In America

January 25, 2004

SP@M SHEN@NIG@NS!!

That Gibberish in Your In-Box May Be Good News

By GEORGE JOHNSON

IF you could sit back with Zen-like detachment and observe the dross piling up in your electronic mailbox, the spam wars might come to seem like a fascinating electronic game. Like creatures running through a maze with constantly shifting walls, spammers dart and weave to sneak their solicitations past ever wilier junk mail filters. They are organisms, or maybe genomes, grinding out one random mutation after another, desperately trying to elude the Grim Reaper.

Viagra becomes "vi@gra" or "v-i-@-g-r-a." Then, as the filters adapt, "v1@gr@" and even "\/l@gr@." Currently, the Internet is swarming with mutants like this: "Cheap Val?(u)m, Viagr@, X(a)n@x, Som@ Di3t Pills Many M3ds RIZfURqgHr77B," the final string of gibberish hanging like an appendage of junk DNA.

Taking a different approach, a come-on for barnyard pornography devolves into "faurm galz bing e rottic." Another pitch promises to reveal "Seakrets of ((eks-eks-eks)) stars."

Dispiriting as it is to start the morning with a hundred of these orthographic monsters crouching in your in-box, there is reason to take heart. Measured in bits and bytes, the sheer volume of spam may not have diminished. But advanced filtering software, which learns to recognize the mercurial traits of junk e-mail, is having an effect. The spammers' messages are becoming harder and harder to decipher. Sense is inevitably degenerating into nonsense, like a pileup of random mutations in an endangered species gasping its last breaths.

Earlier this month, when Internet experts met in Cambridge, Mass., for the 2004 Spam Conference (available as a Web broadcast at spamconference.org), they showed just how far the science of spam fighting has come. For all the recent talk of suing spammers and compiling a national do-not-spam list, most speakers were putting their hopes in technological, not legal solutions. The federal government's new junk e-mail law, the Can Spam Act, barely rated a mention.

Terry Sullivan, a spam researcher with a doctorate in information science, described how he used a "handy 10-dimensional high-fidelity model of historical spam space" to analyze how junk e-mail changes over time. Long stretches of stability are suddenly interrupted by brief bursts of innovation, a pattern he compared to what some evolutionary biologists call punctuated equilibrium. The encouraging news is that there is enough stability - an enduring core of "spamminess" - for the invaders to be quickly identified and destroyed.

Another presentation, called "Cockroaches Hate the Light," considered how to authenticate senders so that spammers can't easily fake their identities. Other speakers proposed eco-electronic solutions like digital postage stamps that would put a price on sending e-mail - trivial for an individual user but making hit-or-miss barrages prohibitively expensive.

Like epidemiologists discussing how to predict and control a biological outbreak, conferencegoers compared the merits of various filtering techniques. Which is better: first-order Bayesian, token grab bag, sparse binary polynomial hash or markovian weighting? The meaning of the terms may be opaque to outsiders, but the underlying message comes through: the spammers are up against some increasingly advanced cybernetic artillery.

Many experts believe that solving the spam problem will require a combination of approaches. But laws take forever to pass and amend. Technological fixes like sender authentication and electronic stamps would also take time to carry out, but filtering is already here - and it is reducing the spammers' messages to feeble signals swamped by a roar of alphanumeric noise.

The turning point came in August 2002 when a computer scientist, Paul Graham, issued a manifesto called "A Plan for Spam," describing how to filter e-mail using a statistical method discovered in the 18th century by the English theologian and mathematician Thomas Bayes. Bayesian e-mail filters had been studied for years, but with Mr. Graham's paper the idea went mainstream.

Presented with thousands of examples of good and bad e-mail, a Bayesian filter compiles a list ranking each word according to how likely it is to appear in junk e-mail. Rising to the top of the roster are high scorers like Valium, Xanax, mortgage, porn and Viagra. Settling toward the bottom are words like deciduous, cashmere and intensify. Hovering in the middle are the vast number of neutral words that can swing either way.

When a new piece of e-mail arrives, the filtering program counts up the words and computes an overall ranking. If the number exceeds a certain threshold, the message is rejected as spam.

A message from a friend saying that she is so worried about refinancing her mortgage that she took a Valium will pique the filter's interest. But most of the text will probably consist of words with neutral or very low rankings, dragging down the score and allowing the e-mail to go through.

If a spam promising "l0w m0rtg@ge rates" slips by, the filter is informed by the user that it has made an error. The mutation is then moved higher on the list, as well as future mutations of the mutation, until the spammer is reduced to sending gobbledygook. A recent e-mail message making the rounds promised "Leacatharsisrn to make a fortcongestiveune on eBay!" (A Web link inside led to a site with information on a money-making auction scheme.)

Increasingly the subject lines convey no meaning at all: "begonia breadfruit extempore defocus purveyor." For the spammer, the hope, slim as it seems, is that a few curious souls will open and read the e-mail, which begins, "I finally was able to lsoe the wieght" and goes on to offer a product "Guanarteed to work or your menoy back!" Read out loud, the message sounds a little like HAL the computer in "2001: A Space Odyssey" sinking into aphasia as its synapses are severed one by one.

In what may be their final death throes, some spammers have begun sending messages consisting of a single image or a one-line sales pitch - "picospams" - with a link to a Web site. Often appended at the end, in an attempt to flummox the filters, is a scrap of Dadaist poetry - "feverish squirt feat transconductance terrify broken trite fascist axis stultify floc bookshelves. " Sometimes this "word salad," as it has come to be called, is rendered in invisible ink - white letters on a white background - or hidden inside an embedded formatting command.

No matter. The filters learn to adapt. If the spammers want to stay in business, ultimately they must convey at least a hint of meaning. After all, you cannot send a completely random message - or one that is blank - and expect many people to click the link.

```
Return-Path: pbbvuoaupdyc@india.com
Return-Path: <pbbvuoaupdyc@india.com>
Received: from ftp.cse.ucsc.edu (ftp.cse.ucsc.edu [128.114.48.173])
        by services.cse.ucsc.edu (8.12.10/8.12.10) with ESMTP id i0PEdBeU024407
        (version=TLSv1/SSLv3 cipher=EDH-RSA-DES-CBC3-SHA bits=168 verify=NO)
        for <draper@ams.ucsc.edu>; Sun, 25 Jan 2004 06:39:11 -0800 (PST)
Received: from 202-178-170-33.cm.apol.com.tw (202-178-170-33.cm.apol.com.tw [202
        by ftp.cse.ucsc.edu (8.12.10/8.12.10) with SMTP id i0PEd4jj028003
        for <draper@ams.ucsc.edu>; Sun, 25 Jan 2004 06:39:09 -0800 (PST)
Received: from [202.178.170.33] by 3002hosting.comIP with HTTP;
        Sun, 25 Jan 2004 12:29:31 -0200
From: "Jasmine Schmidt" <pbbvuoaupdyc@india.com>
To: draper@ams.ucsc.edu
Subject: Re: NUR, the telegram staggered
Mime-Version: 1.0
X-Mailer: mPOP Web-Mail 2.19
X-Originating-IP: [3002hosting.comIP]
Date: Sun, 25 Jan 2004 11:35:31 -0300
Reply-To: "Jasmine Schmidt" <pbbvuoaupdyc@india.com>
Content-Type: multipart/alternative;
        boundary="--ALT--XNVW84170679213703"
Message-Id: <ASWTGFN-0009570489478@cankerworm>
X-Spam-Checker-Version: SpamAssassin 2.60 (1.212-2003-09-23-exp) on
        services.cse.ucsc.edu
X-Spam-Level: **
X-Spam-Status: No, hits=2.5 required=5.0 tests=HTML_MESSAGE,
        RCVD_IN_BL_SPAMCOP_NET,RCVD_IN_RFCI,RCVD_IN_SORBS autolearn=no
        version=2.60
X-Spam-Report:
        *  0.0 HTML_MESSAGE BODY: HTML included in message
        *  2.2 RCVD_IN_BL_SPAMCOP_NET RBL: Received via a relay in bl.spamcop.ne
        *      [Blocked - see <http://www.spamcop.net/bl.shtml?202.178.170.33>]
        *  0.1 RCVD_IN_SORBS RBL: SORBS: sender is listed in SORBS
        *      [202.178.170.33 listed in dnsbl.sorbs.net]
        *  0.1 RCVD_IN_RFCI RBL: Sent via a relay in ipwhois.rfc-ignorant.org
        *      [Inaccurate or missing WHOIS data]

----ALT--XNVW84170679213703
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 8bit

continuity grayish whether crafty
cab sidesaddle coffeepot builtin septennial
midas alligator bluster cryptogram bakelite billings

----ALT--XNVW84170679213703
Content-Type: text/html; charset=us-ascii
Content-Transfer-Encoding: 8bit

<HTML><HEAD>
<BODY>
<p>Fr</citation>ee Ca</tuscany>bleTV!N</bridesmaid>o mo</aerate>re p</apprise>ay
<a href="http://www.3002hosting.com/cable/">
<img border="0" src="http://www.3002hosting.com/fiter1.jpg"></a>
effluvium instant amarillo blackfeet coffin sommelier budapest rupee free chutne
boisterous or beryl immobility abstracter dossier inadvertent digram dew knutson
snob grotesque wintertime behalf brigantine commodity surf kline offertory bobbi
belch court automata dote flinty manic prank ear congregate philistine surrender
creche obstruct bolshevism anyway chorus lucretius tactual portend whipsaw anagr
infra sealant lana dockyard copernican fragmentation spirit vehicular advent bri
```

supplant built adjudge deterrent some glycerol paunchy combinator bristol velvet bucket dey anaglyph importunate convict maternal manhole despond buttery crania flabbergast pet methionine conformance thirteenth slept arhat ltv mockery pollen pauline con muse aldrich pluto commodore areawide crucial hostile directrices el ginmill baffle navigate angelic cellulose edinburgh kept comedy downriver chloro

</BODY>
</HTML>

----ALT--XNVW84170679213703--

A Plan for Spam

August 2002

(This article describes the spam-filtering techniques used in the spamproof web-based mail reader we built to exercise Arc. An improved algorithm is described in Better Bayesian Filtering.)

I think it's possible to stop spam, and that content-based filters are the way to do it. The Achilles heel of the spammers is their message. They can circumvent any other barrier you set up. They have so far, at least. But they have to deliver their message, whatever it is. If we can write software that recognizes their messages, there is no way they can get around that.

To the recipient, spam is easily recognizable. If you hired someone to read your mail and discard the spam, they would have little trouble doing it. How much do we have to do, short of AI, to automate this process?

I think we will be able to solve the problem with fairly simple algorithms. In fact, I've found that you can filter present-day spam acceptably well using nothing more than a Bayesian combination of the spam probabilities of individual words. Using a slightly tweaked (as described below) Bayesian filter, we now miss less than 5 per 1000 spams, with 0 false positives.

The statistical approach is not usually the first one people try when they write spam filters. Most hackers' first instinct is to try to write software that recognizes individual properties of spam. You look at spams and you think, the gall of these guys to try sending me mail that begins "Dear Friend" or has a subject line that's all uppercase and ends in eight exclamation points. I can filter out that stuff with about one line of code.

And so you do, and in the beginning it works. A few simple rules will take a big bite out of your incoming spam. Merely looking for the word "click" will catch 79.7% of the emails in my spam corpus, with only 1.2% false positives.

I spent about six months writing software that looked for individual spam features before I tried the statistical approach. What I found was that recognizing that last few percent of spams got very hard, and that as I made the filters stricter I got more false positives.

False positives are innocent emails that get mistakenly identified as spams. For most users, missing legitimate email is an order of magnitude worse than receiving spam, so a filter that yields false positives is like an acne cure that carries a risk of death to the patient.

The more spam a user gets, the less likely he'll be to notice one innocent mail sitting in his spam folder. And strangely enough, the better your spam filters get, the more dangerous false positives become, because when the filters are really good, users will be more likely to ignore everything they catch.

I don't know why I avoided trying the statistical approach for so long. I think it was because I got addicted to trying to identify spam features

myself, as if I were playing some kind of competitive game with the spammers. (Nonhackers don't often realize this, but most hackers are very competitive.) When I did try statistical analysis, I found immediately that it was much cleverer than I had been. It discovered, of course, that terms like "virtumundo" and "teens" were good indicators of spam. But it also discovered that "per" and "FL" and "ff0000" are good indicators of spam. In fact, "ff0000" (html for bright red) turns out to be as good an indicator of spam as any pornographic term.

Here's a sketch of how I do statistical filtering. I start with one corpus of spam and one of nonspam mail. At the moment each one has about 4000 messages in it. I scan the entire text, including headers and embedded html and javascript, of each message in each corpus. I currently consider alphanumeric characters, dashes, apostrophes, and dollar signs to be part of tokens, and everything else to be a token separator. (There is probably room for improvement here.) I ignore tokens that are all digits, and I also ignore html comments, not even considering them as token separators.

I count the number of times each token (ignoring case, currently) occurs in each corpus. At this stage I end up with two large hash tables, one for each corpus, mapping tokens to number of occurrences.

Next I create a third hash table, this time mapping each token to the probability that an email containing it is a spam, which I calculate as follows [1]:

```
(let ((g (* 2 (or (gethash word good) 0)))
      (b (or (gethash word bad) 0)))
   (unless (< (+ g b) 5)
     (max .01
          (min .99 (float (/ (min 1 (/ b nbad))
                             (+ (min 1 (/ g ngood))
                                (min 1 (/ b nbad))))))))))
```

where `word` is the token whose probability we're calculating, `good` and `bad` are the hash tables I created in the first step, and `ngood` and `nbad` are the number of nonspam and spam messages respectively.

I explained this as code to show a couple of important details. I want to bias the probabilities slightly to avoid false positives, and by trial and error I've found that a good way to do it is to double all the numbers in `good`. This helps to distinguish between words that occasionally do occur in legitimate email and words that almost never do. I only consider words that occur more than five times in total (actually, because of the doubling, occurring three times in nonspam mail would be enough). And then there is the question of what probability to assign to words that occur in one corpus but not the other. Again by trial and error I chose .01 and .99. There may be room for tuning here, but as the corpus grows such tuning will happen automatically anyway.

The especially observant will notice that while I consider each corpus to be a single long stream of text for purposes of counting occurrences, I use the number of emails in each, rather than their combined length, as the divisor in calculating spam probabilities. This adds another slight bias to protect against false positives.

When new mail arrives, it is scanned into tokens, and the most interesting fifteen tokens, where interesting is measured by how far their spam probability is from a neutral .5, are used to calculate the

probability that the mail is spam. If `probs` is a list of the fifteen individual probabilities, you calculate the combined probability thus:

```
(let ((prod (apply #'* probs)))
  (/ prod (+ prod (apply #'* (mapcar #'(lambda (x)
                                         (- 1 x))
                                     probs)))))
```

One question that arises in practice is what probability to assign to a word you've never seen, i.e. one that doesn't occur in the hash table of word probabilities. I've found, again by trial and error, that .4 is a good number to use. If you've never seen a word before, it is probably fairly innocent; spam words tend to be all too familiar.

There are examples of this algorithm being applied to actual emails in an appendix at the end.

I treat mail as spam if the algorithm above gives it a probability of more than .9 of being spam. But in practice it would not matter much where I put this threshold, because few probabilities end up in the middle of the range.

One great advantage of the statistical approach is that you don't have to read so many spams. Over the past six months, I've read literally thousands of spams, and it is really kind of demoralizing. Norbert Wiener said if you compete with slaves you become a slave, and there is something similarly degrading about competing with spammers. To recognize individual spam features you have to try to get into the mind of the spammer, and frankly I want to spend as little time inside the minds of spammers as possible.

But the real advantage of the Bayesian approach, of course, is that you know what you're measuring. Feature-recognizing filters like SpamAssassin assign a spam "score" to email. The Bayesian approach assigns an actual probability. The problem with a "score" is that no one knows what it means. The user doesn't know what it means, but worse still, neither does the developer of the filter. How many points should an email get for having the word "sex" in it? A probability can of course be mistaken, but there is little ambiguity about what it means, or how evidence should be combined to calculate it. Based on my corpus, "sex" indicates a .97 probability of the containing email being a spam, whereas "sexy" indicates .99 probability. And Bayes' Rule, equally unambiguous, says that an email containing both words would, in the (unlikely) absence of any other evidence, have a 99.97% chance of being a spam.

Because it is measuring probabilities, the Bayesian approach considers all the evidence in the email, both good and bad. Words that occur disproportionately rarely in spam (like "though" or "tonight" or "apparently") contribute as much to decreasing the probability as bad words like "unsubscribe" and "opt-in" do to increasing it. So an otherwise innocent email that happens to include the word "sex" is not going to get tagged as spam.

Ideally, of course, the probabilities should be calculated individually for each user. I get a lot of email containing the word "Lisp", and (so far) no spam that does. So a word like that is effectively a kind of password for sending mail to me. In my earlier spam-filtering software, the user could set up a list of such words and mail containing them would automatically get past the filters. On my list I put words like "Lisp" and also my zipcode, so that (otherwise rather spammy-sounding) receipts from online orders would get through. I thought I was being very clever, but I found that the Bayesian filter did the same thing for me, and moreover discovered of a lot of words I hadn't thought of.

When I said at the start that our filters let through less than 5 spams per 1000 with 0 false positives, I'm talking about filtering my mail based on a corpus of my mail. But these numbers are not misleading, because that is the approach I'm advocating: filter each user's mail based on the spam and nonspam mail he receives. Essentially, each user should have two delete buttons, ordinary delete and delete-as-spam. Anything deleted as spam goes into the spam corpus, and everything else goes into the nonspam corpus.

You could start users with a seed filter, but ultimately each user should have his own per-word probabilities based on the actual mail he receives. This (a) makes the filters more effective, (b) lets each user decide their own precise definition of spam, and (c) perhaps best of all makes it hard for spammers to tune mails to get through the filters. If a lot of the brain of the filter is in the individual databases, then merely tuning spams to get through the seed filters won't guarantee anything about how well they'll get through individual users' varying and much more trained filters.

Content-based spam filtering is often combined with a whitelist, a list of senders whose mail can be accepted with no filtering. One easy way to build such a whitelist is to keep a list of every address the user has ever sent mail to. If a mail reader has a delete-as-spam button then you could also add the from address of every email the user has deleted as ordinary trash.

I'm an advocate of whitelists, but more as a way to save computation than as a way to improve filtering. I used to think that whitelists would make filtering easier, because you'd only have to filter email from people you'd never heard from, and someone sending you mail for the first time is constrained by convention in what they can say to you. Someone you already know might send you an email talking about sex, but someone sending you mail for the first time would not be likely to. The problem is, people can have more than one email address, so a new from-address doesn't guarantee that the sender is writing to you for the first time. It is not unusual for an old friend (especially if he is a hacker) to suddenly send you an email with a new from-address, so you can't risk false positives by filtering mail from unknown addresses especially stringently.

In a sense, though, my filters do themselves embody a kind of whitelist (and blacklist) because they are based on entire messages, including the headers. So to that extent they "know" the email addresses of trusted senders and even the routes by which mail gets from them to me. And they know the same about spam, including the server names, mailer versions, and protocols.

If I thought that I could keep up current rates of spam filtering, I would consider this problem solved. But it doesn't mean much to be able to filter out most present-day spam, because spam evolves. Indeed, most antispam techniques so far have been like pesticides that do nothing more than create a new, resistant strain of bugs.

I'm more hopeful about Bayesian filters, because they evolve with the spam. So as spammers start using "c0ck" instead of "cock" to evade simple-minded spam filters based on individual words, Bayesian filters automatically notice. Indeed, "c0ck" is far more damning evidence than "cock", and Bayesian filters know precisely how much more.

Still, anyone who proposes a plan for spam filtering has to be able to answer the question: if the spammers knew exactly what you were doing, how well could they get past you? For example, I think that if checksum-based spam filtering becomes a serious obstacle, the

spammers will just switch to mad-lib techniques for generating message bodies.

To beat Bayesian filters, it would not be enough for spammers to make their emails unique or to stop using individual naughty words. They'd have to make their mails indistinguishable from your ordinary mail. And this I think would severely constrain them. Spam is mostly sales pitches, so unless your regular mail is all sales pitches, spams will inevitably have a different character. And the spammers would also, of course, have to change (and keep changing) their whole infrastructure, because otherwise the headers would look as bad to the Bayesian filters as ever, no matter what they did to the message body. I don't know enough about the infrastructure that spammers use to know how hard it would be to make the headers look innocent, but my guess is that it would be even harder than making the message look innocent.

Assuming they could solve the problem of the headers, the spam of the future will probably look something like this:

```
Hey there.  Thought you should check out the following:
http://www.27meg.com/foo
```

because that is about as much sales pitch as content-based filtering will leave the spammer room to make. (Indeed, it will be hard even to get this past filters, because if everything else in the email is neutral, the spam probability will hinge on the url, and it will take some effort to make that look neutral.)

Spammers range from businesses running so-called opt-in lists who don't even try to conceal their identities, to guys who hijack mail servers to send out spams promoting porn sites. If we use filtering to whittle their options down to mails like the one above, that should pretty much put the spammers on the "legitimate" end of the spectrum out of business; they feel obliged by various state laws to include boilerplate about why their spam is not spam, and how to cancel your "subscription," and that kind of text is easy to recognize.

(I used to think it was naive to believe that stricter laws would decrease spam. Now I think that while stricter laws may not decrease the amount of spam that spammers send, they can certainly help filters to decrease the amount of spam that recipients actually see.)

All along the spectrum, if you restrict the sales pitches spammers can make, you will inevitably tend to put them out of business. That word business is an important one to remember. The spammers are businessmen. They send spam because it works. It works because although the response rate is abominably low (at best 15 per million, vs 3000 per million for a catalog mailing), the cost, to them, is practically nothing. The cost is enormous for the recipients, about 5 man-weeks for each million recipients who spend a second to delete the spam, but the spammer doesn't have to pay that.

Sending spam does cost the spammer something, though. [2] So the lower we can get the response rate-- whether by filtering, or by using filters to force spammers to dilute their pitches-- the fewer businesses will find it worth their while to send spam.

The reason the spammers use the kinds of sales pitches that they do is to increase response rates. This is possibly even more disgusting than getting inside the mind of a spammer, but let's take a quick look inside the mind of someone who responds to a spam. This person is either astonishingly credulous or deeply in denial about their sexual interests. In either case, repulsive or idiotic as the spam seems to us, it is exciting to them. The spammers wouldn't say these things if they didn't sound exciting. And "thought you should check out the following" is just not going to have nearly the pull with the spam

recipient as the kinds of things that spammers say now. Result: if it can't contain exciting sales pitches, spam becomes less effective as a marketing vehicle, and fewer businesses want to use it.

That is the big win in the end. I started writing spam filtering software because I didn't want have to look at the stuff anymore. But if we get good enough at filtering out spam, it will stop working, and the spammers will actually stop sending it.

– – –

Of all the approaches to fighting spam, from software to laws, I believe Bayesian filtering will be the single most effective. But I also think that the more different kinds of antispam efforts we undertake, the better, because any measure that constrains spammers will tend to make filtering easier. And even within the world of content-based filtering, I think it will be a good thing if there are many different kinds of software being used simultaneously. The more different filters there are, the harder it will be for spammers to tune spams to get through them.

Appendix: Examples of Filtering

Here is an example of a spam that arrived while I was writing this article. The fifteen most interesting words in this spam are:

```
qvp0045
indira
mx-05
intimail
$7500
freeyankeedom
cdo
bluefoxmedia
jpg
unsecured
platinum
3d0
qves
7c5
7c266675
```

The words are a mix of stuff from the headers and from the message body, which is typical of spam. Also typical of spam is that every one of these words has a spam probability, in my database, of .99. In fact there are more than fifteen words with probabilities of .99, and these are just the first fifteen seen.

Unfortunately that makes this email a boring example of the use of Bayes' Rule. To see an interesting variety of probabilities we have to look at this actually quite atypical spam.

The fifteen most interesting words in this spam, with their probabilities, are:

```
madam           0.99
promotion       0.99
republic        0.99
shortest        0.047225013
mandatory       0.047225013
standardization 0.07347802
sorry           0.08221981
supported       0.09019077
people's        0.09019077
enter           0.9075001
quality         0.8921298
```

```
organization    0.12454646
investment      0.8568143
very            0.14758544
valuable        0.82347786
```

This time the evidence is a mix of good and bad. A word like "shortest" is almost as much evidence for innocence as a word like "madam" or "promotion" is for guilt. But still the case for guilt is stronger. If you combine these numbers according to Bayes' Rule, the resulting probability is .9027.

"Madam" is obviously from spams beginning "Dear Sir or Madam." They're not very common, but the word "madam" never occurs in my legitimate email, and it's all about the ratio.

"Republic" scores high because it often shows up in Nigerian scam emails, and also occurs once or twice in spams referring to Korea and South Africa. You might say that it's an accident that it thus helps identify this spam. But I've found when examining spam probabilities that there are a lot of these accidents, and they have an uncanny tendency to push things in the right direction rather than the wrong one. In this case, it is not entirely a coincidence that the word "Republic" occurs in Nigerian scam emails and this spam. There is a whole class of dubious business propositions involving less developed countries, and these in turn are more likely to have names that specify explicitly (because they aren't) that they are republics.[3]

On the other hand, "enter" is a genuine miss. It occurs mostly in unsubscribe instructions, but here is used in a completely innocent way. Fortunately the statistical approach is fairly robust, and can tolerate quite a lot of misses before the results start to be thrown off.

For comparison, here is an example of that rare bird, a spam that gets through the filters. Why? Because by sheer chance it happens to be loaded with words that occur in my actual email:

```
perl        0.01
python      0.01
tcl         0.01
scripting   0.01
morris      0.01
graham      0.01491078
guarantee   0.9762507
cgi         0.9734398
paul        0.027040077
quite       0.030676773
pop3        0.042199217
various     0.06080265
prices      0.9359873
managed     0.06451222
difficult   0.071706355
```

There are a couple pieces of good news here. First, this mail probably wouldn't get through the filters of someone who didn't happen to specialize in programming languages and have a good friend called Morris. For the average user, all the top five words here would be neutral and would not contribute to the spam probability.

Second, I think filtering based on word pairs (see below) might well catch this one: "cost effective", "setup fee", "money back" -- pretty incriminating stuff. And of course if they continued to spam me (or a network I was part of), "Hostex" itself would be recognized as a spam term.

Finally, here is an innocent email. Its fifteen most interesting words are as follows:

```
continuation  0.01
describe      0.01
```

```
continuations 0.01
example       0.033600237
programming   0.05214485
i'm           0.055427782
examples      0.07972858
color         0.9189189
localhost     0.09883721
hi            0.116539136
california    0.84421706
same          0.15981844
spot          0.1654587
us-ascii      0.16804294
what          0.19212411
```

Most of the words here indicate the mail is an innocent one. There are two bad smelling words, "color" (spammers love colored fonts) and "California" (which occurs in testimonials and also in menus in forms), but they are not enough to outweigh obviously innocent words like "continuation" and "example".

It's interesting that "describe" rates as so thoroughly innocent. It hasn't occurred in a single one of my 4000 spams. The data turns out to be full of such surprises. One of the things you learn when you analyze spam texts is how narrow a subset of the language spammers operate in. It's that fact, together with the equally characteristic vocabulary of any individual user's mail, that makes Bayesian filtering a good bet.

Appendix: More Ideas

One idea that I haven't tried yet is to filter based on word pairs, or even triples, rather than individual words. This should yield a much sharper estimate of the probability. For example, in my current database, the word "offers" has a probability of .96. If you based the probabilities on word pairs, you'd end up with "special offers" and "valuable offers" having probabilities of .99 and, say, "approach offers" (as in "this approach offers") having a probability of .1 or less.

The reason I haven't done this is that filtering based on individual words already works so well. But it does mean that there is room to tighten the filters if spam gets harder to detect. (Curiously, a filter based on word pairs would be in effect a Markov-chaining text generator running in reverse.)

Specific spam features (e.g. not seeing the recipient's address in the to: field) do of course have value in recognizing spam. They can be considered in this algorithm by treating them as virtual words. I'll probably do this in future versions, at least for a handful of the most egregious spam indicators. Feature-recognizing spam filters are right in many details; what they lack is an overall discipline for combining evidence.

Recognizing nonspam features may be more important than recognizing spam features. False positives are such a worry that they demand extraordinary measures. I will probably in future versions add a second level of testing designed specifically to avoid false positives. If a mail triggers this second level of filters it will be accepted even if its spam probability is above the threshold.

I don't expect this second level of filtering to be Bayesian. It will inevitably be not only ad hoc, but based on guesses, because the number of false positives will not tend to be large enough to notice patterns. (It is just as well, anyway, if a backup system doesn't rely on the same technology as the primary system.)

Another thing I may try in the future is to focus extra attention on specific parts of the email. For example, about 95% of current spam

includes the url of a site they want you to visit. (The remaining 5% want you to call a phone number, reply by email or to a US mail address, or in a few cases to buy a certain stock.) The url is in such cases practically enough by itself to determine whether the email is spam.

Domain names differ from the rest of the text in a (non-German) email in that they often consist of several words stuck together. Though computationally expensive in the general case, it might be worth trying to decompose them. If a filter has never seen the token "xxxporn" before it will have an individual spam probability of .4, whereas "xxx" and "porn" individually have probabilities (in my corpus) of .9889 and .99 respectively, and a combined probability of .9998.

I expect decomposing domain names to become more important as spammers are gradually forced to stop using incriminating words in the text of their messages. (A url with an ip address is of course an extremely incriminating sign, except in the mail of a few sysadmins.)

It might be a good idea to have a cooperatively maintained list of urls promoted by spammers. We'd need a trust metric of the type studied by Raph Levien to prevent malicious or incompetent submissions, but if we had such a thing it would provide a boost to any filtering software. It would also be a convenient basis for boycotts.

Another way to test dubious urls would be to send out a crawler to look at the site before the user looked at the email mentioning it. You could use a Bayesian filter to rate the site just as you would an email, and whatever was found on the site could be included in calculating the probability of the email being a spam. A url that led to a redirect would of course be especially suspicious.

One cooperative project that I think really would be a good idea would be to accumulate a giant corpus of spam. A large, clean corpus is the key to making Bayesian filtering work well. Bayesian filters could actually use the corpus as input. But such a corpus would be useful for other kinds of filters too, because it could be used to test them.

Creating such a corpus poses some technical problems. We'd need trust metrics to prevent malicious or incompetent submissions, of course. We'd also need ways of erasing personal information (not just to-addresses and ccs, but also e.g. the arguments to unsubscribe urls, which often encode the to-address) from mails in the corpus. If anyone wants to take on this project, it would be a good thing for the world.

Appendix: Defining Spam

I think there is a rough consensus on what spam is, but it would be useful to have an explicit definition. We'll need to do this if we want to establish a central corpus of spam, or even to compare spam filtering rates meaningfully.

To start with, spam is not unsolicited commercial email. If someone in my neighborhood heard that I was looking for an old Raleigh three-speed in good condition, and sent me an email

offering to sell me one, I'd be delighted, and yet this email would be both commercial and unsolicited. The defining feature of spam (in fact, its raison d'etre) is not that it is unsolicited, but that it is automated.

It is merely incidental, too, that spam is usually commercial. If someone started sending mass email to support some political

cause, for example, it would be just as much spam as email promoting a porn site.

I propose we define spam as unsolicited automated email. This definition thus includes some email that many legal definitions of spam don't. Legal definitions of spam, influenced presumably by lobbyists, tend to exclude mail sent by companies that have an "existing relationship" with the recipient. But buying something from a company, for example, does not imply that you have solicited ongoing email from them. If I order something from an online store, and they then send me a stream of spam, it's still spam.

Companies sending spam often give you a way to "unsubscribe," or ask you to go to their site and change your "account preferences" if you want to stop getting spam. This is not enough to stop the mail from being spam. Not opting out is not the same as opting in. Unless the recipient explicitly checked a clearly labelled box (whose default was no) asking to receive the email, then it is spam.

In some business relationships, you do implicitly solicit certain kinds of mail. When you order online, I think you implicitly solicit a receipt, and notification when the order ships. I don't mind when Verisign sends me mail warning that a domain name is about to expire (at least, if they are the actual registrar for it). But when Verisign sends me email offering a FREE Guide to Building My E-Commerce Web Site, that's spam.

Notes:

[1] The examples in this article are translated into Common Lisp for, believe it or not, greater accessibility. The application described here is one that we wrote in order to test a new Lisp dialect called Arc that is not yet released.

[2] Currently the lowest rate seems to be about $200 to send a million spams. That's very cheap, 1/50th of a cent per spam. But filtering out 95% of spam, for example, would increase the spammers' cost to reach a given audience by a factor of 20. Few can have margins big enough to absorb that.

[3] As a rule of thumb, the more qualifiers there are before the name of a country, the more corrupt the rulers. A country called The Socialist People's Democratic Republic of X is probably the last place in the world you'd want to live.

Thanks to Sarah Harlin for reading drafts of this; Daniel Giffin (who is also writing the production Arc interpreter) for several good ideas about filtering and for creating our mail infrastructure; Robert Morris, Trevor Blackwell and Erann Gat for many discussions about spam; Raph Levien for advice about trust metrics; and Chip Coldwell and Sam Steingold for advice about statistics.

More Info:

- Plan for Spam FAQ
- Filters that Fight Back
- Japanese Translation
- Chinese Translation
- Better Bayesian Filtering
- Will Filters Kill Spam?
- Spanish Translation
- Probability

Better Bayesian Filtering

January 2003

(This article was given as a talk at the 2003 Spam Conference. It describes the work I've done to improve the performance of the algorithm described in A Plan for Spam, and what I plan to do in the future.)

The first discovery I'd like to present here is an algorithm for lazy evaluation of research papers. Just write whatever you want and don't cite any previous work, and indignant readers will send you references to all the papers you should have cited. I discovered this algorithm after "A Plan for Spam" [1] was on Slashdot.

Spam filtering is a subset of text classification, which is a well established field, but the first papers about Bayesian spam filtering per se seem to have been two given at the same conference in 1998, one by Pantel and Lin [2], and another by a group from Microsoft Research [3].

When I heard about this work I was a bit surprised. If people had been onto Bayesian filtering four years ago, why wasn't everyone using it? When I read the papers I found out why. Pantel and Lin's filter was the more effective of the two, but it only caught 92% of spam, with 1.16% false positives.

When I tried writing a Bayesian spam filter, it caught 99.5% of spam with less than .03% false positives [4]. It's always alarming when two people trying the same experiment get widely divergent results. It's especially alarming here because those two sets of numbers might yield opposite conclusions. Different users have different requirements, but I think for many people a filtering rate of 92% with 1.16% false positives means that filtering is not an acceptable solution, whereas 99.5% with less than .03% false positives means that it is.

So why did we get such different numbers? I haven't tried to reproduce Pantel and Lin's results, but from reading the paper I see five things that probably account for the difference.

One is simply that they trained their filter on very little data: 160 spam and 466 nonspam mails. Filter performance should still be climbing with data sets that small. So their numbers may not even be an accurate measure of the performance of their algorithm, let alone of Bayesian spam filtering in general.

But I think the most important difference is probably that they ignored message headers. To anyone who has worked on spam filters, this will seem a perverse decision. And yet in the very first filters I tried writing, I ignored the headers too. Why? Because I wanted to keep the problem neat. I didn't know much about mail headers then, and they seemed to me full of random stuff. There is a lesson here for filter writers: don't ignore data. You'd think this lesson would be too obvious to mention, but I've had to learn it several times.

Third, Pantel and Lin stemmed the tokens, meaning they reduced e.g. both "mailing" and "mailed" to the root "mail". They may have felt they were forced to do this by the small size of their corpus, but if so this is a kind of premature

optimization.

Fourth, they calculated probabilities differently. They used all the tokens, whereas I only use the 15 most significant. If you use all the tokens you'll tend to miss longer spams, the type where someone tells you their life story up to the point where they got rich from some multilevel marketing scheme. And such an algorithm would be easy for spammers to spoof: just add a big chunk of random text to counterbalance the spam terms.

Finally, they didn't bias against false positives. I think any spam filtering algorithm ought to have a convenient knob you can twist to decrease the false positive rate at the expense of the filtering rate. I do this by counting the occurrences of tokens in the nonspam corpus double.

I don't think it's a good idea to treat spam filtering as a straight text classification problem. You can use text classification techniques, but solutions can and should reflect the fact that the text is email, and spam in particular. Email is not just text; it has structure. Spam filtering is not just classification, because false positives are so much worse than false negatives that you should treat them as a different kind of error. And the source of error is not just random variation, but a live human spammer working actively to defeat your filter.

Tokens

Another project I heard about after the Slashdot article was Bill Yerazunis' CRM114 [5]. This is the counterexample to the design principle I just mentioned. It's a straight text classifier, but such a stunningly effective one that it manages to filter spam almost perfectly without even knowing that's what it's doing.

Once I understood how CRM114 worked, it seemed inevitable that I would eventually have to move from filtering based on single words to an approach like this. But first, I thought, I'll see how far I can get with single words. And the answer is, surprisingly far.

Mostly I've been working on smarter tokenization. On current spam, I've been able to achieve filtering rates that approach CRM114's. These techniques are mostly orthogonal to Bill's; an optimal solution might incorporate both.

"A Plan for Spam" uses a very simple definition of a token. Letters, digits, dashes, apostrophes, and dollar signs are constituent characters, and everything else is a token separator. I also ignored case.

Now I have a more complicated definition of a token:

1. Case is preserved.
2. Exclamation points are constituent characters.
3. Periods and commas are constituents if they occur between two digits. This lets me get ip addresses and prices intact.
4. A price range like $20-25 yields two tokens, $20 and $25.

5. Tokens that occur within the To, From, Subject, and Return-Path lines, or within urls, get marked accordingly. E.g. "foo" in the Subject line becomes "Subject*foo". (The asterisk could be any character you don't allow as a constituent.)

Such measures increase the filter's vocabulary, which makes it more discriminating. For example, in the current filter, "free" in the Subject line has a spam probability of 98%, whereas the same token in the body has a spam probability of only 65%.

Here are some of the current probabilities [6]:

```
Subject*FREE        0.9999
free!!              0.9999
To*free             0.9998
Subject*free        0.9782
free!               0.9199
Free                0.9198
Url*free            0.9091
FREE                0.8747
From*free           0.7636
free                0.6546
```

In the Plan for Spam filter, all these tokens would have had the same probability, .7602. That filter recognized about 23,000 tokens. The current one recognizes about 187,000.

The disadvantage of having a larger universe of tokens is that there is more chance of misses. Spreading your corpus out over more tokens has the same effect as making it smaller. If you consider exclamation points as constituents, for example, then you could end up not having a spam probability for free with seven exclamation points, even though you know that free with just two exclamation points has a probability of 99.99%.

One solution to this is what I call degeneration. If you can't find an exact match for a token, treat it as if it were a less specific version. I consider terminal exclamation points, uppercase letters, and occurring in one of the five marked contexts as making a token more specific. For example, if I don't find a probability for "Subject*free!", I look for probabilities for "Subject*free", "free!", and "free", and take whichever one is farthest from .5.

Here are the alternatives [7] considered if the filter sees "FREE!!!" in the Subject line and doesn't have a probability for it.

```
Subject*Free!!!
Subject*free!!!
Subject*FREE!
Subject*Free!
Subject*free!
Subject*FREE
Subject*Free
Subject*free
FREE!!!
Free!!!
free!!!
FREE!
Free!
free!
FREE
Free
free
```

If you do this, be sure to consider versions with initial caps as well as all uppercase and all lowercase. Spams tend to have more sentences in imperative mood, and in those the first word is a verb. So verbs with initial caps have higher spam probabilities than they would in all lowercase. In my filter, the spam probability of "Act" is 98% and for "act" only 62%.

If you increase your filter's vocabulary, you can end up counting the same word multiple times, according to your old definition of "same". Logically, they're not the same token anymore. But if this still bothers you, let me add from experience that the words you seem to be counting multiple times tend to be exactly the ones you'd want to.

Another effect of a larger vocabulary is that when you look at an incoming mail you find more interesting tokens, meaning those with probabilities far from .5. I use the 15 most interesting to decide if mail is spam. But you can run into a problem when you use a fixed number like this. If you find a lot of maximally interesting tokens, the result can end up being decided by whatever random factor determines the ordering of equally interesting tokens. One way to deal with this is to treat some as more interesting than others.

For example, the token "dalco" occurs 3 times in my spam corpus and never in my legitimate corpus. The token "Url*optmails" (meaning "optmails" within a url) occurs 1223 times. And yet, as I used to calculate probabilities for tokens, both would have the same spam probability, the threshold of .99.

That doesn't feel right. There are theoretical arguments for giving these two tokens substantially different probabilities (Pantel and Lin do), but I haven't tried that yet. It does seem at least that if we find more than 15 tokens that only occur in one corpus or the other, we ought to give priority to the ones that occur a lot. So now there are two threshold values. For tokens that occur only in the spam corpus, the probability is .9999 if they occur more than 10 times and .9998 otherwise. Ditto at the other end of the scale for tokens found only in the legitimate corpus.

I may later scale token probabilities substantially, but this tiny amount of scaling at least ensures that tokens get sorted the right way.

Another possibility would be to consider not just 15 tokens, but all the tokens over a certain threshold of interestingness. Steven Hauser does this in his statistical spam filter [8]. If you use a threshold, make it very high, or spammers could spoof you by packing messages with more innocent words.

Finally, what should one do about html? I've tried the whole spectrum of options, from ignoring it to parsing it all. Ignoring html is a bad idea, because it's full of useful spam signs. But if you parse it all, your filter might degenerate into a mere html recognizer. The most effective approach seems to be the middle course, to notice some tokens but not others. I look at a, img, and font tags, and ignore the rest. Links and images you should certainly look at, because they contain urls.

I could probably be smarter about dealing with html, but I don't think it's worth putting a lot of time into this. Spams full of html are easy to filter. The smarter spammers already avoid it. So performance in the future should not depend much on how you deal with html.

Performance

Between December 10 2002 and January 10 2003 I got about 1750 spams. Of these, 4 got through. That's a filtering rate of about 99.75%.

Two of the four spams I missed got through because they happened to use words that occur often in my legitimate email.

The third was one of those that exploit an insecure cgi script to send mail to third parties. They're hard to filter based just on the content because the headers are innocent and they're careful about the words they use. Even so I can usually catch them. This one squeaked by with a probability of .88, just under the threshold of .9.

Of course, looking at multiple token sequences would catch it easily. "Below is the result of your feedback form" is an instant giveaway.

The fourth spam was what I call a spam-of-the-future, because this is what I expect spam to evolve into: some completely neutral text followed by a url. In this case it was was from someone saying they had finally finished their homepage and would I go look at it. (The page was of course an ad for a porn site.)

If the spammers are careful about the headers and use a fresh url, there is nothing in spam-of-the-future for filters to notice. We can of course counter by sending a crawler to look at the page. But that might not be necessary. The response rate for spam-of-the-future must be low, or everyone would be doing it. If it's low enough, it won't pay for spammers to send it, and we won't have to work too hard on filtering it.

Now for the really shocking news: during that same one-month period I got three false positives.

In a way it's a relief to get some false positives. When I wrote "A Plan for Spam" I hadn't had any, and I didn't know what they'd be like. Now that I've had a few, I'm relieved to find they're not as bad as I feared. False positives yielded by statistical filters turn out to be mails that sound a lot like spam, and these tend to be the ones you would least mind missing [9].

Two of the false positives were newsletters from companies I've bought things from. I never asked to receive them, so arguably they were spams, but I count them as false positives because I hadn't been deleting them as spams before. The reason the filters caught them was that both companies in January switched to commercial email senders instead of sending the mails from their own servers, and both the headers and the bodies became much spammier.

The third false positive was a bad one, though. It was from someone in Egypt and written in all uppercase. This was a direct result of making tokens case sensitive; the Plan for Spam filter wouldn't have caught it.

It's hard to say what the overall false positive rate is, because we're up in the noise, statistically. Anyone who has worked on filters (at least, effective filters) will be aware of this problem. With some emails it's hard to say whether they're spam or not, and these are the ones you end up looking at when you get filters really tight. For example, so far the filter has caught two

emails that were sent to my address because of a typo, and one sent to me in the belief that I was someone else. Arguably, these are neither my spam nor my nonspam mail.

Another false positive was from a vice president at Virtumundo. I wrote to them pretending to be a customer, and since the reply came back through Virtumundo's mail servers it had the most incriminating headers imaginable. Arguably this isn't a real false positive either, but a sort of Heisenberg uncertainty effect: I only got it because I was writing about spam filtering.

Not counting these, I've had a total of five false positives so far, out of about 7740 legitimate emails, a rate of .06%. The other two were a notice that something I bought was back-ordered, and a party reminder from Evite.

I don't think this number can be trusted, partly because the sample is so small, and partly because I think I can fix the filter not to catch some of these.

False positives seem to me a different kind of error from false negatives. Filtering rate is a measure of performance. False positives I consider more like bugs. I approach improving the filtering rate as optimization, and decreasing false positives as debugging.

So these five false positives are my bug list. For example, the mail from Egypt got nailed because the uppercase text made it look to the filter like a Nigerian spam. This really is kind of a bug. As with html, the email being all uppercase is really conceptually one feature, not one for each word. I need to handle case in a more sophisticated way.

So what to make of this .06%? Not much, I think. You could treat it as an upper bound, bearing in mind the small sample size. But at this stage it is more a measure of the bugs in my implementation than some intrinsic false positive rate of Bayesian filtering.

Future

What next? Filtering is an optimization problem, and the key to optimization is profiling. Don't try to guess where your code is slow, because you'll guess wrong. Look at where your code is slow, and fix that. In filtering, this translates to: look at the spams you miss, and figure out what you could have done to catch them.

For example, spammers are now working aggressively to evade filters, and one of the things they're doing is breaking up and misspelling words to prevent filters from recognizing them. But working on this is not my first priority, because I still have no trouble catching these spams [10].

There are two kinds of spams I currently do have trouble with. One is the type that pretends to be an email from a woman inviting you to go chat with her or see her profile on a dating site. These get through because they're the one type of sales pitch you can make without using sales talk. They use the same vocabulary as ordinary email.

The other kind of spams I have trouble filtering are those from companies in e.g. Bulgaria offering contract programming services. These get through because I'm a programmer too, and the spams are full of the same words as my real mail.

I'll probably focus on the personal ad type first. I think if I look closer I'll be able to find statistical differences

between these and my real mail. The style of writing is certainly different, though it may take multiword filtering to catch that. Also, I notice they tend to repeat the url, and someone including a url in a legitimate mail wouldn't do that [11].

The outsourcing type are going to be hard to catch. Even if you sent a crawler to the site, you wouldn't find a smoking statistical gun. Maybe the only answer is a central list of domains advertised in spams [12]. But there can't be that many of this type of mail. If the only spams left were unsolicited offers of contract programming services from Bulgaria, we could all probably move on to working on something else.

Will statistical filtering actually get us to that point? I don't know. Right now, for me personally, spam is not a problem. But spammers haven't yet made a serious effort to spoof statistical filters. What will happen when they do?

I'm not optimistic about filters that work at the network level [13]. When there is a static obstacle worth getting past, spammers are pretty efficient at getting past it. There is already a company called Assurance Systems that will run your mail through Spamassassin and tell you whether it will get filtered out.

Network-level filters won't be completely useless. They may be enough to kill all the "opt-in" spam, meaning spam from companies like Virtumundo and Equalamail who claim that they're really running opt-in lists. You can filter those based just on the headers, no matter what they say in the body. But anyone willing to falsify headers or use open relays, presumably including most porn spammers, should be able to get some message past network-level filters if they want to. (By no means the message they'd like to send though, which is something.)

The kind of filters I'm optimistic about are ones that calculate probabilities based on each individual user's mail. These can be much more effective, not only in avoiding false positives, but in filtering too: for example, finding the recipient's email address base-64 encoded anywhere in a message is a very good spam indicator.

But the real advantage of individual filters is that they'll all be different. If everyone's filters have different probabilities, it will make the spammers' optimization loop, what programmers would call their edit-compile-test cycle, appallingly slow. Instead of just tweaking a spam till it gets through a copy of some filter they have on their desktop, they'll have to do a test mailing for each tweak. It would be like programming in a language without an interactive toplevel, and I wouldn't wish that on anyone.

Notes

[1] Paul Graham. "A Plan for Spam." August 2002. http://paulgraham.com/spam.html.

Probabilities in this algorithm are calculated using a degenerate case of Bayes' Rule. There are two simplifying

assumptions: that the probabilities of features (i.e. words) are independent, and that we know nothing about the prior probability of an email being spam.

The first assumption is widespread in text classification. Algorithms that use it are called "naive Bayesian."

The second assumption I made because the proportion of spam in my incoming mail fluctuated so much from day to day (indeed, from hour to hour) that the overall prior ratio seemed worthless as a predictor. If you assume that P(spam) and P(nonspam) are both .5, they cancel out and you can remove them from the formula.

If you were doing Bayesian filtering in a situation where the ratio of spam to nonspam was consistently very high or (especially) very low, you could probably improve filter performance by incorporating prior probabilities. To do this right you'd have to track ratios by time of day, because spam and legitimate mail volume both have distinct daily patterns.

[2] Patrick Pantel and Dekang Lin. "SpamCop-- A Spam Classification & Organization Program." Proceedings of AAAI-98 Workshop on Learning for Text Categorization.

[3] Mehran Sahami, Susan Dumais, David Heckerman and Eric Horvitz. "A Bayesian Approach to Filtering Junk E-Mail." Proceedings of AAAI-98 Workshop on Learning for Text Categorization.

[4] At the time I had zero false positives out of about 4,000 legitimate emails. If the next legitimate email was a false positive, this would give us .03%. These false positive rates are untrustworthy, as I explain later. I quote a number here only to emphasize that whatever the false positive rate is, it is less than 1.16%.

[5] Bill Yerazunis. "Sparse Binary Polynomial Hash Message Filtering and The CRM114 Discriminator." Proceedings of 2003 Spam Conference.

[6] In "A Plan for Spam" I used thresholds of .99 and .01. It seems justifiable to use thresholds proportionate to the size of the corpora. Since I now have on the order of 10,000 of each type of mail, I use .9999 and .0001.

[7] There is a flaw here I should probably fix. Currently, when "Subject*foo" degenerates to just "foo", what that means is you're getting the stats for occurrences of "foo" in the body or header lines other than those I mark. What I should do is keep track of statistics for "foo" overall as well as specific versions, and degenerate from "Subject*foo" not to "foo" but to "Anywhere*foo". Ditto for case: I should degenerate from uppercase to any-case, not lowercase.

It would probably be a win to do this with prices too, e.g. to degenerate from "$129.99" to "$--9.99", "$--.99", and "$--".

You could also degenerate from words to their stems, but this would probably only improve filtering rates early on when you had small corpora.

[8] Steven Hauser. "Statistical Spam Filter Works for Me." http://www.sofbot.com.

[9] False positives are not all equal, and we should remember this when comparing techniques for stopping spam. Whereas many of the false positives caused by filters will be

near-spams that you wouldn't mind missing, false positives caused by blacklists, for example, will be just mail from people who chose the wrong ISP. In both cases you catch mail that's near spam, but for blacklists nearness is physical, and for filters it's textual.

In fairness, it should be added that the new generation of responsible blacklists, like the SBL, cause far fewer false positives than earlier blacklists like the MAPS RBL, for whom causing large numbers of false positives was a deliberate technique to get the attention of ISPs.

[10] If spammers get good enough at obscuring tokens for this to be a problem, we can respond by simply removing whitespace, periods, commas, etc. and using a dictionary to pick the words out of the resulting sequence. And of course finding words this way that weren't visible in the original text would in itself be evidence of spam.

Picking out the words won't be trivial. It will require more than just reconstructing word boundaries; spammers both add ("xHot nPorn cSite") and omit ("P#rn") letters. Vision research may be useful here, since human vision is the limit that such tricks will approach.

[11] In general, spams are more repetitive than regular email. They want to pound that message home. I currently don't allow duplicates in the top 15 tokens, because you could get a false positive if the sender happens to use some bad word multiple times. (In my current filter, "dick" has a spam probabilty of .9999, but it's also a name.) It seems we should at least notice duplication though, so I may try allowing up to two of each token, as Brian Burton does in SpamProbe.

[12] This is what approaches like Brightmail's will degenerate into once spammers are pushed into using mad-lib techniques to generate everything else in the message.

[13] It's sometimes argued that we should be working on filtering at the network level, because it is more efficient. What people usually mean when they say this is: we currently filter at the network level, and we don't want to start over from scratch. But you can't dictate the problem to fit your solution.

Historically, scarce-resource arguments have been the losing side in debates about software design. People only tend to use them to justify choices (inaction in particular) made for other reasons.

Thanks to Sarah Harlin, Trevor Blackwell, and Dan Giffin for reading drafts of this paper, and to Dan again for most of the infrastructure that this filter runs on.

Related:

- A Plan for Spam
- Plan for Spam FAQ
- 2003 Spam Conference Proceedings
- Japanese Translation
- Chinese Translation