient of a message, so it makes sense to allow the immune system to treat them as the same string [OW03a].

The idea here is similar to the concept of virus "signatures" used by anti-virus companies. The spam immune system was inspired by work with immune systems for computer viruses [WSP+02], so it helps to think of the system from this perspective. A ***virus signature*** is defined as follows:

> A unique string of bits, or the binary pattern, of a virus. The virus

> signature is like a fingerprint in that it can be used to detect and identify specific viruses. Anti-virus software uses the virus signature to scan for the presence of malicious code. [vir04]

Viruses have different minor mutations; email viruses in particular often come with different subject and from headers. But the signature of the virus, a small subset of the whole virus, is likely to remain the same [Sta99]. The spam immune system attempts to find patterns that, like signatures, will appear in many different mutations of spam.

See Section 3.2.1 for a description of how lymphocytes and their antibodies are created.

3.1.2.2 Weights

Each lymphocyte retains two pieces of weight information:

- *spam_matched*: the cumulative weighted number of spams matched by this lymphocyte.
- *msg_matched*: the cumulative weighted number of messages matched by this lymphocyte.

Both of these numbers are initialized to zero. When the detector matches a message, *msg_matched* is incremented, usually by 1. If that message is known to be spam, *spam_matched* is also incremented. These numbers vary depending upon whether the message is looked at during training, regular operation, or re-training because an erroneous classification was made:

Training During initial training, the messages seen by the system have already been classified, so *spam_matched* is incremented by 1 if the message in question is spam, and 0 if it is legitimate mail.

The system may also be trained on messages that have been classified by another system. In this case, the numbers used are set by the user and indicate the likelihood that the classification is correct. For example, if system A classified a spam message and system A has a 10% chance of being wrong, you might increment *spam_matched* by 0.9 instead of by 1. Similarly, if the message was classified by system A as non-spam, then you would increment *spam_matched* by 0.1 instead of 0.

Regular Operation During regular operation, nothing is known about each message that comes in except what the system determines itself, so *spam_matched* is incremented by the score determined by the system.

If more is known about the probability that an unknown message is spam, this information could be factored in, but since the work with Bayesian systems implies that such information is not necessary to achieve reasonable results [Gra02], it is not considered necessary for the spam immune system.

It would also be possible to use Shafer-Dempster theory to allocate these values, but this is not done with the current system.

Re-training If we are re-training the system, then it has made a mis-classification, and was trained upon the basis of that mis-classification. Thus, we need to compensate for that mis-training as well as adding a new training value. Retraining is equivalent to repeated regular training, except that to retrain you must know the value originally assigned to the mis-trained lymphocytes, so that it can be changed to the value used for spam or non-spam as indicated by the user. The weight assigned to a retraining is the number of regular trainings to which it is equivalent.

Testing mode There is also a special testing mode where neither *msg_matched* nor *spam_matched* is updated.

Section 3.2.3 explains in more detail how the weights are updated over the lifespan of the lymphocyte.

3.1.3 The Library

The gene library contains partial patterns used to build the full patterns used in lymphocytes. The partial patterns, in this case, are small patterns which can be combined with other small patterns to create a larger one, or can stand alone as complete patterns by themselves if necessary. For the spam immune system, we chose to use small patterns which represented heuristics used for finding spam or for finding non-spam, but this was not the only reasonable choice.

There is one primary aim in building the library: it should produce lymphocytes which actually detect messages (spam or non-spam). Lymphocytes which do not ever match any messages are not useful to the system, since they cannot learn and contribute to the decision as to whether a message is spam or not spam. A ***"useful" lymphocyte*** is one which has matched at least one message and has a *msg_matched* value that is larger than zero. Any lymphocyte which has a *msg_matched* value of zero does not contribute to any message scores.

The secondary aim is that the library produce such matching lymphocytes with a high frequency. Again, the more lymphocytes which match a message, the more likely it is that some of them will make a noticeable distinction between spam and non-spam, making the classification better. By having fewer "useless" lymphocytes, we are wasting less processing time and memory. Note that useless is in quotes here: a lymphocyte which matches nothing for a long time may at any point become useful if the right messages start to come in to the system, so it's unfair to discount them entirely. The lifetime of the lymphocyte is important, and the lifespan of each lymphocyte is altered depending upon the number of times the associated antibody matches a message.

Ideally, we want to produce many lymphocytes which do match and few which remain unused. Several different libraries were tested in the development of this system.

3.1.3.1 All possible characters

It would be possible to use a library which contained every possible character in email. If such a library were used, it would be necessary to train a very large number of lymphocytes, since most of the lymphocytes created as random character strings would not match very many messages. For example, consider the string "kqpcrmd" versus the string "this is" – those two strings are equally likely to occur given a random selection process, yet the latter is much more likely to match an email. However, the number of random strings which form actual words is much lower than the number of random strings which don't.

Although it would be possible to use this very general library and train the lymphocytes it could produce, a very high number of lymphocytes would need to be created in order for there to be enough that match. Fortunately, some information about the structure of email is known in advance, and this can be used to reduce the number of lymphocytes that need to be generated. For example, we know that most emails (spam and non-spam) contain actual words, not just random character strings. The next libraries use this information to reduce the number of lymphocytes which need to be generated.

3.1.3.2 A dictionary of English words

For the personal email of an English speaker, most of the messages he or she receives will likely be in English. As such, the first library attempted was a list of American English words, taken from version 5-4 of the Debian package wamerican. This dictionary contains 96274 words.

The library of English words was attempted because most of the corpus of email being trained is English; the library would not have extended well to non-English messages. Results from this library can be found in Section 4.2.4.

3.1.3.3 Bayesian-style tokens

Another library used Bayesian-style tokens. The SpamBayes Bayesian-style tokenizer was used to parse a training set of emails into tokens, which were then used as the gene library. The Bayesian tokenizer divides a mail up into separate components, usually individual words. This was described further in Section 2.1.3.11.

The SpamBayes [spa04a] tokenizer was used to produce a set of tokens from the training set of messages. Their implementation is based upon the work of Paul Graham [Gra02], but includes many additions not found in his work [spa04a]. Each modification was tested and has to be shown to make a noticeable increase in classification accuracy before it was incorporated into the SpamBayes code [spa04b].

3.1.3.4 Heuristics

The library which gained the best results is a library of heuristics. Using full libraries of words wasted valuable knowledge that was available about spam and non-spam messages. For example, although both messages contain common words like "the" the presence or absence of such common words tells us little about the likelihood of the message being spam. By concentrating on words and phrases which are more likely to indicate a classification for the message, the system produces more "useful" detectors and can achieve results with a much smaller set of detectors.

The heuristic library is much smaller than its counterparts. The heuristics used are drawn from SpamAssassin [sa-04d], information about the training results of Bayes classifiers [Gra02] [Gra03], as well as directly from examination of spam. They may have been created by humans based on empirical evidence or created by machines

(for example, the Bayes results are machine-generated). Some potential heuristics include:

- Does the message claim that you can unsubscribe from the list by replying with the word "remove" in the subject line?
- Does the string "NO QUESTIONS ASKED" appear anywhere in the message?
- Does the message claim that it is not spam?
- Does this message claim that the product has been seen on well-known news source such a large TV network?

For that last example (the message claims that the product has been seen on well-known news source) the heuristic is as follows:

```
seen on\b\s*(?:T\.?V\.?|ABC|NBC|CBS|CNN|Oprah|USA Today|48 Hours|
New York Times|\w+\s+T\.?V\.?|:)
```

Although the actual heuristic does not have a line-break in it. This particular heuristic was taken directly from SpamAssassin 2.64 [sa-04d].

This heuristic pattern would match the following phrases:

- seen on TV
- seen on T.V.
- seen on New York Times
- seen on:
- seen on CBS
- seen on CBS

And many other variations.

Figure 3.1 shows 20 of the gene fragments which can be found in the complete heuristic library. These are all partial regular expressions, although some contain no meta-characters.

```
reply.{1,15}remove.{1,15}subject
ff0000
\<BODY.*bgcolor="#?[^f]
LOSE WEIGHT
100% GUARANTEED
\bno (?:cost|charge)\b
Dear [A-Za-z0-9_-]+\@
subject to credit approval
\b(?:boost|increase|grow|larger|bigger|higher) (?:traffic|sales)\b
e-*mail marketing
you (?:do not|no longer) wish to receive
javascript:
This.{0,30}is not (?:a )?spam
-----BEGIN PGP SIGNATURE-----
\b(?:college|university)\s+diplomas
You won'?t be diss?app?ointed
(?:100%|completely|totally|all) natural
(?:toner|ink(?:[-\s]*jet)?|fax|copier)[-\s]+cartridge
(?:\$|US\$|usd?).?\d{2,3}(?:\.\d)?.?(?:m|millions?)
FrontPage.Editor
```

Figure 3.1: Sample gene fragments from the heuristic library

Using a smaller library has advantages for speed, but there are also drawbacks. One of the most significant problems for learning occurs when a message is found that no detector matches. With a library that is not utterly comprehensive, it may be possible that no gene combination could even produce such a detector.

3.1.3.5 Other Libraries

Although most of the work with the spam immune system has been done using a heuristic library, this is not the only possibility.

For example, it would be possible to use multiple libraries at once. The human immune system already selects from three distinct categories to build its detectors [Pla84]. The heuristic library already combines information from many sources, but if there were specific libraries whose gene fragments work better together, it would be possible to generate some lymphocytes from one library, some lymphocytes from another library, and so on. If one library is found to produce few useful lymphocytes, it could be removed from the set of libraries in use, or the number of lymphocytes created with it could be reduced.

This multiple-libraries approach still uses static libraries, but another interesting idea involves making the library more adaptive. Although this is not normally done with an artificial immune system, this may be a way to compensate for the incomplete library being used. The static heuristic library, for example, is based upon knowledge of previous spam messages, and it would be convenient if we could also use the most recent data available by creating detectors from the messages seen by the immune system while it is running. It would also potentially be interesting to incorporate feedback about the utility of the detectors so that the utility of particular gene fragments could be known and the random recombination process biased to use the most useful fragments most frequently. Adaptive libraries are a potential future direction for this research and have not been attempted in the current implementation.

Extending the idea of using Bayes tokens, it would be possible to run the system with a Bayes tokenizer adding new information to the library. A list of Bayes tokens could be kept and then the most recent list of tokens would be used as the library when new lymphocytes are created.

The problem with this idea, as with the static library of Bayes tokens, is that the library could be so large that the chances of producing a useful lymphocyte through random recombination is fairly low. The chances could be increased by using the Bayes scores as weights so the heavier-weighted Bayes tokens would be more likely to

be selected.

It would also be reasonable to use a tool which produces fewer gene fragments than a Bayesian tokenizer. The keyword extraction algorithm by Peter Turney [Tur03], for example, could be used to produce a list of keywords or keyphrases from each training mail. These keywords or phrases approximately correspond to the most important parts of the message, such as the name of the product or products being sold, or key features of said products. These core parts are more likely to appear in other messages, since spam is frequently selling the same or similar items.

These libraries have not yet been fully explored. The only three libraries which have been attempted are the English library described in Section 3.1.3.2, the Bayesian token library described in Section 3.1.3.3, and the heuristic library described in Section 3.1.3.4. The results from these libraries is described in Section 4.2.4.

3.2 Lifecycle

The lifecycle of a digital lymphocyte starts when the lymphocyte is created. This is described in Section 3.2.1.

Once the lymphocyte is created, it can be used to match messages. This process is described in Section 3.2.3.

After some time has passed (this time interval is chosen by the user), the lymphocyte is aged and potentially expired. The amount of time is a parameter of the system, defined by the user. The idea is to give the lymphocytes a chance to match and be assigned a weight, but not so much time that the system is wasting time on lymphocytes that do not match any messages. Thus, this time interval will depend upon the number of messages that the user receives every day. Although it is probably easiest to keep this value static, it could be reduced in times when the system has to learn more quickly because it has been making mistakes, or lengthened when

the system is doing well.

Section 3.2.4 describes the ageing process and the procedure involved when a lymphocyte dies.

This lifecycle is described more precisely in Algorithm 1.

Algorithm 1 Spam Immune System

Require: *update_interval* $\Leftarrow$ a time interval after which the system will age. {chosen by user} {e.g. 10 days from now}

repertoire $\Leftarrow \phi$ {Initialize repertoire (list) of lymphocytes to be empty}
update_time $\Leftarrow$ *currenttime* + *update_interval* {time of next lymphocyte update}

Generate lymphocytes (See Algorithm 2)
Do initial training (See Algorithm 3)
while Immune System is running **do**
  if *message* is received **then**
    Apply lymphocytes (See Algorithm 4)
  end if
  if current time > *update_time* **then**
    Cull lymphocytes (See Algorithm 5)
    Generate lymphocytes to replace those lost by culling (See Algorithm 2)
    update_time $\Leftarrow$ *currenttime* + *update_interval* {t}ime of next lymphocyte update
  end if
end while

The life cycle of an individual digital lymphocyte is represented graphically in Figure 3.2

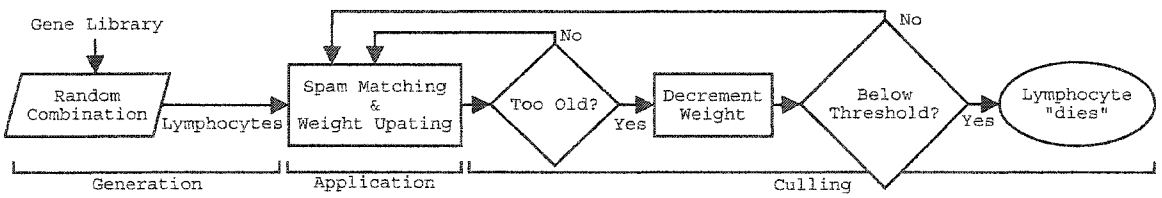

Figure 3.2: The life cycle of a digital lymphocyte.

When the system starts, it needs to initialize the repertoire, the list of available antibodies. It also needs to determine the time of the next update based upon the interval provided by the user. This *update_time* determines how frequently culling will

occur and new lymphocytes will be created. The best choice of *update_interval* will depend upon the volume of email received by the user as well as the user's personal preference for the system. This value is related to the rate at which spam and regular email is changing. It might be interesting future work to define this parameter using a genetic algorithm, but currently it is a static value. We want to give the lymphocytes a chance to be trained and keep them as long as they can be useful, but we don't want old lymphocytes to stick around the system when they are no longer being used or have become incorrect.

Next, the system generates the lymphocytes (See Section 3.2.1) and does initial training of these lymphocytes (See Section 3.2.2).

Once training is over, the system waits for messages. Each time a new message is received, the lymphocytes are applied to it (See Section 3.2.3) and it is assigned a score based upon the lymphocytes. This score determines whether each message is spam or non-spam.

Most of the time, the system does nothing while it is waiting for a new message. In fact, the process may just be invoked when a new mail arrives rather than taking up space in memory.

When the *update_time* passes, the system "ages" the lymphocytes, culling those that aren't fit enough to survive (See Section 3.2.4). New lymphocytes are generated (See Section 3.2.1) to replace those which die. Finally, a new *update_time* is set based, again, upon the interval (*update_interval*) provided by the user. The system then goes back into its waiting state until a new message arrives.

3.2.1 Creation of Lymphocytes and their Antibodies

As described in Section 3.1.2, a lymphocyte contains weighting information and an antibody. The weighting information is simply initialized to zero, but the antibody must be created from a gene library such as those described in Section 3.1.3.

The antibodies are created completely randomly for simplicity. As described in Algorithm 2, each antibody starts with a gene fragment randomly chosen from the library. A random number between 0 and 1 is generated from a uniform distribution, and if that number is smaller than the probability for appending to occur, then another randomly chosen gene fragment is appended to the antibody. It continues to grow in this manner until the random number generated is larger than or equal to the probability of appending. Between each gene fragment a wildcard (a pattern which matches 0 or more characters, in this case) is placed.

Algorithm 2 Generation of lymphocytes

Require: *library* $\Leftarrow$ a gene fragment library (cannot be empty)
Require: *repertoire* $\Leftarrow$ the list of existing lymphocytes (may be empty)
Require: *p_appending* $\Leftarrow$ the probability of appending to *antibody* {chosen by user}

```
while repertoire is smaller than the required size do
  lymphocyte ⇐ a new empty memory structure with space for an antibody, and
  the numbers msg_matched and spam_matched
  antibody ⇐ '' {An empty string to start the new antibody being created. This
  will be a regular expression made up of genes and wildcards.}

  lymphocyte.msg_matched ⇐ 0
  lymphocyte.spam_matched ⇐ 0
  repeat
    antibody ⇐ randomly chosen gene fragment from library
    x ⇐ randomly chosen number between 0 and 1 {uniform distribution}
    while x < p_appending do
      newgene ⇐ new randomly chosen gene fragment from library
      antibody ⇐ concatenate antibody, an expression that matches 0 or more
      characters, and newgene
      x ⇐ new randomly chosen number between 0 and 1 {uniform distribution}
    end while
  until an antibody is created that does not not match any in the repertoire

  lymphocyte.antibody ⇐ antibody
  Add lymphocyte to repertoire of lymphocytes
end while
```

The system must start with a gene fragment library (as discussed in Section 3.1.3) and the repertoire (a list of existing lymphocytes, which may be an empty list). In ad-

dition, it needs a probability of appending to a new antibody. This value, *p_appending* is chosen by the user. Generally, a value of 0.5 is reasonable – this means that there is a 50-50 chance that the antibody will get longer with each step – but other values can be tried to encourage formation of longer or shorter antibodies. The optimal value for this will depend upon the library being used. A simpler library such as one which contains individual characters will probably generate more useful lymphocytes if *p_appending* is higher. There is no maximum antibody length, although the chances of producing longer antibodies is lower than that of producing shorter antibodies.

The random recombination algorithm can be biased so that the random selection is weighted and some fragments have a higher chance of being selected than others. In this manner, production of combinations could be biased based upon the utility of the fragments, should some information about the fragments be known. This is not currently used and tested with this implementation of the spam immune system, but could be a useful extension for future use.

From there, the system creates new lymphocytes until the repertoire is filled. First, the *msg_matched* and *spam_matched* values are initialized to zero. Then, the antibody is created.

The creation of the antibody starts with the choice of a random gene fragment from the library. A random number is generated, and if it is less than *p_appending*, another gene fragment is chosen and appended to the antibody with a separator between the two fragments. This appending process is repeated until the random number generated is more than *p_appending*

This process may be clearer with an example. Suppose that our library consists of only three gene fragments:

$$library = \{A, B, C\}$$

And the probability of appending is 0.7 or 70%.

$$p\_appending = 0.7$$

One of the gene fragments is randomly chosen to be the first gene fragment in the *antibody*. Suppose that the one chosen is B.

$$antibody = B$$

A random number is generated from a uniform distribution between 0 and 1. On this first round, the number is 0.3. This is smaller than *p_appending* so another gene fragment is added to the antibody, along with a wildcard (.*) to separate the two:

$$antibody = B. * A$$

Another random number is generated from the uniform distribution between 0 and 1. In this second round, the number is 0.62 so another gene fragment and wildcard are added:

$$antibody = B. * A. * A$$

There is no problem with the same gene fragment appearing multiple times in the final antibody. Another random number is generated, but this time the number is 0.87 so the antibody is now finished and its associated lymphocyte will be added to the repertoire of the immune system.

The random selection of gene fragments from the library is such that each gene fragment has an equal chance of being selected each time, but this selection process could be altered should there be a good reason to favour some gene fragments over others. For example, it may be found that a given gene fragment does not occur in lymphocytes that match, in which case we may want to reduce the number of

lymphocytes made using that fragment. To do this, the library must include a number giving the relative weight of each gene fragment, and the selection process would be biased to use this. This is a potential future direction for the system, but has not yet been implemented. At the moment, all genes are equally weighted.

Finally, it should be noted that the repertoire of the immune system does not contain duplicates. These were allowed in earlier versions of this system, but it was concluded that doing so required the system to waste processing time applying the same antibody to a message multiple times.

3.2.1.1 Regular Expressions

In order to simulate the inexect matching of the biological immune system, the spam immune system uses antibodies composed of regular expressions. They can be complex or very simple. The simplest regular expression (regex) is a string of characters. A regex consisting of a word matches any string that contains that word.

Regex	Sample Matching Strings
Zzuf	Zzuf
	(Zzuf)
	What is Zzuf?

There are some characters, called ***meta-characters*** which are reserved for special use in a regular expression. These characters are as follows:

```
{}[]()^$.|*+?\
```

Any of these can be matched by "escaping" the characters with a backslash:

Regex	Sample Matching Strings
\+	+
	2 + 2 = 4

There are two of these characters which are mentioned in the previous section. The "." means "match any single character."

Regex	Sample Matching Strings
d.g	dog
	big dig
	d g

The "*" means "match zero or more of the previous item." An item may be a single character, a meta-character, or a grouping created using the meta-characters. ("One or more" is designated by "+")

Regex	Sample Matching Strings	Regex	Sample Matching Strings
ta*r	tar (one a)	d.*g	dog
	taardvark (two a's)		drought
	tricycle (no a's)		dg

As explained in the previous section, when the two characters are combined to create ".*" they can match 0 or more of any characters. By itself, ".*" can match any possible string, including the one containing no characters.

While a regex containing only normal characters can match any message which has that sequence of characters in it in exactly the sequence written, each regex containing meta-characters can match a number of different strings with variations within the pattern described. The regex "a" matches anything with an a in it, but the regex "." matches any message with any character in it, which would be all strings other "", the string with no characters in it.

The other meta-characters allow for more complex groupings, such as "any digit" or "between two and four letters from the set {a, b, c}". For more information on how regular expressions work, see [Kva00a], [per00] [Kva00b], and [Fri02].

3.2.2 Training of Lymphocytes

Algorithm 3 describes the training phase of the algorithm. A training phase should be run before the immune system is used on actual email, since without this training phase the system will be unable to distinguish spam and non-spam. To do training, it is necessary to compile a set of pre-classified messages. In initial training, these should be human-classified or a human-verified classification – the more accurately classified these training messages are, the more accurate the trained immune system will be. However, it is also possible to use machine-classified messages if more data is needed. In this case, the accuracy of each classification may not be 100%, so the training algorithm can weight these trainings differently to reflect this lack of accuracy.

Algorithm 3 Training of lymphocytes

Require: *repertoire* $\Leftarrow$ the list of lymphocytes (cannot be an empty list)
Require: *message* $\Leftarrow$ a message which has been marked as spam or non-spam

```
if the message is user-determined spam then
  spam_increment ⇐ 1
else if the message is user-determined non-spam then
  spam_increment ⇐ 0
else
  spam_increment ⇐ a number between 0 and 1 indicating how likely the message
  is to be spam {Chosen by user}
end if

for each lymphocyte in the repertoire do
  if lymphocyte.antibody matches the message then
    lymphocyte.msg_matched ⇐ lymphocyte.msg_matched + 1
    lymphocyte.spam_matched ⇐ lymphocyte.spam_matched + spam_increment
  end if
end for
```

In the initial training phase, all the training messages have already been identified as spam or non-spam. For each message that matches a given antibody, the associated lymphocyte's *msg_matched* score is incremented by one. If the message is known spam, then the lymphocyte's *spam_matched* score is also incremented by one.

Since it is possible to train using messages which were not classified by the user, we may want to indicate to the system that these messages are not necessarily classified correctly. In that case, the *spam_increment* used indicates the confidence the user has in the classification: If the user has a system which has a 5% failure rate, a message classified as spam by that system could be trained using an increment of 0.95, and a message classified as non-spam by that system would be incremented by 0.05.

Re-training is very similar to regular training, but uses higher increment values to compensate for the original mis-training.

3.2.3 Application and weighting of lymphocytes

The two numbers *spam_matched* and *msg_matched* can be used to give a weighted percentage of the time an antibody detects spam. The field *msg_matched* gives an indication of how often this antibody has been used, which helps determine how important it should be in the final weighting. An antibody that matches with a rate of 100% over a sample of 2 messages is probably not as useful as one that matches with accuracy 80% over a sample of 1000 messages.

Earlier weighting schemes for the spam immune system used a slightly differently weighted version of the *spam_matched* value to create a final score for the message: The *spam_matched* value was incremented or decremented depending upon the classification of the message being trained, and the values were sometimes in excess of 1 or -1. Specifically, it used the same algorithm eliminating the *msg_matched* value and incrementing the *spam_matched* value by 5 for known spam and -5 for known non-spam during initial training, 1 and -1 during operation of the AIS, and 10 and -10 for false positives and negatives during retraining [OW03a]. The final score used, a ***straight sum*** is simply a sum of the *spam_matched* values from all matching lymphocytes. This can be seen in Equation 3.1.

$$Straight\ sum = \sum_{matching\ lymphocytes} spam\_matched \tag{3.1}$$

This is equivalent to the scoring system used in [OW03a]. The problem with this scheme was that one highly weighted lymphocyte could easily overpower the sum for long periods of time, even if the lymphocyte was no longer matching much spam. A variant which helps to solve this problem is a ***weighted average***, which is the sum of the *spam_matched* values from all matching lymphocytes divided by the sum of all the *msg_matched* values from all matching lymphocytes. This can been seen in Equation 3.2.

$$Weighted\ average = \frac{\sum_{matching\ lymphocytes} spam\_matched}{\sum_{matching\ lymphocytes} msg\_matched} \tag{3.2}$$

This is the weighting scheme described in [OW03b]. This weighted average allows lymphocytes that have matched more often to have more effect on the final score than those that only match occasionally. It is also worth noting that the weighted sum has bounded results (all results are between 0 and 1, inclusive). Because the results are bounded, a single entity cannot dominate or skew the results as easily it could when using the unbounded straight sum.

It is also possible to use a Bayesian weighting scheme, using these *msg_matched* and *spam_matched* values. This would be as shown in Equation 3.3. The Bayes score is also bounded. More about Bayesian scoring can be found in Section 2.1.3.11

$$Bayes\ score = \frac{\prod_{matching\ lymphocytes} \frac{spam\_matched}{msg\_matched}}{\prod_{matching\ lymphocytes} \frac{spam\_matched}{msg\_matched} + \prod_{matching\ lymphocytes} 1 - \frac{spam\_matched}{msg\_matched}} \tag{3.3}$$

The algorithm finally selected for use with this system can be seen in Algorithm 4. This algorithm requires an existing repertoire of antibodies and a message which

Algorithm 4 Application of antibodies with dynamically updated weights

Require: *repertoire* $\Leftarrow$ the list of antibodies (cannot be an empty list)
Require: *message* $\Leftarrow$ a message to be marked
Require: *threshold* $\Leftarrow$ a cutoff point valued between 0 and 1 inclusive; anything with a higher score than this is spam {chosen by user}

Require: *increment* $\Leftarrow$ increment used to update lymphocytes
Or...
Require: *confidence* $\Leftarrow$ a value between 0 and 1 inclusive, depending upon the user's confidence in the system. {chosen by user}

total_spam_matched $\Leftarrow$ 0 {initialize # of spams matched to 0}
total_msg_matched $\Leftarrow$ 0 {initialize # of messages matched to 0}
matching_lymphocytes $\Leftarrow \phi$ {Initialize empty list of matching lymphocytes}

for each *lymphocyte* in the *repertoire* **do**
  if *lymphocyte.antibody* matches *message* **then**
    total_spam_matched $\Leftarrow$ *total_spam_matched* + *lymphocyte.spam_matched*
    total_msg_matched $\Leftarrow$ *total_msg_matched* + *lymphocyte.msg_matched*
    lymphocyte.msg_matched $\Leftarrow$ *lymphocyte.msg_matched* + 1 {increment the # of messages matched by this antibody}
    add *lymphocyte* to *matching_lymphocytes*
  end if
end for

score $\Leftarrow \frac{total\_spam\_matched}{total\_msg\_matched}$ {Determine the score using a weighted sum}
if *score* < *threshold* **then**
  Message is spam
  for each *lymphocyte* in *matching_lymphocytes* **do**
    if *confidence* is set **then**
      increment $\Leftarrow$ *confidence* $*$ *score*
    else
      {*incrementhasbeensuppliedbytheuser*}
    end if
    lymphocyte.spam_matched $\Leftarrow$ *lymphocyte.spam_matched* + *increment*
  end for
else
  Message is not spam
end if

is to be assigned a score.

The *increment* can be chosen directly by the user, or a *confidence* value can be used and is used to update lymphocytes automatically. The *confidence* value represents the likelihood that a message is spam given that the system thinks it is spam. A neutral response would be 0.5 (meaning that the user doesn't think it's more likely either way), and a affirmative response would be 1 (the user thinks that anything tagged by the system is guaranteed spam). There is little use for a response of 0 (meaning that the user is sure that the message is non-spam if the system thinks it is spam) during regular learning, but it is allowed for re-training purposes.

For current tests, the confidence value is always set to 1, meaning that the user believes that the system has correctly classified the messages, but this value could be used for fine tuning in future work.

3.2.3.1 But what if none of the antibodies match?

If none of the antibodies match, the straight sum and weighted average scores would both be 0. This isn't necessarily bad: if the message is non-spam, a low score is simply an indicator that the message is non-spam, so a score of 0 is not a problem.

However, if the message is spam, this is a problem, since the system is not able to learn anything about this message. This is an indication that the repertoire of the spam immune system does not cover the area of interest. This may occur when the library used does not does not contain enough information, or that the information contained therein is not useful for covering the area of interest. It may also occur when not enough lymphocytes have been produced. Because of the stochastic nature of the system, this may also happen through random chance even if there are sufficient numbers of lymphocytes and a good library.

Currently, this case is not handled by the spam immune system – the system simply does not learn. However, there are potential ways in which this could be

handled:

Ideally, we would like to learn more from messages that have been mis-classified, so it is a problem if no antibodies match. In the biological immune system, it is generally possible to find an antibody which is sufficiently close and mutate from there to find an antibody which has a closer match, although this may take considerable time (and during this time, the host feels sick). This process, called hypermutation, happens regularly in the biological immune system. When a B-cell is activated by the system, it undergoes hypermutation. But because this process is CPU-intensive, it is not done in the basic spam immune system.

However, in order to allow the system to learn from unmatched messages, it would be valuable to have some sort of hypermutation process. This idea is somewhat linked to Danger Theory [SFT03], which suggests that T-cells must be activated through alarm signals from damaged tissues. In this case, a special process is being activated through an alarm signal from the user, indicating that an incorrect classification has been made. There are existing methods for mutation of regular expressions such as those used in the spam immune system, and it may be possible to incorporate these methods into future implementations of the system.

3.2.4 Culling of antibodies: Ageing and Death

To cope with the fact that both the self of legitimate messages and the non-self of spam change constantly, the spam immune system needs to be able to unlearn as well as learn things. Forgetting can be useful in a system with limited resources [Str93]. The memory of the spam immune system is a hybrid of the two memory models described for the human immune system. (See Section 2.2.2.4). In this case, longer lived "memory cell" lymphocytes exist in the system, but it is reinfection that determines the lifespan of these longer-lived lymphocytes.

Each lymphocyte stores the information about the weighted number of messages

and spam messages it matches. Periodically (perhaps once per month), the system looks at all the lymphocytes and culls those that haven't been used as much recently, although those that have matched many in the past still have an advantage. For the spam immune system, the system goes through this cycle once per month, since it for long periods of time, spam changes very little, but it does occasionally have periods of rapid change [Sul04]. If the user feels that the system is not adapting quickly enough and accuracy is falling below expectations, he or she could force a retraining and culling cycle more frequently in order to speed the process of learning and forgetting. User controlled culling cycles can help the system adapt in periods of rapid change in either spam, legitimate mail, or both, but they have not been tested with this implementation of the spam immune system, since the spam corpus used is not known to exhibit rapid change.

Lymphocytes also "age" during this culling process. The two values *spam_matched* and *msg_matched* are decreased by a percentage (so that the ratio between the two stays the same). Eventually, if the lymphocyte does not match new messages, the value of *msg_matched* will become small enough that the lymphocyte will be culled.

In this way, any lymphocyte which matches many messages can potentially become a "memory cell" and in order to stay as a memory cell, the system must experience "re-infection" of similar spam messages.

3.3 Layering revisited

As seen in Section 2.2.1, the biological immune system has many layers which helps it to achieve impressive accuracy. Similarly, many anti-spam products use multiple methods together in order to improve their classification results.

Figure 3.3 (an extension of Figure 2.2) shows how the spam immune system fits in to the process of handling email.

Algorithm 5 Culling of antibodies: ageing and death

Require: *matched_threshold* $\Leftarrow$ any lymphocyte with a *msg_matched* value below this threshold will be killed {chosen by user}

Require: *decrement* $\Leftarrow$ amount by which to decrement ageing antibodies {chosen by user}

for each *lymphocyte* in the *repertoire* (list of all lymphocytes) **do**
  $lymphocyte.spam\_matched \Leftarrow \frac{lymphocyte.spam\_matched}{lymphocyte.msg\_matched} * (lymphocyte.msg\_matched - decrement)$
  {the ratio between the two weights stays the same as it was before the ageing}
  $lymphocyte.msg\_matched \Leftarrow lymphocyte.msg\_matched - decrement$
  if $lymphocyte.msg\_matched < threshold$ **then**
    remove antibody from data store
  end if
end for

This system could also work with other layers in order to achieve higher accuracy. For example, one might choose to use Blacklisting (described in Section 2.1.3.7) as a layer that goes before the spam immune system. Thus, some spam messages would be blacklisted and removed from the queue before they reach the spam immune system. The system should be able to work with any solution at another level, with varying degrees of integration, and may also work with solutions at the same level. For example, weighted tokens of a Bayesian system could be used to generate new lymphocytes. The system could also use messages which have been classified by other systems as messages for training. Section 3.2.2 describes how this would be done.

3.4 Comparing the Spam Immune System to other artificial immune systems

This application of the artificial immune system has elements in common with several other applications. It can be considered as a two-class version of a text classifier, where the categories are spam and non-spam. Similar to anomoly detection systems, it looks for the "anomalous" spam among the legitimate mail. The spam immune system is

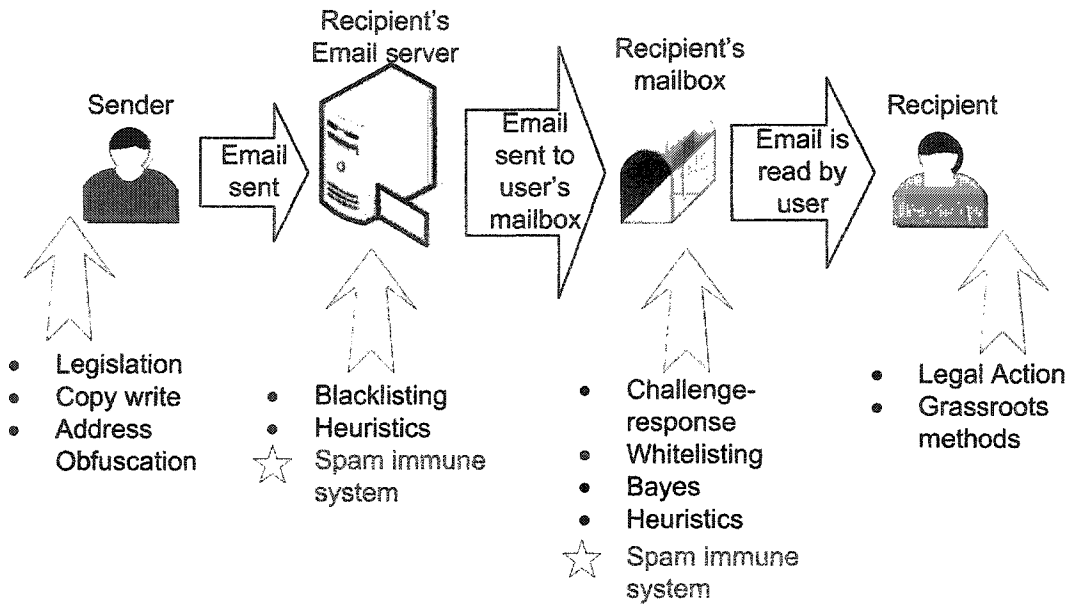

Figure 3.3: How the spam immune system fits in with other solutions

also a simplistic pattern recognition system, recognizing the patterns of spam and non-spam mail, as well as a learning system. (These general categories of immune systems are drawn from [dCZ00])

3.5 Summary

The spam immune system works as follows:

First a library of appropriate gene fragments must be collected. Then a collection of email that is already classified as spam or non-spam should be assembled for training. Lymphocytes are generated and initially trained using these two things.

Once the system is running normally, lymphocytes learn continually as new emails come in and are assigned a spam score by the system. Periodically, the lymphocytes are culled and aged, and new lymphocytes are created to replace those lost.

In this way, while the system is distinguishing self and non-self, it can also both learn and un-learn information, adapting to the changing nature of email.

The next chapter demonstrates the function of this spam immune system.

Chapter 4

Testing the Spam Immune System

4.1 Test Setup

This section describes the libraries and parameters used in the testing of the spam immune system.

4.1.1 Spam Corpuses

In order to test and train the classifier, it is necessary to have a corpus of email. Many people use their own personal email for training classifiers, since it's easily available, is usually already sorted by the individual, and it likely contains the characteristics that the user would like a classifier to learn. This is great when the aim is to show "it works for me," but to give more broad-ranging evidence that a system will work, there are other characteristics that should be kept in mind when choosing a corpus.

The ideal corpus of email...

- ...should be publicly available.

  This makes it easier for others to verify results by testing the filter against the same corpus. Many people use their own email to test and train classifiers, since the idea is to use the classifier on personal mail, however this makes it difficult

for others to repeat the experiment, as each individual's mail may have different characteristics.

- ...must contain spam and non-spam.

  It is easy to get spam, especially given the existence of sites such as Spam Archive [spa03b]. People are quite willing to donate spam, since it doesn't generally contain any personal information. It is harder to obtain non-spam from people, since it would require additional sorting and a willingness to share personal email with the world.

- ...must be sorted.

  This is necessary because it makes it easier for results to be duplicated when there is an existing consensus upon the classification of each message.

- ... should have been gathered by one email address or several email addresses that got both spam and non-spam.

  Some spam senders attempt to have targeted lists: It is possible to buy lists that claim to be targeted. Even if targeting is not very successful, not every list contains every email address, so we can assume that spam actually does vary from person to person, and it may vary in a meaningful way. Using mail from a single source or small number of sources means that the legitimate mail is likely to exhibit more common characteristics, and it's possible that the spam will also have more common characteristics. In theory, commonalities in one or both classifications will make the messages easier to classify.

  There are also two interesting points to keep in mind here: using email from one source will mean that the results are more likely to be similar for other single sources. However, using email from multiple sources shows that the classifier may not need these additional common characteristics to make a determination,

which means the classifier could probably be run without separate databases for each user.

- ... should be as recent as possible.

  Over time, techniques used by spammers have changed. For example, the rise of HTML-aware email clients has made HTML significantly more popular in spam, and the increased use of Bayesian filters has inspired more spam with random words and texts included. Since there's little benefit to being able to filter the spam of the past – we want to be able to filter current and future spam – it seems most logical to use a recent corpus for testing and training.

  Having the most recent corpus is not necessarily essential to ensure relevance. Sullivan found that spam does undergo rapid changes, but may remain stable for long periods of time [Sul04], so as long as the corpus doesn't encompass a drastic change, it may continue to be relevant for months or years.

- ... must be altered as little as possible.

  The full headers of email contain useful information about the sender of the message. It has been shown that using header information increases the accuracy of a Bayesian filter [O'B03]. Since the messages are to be found in a public corpus, it us understandable that headers may be edited somewhat for privacy, but ideally the message should be as unadulterated as possible. Older spam corpora such as LING-SPAM [SDHH98] removed a lot of potentially valuable header information.

4.1.1.1 SpamAssassin Public Corpus

The SpamAssassin public corpus of email [Spa03d] meets many of these criteria, although not all of them. It is indeed publicly available, and contains sorted spam and non-spam. Unfortunately, the messages are not from limited sources, so sorting

this corpus is more like sorting all mail for an Internet Services Provider than it is like sorting mail for an individual. This is quite a bit more difficult, and the filter is not expected to achieve the same accuracy as a result. The corpus contains mostly messages from 2002 is relatively recent, but not as up-to-date as possible. The messages are, however, relatively unaltered. Only some anonymizing was done and the headers are mostly available. This information is summarized in Table 4.1

Desirable quality	True?	Notes
Publicly available	yes	`http://spamassassin.org/publiccorpus/`
Contains spam and non-spam	yes	4150 non-spam and 1900 spam messages
Messages are sorted	yes	
Limited sources	no	Messages were obtained from a variety of mailing lists and donations, not necessarily all from one person
Recent	somewhat	The bulk of the corpus is from 2002.
Unaltered	somewhat	Messages were somewhat altered to preserve privacy and remove information added when they were donated

Table 4.1: Summary of Good Qualities of the SpamAssassin public corpus

It should be noted that SpamAssassin's public corpus is known to be a difficult collection of email to filter. The author of CRM114 mentions that he (as a human) can rarely achieve 90% accuracy sorting this corpus, but generally he achieves 99.84% accuracy sorting "real life" email [Yer04]. As such, very high accuracy is not expected when training and testing on this corpus.

4.1.1.2 Preparing the Corpus

The corpus was divided up by the information found in the `Date:` email header. This information is not necessarily accurate, since it relied on the configuration of the clock of the sender, which may not always be correct, but since no more accurate date information was available this seemed an appropriate compromise.

Messages whose date field were clearly inaccurate (such as messages where the year was listed as 2028) were discarded, and since all of the ham was sent during 2002, only the spam for that year was used. The breakdowns are shown in Table 4.2 and Figure 4.1. The SpamAssassin corpus is divided into five parts: easy_ham, easy_ham_2 and hard_ham contain the non-spam messages, and spam and spam_2 contain the spam messages. The parts marked _2 indicate more recent additions to the collection, and the easy and hard markings indicate which messages have features which make them seem more like spam, such as phrases which are common to spam or unusual HTML markup [Spa03d].

	easy_ham	easy_ham_2	hard_ham	Total Non-Spam	spam	spam_2	Total Spam
Total[a]	2500	1400	250	4150	500	1400	1900
Jan	1	0	1	2	0	0	0
Feb	43	0	0	43	1	11	12
Mar	0	0	0	0	0	18	18
Apr	0	0	0	0	0	18	18
May	0	0	2	2	0	320	320
Jun	0	2	5	7	1	140	141
Jul	0	558	158	716	5	498	503
Aug	455	833	34	1322	183	150	333
Sep	1220	0	25	1245	270	0	270
Oct	712	0	9	721	5	1	6
Nov	16	6	15	37	1	8	9
Dec	52	1	1	54	10	15	25

[a]Before date parsing. After parsing, some messages were discarded.

Table 4.2: Breakdown of the SpamAssassin public corpus by month

It should be noted that the SpamAssassin public corpus, as well as being difficult to classify, does not necessarily reflect normal ratios of spam to non-spam. Looking at Figure 4.1 we can see that the corpus contains few non-spam messages from January to June, then suddenly jumps to receiving over 600 messages/month for the next few months before returning to smaller numbers. While it is possible that real mail would

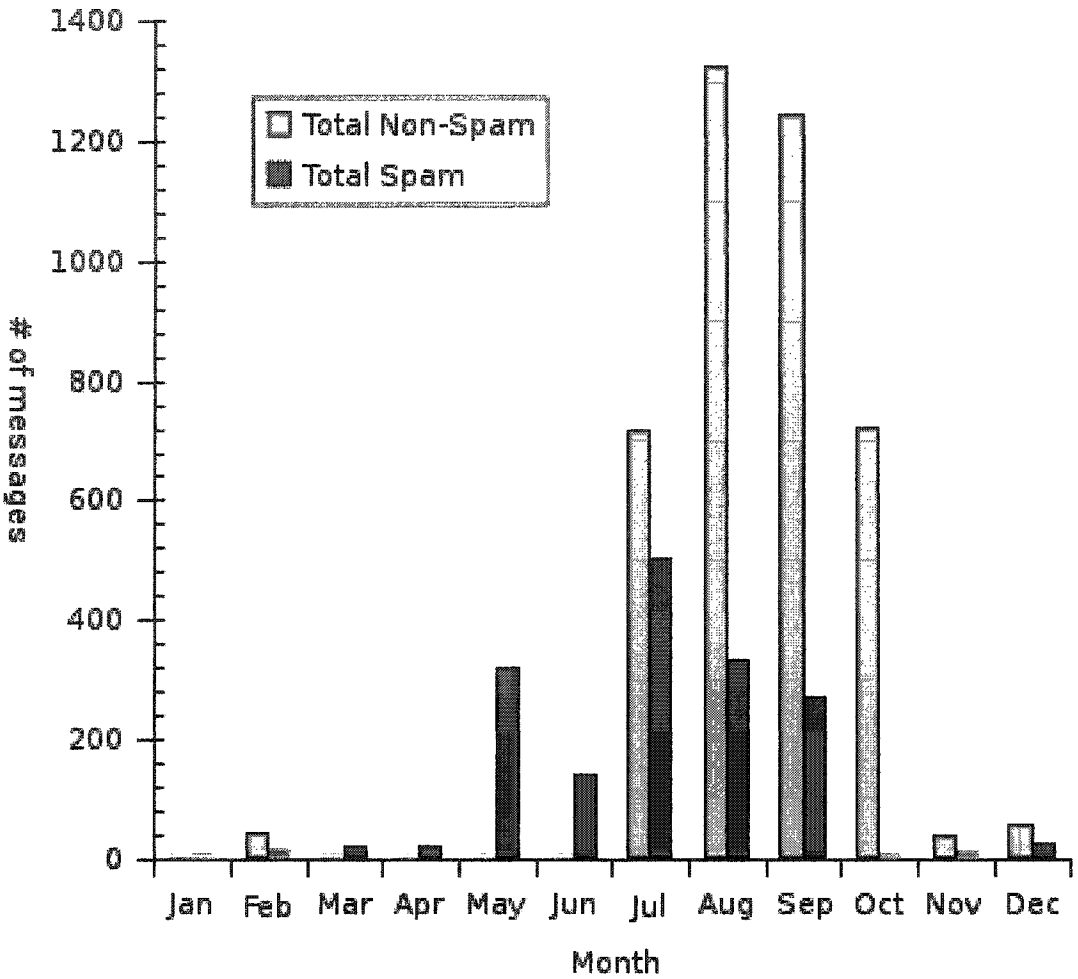

Figure 4.1: Messages by month from the SpamAssassin public corpus

exhibit this behaviour, it is probably not typical behaviour for an individual mailbox or group of mailboxes. This pattern is more likely due to the way in which the mails were collected by the SpamAssassin team: Presumably, most of the collection was done during the months of June to October.

4.1.2 Running the Spam Immune System

4.1.2.1 Initial Training

The Training phase

1. A number of lymphocytes were generated from the chosen library. The libraries

used are described in Table 4.3. (See Algorithm 2 in Section 3.2.1.)

2. These lymphocytes were trained on the messages from January to July, inclusive. This is 770 non-spam and 1012 spam messages. (See Algorithm 3 in Section 3.2.2.)

Since a spam filter must be able to correctly classify future messages based upon current and past messages, the training set entirely precedes the testing set based upon their dates. The messages from January and July were chosen to be the training set because it was only in July that there were enough non-spam messages in the corpus for sufficient training. Before July, there were fewer than 100 messages in the non-spam part of the corpus.

Several sets of lymphocytes were trained with varying libraries as described in Table 4.3. The English library is described in Section 3.1.3.2, the Bayesian library is described in Section 3.1.3.3 and the heuristic library is described in Section 3.1.3.4.

Gene types	Number of genes
English words	96274
Bayesian tokens	105248
heuristics	201

Table 4.3: Libraries used by the spam immune system

Sets of lymphocytes were created, trained and saved so that the same initial set could be used with varying runtime parameters, as described in Section 4.1.2.2.

4.1.2.2 Testing and ongoing learning

The messages from August to December inclusive were used as the testing set. This range includes 3379 non-spam and 643 spam.

1. For each message, the lymphocytes were applied to the message and the weights were updated appropriately. (See Algorithm 4 in Section 3.2.3.)

- Weighting can be dynamic, where the weight assigned to each lymphocyte is linked to the final weight of the message, or with a static value of 0.5.

2. At the end of the month, retraining can occur.

   - Retraining weights can vary. The base method weights a re-training as 2 times an initial training, but values of 5 and 10 were also tried.

3. Once retraining has happened, culling may occur. (See Algorithm 5 in Section 3.2.4.)

   - The parameters used for culling may vary. The first parameter indicates that lymphocytes with *msg_matched* values below this will be killed. The second parameter indicates how much the other lymphocytes will be aged, that is, how much their *msg_matched* values will be decremented. *spam_matched* is also decremented such that the ratio between the two remains the same. Values of 1 and 10 were tried for each of these parameters.

4. If culling has occurred, new lymphocytes were generated using the same library as the original lymphocytes. (See Algorithm 2 in Section 3.2.1.)

Runs were attempted with several different values used for training upon application, with and without retraining and with several values for retraining, and with and without culling. These runs are summarized in Table 4.4.

4.2 Results

4.2.1 Baseline Test

The baseline result used for comparison is a repertoire of 500 lymphocytes from the heuristic library, trained dynamically, retrained with a weight of 2 (meaning each

Number of Lymphocytes	Library	Training Increment	Retraining	Retraining Parameters	Culling	Culling parameters
Baseline test						
500	heuristic	dynamic	yes	2	yes	1 1
Population size						
200	heuristic	dynamic	yes	2	yes	1 1
300	heuristic	dynamic	yes	2	yes	1 1
400	heuristic	dynamic	yes	2	yes	1 1
500	heuristic	dynamic	yes	2	yes	1 1
600	heuristic	dynamic	yes	2	yes	1 1
700	heuristic	dynamic	yes	2	yes	1 1
800	heuristic	dynamic	yes	2	yes	1 1
900	heuristic	dynamic	yes	2	yes	1 1
1000	heuristic	dynamic	yes	2	yes	1 1
Alternate libraries						
500	heuristic	dynamic	yes	2	yes	1 1
500	Bayes	dynamic	yes	2	yes	1 1
500	English	dynamic	yes	2	yes	1 1
Training during regular operation						
500	heuristic	dynamic	yes	2	yes	1 1
500	heuristic	0.5	yes	2	yes	1 1
Lifecycle Tests						
500	heuristic	dynamic	yes	2	yes	1 1
500	heuristic	dynamic	no	2	no	1 1
500	heuristic	dynamic	yes	2	no	1 1
500	heuristic	dynamic	no	2	yes	1 1
Retraining parameters						
500	heuristic	dynamic	yes	2	yes	1 1
500	heuristic	dynamic	yes	5	yes	1 1
500	heuristic	dynamic	yes	10	yes	1 1
Culling parameters						
500	heuristic	dynamic	yes	2	yes	1 1
500	heuristic	dynamic	yes	2	yes	1 10
500	heuristic	dynamic	yes	2	yes	10 1
500	heuristic	dynamic	yes	2	yes	10 10

Table 4.4: Spam immune system parameters

retraining is equal to two trainings, once to reverse the original training and once as a new training), and culled if the *msg_matched* value falls below 1 and aged by 1 if the value is higher. Unless otherwise specified, these are the parameters used for each test.

This baseline was not chosen to be the best of the tests: as shown in Section 4.2.3, better classifications can be achieved by using larger populations. The benefit to using a non-optimal baseline is that there is more room to improve, so it is more evident if a given technique actually improves the results.

The average accuracy for the baseline test is 91.9% with 2.4% false positives. The standard deviation of this accuracy is 3.0%.

4.2.2 Comparing Scoring Systems

As described in Section 3.2.3, three different weighting schemes have been used with the spam immune system. The aim is to find a scoring system where the weights of spam messages and the weights of non-spam messages can be separated easily with a threshold. Ideally, everything above this threshold would be spam, everything below would be non-spam, but realistically there will be some outlying messages that get sorted into the wrong category.

Each of the three systems produces a very different pattern of scores when applied to the messages. Figures 4.2, 4.3 and 4.4 show these scores for one instance of the baseline test. Only the first month (August) is graphed to avoid showing any effects related to culling and retraining.

Figure 4.2 shows the pattern of the straight sum scoring system. In this graph, you can see that there is little clear division between the spam and the non-spam messages. Although it is not immediately apparent from this graph, the messages have a much wider range of scores than you would find in the bounded Bayes or weighted average scores. Note that there is a large spike of spam and smaller spike

of non-spam at the bottom end of the range – these represent messages for which few or no lymphocytes matched. The average best threshold is at score 3808, with an average error rate of 20.11%. However, this error rate is almost identical to the rate of spam in the portion of the corpus being tested, so effectively the straight sum is not distinguishing any messages.

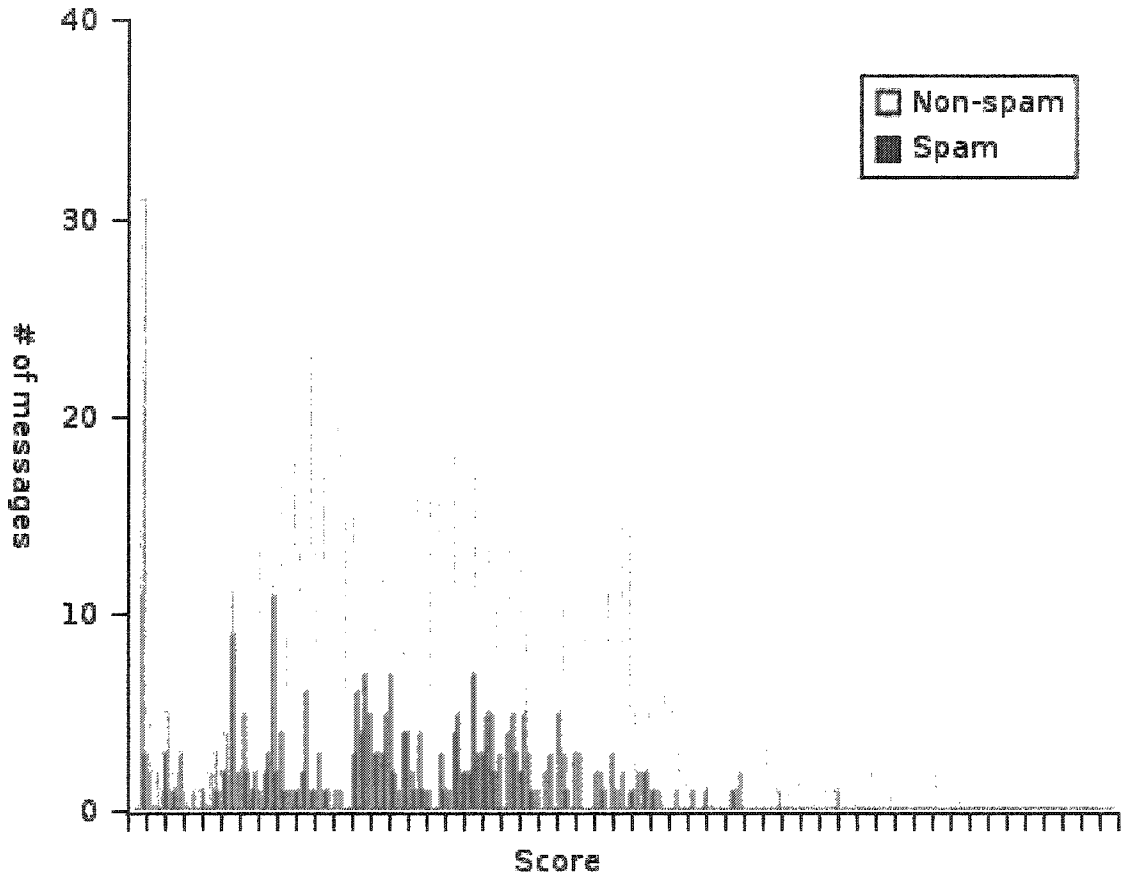

Figure 4.2: Straight Sum Score Distribution

Figure 4.3 shows the pattern of the Bayesian scoring system. This bowl-shaped distribution shows mostly spam at the top of the score range, and mostly non-spam at the bottom of the range. There is a spike in the middle of the distribution: this occurs because a weight of 0.5 is assigned to any message about which nothing is known. The average best threshold is at 0.62 and the average best error rate is 7.08%.

Figure 4.4 shows the pattern of the Weighted Average scoring system. The scores

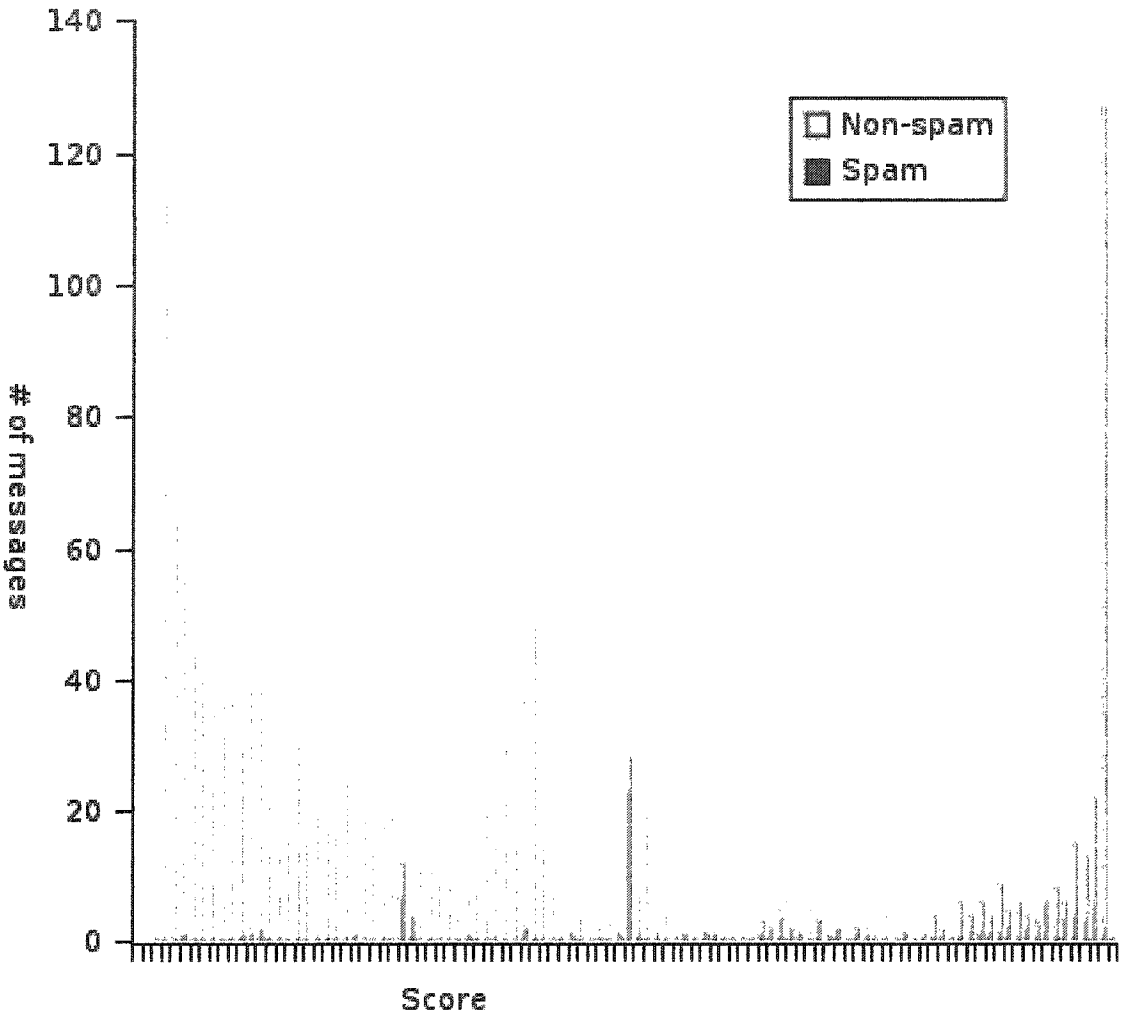

Figure 4.3: Bayes Score Distribution

of the spam messages and the non-spam messages are somewhat distinct, falling in two bell-curves that partially overlap at the edges. As with the Straight Sum, there is a spike of messages at 0 because this is the score assigned to messages about which nothing is known. The average best threshold was 0.55, with an average best error rate of 4.96%.

The spam immune system assigns scores to every message, but it is the threshold that determines which scores result in a spam classification and which results in a non-spam classification. As one would probably imagine, finding a good threshold for the straight sum scores was nearly impossible, as the spam and non-spam scores

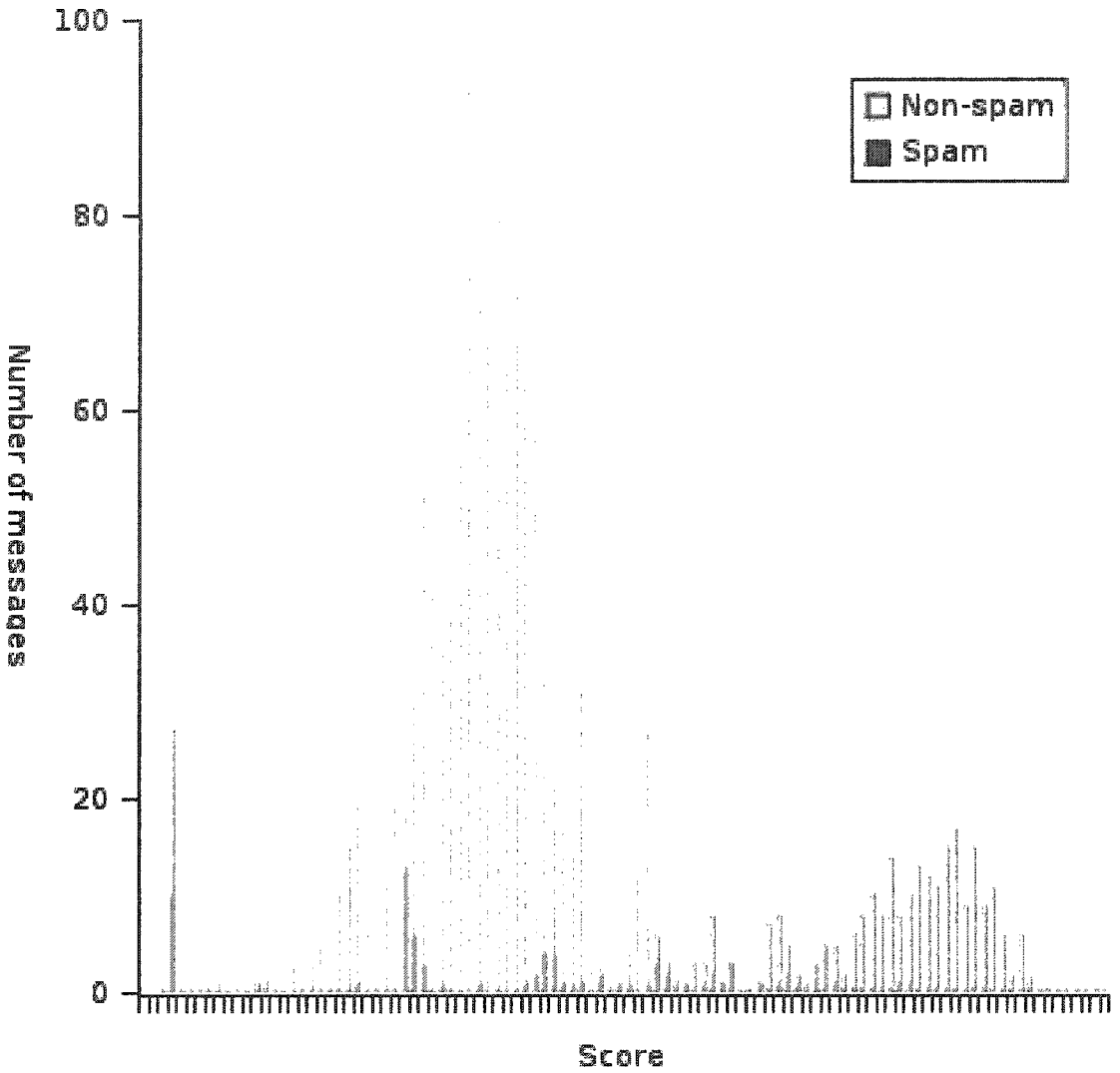

Figure 4.4: Weighted Average Score Distribution

are thoroughly intermingled. The Bayes scores and weighted average proved more effective in determining the two-class classification.

The error rate for each threshold was determined by counting the false positives and false negatives for each message and summing these values. If a lower number of false positives was desired, the error rate could be calculated using a higher weight for that type of error. However, since there is no known proven value for such a weight, the unweighted error was used to calculate the best threshold.

As shown in Table 4.5, the weighted average scores not only provided a lower

Scoring System	Threshold	Percent Error	Standard Deviation of Threshold
Straight Sum	3808	20.11	772.62
Bayes	0.62	7.08	0.12
Weighted Average	0.55	4.96	0.01

Table 4.5: Average threshold values for the three scoring systems

error rate on average, but the standard deviation of the threshold was smaller, which makes it easier to assume that future tests at the same threshold will yield similarly good results. As such, it is this scoring system that has been used for further tests.

4.2.3 Comparing Population size

Using the heuristic library, lymphocytes were generated in batches of 1000, 900, 800, 700, 600, 500, 400, 300, 200, and 100. Each one was tested against all the messages of the testing set, using the parameters for the baseline test other than the number of lymphocytes in the repertoire. The tables and figures all show the average value for all runs.

# of lymphocytes	False positives	False negatives	Total Error
200	8.29	6.88	15.17
300	3.10	6.69	9.79
400	3.94	6.12	10.07
500	2.44	5.63	8.07
600	1.18	5.42	6.60
700	1.09	5.28	6.37
800	1.72	5.80	7.52
900	1.06	5.16	6.22
1000	1.15	5.22	6.37

Table 4.6: Percent error versus population size

Figure 4.5 and Table 4.6 show the percent error in classification as a function of population size. With fewer lymphocytes, the error rate is higher, but the accuracy seems to plateau around 600 lymphocytes. This is less true of the false negative rate, which remains significantly more constant, probably because there are a number of spam messages which are being missed by the system. These messages may be largely

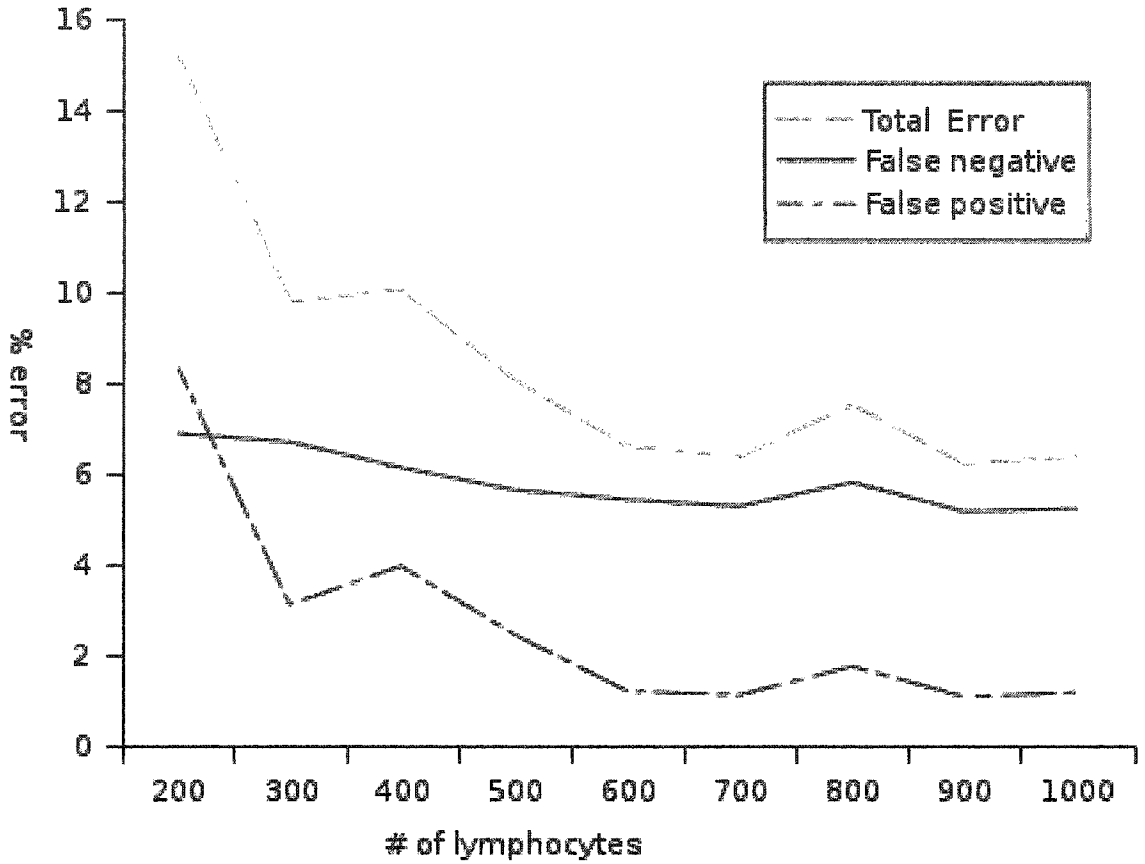

Figure 4.5: Percent error versus population size

invisible to the system as they are matched by few or no lymphocytes.

Recall also that the false positive rate needs to be especially low. False positives are the legitimate messages which have been tagged as spam by the classifier. It can usually be assumed that the user will notice false negatives, since they will appear in his or her regular mailbox, but false positives are more easily missed since they appear alongside spam, which the user may not read, or may even have automatically deleted.

Figure 4.6 shows the standard deviation of each of the previous data points. This shows that as the number of lymphocytes goes up, the amount of variability in the classification accuracy tends to go down, which is important in a spam classification system, since classification of this sort is generally one-shot, unlike optimization where one might try to use a system multiple times to get multiple good answers.

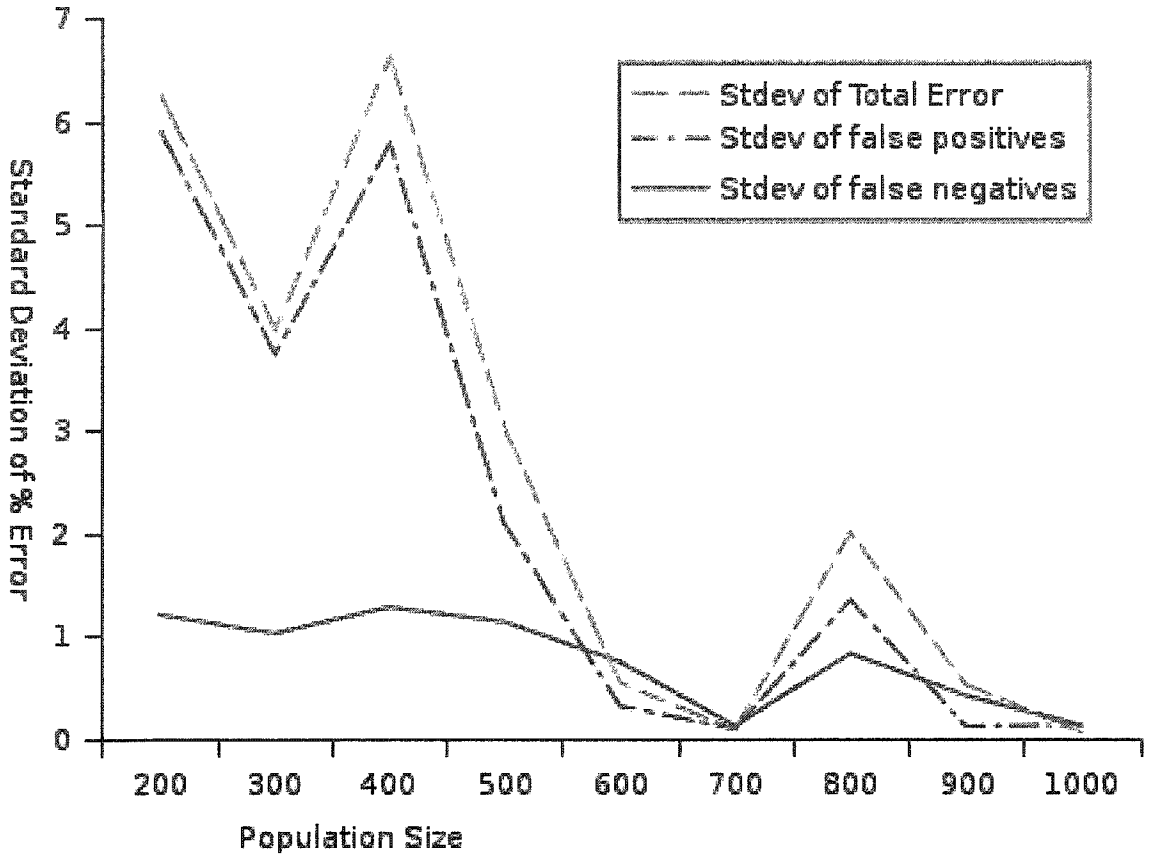

Figure 4.6: Standard deviation of percent error versus population size

Another useful piece of information about these results is the number of useful lymphocytes in each of these populations – the number of lymphocytes which have scores larger than zero. These numbers, grouped by month and population size, are shown in Figure 4.7. The graph was created by looking at the population of lymphocytes with any weight after the culling step of the lifecycle. Near the top of this graph, the lines for various population sizes converge, implying that we may have reached an optimal number of lymphocytes from this library for this corpus. Figure 4.8 shows this convergence more clearly, as it shows the number of useful lymphocytes graphed directly against the number of lymphocytes in the repertoire.

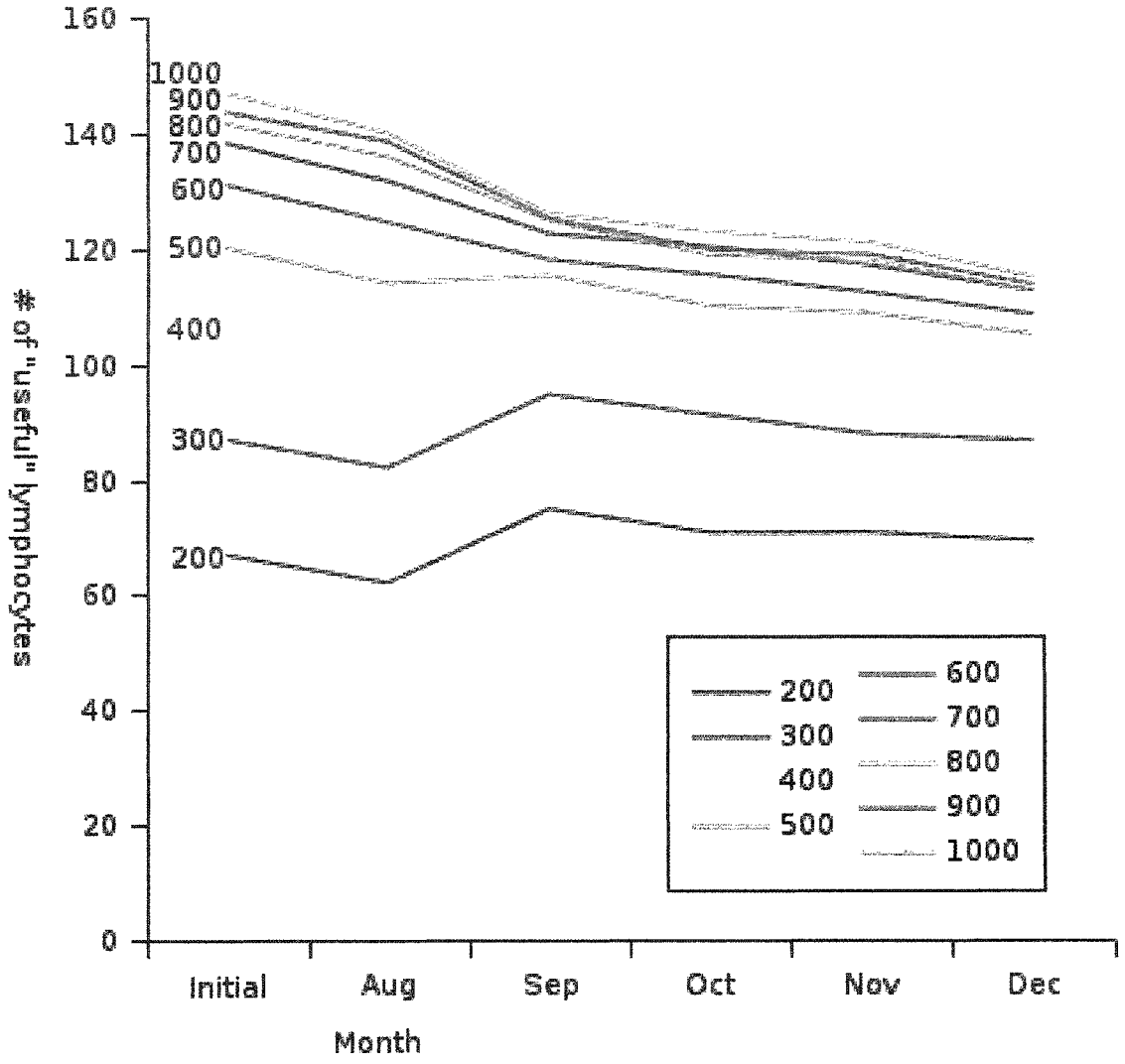

Figure 4.7: The number of useful lymphocytes in each population

4.2.4 Comparing alternate libraries

In order to compare libraries, sets of 500 lymphocytes were generated for each of the three libraries described in Section 4.1.2.1: the heuristic library, an English dictionary, and a Bayesian library. These populations were then trained on the standard set of messages, also as described in Section 4.1.2.1. From there, each set was run through the rest of the messages using the standard baseline parameters. The results are shown in Tables 4.7 and 4.8.

The Bayes and English libraries perform very similarly, which is expected since the

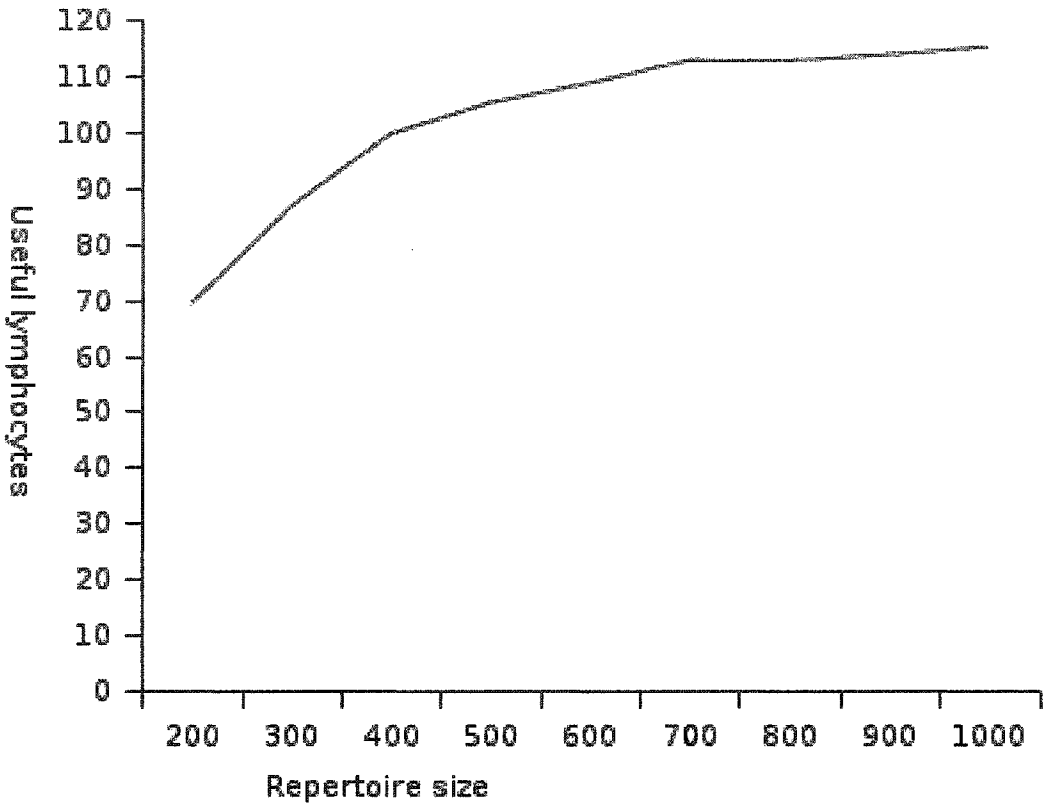

Figure 4.8: Number of useful lymphocytes versus repertoire size

	False Positives	False Negatives	Total Error
Bayes	18.00	11.20	29.19
English	18.08	11.36	29.44
Heuristic	2.44	5.63	8.07

Table 4.7: Error for the three libraries

Bayesian tokens are largely English words. The heuristic library, as well as having better performance, has a much smaller variance in score, making it overall more useful for a spam detection system.

The pattern of "useful" lymphocytes is shown in Figure 4.9. These are all averages by library from repertoires of 500 lymphocytes. The Bayes library starts with more useful lymphocytes than the English library, which is to be expected since the Bayes library was produced from the training set – all of the individual genes can be found in that training set. There are even more heuristic lymphocytes, probably because each of the genes used is *designed* to match multiple common phrases, whereas the

	False Positives	False Negatives	Total Standard Deviation
Bayes	11.13	2.69	8.94
English	9.84	3.16	8.18
Heuristic	2.10	1.13	3.02

Table 4.8: Standard deviation of error for three libraries

other libraries are more naive about the messages.

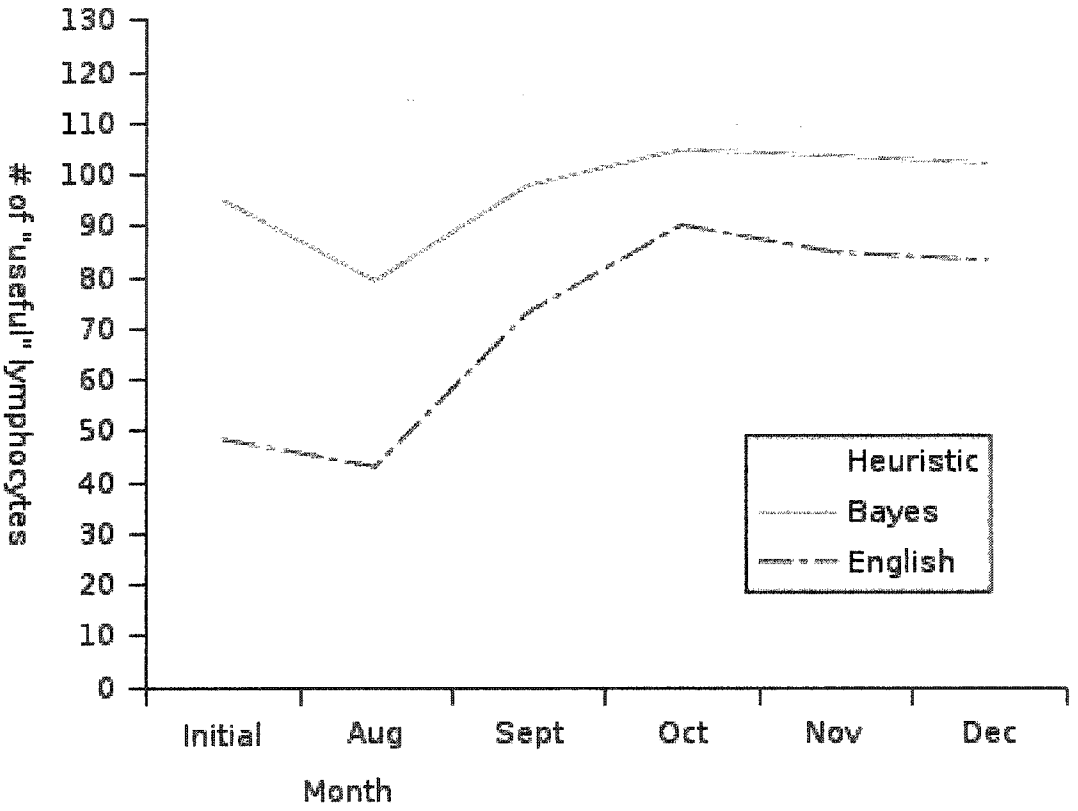

Figure 4.9: Average number of "useful" lymphocytes from the three libraries

4.2.5 Training during regular operation

Each time a lymphocyte matches a message, the system assigns new weights to that lymphocyte as described in Section 3.2.3. The increment used for the *spam_matched* value can be dynamically set as the score given by a weighted average, or it can be set at a fixed increment of 0.5, meaning that the system thinks there is an equal probability that the message is spam or non-spam. This value is chosen because this

assumption seems to work well for Bayesian classifiers. (See Section 2.1.3.11 for more information about how this assumption is used in Bayes classifiers.)

To test these training numbers, the baseline test was run along with tests that had the increment statically set at 0.5. The average results are given in Table 4.9, where you can see that the error rate for the dynamic weighting is slightly below that of the static weighting of 0.5. However, a t-test with an α value of 0.05 shows that this difference is not significant.

	False Positives	False Negatives	Total Error
dynamic	2.44	5.63	8.07
0.5	3.92	5.57	9.49

Table 4.9: Dynamic training increment versus static increment of 0.5

4.2.6 Lifecycle Tests

There are several parts of the complete lifecycle used by the spam immune system which are not, strictly speaking, necessary for the system to function: retraining and culling. These are intended to allow the system to adapt more readily to new messages.

The baseline test is compared against a test where no retraining and no culling occurs, one where retraining occurs but culling does not, and one where culling occurs but retraining does not.

According to Table 4.10, there is a marginal increase in accuracy when no retraining is done and a decrease when it is done, however this difference is well within the

Retrain	Cull	False Positives	False Negatives	Total Error
retrain	cull	2.44	5.63	8.07
no retrain	no cull	1.40	6.66	8.07
no retrain	cull	1.06	6.53	7.59
retrain	no cull	1.76	6.67	8.43

Table 4.10: Lifecycle tests

Retrain	Cull	False Positives	False Negatives	Total Error
retrain	cull	2.10	1.13	3.02
no retrain	no cull	1.18	3.37	2.93
no retrain	cull	0.74	3.46	3.11
retrain	no cull	1.11	3.38	3.08

Table 4.11: Standard deviation of lifecycle tests

Retraining Parameter	False Positives	False Negatives	Total Error
2	2.44	5.63	8.07
5	2.48	5.64	8.12
10	2.76	5.81	8.57

Table 4.12: Retraining parameter tests

standard deviations shown in Table 4.11, and as such, is probably just due to randomness as opposed to a flaw in the idea of retraining. It seems that culling provides a marginal improvement, but this is also within the standard deviations. As expected, t-tests with an α value of 0.05 against the baseline test show that the differences are not significant.

It is possible that the spam corpus used does not include spam which is changing suffciently. If the messages are similar throughout the corpus, then there is no significant advantage to adaptation. Section 4.3 provides further explanation of why so little effect was seen with retraining and culling.

4.2.7 Retraining parameters

There is only one parameter to retraining, and this parameter represents the weight of the retraining. Retraining is equivalent to repeated regular training, so this weight represents the number of repetitions. First, the initial training during regular operation is reversed, then the lymphocytes are trained as spam or non-spam as the user indicates.

At first glance, Table 4.12 may appear show that retraining with higher weights has actually decreased the accuracy. However, Table 4.13 shows that these differences

Retraining Parameter	False Positives	False Negatives	Total Error
2	2.10	1.13	3.02
5	2.04	1.14	2.98
10	2.30	0.94	3.07

Table 4.13: Standard deviation of retraining parameter tests

min *msg_matched*	ageing	False Positives	False Negatives	Total Error
1	1	2.44	5.63	8.07
10	10	2.91	5.70	8.61
10	1	2.28	5.62	7.90
1	10	2.82	5.70	7.97

Table 4.14: Culling parameters

are well within the standard deviation, and t-tests with an α value of 0.05 confirm that the differences in the averages are not significant. Section 4.3 describes reasons that no difference may be seen despite the different parameters used.

4.2.8 Culling parameters

There are two parameters that define the actions of the system during the culling phase. The first indicates a minimum *msg_matched* value, and any lymphocyte whose *msg_matched* score is below this will be "killed" and removed from the database of lymphocytes. The second parameter indicates how much each lymphocyte will be "aged" – how much the *msg_matched* value will be decremented for ageing. The *spam_matched* value is decremented in proportion so that the ratio between the two remains the same. Each parameter was tested at the values 1 and 10, making a total of 4 different combinations that were tested.

min *msg_matched*	ageing	False Positives	False Negatives	Total Error
1	1	2.10	1.13	3.02
10	10	2.97	1.18	3.91
10	1	1.78	1.12	2.69
1	10	2.79	1.18	4.27

Table 4.15: Standard deviation for culling parameter tests

Table 4.14 shows the results of tests with several different parameter values. There appears to be little difference in the results depending upon the parameters used, and Table 4.15 shows that the differences in errors between the different runs are all within the standard deviations of each value. The variation in parameter shows no statistically significant effect upon the end results, and this is confirmed with a t-test using an α value of 0.05. In Section 4.3, reasons for this lack of difference are described in further detail.

4.3 Discussion

Although the overall results are good (see Section 4.4), the test results for many of the lifecycle and parameter related tests were inconclusive. While it is possible that many of these parameters have little effect upon the system, it is also possible that this is a function of the corpus being used. The SpamAssassin public corpus, although it has many good qualities, only has four months that include high numbers of messages (See Figure 4.1). Since one of these must be used as part of the training set, this leaves only three months for testing. It is possible that lifecycle tests do not exhibit any interesting behaviour on such a small set, but they might have more effect on the results over a longer time period. Section 5.2 describes some ways in which a more suitable corpus could be assembled for use in future experimentation.

It is also possible that many of the lymphocytes which have been retrained are not being used again by the system. As such, the varied parameters and retraining would have very little effect, since these updated weights are not being used very frequently after they are updated. This hypothesis should be tested in future work.

Similarly, culling involves removing lymphocytes which are known not to be useful, so it will only be helpful if new useful lymphocytes are generated. Unfortunately, Figure 4.7 shows that with 500 lymphocytes in the repertoire, we are probably close

to attaining the maximum number of useful lymphocytes for this library with this spam corpus, so the chance of producing a new good lymphocyte is relatively slim. As such, culling has little effect. Over longer periods of time, with more messages, the chances of needing a new lymphocyte will increase, so presumably with a larger corpus culling would have a more noticeable effect.

Finally, recall spam tends to remain relatively similar for long periods of time punctuated by occasional periods of rapid change [Sul04]. Tests related to retraining and culling may have more effect when a more volatile time period is investigated. Future work should include experimentation with more suitable spam corpora, as well as more varied parameter values.

4.4 Overall Results

The spam immune system achieved accuracy of 93.6% with 1.1% false positives using the baseline set of parameters with 700 lymphocytes.

While this may seem unexciting in the face of advertising claims of 99+% accuracy, it is important to note that these advertising claims often represent best numbers. In the case of Brightmail, their accuracy rate for spam is based upon their false positive rate of 1/1 000 000, but this accuracy number disregards their false negative rate entirely. The independent review of spam products done by Network Computing, however, suggests that accuracy rates in the low to mid-ninety percent range are much more common [And04]. Their unweighted accuracy for Brightmail is 94%. The lowest unweighted score in their top ten products was 92.9%, implying that the spam immune system is achieving the sort of accuracy demonstrated by professional anti-spam products. In another study conducted by PC Magazine, even the best product achieved only 90% accuracy with 3% false positives [Met04].

This is even more promising when you consider the features available in the anti-

spam products, which often include use of blacklists and Bayesian filters as part of a complete solution. By comparison, my spam immune system uses only one method to achieve similar results. It is also worth noting that the spam corpus used is known for being difficult [Yer04], although this may also be true of the corpora used by the magazines to review products.

Chapter 5

Conclusions & Future Work

5.1 Conclusions

5.1.1 Overall

The spam immune system successfully adapts the artificial immune system model for use in spam detection. At 700 heuristic lymphocytes, the system averages 93.6% accuracy with 1.1% false positives. Thus, the spam immune system achieves accuracy comparable to that of commercial anti-spam solutions according to third-party reviewers of said products [And04] [Met04]. Accuracy numbers cited by vendors are often higher than these numbers, but these third party reviews are probably closer to the accuracy that would be seen by typical users.

This system is even more compelling in that it uses only a single approach (the immune system approach) to achieve this accuracy. As shown in [And04], many of the products they tested use multiple approaches, such as blacklisting combined with URL analysis. Presumably, if these approaches were added into a complete system including the spam immune system, it would be possible to achieve even higher accuracy. This demonstrates that not only is it possible to apply the artificial immune system model to spam detection, but it is also a viable alternate anti-spam

solution.

One would expect that, like the Bayesian systems, the immune system will tend to perform best when used on one user's personal mail. This is based upon the assumption that one user's legitimate mail "self" will very less wildly than a self produced by multiple users' messages. However, even though the SpamAssassin public corpus is composed of mail from more than one source, reasonable classification rates were observed.

These results imply that the system may be more useful than expected at the server level, which means that it could be deployed without requiring separate repertoires and training data for each user whose mail is to be classified. As well, it uses a relatively small set of 700 lymphocytes for matching, similar to the number used by SpamAssassin [sa-04d], and significantly less than the 23 000 tokens used by Graham's Bayesian filter [spa04a]. Thus, the system could be a fairly lightweight solution for servers.

5.1.2 Scoring

The scoring system which produced these results was the weighted average, originally proposed in [OW03b]. The straight sum scoring system failed to achieve significant accuracy. The Bayes system achieved similar results, but the larger variance between runs made it less attractive for a system which users would want to be relatively stable.

5.1.3 Libraries

Three libraries were tested, but it was the heuristic library, originally proposed in [OW03a] which emerged as the most accurate for classification. The Bayesian token and English word libraries performed significantly less accurately on average, and although the higher variance between runs implies that they could occasionally do as

well as the heuristic library, most users will not be content with a system that "might" work – they want something which will work consistently for them on a given run.

5.1.4 Lifecycle

The tests related to the lifecycle were inconclusive as to whether the proposed system of ageing, culling and retraining outperforms the system without these evolutionary aids. As such, it is hard to see if the system will adapt over time or if it will work successfully for long periods without adaptation. Another problem with this corpus is that there are very few spam messages in the last months: only 40 messages from October to December, compared to over 600 in August and September. This gives the lymphocytes little time to adapt to any new types of spam found in those months, and even if they adapt quickly, these months have little effect on the final averages. Since one of the causes of this inconclusive result may be the corpus used, future work will include tests against more suitable corpora.

5.2 Future Work

Many future ideas were described in Chapter 3 as design decisions were discussed. These included the following:

- Extensions to deal with images and attachments other than HTML or plain text. (See Section 3.1.1.)

  Currently, the system deals only with content as it has been translated to plain text or HTML. Since some spam is sent where most of the message is in an image, it would be worth looking at ways in which images and other attachments could be examined by the system. These could include algorithms which extract text from the attachment, or more complex analysis of the information contained within the attachment.

- Adaptive gene libraries. (See Section 3.1.3.5.)

  Currently, the system uses a library that is prepared in advance and does not change. However, as the system sees more spam, it could be gathering information that could be used to create new gene fragments. This could be done, for example, by looking at the Bayesian tokens found in messages. By adding this ability to adapt the gene library, it would be possible to make the system adapt to messages which are not matched by any current gene fragment. Removing this limitation from the system by allowing adaptive gene libraries could potentially have a large effect upon the final accuracy of the system.

- Weighted gene libraries (pre-weighted or adaptive weights). (See Section 3.2.1.)

  When antibodies are generated, the entire library of gene fragments has an equal chance of being used, but it would be possible to weight the fragments so that those more likely to produce useful lymphocytes could be used more frequently. The weights could be determined in advance from previous runs, or they could be done adaptively, using the *spam_matched* and *msg_matched* scores of lymphocytes whose antibodies include those fragments.

- Parameters set using a genetic algorithm or other automated methods. (See Section 3.2.)

  The parameters are currently user-selected based on the user's knowledge of their email – be it prior knowledge or current knowledge about misclassified messages. These parameter choices were not shown to have a huge effect upon the system, but it would be interesting to see if a genetic algorithm or other automated method could be used to produce a set of parameters which has a larger effect upon the final results.

- Varying confidence values. (See Section 3.2.3.)

The system currently assumes complete confidence in its scoring of the message and uses this value, unaltered, as the base for training all matching lymphocytes. A percentage confidence value could be used to lessen the effect of the system on itself, which might be valuable in times when the system does not seem to be performing well.

- Mutations of lymphocytes. (See Section 3.2.3.1.)

  One of the limitations of this system is that some messages remain completely unmatched by any lymphocyte in the system. The human immune system uses a process called hypermutation to counter this: an antibody which has a weak match is mutated until it produces a strong match. Although the system does not currently have a perception of a weak and strong match, making this difficult although not impossible to implement. It would be worth finding ways to accomplish this or something that serves the same purpose, given that this inability to learn from unmatched messages is a limitation of the current system.

In addition to these ideas, there are ideas stemming from the experiments that would be worth exploring.

It would be beneficial to try this system with other spam corpora, given the shortcomings of the one used. This may necessitate creating a new publicly available corpus. One easy method for doing so would be to gain the permission of a publicly available mailing list and collect the spam sent to that list. For many lists, the spam is already sorted out by hand and would only need to be stored together with the legitimate messages to produce a reasonable one-source corpus. Although this will likely not show all the characteristics of an individual user's mailbox (for example, mailing list topics tend to be more limited than the discussion topics of one user), it should be reasonably close. In addition, sorting solutions for mailing lists could save list administrators some time (many do the sorting manually at this time), and

integration of anti-spam products with mailing list management tools is an interesting problem in its own right.

Given a better corpus, the system should be tested over a longer period than the one-year one available in the SpamAssassin public corpus. In this way, it should be evident what sort of difference the parameters and lifecycle steps make upon classification over a longer period.

As well as a longer period, it would be good to test against a more recent corpus, as spam characteristics have probably changed since 2002. It would also be desirable to have a corpus which contains a higher ratio of spam to non-spam, since the SpamAssassin corpus, with approximately 15% spam, no longer reflects the numbers reported for the global ratio, where spam now outnumbers legitimate mail.

Once other corpora are available for testing, the parameters and lifecycle tests should be repeated to see if they have more effect. It would be good to test these things against messages that exhibit periods of volatility as described in [Sul04], as well as messages that are known to exhibit fairly stable characteristics.

The spam immune system has potential for use as part of a more complete anti-spam product. It would be interesting to look at how best to incorporate it with other techniques to produce a complete product. For example, setting so that information from a blacklist can be incorporated into the final score may yield a higher classification accuracy.

It would also be good to compare the spam immune system with other anti-spam offerings, both for actual classification accuracy and for memory, processor and other resource usage. It seems likely that it will be a more lightweight solution, although it may require some work to optimize it. For example, the database currently used could be replaced with one whose performance is higher.

With work as described above, it should be possible to turn the spam immune system into a viable alternative anti-spam solution.

Glossary

antibody: The antibody is the actual detector used by the immune system. Each lymphocyte's surface is covered by many copies of the same antibody. (Introduced on page 43.)

antigens: The surface proteins of a pathogen. These are what the lymphocytes detect. (Introduced on page 43.)

autoimmune reaction: A reaction wherein the body makes a mis-classification and starts to attack self. (Introduced on page 44.)

B-cells: B-cells are lymphocytes named for the "Bursa of Fabricius (a lymphoid organ found in birds), which is where they mature. Humans do not have this organ, and the B-cells originate and mature largely in the bone marrow of humans. (Introduced on page 44.)

Bayes score: A weighting scheme based upon the naive Bayes rule. Using the weights of the spam immune system, this is

$$\frac{\prod_{matching\ lymphocytes} \frac{spam\_matched}{msg\_matched}}{\prod_{matching\ lymphocytes} \frac{spam\_matched}{msg\_matched} + \prod_{matching\ lymphocytes} 1 - \frac{spam\_matched}{msg\_matched}}$$

(Introduced on page 74.)

body of email: In an email context, the body is the actual message text. (Introduced on page 15.)

digital antibody: A regular expression used as a detector for patterns in an email message. (Introduced on page 54.)

digital lymphocyte: The set including the digital lymphocyte as well as weighting information for that lymphocyte. (Introduced on page 54.)

false negative: In spam classification, a spam message that has been mistakenly classified as non-spam. (Introduced on page 17.)

false positive: In spam classification, a non-spam message that has been mistakenly classified as spam (Introduced on page 17.)

ham: In the context of email classification, ham is a slang term for non-spam. (Introduced on page 6.)

headers of email: In an email context, the headers are a set of meta-information found at the top of the email. They take the form of `<Name>: <value>` and while some of them are standard, more optional headers can be set. They include information about when a message was sent, to whom the message is being sent, from whom it was sent, what the subject of the message is, along with optional information such as the name of the program used to send the mail. (Introduced on page 15.)

lymphocytes: Specialized white blood cells which act as detectors for the immune system. B-cells and T-cells are lymphocytes. (Introduced on page 43.)

memory cells: One theory of immunological memory suggests that these long-lived lymphocytes are created when an infection has been found. They stay in the body forever, always available to react to re-infection by the same pathogen. (Introduced on page 46.)

meta-characters: In a regular expression, meta-characters are characters which have special uses for creating more complex expressions. (Introduced on page 70.)

non-self: Anything which is not self. Pathogens are non-self. (Introduced on page 41.)

pathogens: Dangerous non-self agents such as bacteria, viruses, and other foreign life forms and substances. (Introduced on page 41.)

repertoire: The population of different antibodies (and thus, lymphocytes) available in the body at any given time. (Introduced on page 43.)

self: Given an organism which is protected by an immune system, this is all the parts of said organism, including the immune system itself. (Introduced on page 41.)

spam: In email filtering, spam is what the recipient considers to be junk mail and does not wish to receive. This definition is not strict enough to be used as a legal definition, but is clear and flexible enough for email classification for an individual. (Introduced on page 9.)

straight sum: The sum of *spam_matched* values from all matching lymphocytes. That is, $\sum_{matching\ lymphocytes} spam\_matched$. (Introduced on page 73.)

T-cells: T-cells are lymphocytes named for the thymus, which is where they mature (although they originate in the bone marrow, like the B-cells). The thymus contains many different types of self-proteins, and as the T-cells mature, those which match self are killed off or are not selected to reproduce. (Introduced on page 44.)

unsolicited bulk email (UBE): Unsolicited Bulk Email or UBE consists of messages which have been sent to many parties without their request or consent.

(Introduced on page 7.)

unsolicited commercial email (UCE): Unsolicited Commercial Email or UCE consists of advertisements which have been sent out without request or consent of the recipients. (Introduced on page 7.)

useful lymphocyte: A useful lymphocyte, in a spam immune system, is a lymphocyte which has matched at least one message thus has a *msg_matched* value that is larger than zero. Any lymphocyte which has a *msg_matched* value of zero does not contribute to any message scores. (Introduced on page 58.)

virus signature: A unique string of bits, or the binary pattern, of a virus. The virus signature is like a fingerprint in that it can be used to detect and identify specific viruses. Anti-virus software uses the virus signature to scan for the presence of malicious code. [vir04] (Introduced on page 56.)

weighted average: The sum of the *spam_matched* values from all matching lymphocytes divided by the sum of all the *msg_matched* values from all matching lymphocytes. That is, $\frac{\sum_{matching\ lymphocytes} spam\_matched}{\sum_{matching\ lymphocytes} msg\_matched}$ (Introduced on page 74.)

Bibliography

[And04] Ron Anderson. Filters take a bite out of spam. *Network Computing*, May 13 2004.

[Art04] Artists Against 419. Welcome to the 5th international 419 flash mob on monday, may 3rd 2004 ! *Artists Against 419 Website*, May 4 2004.

[Asa04a] Amit Asaravala. ISP files first can-spam lawsuit. *Wired News*, March 6 2004.

[Asa04b] Amit Asaravala. With this law, you can spam. *Wired News*, January 23 2004.

[Atk03] Steve Atkins. Size and cost of the problem. In *Proceedings of the Fifty-sixth Internet Engineering Task Force (IETF) Meeting*, San Francisco, CA, March 16-21 2003. SpamCon Foundation.

[Bra04] Marshall Brain. How your immune system works. *HowStuffWorks*, 2004.

[Bri03] Brightmail Inc. The spam problem and the brightmail filtering engine. *Brightmail website*, 2003.

[Bri04] Brightmail Inc. Spam statistics: Spam percentages and spam categories. *Brightmail website*, April 2004.

[BSL96] Eli Benjamini, Geoffrey Sunshine, and Sidney Leskowitz. *Immunology: A Short Course*. Wiley-Liss, Inc., third edition, 1996.

[can03] Controlling the assault of non-solicited pornography and marketing (CAN-SPAM) act of 2003, November 25 2003. S. 877 as it was passed by the US Senate.

[Car03] Devin Carraway. Sugarplum – spam poison, September 25 2003. `http://www.devin.com/sugarplum/`.

[CAU04] CAUCE. How do you define "spam"? *Coalition Against Unsolicited Commercial Email Frequently Asked Questions*, Accessed July 2004.

[CL04] Gordon Cormack and Thomas Lynam. A study of supervised spam detection applied to eight months of personal email. *unpublished*, July 1 2004.

[CM01] Xavier Carreras and Lluís Márquez. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, BG, 2001.

[Dan04] Hadmut Danisch. The RMX DNS RR and method for lightweight smtp sender authorization. Technical report, May 2004.

[Das98] Dipankar Dasgupta, editor. *Artificial Immune Systems and Their Applications*. Springer, 1998.

[dCZ00] Leandro Nunes de Castro and Fernando Jos Von Zuben. Artificial immune systems: Part ii - a survey of applications. Technical Report DCA-RT 02/00, Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, SP, Brazil, February 2000.

[Del04] Mark Delany. Domain-based email authentication using public-keys advertised in the dns (domainkeys). Technical report, Yahoo! Inc., August 2004.

[DNS02] DNSRBL. Domain name system real-time black list, 2002. `http://www.dnsrbl.com/index.html`.

[dns04] Antispam FAQ. *AntiSpam DNSBL*, Accessed May 10 2004.

[DWV99] Harris Drucker, Donghui Wu, and Vladimir N. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, September 1999.

[Fal03] Deborah Fallows. Spam: How it is hurting email and degrading life on the internet. *Pew Internet & American Life Project*, October 22 2003.

[Fec04] Gordon Fecyk. Designated mailers protocol. Technical report, May 2004.

[Fei02] Patrick Feisthammel. Explanation of the web of trust of PGP. June 19 2002. `http://www.rubin.ch/pgp/weboftrust.en.html`.

[FHS97] Stephanie Forrest, Steven A. Hofmeyr, and Anil Somayaji. Computer immunology. *Communications of the ACM*, 40(10):88–96, 1997.

[Fon03] John Fontana. Tallying the true cost of spam. *Network World*, November 17 2003.

[Fri02] Jeffrey E. F. Friedl. *Mastering Regular Expressions.* Second edition, July 2002.

[GC03] Julie Greensmith and Steve Cayzer. An artificial immune approach to semantic document classification. In Jon Timmis, Peter Bentley, and Emma Hart, editors, *Artificial Immune Systems. Second International Conference, ICARIS 2003 Proceedings*, number 2787 in Lecture Notes In Computer Science. Springer, September 2003.

[Gra02] Paul Graham. A plan for spam. *Hackers & Painters: Big Ideas From the Computer Age*, August 2002.

[Gra03] Paul Graham. Better bayesian filtering. In *2003 Spam Conference*, January 2003.

[Gre04] Peter H. Gregory. Security predictions for 2004. *Computerworld*, January 5 2004.

[Hab03] Habeas. How sender warranted email™ works. 2003.

[Han03] Saul Hansell. The high, really high or incredibly high cost of spam. *The New York Times*, July 29 2003.

[Hor04a] Hormel Foods Corporation. SPAM: SPAM and the internet. *http://www.spam.com/*, Accessed January 17 2004.

[Hor04b] Hormel Foods Corporation. WWW.SPAM.COM: Legal & copyright info. *http://www.spam.com/*, Accessed January 17 2004.

[IF04] Ipsos-Reid and Forge Marketing. Spam volume doubles. *Canadian Inter@ctive Reid Report*, March 14 2004.

[Jes04] Anick Jesdanun. Paying for e-mail may be anti-spam tactic. *Associated Press*, March 04 2004.

[Kva00a] Mark Kvale. Perl regular expressions quick start. *Perl Programmers Reference Guide*, 2000.

[Kva00b] Mark Kvale. Perl regular expressions tutorial. *Perl Programmers Reference Guide*, 2000.

[Lou03] Greg Louis. Really bayesian bogofilter, August 2003.

[McW03] Brian McWilliams. Swollen orders show spam's allure. *Wired News*, August 6 2003.

[Met04] Cade Metz. Spam blockers. *PC Magazine*, Feburary 17 2004.

[MM03] Greg Mori and Jitendra Malik. Breaking a visual captcha. December 15 2003.

[Mye99] J. Myers. Smtp service extension for authentication. Technical Report 2554, Netscape Communications, March 1999.

[O'B03] Cormac O'Brien. Compiling a corpus of e-mail to evaluate spam filters. 2003. Unpublished.

[ORD04] ORDB. Open relay database - faq. *http://www.ordb.org/faq*, Accessed February 18 2004.

[OW03a] Terri Oda and Tony White. Developing an immunity to spam. In *Genetic and Evolutionary Computation Conference (GECCO 2003), Proceedings, Part I*, volume 2723 of *Lecture Notes in Computer Science*, Chicago, July 2003.

[OW03b] Terri Oda and Tony White. Increasing the accuracy of a spam-detecting artificial immune system. In *Congress on Evolutionary Computation (CEC 2003), Proceedings*, volume 1, pages 390–396, Canberra, Australia, December 2003.

[per00] Perl regular expressions. *Perl Programmers Reference Guide*, 2000.

[PL98] Patrick Pantel and Dekang Lin. Spamcop: A spam classification & organization program. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[Pla84] J.H.L. Playfair. *Immunology at a Glance*. Blackwell Scientific Publications, third edition, 1984.

[Pra04] Vipul Ved Prakash. Vipul's razor, 2004.

[Pro04] Prophis Research and Consulting. Perceptions of internet-based marketing: How annoying is that? *www.mythoughtsmatter.com research panel*, March 4 2004.

[Ras04] Wayne Rash. Virginia seeks to send spammer to slammer. *eWEEK: Enterprise news and reviews*, November 4 2004.

[Raz04] Uri Raz. How do spammers harvest email addresses? Accessed June 2004.

[RBM98] Ivan Roitt, Jonathan Brostoff, and David Male. *Immunology.* Mosby International Ltd., fifth edition, 1998.

[Rob02] Sara Robinson. Up to the challenge: Computer scientists crack a set of ai-based puzzles. *SIAM News*, 35(9), November 2002.

[Ros03a] Chip Rosenthal. Popular spam protection technique doesn't work. *It's Just this Little Chromium Switch Here*, November 26 2003.

[Ros03b] Chuq Von Rospach. Aol "spam blocking". *Teal Sunglasses*, February 2003.

[sa-04a] The auto-whitelist. *SpamAssassin Wiki*, March 12 2004.

[sa-04b] Bayes in spam assassin. *SpamAssassin Wiki*, April 28 2004.

[sa-04c] DNS blocklists. *SpamAssassin Wiki*, April 1 2004.

[sa-04d] Tests performed. *SpamAssassin website*, Accessed May 10 2004.

[SC01] Lee A. Segel and Irun R. Cohen, editors. *Design Principles for the Immune System and Other Distributed Autonomous Systems.* Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 2001.

[SDHH98] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Catego-*

rization: Papers from the 1998 Workshop, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[sen04a] Bill gates outlines technology vision to help stop spam. Technical report, Microsoft Corporation, Feburary 2004.

[sen04b] Sender id framework at a glance. Technical report, Microsoft Corporation, September 2004.

[SFT03] Andy Secker, Alex A. Freitas, and Jon Timmins. A danger theory inspired approach to web mining. In Jon Timmis, Peter Bentley, and Emma Hart, editors, *Artificial Immune Systems. Second International Conference, ICARIS 2003 Proceedings*, number 2787 in Lecture Notes In Computer Science, pages 157–167. Springer, September 2003.

[sho99a] Two spammers murdered in new jersey. *Slashdot*, October 30 1999.

[sho99b] Two spammers shot dead. *CNN*, October 28 1999.

[Smi04] Steve Smith. Can the spam: 2004 antispam editions still need some work. *Computer Power User*, 4(1):65–66, January 2004.

[Som03] Lauren Sompayrac. *How the Immune System Works, 2nd Edition.* Blackwell Publishing, 2003.

[spa00] Spam. *The American Heritage Dictionary of the English Langugae*, 2000.

[spa03a] Spam. *The Free On-line Dictionary of Computing*, September 21 2003.

[spa03b] Spam archive: Donate your spam to science, March 2003. `http://spamarchive.org/`.

[spa03c] The spamhaus project. 2003. `http://www.spamhaus.org/`.

[Spa03d] SpamAssassin. SpamAssassin public corpus, February 28 2003. The corpus is found at `http://spamassassin.org/publiccorpus/`. A readme file detailing the contents can be found at `http://spamassassin.org/publiccorpus/readme.html`.

[spa04a] Spambayes: Bayesian anti-spam classifier written in python, 2004. Accessed October 13, 2004. `http://spambayes.sourceforge.net/`.

[spa04b] Spambayes: Frequently asked questions, August 11 2004. Version 1.81.

[spa04c] Spamnet overview, 2004.

[spf04] An interview with the lead developer of spf. *CircleID*, June 29 2004.

[Sta99] William Stallings. *Cryptography And Network Security, Principles and Practice*, chapter 15.2, pages 501–514. Prentice Hall, second edition, 1999.

[Str93] Martin Strecker. Forgetting in intelligent systems. April 21 1993.

[Sul04] Terry Sullivan. The myth of spam volatility. *QAQD.com White Paper*, 2004. This is a summary of the paper presented at the 2004 MIT Spam Conference.

[TBH03] Jon Timmis, Peter Bentley, and Emma Hart, editors. *Artificial Immune Systems. Second International Conference, ICARIS 2003 Proceedings.* Number 2787 in Lecture Notes In Computer Science. Springer, September 2003.

[Tec03] The Center For Democracy & Technology. Why am i getting all this spam? March 2003.

[Tur03] Peter D. Turney. Coherent keyphrase extraction via web mining. In *The Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), (2003), Acapulco, Mexico*, pages 434–439, 2003.

[Ula04] Lance Ulanoff. Spam: A reality check. *PC Magazine*, February 18 2004.

[vABL04] Luis von Ahn, Manuel Blum, and John Langford. Telling humans and computers apart automatically. *COMMUNICATIONS OF THE ACM*, 47(2):57–60, February 2004.

[vir04] What is a virus signature? *Webopedia Computer Dictionary*, May 18 2004.

[Vix00] Paul Vixie. MAPS RBL rationale. *Mail Abuse Prevention System (MAPS) Website*, July 19 2000.

[WC02] Darla J. Wise and Gordon R. Carter. *Immunology: A Comprehensive Review.* Iowa State University Press, first edition, 2002.

[Wen02a] Mike Wendland. Internet spammer can't take what he dishes out. *Detroit Free Press*, December 6 2002.

[Wen02b] Mike Wendland. Spam king lives large off others' e-mail troubles. *Detroit Free Press*, November 22 2002.

[WSP+02] Steve R. White, Morton Swimmer, Edward J. Pring, William C. Arnold, David M. Chess, and John F. Morar. Anatomy of a commercial-grade immune system. Technical report, IBM Thomas J. Watson Research Center, Accessed 2002.

[Yah04] Yahoo! Inc. Domainkeys. *Yahoo! Anti-Spam Resource Centre*, 2004.

[Yer04] William S. Yerazunis. The spam-filtering accuracy plateau at 99.9% accuracy and how to get past it. In *2004 MIT Spam Conference*, January 18 2004.

[Zdz04] Jonathan A. Zdziarski. Advanced language classification using chained tokens. In *MIT Spam Conference 2004 Proceedings*, February 2004.