Better Bayesian Filtering

paulgraham.com · Paul Graham · 2003-01 · source

|
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:

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%. Case is preserved.

Exclamation points are constituent characters.

Periods and commas are constituents if they occur
between two digits. This lets me get ip addresses
and prices intact.

A price range like $20-25 yields two tokens,
$20 and $25.

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.)

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.

[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: