Introduction to Bayesian Filtering

Bayesian Filtering Defined

Just a few short years ago, Bayes' Formula was found mostly in university-level statistics textbooks. This whitepaper gives a basic explanation of what Bayes' Formula is, and how it can be used to keep spam out of your Inbox. In its simplified form, Bayes' Formula is:

$$P(E_j|F) = \frac{P(F|E_j)\,P(E_j)}{\sum P(F|E_i)\,P(E_i)}$$

So what does that formula do in simple terms? It lets us combine the probability of multiple independent events into one number. For example, suppose I know that when my car makes a loud clicking sound, there's a 75% chance it's going to break down. I also know that if it's more than 80 degrees Fahrenheit outside, my car only has a 15% chance of breaking down. If it's more than 80 degrees outside and my car is making a loud clicking sound, I can use Bayes' Formula to figure out how likely it is that I'm going to be walking to work that morning (34.6%, if you're curious). Since most people get rid of a car when it breaks down as much as mine does (rather than figuring out how often it's going to break down), we're going to use the formula to figure out the probability that a message is spam based on the words appearing in it.

Our goal is to have our filter:

1. List every word in an incoming mail message
2. Determine the odds of each word appearing in a spam message, and
3. Use those odds as input to Bayes' Formula to determine if the message is spam or not.

The first thing we need to do is teach our filter the difference between spam and non-spam messages. When we humans accidentally read a spam message, we almost immediately recognize it as spam because of certain key words (such as "viagra" and "mortgage") or phrases (such as "Get your free porn here!"). Instinctively, we know that a message containing these words or phrases is spam because of our experience in dealing with junk mail. The opposite is true as well – we can almost instantly look at a message from our mother (containing phrases such as "When are you going to get married so I can have grandchildren?") or our boss (containing phrases such as "Less computer solitaire and more work if you want a paycheck this Friday") and know they're not spam.

Our filter doesn't have the benefit of our years of experience, so we have to teach it what spam messages look like, and how they differ from non-spam messages. We "train" the filter by showing it a bunch of mail messages, and telling it whether the message is spam. Whenever we show a message to the filter, it finds every word in the message and stores it (along with how many times it occurred) in a database.

Separate databases are kept for spam and non-spam messages. The filter uses a looser definition of a word than humans do – a word (more properly called a token) can also be an IP address, a host name, an HTML tag, or a price (such as “$99.99”). The things a token can’t be are random strings, words less than three characters long, and numbers.

As a simple example, let’s train the filter on these three very short spam messages:

```
Get Viagra here without a prescription.
Click here to get your free porn!
Free mortgage consultations available now.
```

Our spam token database should now look a little like this:

```
# PreciseMail Token Database Msg Count: 3
available 1
click 1
consultations 1
free 2
get 2
here 2
mortgage 1
now 1
porn 1
prescription 1
viagra 1
without 1
your 1
```

Now let’s train it on these three short non-spam messages:

```
Important meeting today at noon.
When is the next time you’re coming home to visit?
Let’s all meet at the diner for breakfast.
```

After we do that, our non-spam token database should look a little like this:

```
# PreciseMail Token Database Msg Count: 3
all 1
breakfast 1
coming 1
diner 1
home 1
important 1
let’s 1
meet 1
meeting 1
next 1
noon 1
```

```
time 1
today 1
visit 1
when 1
you’re 1
```

At this point, the filter is very minimally trained – using it to filter spam would be like letting a 3-year old drive on a major highway. (In real life, our filter is smart enough to know when it’s been sufficiently trained.) It’s important that you train the filter with approximately equal amounts of spam and non-spam messages, as we’re doing in this example. Many people (including several software vendors that really should know better) tend to train their Bayesian filter only on spam messages. There’s an old saying that goes something like: “If the only tool you have is a hammer, every problem looks like a nail.” With a Bayesian filter, if it’s only been trained with spam messages, every message looks like spam.

It’s also important that the Bayesian filter be trained on spam and non-spam messages from your site, and your site only. If a Bayesian filter is pre-trained on messages from another site, it won’t be able to identify features specific to messages destined for your site. This can easily lead to large numbers of false positives and a low spam detection accuracy. Even with this knowledge, many people choose to use pre-trained products because they don’t want to take the time and effort to properly train their own Bayesian filter. To counter this, several new products on the market, including Process Software’s PreciseMail Anti-Spam Gateway, can automatically train the built-in Bayesian filter on spam and non-spam messages. This lets the systems administrator “install and forget” the Bayesian component of the spam filtering system.

Now we’re going to let the filter try to decide if a message is spam or not, based on what we’ve told it about what spam looks like and how it differs from non-spam. Let’s pretend a mail message is sent to the filter, which looks like this:

```
Hi,
Just a reminder: don’t forget your allergy prescription when you visit New York
City today.

Mom
```

The filter scans through the message, creating a list of every word it knows about (in other words, every word in the message that’s also in the token databases). In this example, the words it knows about are “prescription”, “when”, “today”, “visit”, and “your”. Once the filter has the list of words it knows about, for each word it calculates the probability that the word appears in spam based on the frequency data in the token databases.

This probability value assigned to each word is commonly referred to as spamicity, and ranges from 0.0 to 1.0. A spamicity value greater than 0.5 means that a message containing the word is likely to be spam, while a spamicity value less than 0.5 indicates that a message containing the word is likely to be ham. A spamicity value of 0.5 is neutral, meaning that it has no effect on the decision as to whether a message is spam or not.

In simplest terms, the spamicity is based on the number of times a word occurs in spam messages as opposed to the number of times it occurs in non-spam messages. For example, if a word has occurred 50 times in spam messages but only 2 times in non-spam messages, a message that contains it has a good chance of being spam. The opposite is true as well – if a word appears 50 times in non-spam messages but only 2 times in spam messages, a message that contains it isn't very likely to be spam. Common words, such as "for" and "what", tend to occur about equally often in both spam and non-spam messages, so they have neutral spamicities.

Back to our example: after looking at the token databases, our filter comes up with the following spamicities for each of the known words in the incoming message:

```
prescription      0.990000
today             0.010000
when              0.010000
visit             0.010000
your              0.990000
```

Now that we have a spamicity value for each word, the filter is going to use Bayes' Formula to combine them all together to get an overall spamicity for the whole message. In the case of our example, the overall spamicity is 0.00. Since this is certainly less than 0.5, the filter has decided that the message isn't spam, and allows the mail server to deliver it to the intended recipient.

A key point to take away from this example is that a couple words with a high spamicity don't automatically doom an incoming message to be branded as spam – there have to be enough words with a high spamicity to outweigh the words with a low spamicity. This means that a message from your spouse discussing taking out a second mortgage on the house to finance some kitchen remodeling will most likely make it through, as will a message from your best buddy talking about how happy he is now that he's starting taking Viagra.

To make sure we don't accidentally mark a non-spam message as spam, we double the weight of non-spam words in our calculations. This means that it takes two words with a high spamicity to outweigh one word with a low spamicity. Doing this doesn't have much of an effect on the filter's spam detection accuracy (spam messages tend to contain lots of words with high spamicities), but it does significantly lower the false positive rate.

Bayesian Filtering Examples

Now that we have a basic understanding of how Bayesian filtering works, I'm going to use my personal email account for some real-world examples of what Bayesian filtering looks like. The token databases used in the below examples have been trained with around a thousand spam messages and a thousand non-spam messages.

The first example below is the output from running the Bayesian filter on a message from a friend who just moved to the Los Angeles area. Most of the message is complaints about smog and traffic, followed by his new contact information.

```
SPAMICITY:         0.002844
final              0.378383
leave              0.327315
area               0.683825
New                0.740312
info               0.786698
here               0.786764
I'll               0.201614
contact            0.846948
his                0.148226
left               0.853779
transit            0.121959
maybe              0.112914
ext                0.978168
capped             0.010000
smog               0.010000
```

This particular message has a mix of spam and non-spam words, but the non-spam words outweigh the spam words.

You've probably noticed that the output from the filter only contains 15 words – my friend hates typing, but he doesn't hate it so much he only uses 15 words per message. Our Bayesian filter uses only the 15 most "interesting" words to calculate the message's overall spamicity. These 15 words are the words in the message that have either the highest or lowest spamicity (i.e. are closest to 0 or 1 in value). We ignore these other words in the message because they don't have any real impact over and above the 15 most "interesting" words. This keeps the Bayesian filter running quickly, even if somebody feels the urge to send you the complete text of the first act of Hamlet.

The next example output is from a spam message so obscene it would make anyone blush. And that doesn't even cover the photos that the spammer thoughtfully included. This was one of those messages that you delete as quickly as possible so someone walking by doesn't see it on your screen and report you to Human Resources.

```
SPAMICITY:         1.000000
great              0.649814
64.119.221.136     0.990000
Thank              0.744272
our                0.766728
you'll             0.819132
understanding      0.102168
offer              0.948124
```

```
SIZE                0.957696
privacy             0.989979
enjoiy              0.990000
teenies             0.990000
e.gif               0.990000
FF0000              0.990000
TeenParadiso        0.990000
TEEN                0.990000
```

This example demonstrates some interesting features of Bayesian filtering. Since we look at everything in the message, things like IP addresses, HTML attributes, and even image file names are included in the Bayesian analysis. Interesting things of note in this message are:

- The misspelled word "enjoiy" has a very high spamicity. As a whole, spammers are shockingly bad spellers so misspelled words tend to have high spamicities. (If you're a really lousy speller, this might be a good time to get in the habit of using a spell checker.)
- `FF0000` is the HTML attribute for the color red. The human eye is unconsciously drawn to the color red, a fact of which spammers are well aware. Most people never use red text in an email message, but spammers use the color more often than sports car manufacturers.
- The IP address 64.119.221.136 is referenced by the message. This particular address is used by a spammer that sends several messages a day in my direction, so it ends up being a handy way to quickly identify a spam message. Even if the spammer sends me a message that contains no words with a high spamicity, the IP address will give him away.
- An external image file is referenced (`e.gif`). Embedded links to images on a web server are almost never found in non-spam email, so they quickly become a way for the Bayesian filter to differentiate spam messages.

It's important to remember that the Bayesian filter adapts itself to your site. For example, just because I never receive non-spam mail containing red text doesn't mean everyone doesn't. If red text is common at your site, then the Bayesian filter will learn that and not use it as a way to identify spam.

The final example is the output from running the Bayesian filter against an email from another friend, who's a medical student. She's training to be a forensic pathologist, and likes sending me email describing horrible medical conditions in an attempt to make me nauseous. As a result, mail from her tends to contain mostly medical terminology that the filter has never seen before. As an example, here's the output from running the filter against an email describing how the Ebola virus destroys your internal organs:

```
SPAMICITY:          0.000000
Antigen             0.400000
aerosols            0.400000
re-insertion        0.400000
Nosocomial          0.400000
virus               0.647605
PCR                 0.333311
```

```
isolation        0.080738
reaction         0.070790
polymerase       0.010000
health-care      0.010000
secretions       0.010000
hypothesized     0.010000
amplification    0.010000
primates         0.010000
chimpanzees      0.010000
```

This output from the filter contains medical terms that most people have probably never heard before, even if they religiously watch medical dramas on television. Most people don't casually drop words like "antigen" and "nosocomial" in their conversations, and they don't use them in email messages. As a result, the Bayesian filter has no idea what to do with the word. This particular message presents a unique problem to a Bayesian filter – it's almost nothing but medical terms and very common words that have neutral spamicities.

Since the Bayesian filter doesn't have any training data available for these words, it assigns them a default spamicity of 0.4. This value is slightly non-spam, since it's unusual for a spam message to contain a significant number of words that haven't appeared in previous spam messages. If a spammer uses uncommon words, he's going to confuse his target audience (namely those who respond to spam messages).

As we've seen, when properly used a Bayesian filter can be an extraordinarily accurate way to identify and discard spam messages. Large-scale spammers are starting to learn how effective it is the hard way, and they're busily working on ways to circumvent it.

The first circumvention technique spammers tried was a modification of the one that rendered checksum-based message signature filtering useless: including strings of random text in the message. The theory was that enough of these random character strings would increase the size of the token databases to the point where they made the filter so slow that it was useless. In response, smart developers of anti-spam filters (i.e. the "Good Guys") implemented a system that removed infrequently used tokens from the token database. Even though all of the random strings from spam messages would be placed in the token database, they would be removed as soon as it was obvious that they didn't appear in other messages.

The current circumvention technique the spammers are trying is to include obviously non-spam words in their spam messages. For example, last week a spammer sent me an advertisement for "penile growth supplements" that included the words "congresswoman" and "soybean" in the subject and message body. The Bayesian filter promptly marked it as spam since the words "penile" and "supplements" pretty much condemn any message sent to my account as spam, but the spammer had two goals in sending me this message.

The first (and obvious) goal was to see if the spammer could sneak the message past the Bayesian filter by including obviously non-spam words. In this, he failed. The secondary goal was to try to get the filter to start recognizing the words “congresswoman” and “soybean” as words with a high spamicity. If spammers can get the filter to assign a high spamicity to enough words that commonly appear in non-spam messages, they can render the filter useless.

Luckily, while spammers may be smart enough to come up with these circumvention ideas, they’re not very good at coordinating their efforts. As long as every spammer that sends me mail doesn’t start including the words “congresswoman” and “soybean” in their spam messages, they won’t appear often enough in spam to significantly increase their spamicities. (In my case, I’ve never received a mail message containing the words “congresswoman” or “soybean”, so the filter ignored them altogether.)

This circumvention method also has limited utility for another good old-fashioned marketing reason. If spammers start including a large pile of legitimate text (such as an article from the CNN website) in each message, they’d confuse their target audience. A spam message advertising penile growth supplements that also contains an article about the relative value of the Euro is going to confuse the audience so much they just ignore the message.

Summary

While other filtering methods may be as effective for now, Bayesian filtering includes an element of changeability that makes it very difficult for spammers to circumvent. Savvy spammers with a lot of resources try to buy every anti-spam filtering product they can get their hands on, and test their spam messages against it so they can be sure the spam will get through to the recipient. Since every site with a Bayesian filter installed has a different set of tokens in their database, it makes it impossible for the spammer to deliberately craft a message that will bypass a site’s Bayesian filter. Because the Bayesian filter is constantly being updated, every sneaky change in message wording made by spammers will be quickly identified and rendered ineffective. This makes Bayesian filtering the bane of every spammer, and something every site serious about stopping spam should have.

About PreciseMail Anti-Spam Gateway

PreciseMail Anti-Spam Gateway is an enterprise software solution that eliminates spam, phishing and virus threats at the Internet gateway or mail server. It has a proven 98% spam detection accuracy rate out-of-the-box without filtering legitimate messages. PreciseMail Anti-Spam Gateway has a highly sophisticated filtering engine is based on a combination of proven heuristic, DNS blacklisting, and Bayesian artificial intelligence technologies, which automatically learn how to separate spam messages from legitimate email. As a result, PreciseMail Anti-Spam Gateway can determine whether email is spam instead of passively reacting to known spammers by creating rules that block them after a spam attack occurs.

About Process Software

Process Software has been a premier supplier of communications software solutions to mission critical environments for twenty years. We were early innovators of email software and anti-spam technology. Process Software has a proven track record of success with thousands of customers, including many Global 2000 and Fortune 1000 companies.

PROCESS™ SOFTWARE
U.S.A.: (800) 722-7770 • International: (508 879-6994 • Fax: (508) 879-0042
E-mail: info@process.com • Web: http://www.process.com/