A Plan for Spam

Like to build things? Try Hacker News.

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 (&lt; (+ 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 thecombined 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.