Immunity from spam: an analysis of an artificial immune system for junk email detection

Terri Oda and Tony White

Carleton University, Ottawa ON, Canada
terri@zone12.com, arpwhite@scs.carleton.ca

Abstract. Despite attempts to legislate them out of existence, spam messages (junk email) continue to fill electronic mailboxes around the world. With spam senders adapting to each technical solution put on the market, adaptive solutions are being incorporated into new products. This paper undertakes an extended examination of the spam-detecting artificial immune system proposed in [1, 2], focusing on comparison of scoring schemes, the effect of population size, and the libraries used to create the detectors.

1 Introduction

The first junk email was sent in 1978 [3]. Junk email messages were merely a curiousity in the early 1990's, but they soon became a nuisance, and then a serious problem to many people. Junk emails may account for 75-85% of email [4, 5]. Despite attempts at legislation such as the CAN-SPAM act in the US [6], the problem does not seem to have lessened significantly, and may even be getting worse [7].

Artificial immune systems have been used for a diverse set of things, including spam detection [1, 2] and email classification [8]. This paper focuses upon extending the work of [1, 2]. The initial papers on a spam-detecting immune system showed positive results, but did not look at how the system performed over a longer period of time, or the effects of different alternatives such as variant libraries or varying population sizes. This paper compares results from different setups. In addition, this paper gives an algorithmic treatment of the spam immune system used, making it more clear what other parameters might be altered and which parts of the algorithm could be changed.

This system differs from AISEC [8] in several important ways, although they are both immunologically-inspired email classification tools. Firstly, it is specifically geared towards spam detection rather than a more general model of text classification. As such, it does not take into account known heuristics for email classification, and thus has a broader application field. The representation for AISEC is based upon vectors of words found in the subject and sender header fields of email, whereas the representation for this system can match upon any part of a given message. In part because of this representation, AISEC has the ability to do clonal mutations, something which is not seen in this system. The

two systems, while performing similar functions, reach their classifications by very different means.

Section 2 gives a short overview of spam detection: what makes it an interesting problem, and how adaptive solutions can help. Section 3 describes the spam immune system as it was tested. The results of these tests are given in Section 4. Conclusions are discussed in Section 5, and some ideas for future work are outlined in Section 6.

2 Spam

While defining spam for the lawyers can be tricky, defining spam for the purpose of filtering is easy: Spam is what the recipient considers to be junk mail and does not wish to receive.

Spam is basically a two-class problem where the two classes are spam and non-spam (legitimate mail). Spam changes over time as new products become available or popular, but it also changes because the problem is co-evolutionary: spammers adapt to filters, and filters adapt to spam.

Although it does change, spam is not completely volatile: it tends to have many stable features and occasionally undergoes periods of rapid change [9]. This means that a semi-static solution will work for long periods, then break seemingly all at once, letting through a flood of messages. Obviously, this is not desirable. The hope with adaptive spam solutions is that they will be able to adapt to both slow and rapid changes.

Adaptive systems such as this one are also inherently diverse from one instance to the next. Although for the individual, diversity may not have immediate benefits (a given spam message might still go through an individual's filter), diversity in spam filters has a impact on the industry as a whole. If it is impossible for a spam sender to craft a message which will go through enough filters, then it will cost more to send messages than the spam senders can make in profit. This sort of economic disincentive may prove to be the only significant deterrent to spam, given the lack of success so far with legislation [7].

2.1 Spam technologies

There are two broad classes of solutions to spam: those which are technological in nature, such as the many anti-spam products available, and those which are more social solutions, such as the legislations surrounding unsolicited email. Two of the technological solutions have lent ideas to the spam immune system, so these are described briefly here:

- **SpamAssassin** SpamAssassin [10] is an excellent open source spam filter which uses a number of interesting heuristic techniques, including Bayesian style filtering, lookup in blacklists, and many others. Of particular interest to this paper are the text-based heuristics it uses, which are Perl regular expressions.

– **Bayesian-inspired spam filters** The idea of using Bayes rule to sort spam was introduced in 1998 [11, 12], but the idea became much more popular after a paper in 2002 [13] which boasted extremely high accuracies. Bayes rule is a result from probability theory that helps predict the classification of a given item based on features it has. ("Give me the probability that this message is spam, given that it contains the tokens 'Rolex' and 'replica'".)

3 The Spam Immune System

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.

3.1 Detectors: Lymphocytes and Antibodies

The central part of the spam immune system is its detectors, which are regular expressions made by randomly recombining information from a set of libraries, as described in Section 3.2. These regular expressions match patterns in the entire message.

The digital lymphocyte consists of an antibody and two associated weights detailing what has been matched by that particular lymphocyte. Both of these weights are initialized to zero.

– *spam_matched*: the cumulative weighted number of spams matched
– *msg_matched*: the cumulative weighted number of messages matched

3.2 Libraries

The gene library contains partial patterns used to build the full patterns used in lymphocytes. (Algorithm 2 describes how this is done.) In order to create antibodies which match spam, a few different libraries were tested:

Dictionary of English Words For the personal email of an English speaker, most messages are written in English. This is the case for the corpus used for testing and training. 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.

Bayesian-style Tokens The Bayesian tokenizer divides a mail up into separate components, usually individual words. The SpamBayes [14] tokenizer was used to parse a training set of emails into Bayes tokens. Their implementation is based upon the work of Paul Graham [13], but includes many additions not found in his work [14]. This library contains 105248 tokens.

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.

Figure 1 gives some example heuristics. The syntax used is that of Perl regular expressions. The first of these looks for a pattern where the words "reply", "remove" and "subject" appear fairly close together (eg: "send a reply with remove in the subject"). The second is a simple string which represents the colour red in hexadecimal (this string might appear in HTML-formatted mail). The third contains the code for setting the background colour of an HTML document. Finally, the last matches strings such as "college diplomas" or "university diplomas" because these are periodically offered through spam messages.

```
reply.{1,15}remove.{1,15}subject
ff0000
\<BODY.*bgcolor="#?[^f]
\b(?:college|university)\s+diplomas
```

Fig. 1. Some heuristics from the Heuristic gene fragment library

The heuristic library is much smaller than its counterparts, with only 201 fragments. The heuristics used are drawn from SpamAssassin [10], information about the training results of Bayes classifiers [13] [15], as well as directly from examination of spam.

3.3 Assigning scores to messages: Is it spam?

Given a set of weighted antibodies which have matched a given message, how do we make a determination as to whether that message is spam? First we combine all the individual antibody scores to assign a score to the message, and then we must set a threshold so that scores on one side of this threshold are spam, and those on the other are not.

In the first paper, scoring was done with a simple sum of the messages matched by each lymphocyte [1], as shown in Equation 1. Later work used a "weighted average" where this score was divided by the number of messages matched by all lymphocytes [2], as shown in Equation 2. Given the information stored by each lymphocyte, it is also possible to use a Bayes Score, as shown in Equation 3. In each of these, the sum or product is taken over all matching lymphocytes, so only the spam_matched and msg_matched values from those

lymphocytes are used in the score. The results of testing these equations can be found in Section 4.2.

$$Straight\ sum = \sum_{matching\ lymphocytes} spam\_matched \tag{1}$$

$$Weighted\ average = \frac{\sum_{matching\ lymphocytes} spam\_matched}{\sum_{matching\ lymphocytes} msg\_matched} \tag{2}$$

$$Bayes\ score = \frac{\prod_{matching\ lymphocytes} \frac{spam\_matched}{msg\_matched}}{\prod_{matching\ lymphocytes} \frac{spam\_matched}{msg\_matched} + \prod_{matching\ lymphocytes} 1 - \frac{spam\_matched}{msg\_matched}} \tag{3}$$

Ideally all spam would be on one side of the threshold and all non-spam on the other. Doing the threshold selection after initial training allows the user some control over the accuracy of the system. Some users may be willing to lose a few legitimate messages if it means they don't have to deal with all the spam, while others will prefer to sort through more spam rather than risk losing any legitimate mail. Although it has been suggested that a false positive should be weighted more heavily as an error than a false negative [13], there does not seem to be a consensus on an appropriate value for this weight. As a result, these tests have been done using a sum of the false positive and false negative scores to give a total error. The threshold was determined based on the score that gave a minimum total error over an average of all runs. For most tests, 20 runs were conducted. The results from this threshold determination are described in Section 4.2.

3.4 Lifecycle

The lifecycle of a digital lymphocyte starts when the lymphocyte is created and initialized (as described in Section 3.1). Once it has been created and initialized, it can be used to match messages. It is usually trained first on a set of pre-classified messages, then allowed to work with real, unclassified messages. The lymphocytes are culled periodically (on an interval set by the user, perhaps once a month or every two weeks), and new lymphocytes are generated. Algorithm 1 describes the overall functioning of the spam immune system.

The sub-algorithms describe the phases of the lifecycle in more detail: Algorithm 2 explains the generation of new lymphocytes, Algorithm 3 describes their initial training phase, Algorithm 4 explains the application of lymphocytes to messages, and Algorithm 5 details the process of culling and ageing of old lymphocytes.

Algorithm 1 Spam Immune System

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

repertoire ⇐ ϕ {Initialize repertoire (list) of lymphocytes to be empty}
update_time ⇐ *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 ⇐ *currenttime* + *update_interval* {t}ime of next lymphocyte update
  end if
end while

Algorithm 2 Generation of lymphocytes

Require: *library* ⇐ a gene fragment library (cannot be empty)
Require: *repertoire* ⇐ the list of existing lymphocytes (may be empty)
Require: *p_appending* ⇐ 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 ⇐ randomly chosen gene fragment from *library* {This starts 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
    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

Algorithm 3 Training of lymphocytes

Require: *repertoire* ⇐ the list of lymphocytes (cannot be an empty list)
Require: *message* ⇐ 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
```

4 Results

The system was tested against [16] because it is publicly available, contains sorted spam and non-spam which is relatively unaltered (messages are altered to preserve privacy and remove information added when they were donated). It is no longer very recent (the bulk of the messages are from 2002), but it should be sufficiently recent for testing purposes.

The corpus was divided up by the information found in the `Date:` email header, as it was the only date information available. Messages whose date field were clearly inaccurate (such as messages where the year was listed as 2028) were discarded, and since all of the non-spam was sent during 2002, only the spam for that year was used. The messages were grouped by month.

4.1 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 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 4.3, 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%.

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 score great than or equal to 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

Algorithm 5 Culling of antibodies: ageing and death

Require: *repertoire* ⇐ the list of antibodies (cannot be an empty list)
Require: *matched_threshold* ⇐ any lymphocyte with a *msg_matched* value below this threshold will be killed {chosen by user}
Require: *decrement* ⇐ 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

4.2 Scoring

As described in Section 3.3, three different weighting schemes have been used with the spam immune system. Each of the three systems produces a very different pattern of scores when applied to the messages. Figures 2, 3 and 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 2 shows the pattern of the straight sum scoring system. There is little clear division between the spam and the non-spam messages and there is a much wider range of scores. 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 3 shows the bowl-shaped pattern of the Bayesian scoring system. There is mostly spam at the top of the score range, and mostly non-spam at the bottom of the range, with a spike in the middle of the distribution. (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 shows the pattern of the Weighted Average scoring system. The scores 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%.

Table 1 shows the thresholds as determined experimentally. Not only did the weighted average scores give a lower error rate on average, but the standard deviation of the best threshold was smaller, which makes it easier to assume

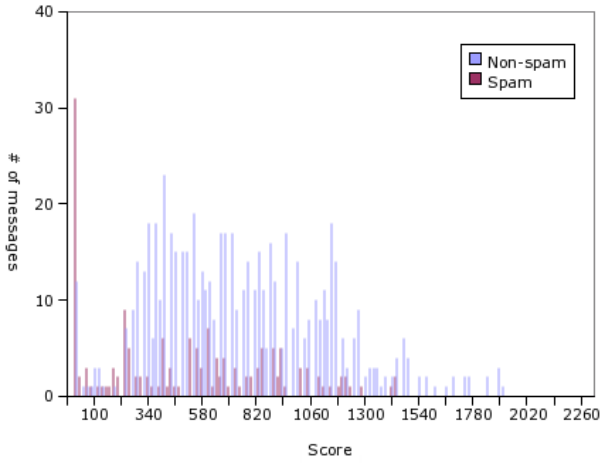

Fig. 2. Straight Sum Score Distribution

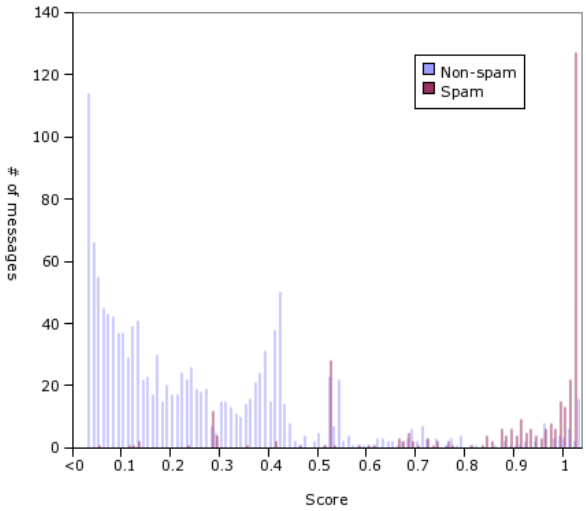

Fig. 3. Bayes Score Distribution

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 1. Average threshold values for the three scoring systems

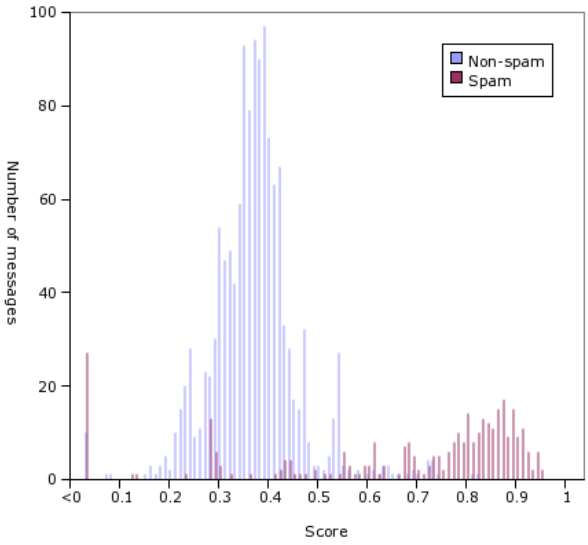

Fig. 4. Weighted Average Score Distribution

that future tests at this threshold will yield similarly good results. As such, the weighted average is the scoring system used in the other tests.

4.3 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. Figure 5 shows the percent error in classification as a function of population size. All the values shown are averages for that population size.

"Useful" lymphocytes are those that have matched some messages and thus have scores larger than zero. These are shown in Figure 6. 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.

4.4 Libraries

The libraries were tested as with the baseline test, only with different libraries used. The results, in Table 2, show that the accuracy of the heuristic library is much higher. The numbers in brackets are the standard deviations of error for each of the libraries tested. Because the standard deviation of the error using

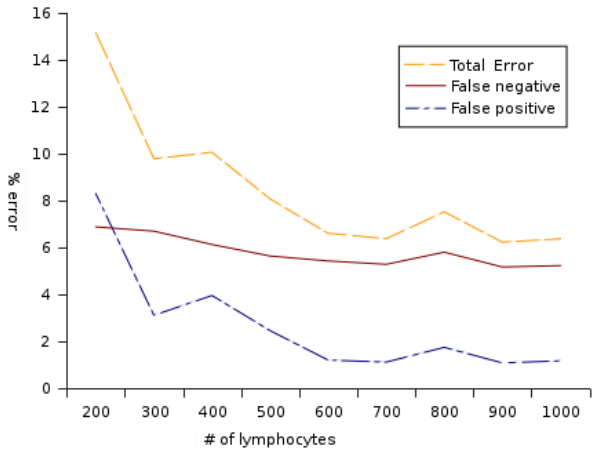

Fig. 5. Percent error versus population size

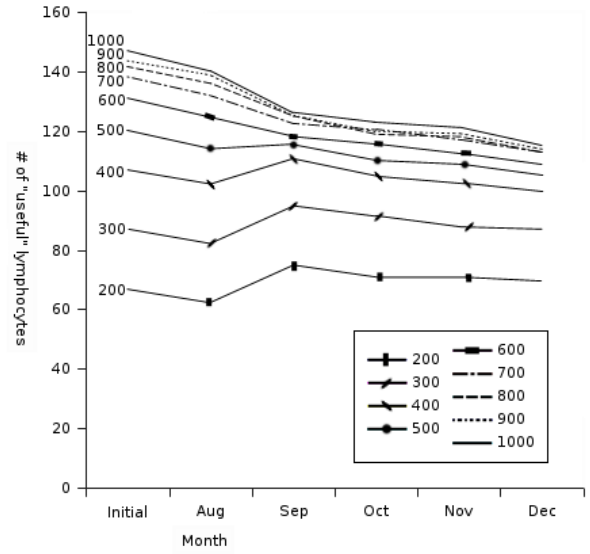

Fig. 6. The number of useful lymphocytes in each population

	False Positives	False Negatives	Total Error
Bayes	18.00 (11.13)	11.20 (2.69)	29.19 (8.94)
English	18.08 (9.84)	11.36 (3.16)	29.44 (8.18)
Heuristic	2.44 (2.10)	5.63 (1.13)	8.07 (3.02)

Table 2. Error for the three libraries

the heuristic library is low, we can reasonably assume that the error will be consistent over a range of runs.

5 Conclusions

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 [17] [18]. 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 to achieve this accuracy. As shown in [17], 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.

The scoring system which produced the best results was the weighted average, originally proposed in [2]. While Bayes system achieved similar results, the larger variance between runs made it less attractive for a system which users would want to be relatively stable.

Three libraries were tested, but it was the heuristic library, originally proposed in [1] which emerged as the most accurate for classification. The Bayesian token and English word libraries performed significantly less well. 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.

6 Future Work

In order for research to continue, work should be done to produce a suitable corpus of messages that is more up-to-date and has a better distribution than the SpamAssassin corpus. In the past, mailing lists have been used as ways to gather spam and non-spam [12]. We have explored making modifications to the popular open source list management software Mailman [19], so that collection could be done with little additional work on the part of the list administrators, but this has not yet been explored on a live mailing list.

Ideally, the new corpus would be gathered over a period longer than the one-year span of the SpamAssassin corpus. If possible, it would be nice to have a higher ratio of spam, reflecting the greater ratio of spam found in the world currently. It would be nice to have a spam corpus which exhibits periods of volatility as described in [9], as well as messages known to be relatively stable.

Once a better corpus is prepared, other gene libraries should be explored:

Adaptive gene libraries 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.

Weighted gene libraries 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 "usefulness" of fragments could be based upon the weights assigned to lymphocytes that use them.

Other ideas include allowing mutations of lymphocytes, managing parameter settings adaptively (for example, using a genetic algorithm), varying the confidence values for training during application.

References

1. Oda, T., White, T.: 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 (2003) 231–242
2. Oda, T., White, T.: Increasing the accuracy of a spam-detecting artificial immune system. In: Proceedings of the Congress on Evolutionary Computation. Volume 1., Canberra, Australia (2003) 390–396
3. Templeton, B.: Reflections on the 25th anniversary of spam. (2003)
4. Postini Inc.: Postini - email stats (2005) accessed April 2005.
5. MessageLabs Ltd.: Monthly report: February 2005. Intelligence Newsletter (2005)
6. : Controlling the assault of non-solicited pornography and marketing (CAN-SPAM) act of 2003 (2003) S. 877 as it was passed by the US Senate.
7. Asaravala, A.: With this law, you can spam. Wired News (2004)
8. Secker, A., Freitas, A., Timmis, J.: AISEC: An Artificial Immune System for E-mail Classification. In: Proceedings of the Congress on Evolutionary Computation, Canberra. Australia, IEEE (2003) 131–139
9. Sullivan, T.: The myth of spam volatility. QAQD.com White Paper (2004) Presented at the 2004 MIT Spam Conference.
10. Apache Software Foundation: Spamassassin (2005) http://spamassassin.org.
11. Pantel, P., Lin, D.: Spamcop: A spam classification & organization program. In: Learning for Text Categorization: Papers from the 1998 Workshop, Madison, Wisconsin, AAAI Technical Report WS-98-05 (1998)
12. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A bayesian approach to filtering junk E-mail. In: Learning for Text Categorization: Papers from the 1998 Workshop, Madison, Wisconsin, AAAI Technical Report WS-98-05 (1998)
13. Graham, P.: A plan for spam. Hackers & Painters (2002)
14. : SpamBayes: Bayesian anti-spam classifier written in python (2004) Accessed October 13, 2004. http://spambayes.sourceforge.net/.
15. Graham, P.: Better bayesian filtering. In: 2003 Spam Conference. (2003)
16. : SpamAssassin public corpus (2003) http://spamassassin.org/publiccorpus/.
17. Anderson, R.: Filters take a bite out of spam. Network Computing (2004)
18. Metz, C.: Spam blockers. PC Magazine (2004)
19. Free Software Foundation: Mailman (2005) http://www.list.org.