Fighting spam with statistics

Spam—unsolicited commercial e-mail—is a complex and growing problem, and threatens to derail the internet revolution. **Joshua Goodman** and **David Heckerman** of Microsoft Research describe some statistics-based methods for blocking spam, first by distinguishing it from wanted mail, and then by constructing puzzles they propose to use to challenge suspected spammers.

Some statistics about spam

Spam is a huge problem. A few statistics can tell us just how huge.

- Brightmail has reported that over 50% of mail on the internet is now spam.
- A recent Infoworld poll identified spam as the number 1 "information technology disaster of the past year".
- From a report by the United States Federal Trade Commission[1]:
  - 66% of spam had false information somewhere in the message;
  - 18% of spam advertises "adult" material.
- From a report by the Pew Internet and American Life Project[2]:
  - 25% of e-mail users say that spam has reduced their use of e-mail;
  - 12% of users spend half an hour or more per day dealing with spam.

Statistics also help to explain why spam is such a problem.

- It costs only about 0.01 ¢ to send spam.
- 7% of email users say they have bought a product advertised in unsolicited e-mail[2].

Given the tiny costs of spamming, even a tiny response rate makes spam economically viable.

Identifying spam—harder than it looks

Around a year ago, MSN's Hotmail service proposed blocking all spam entering their system that was allegedly from Hotmail. Spammers often fake the address that their mail is from, including pretending that such mail is from Hotmail itself. Because there are no standards on the internet to say which IP addresses are allowed to send mail from which domain names, we usually cannot tell when a sender's address has been faked. (Recent proposals such as Microsoft's Caller-ID standard will help to fix this problem in the future.) The one exception today is that Hotmail knows its own valid internet addresses, and knows that mail from outside those addresses that says "From Hotmail" is fake. It sounded reasonable to block all such deceptive mail.

There are, however, a few cases where this faking may be legitimate. For instance, when your friend sends you an on-line birthday card via e-mail, the "From" address has your friend's e-mail address (perhaps at Hotmail), even though the e-mail actually originates on the greeting card company's servers, and comes from their internet address. Fortunately, we had acquired a data set of spam and good mail. We could show that, for every four spams allegedly from Hotmail that would be blocked, one good message would be lost—the exceptions were not as rare as had been thought. A 4-to-1 trade-off is not close to tolerable. This simple analysis was key to preventing an unacceptably large number of deleted good messages, including those birthday cards.

"Given the tiny costs of spamming, even a tiny response rate makes spam commercially viable"

Although there are no hard-and-fast rules for marking mail as spam—even a faked "From" address is only moderately bad—by combining different indicators together, we can often be almost certain that mail is spam, and safely delete it, or very confident that it is spam, and place it in a junk folder. The question is how to combine these indicators.

One of the most popular techniques for stopping spam is the *naïve Bayes method*[3], often mistakenly simply called *Bayesian spam filtering*. In the naïve Bayes method, we try to determine the probability that a given message is spam, or good. We start by using the Bayes rule, which in this case tells us that

$$P(\text{spam}|\text{message}) = \frac{P(\text{message} \mid \text{spam}) \times P(\text{spam})}{P(\text{message})}$$

where

$$P(\text{message}) = P(\text{message} \mid \text{spam}) \times P(\text{spam})$$
$$+ P(\text{message} \mid \text{not spam}) \times P(\text{not spam}).$$

$P(\text{spam})$ is simply the prior probability that any given message is spam. For instance, if half the messages you receive are spam, $P(\text{spam}) = \frac{1}{2}$, and similarly for $P(\text{not spam})$. Also, we need to compute $P(\text{message}|\text{spam})$ and $P(\text{message}|\text{not spam})$—these are respectively the probability of receiving any of the billions of possible spam messages and the probability of receiving any particular good piece of e-mail, of the infinite possibilities. Again this is difficult, if not impossible. However, we can make some approximations. In particular, we shall assume that the probability of every word in the message is independent of every other word, conditioned on knowing whether the message is spam. For instance, we assume that the probability of "click" is independent of the probability of "here", given that the message is spam. Clearly, this approximation is not a great one, which is why the technique is called naïve. However, once we have made this assumption, it is easy to compute the message probabilities. In particular,

$$P(\text{message} \mid \text{spam}) = P(\text{first word} \mid \text{spam})$$
$$\times P(\text{second word} \mid \text{spam})$$
$$\times \dots \times P(\text{last word} \mid \text{spam})$$

$$P(\text{message} \mid \text{not spam}) = P(\text{first word} \mid \text{not spam})$$
$$\times P(\text{second word} \mid \text{not spam})$$
$$\times \dots \times P(\text{last word} \mid \text{spam}).$$

What is the probability of a particular word given spam? For each word, e.g. "click", we simply count the number of times the word occurred in all spam messages in a "training corpus" (a set of messages known to be spam or good) and how many spam messages there were overall. The ratio is an estimate of the probability that any particular spam word is the word "click". We can do the same for words in non-spam messages.
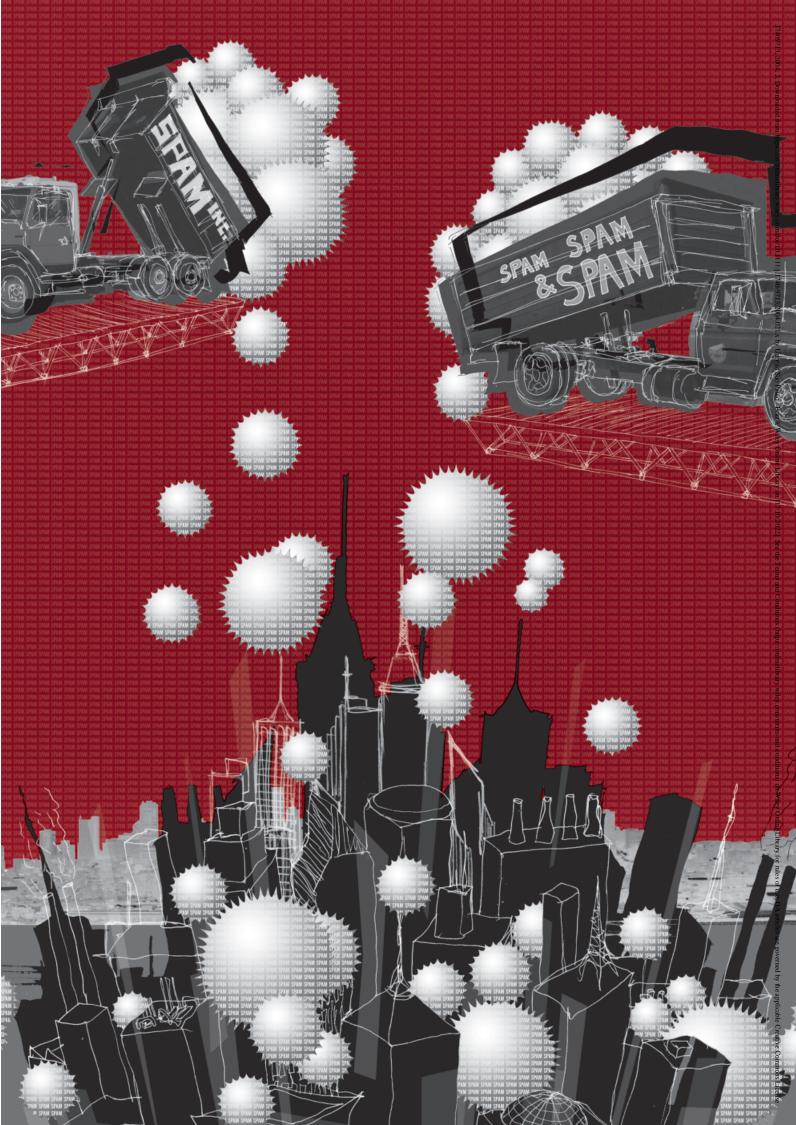
Naïve Bayes is just one of many machine learning techniques, often used because it is the easiest to implement. It can be surprisingly effective, despite the simplicity of the model. If you are willing to label carefully thousands of your messages as spam or good to train the model, it can result in excellent performance for you on your own mail. However, the independence assumption in naïve Bayes limits its effectiveness. A naïve Bayes model will tend to be confused by common spam phrases like "click here to unsubscribe" that sometimes also occur in good mail, because naïve Bayes does not model the fact that these words occur together. If each word is 10 times as likely in spam as in good mail, a naïve Bayes model will think any such message is 10 000 times more likely to be spam than good ($10 \times 10 \times 10 \times 10$)—a large overestimate of the true ratio. Other model types, such as neural networks, graphical models, logistic regression and support vector machines all make fewer assumptions and can model these kinds of relationships between words implicitly or explicitly, at the expense of more complexity. At Microsoft, we use these more complex model types.

Challenge–response systems

Probabilistic filters can do an excellent job of identifying spam. Nonetheless, like all filtering techniques, they are imperfect: they always occasionally mark some good mail as spam. At Microsoft Research, we have been exploring and adapting an idea due to Cynthia Dwork (now at Microsoft Research, Silicon Valley) and Moni Naor (at the Weizmann Institute of Science in Israel). They proposed using "computational puzzles" to stop spam[4]. We are researching a variation on this idea, in which it is used in combination with a probabilistic filter[5]. If you send me mail, my filter examines it. If it appears good, it goes in my inbox. If it looks like junk, it goes in my junk folder. To make sure that I eventually see any good mail sitting in my junk folder, my e-mail program automatically sends you a "challenge": a request to solve a computational puzzle. Your e-mail program must spend perhaps 15 seconds to solve this puzzle. It solves this puzzle automatically, in the background, without you even noticing, and then sends the puzzle solution back to my program, which verifies the solution, and moves your original message to my inbox.

If you are a legitimate sender, you are willing to spend the time to do this. But, if you are a spammer, you cannot afford to solve these puzzles—you would be able to answer fewer than 6000 puzzles per day per computer, compared with the millions of messages you can normally send.

What do these puzzles look like? One solution is to use a hash function. A hash function takes a string of letters as input and returns a number. The number is a deterministic, but nearly random, function of the input string. Slight changes in the input string result in large and unpredictable changes in the output of the function. The message recipient—the challenger—sends a challenge string based on the actual message to the message sender. The message sender must find some other string that, when put in front of the first string, results in a hash value of, say, 0. If the hash function returns a

Charles Trevelyan

How to get less spam

- Choose an e-mail address that is hard to guess. Many spammers use "dictionary attacks" where they try billions of common words separately and in combination, e.g. Bob, Bob1970, StatisticsNut, etc. Choose a longer name with more words, like BobStatisticsNut1970, and a spammer is much less likely to guess it.

- Do not give your e-mail address to a company you do not trust. Some companies sell the addresses they collect.

- Do not put your e-mail address on web pages. Many spammers "crawl" the web, looking for addresses that they then add to their lists. If you need to put your e-mail address on the web, disguise it, or put it in an image. Spammers do not use image recognition software (yet!).

number between 1 and 15 million, the sender must try about 15 million different strings to find one that results in a zero value. A typical hash function might require 1 microsecond—a millionth of a second. It would thus take, on average, 15 seconds to find a puzzle solution. Meanwhile, the recipient only needs to verify that the solution is correct, meaning running the hash function once, which can be done in just 1 microsecond.

Designing better puzzles

There is a problem with the simple puzzle described above: it has high variance. On average, the puzzle will require 15 seconds to solve, but sometimes it will require only 1 second, sometimes 30 seconds, sometimes 45 seconds, and even occasionally more than a minute. This can be solved by using a suggestion from Cynthia Dwork and Andrew Goldberg. We can use an easier puzzle, perhaps one that returns a random number between 1 and 1 million, but require the sender to find 15 solutions. This lowers the variance by about a factor of 15. Variance is reduced linearly as the number of puzzles increases (assuming that we keep the total expected time constant). If we used a huge number of puzzles, say 100, the variance would become negligible. But the size of the solution would also be large, even larger than the original message in some cases.

An alternative solution, also due to Dwork and Goldberg, is to use a hash which returns a number between, say, 0 and 1 billion and then require the sender to find, say, 15 solutions all with the same value, and with that value between 0 and 1000. For instance, the sender could find 15 solutions with a value of 0, or 15 solutions with a value of 1, etc. We can think of this as balls dropping into bins at random, and waiting for one of the bins to collect 15 balls in it. This problem turns out to have much lower variance than simply finding 15 solutions.

Conclusion

Spam is an extremely complex topic, and no single article can hope to describe all the ways to fight it. We have described here those techniques that rely most heavily on statistics, but we are exploring and pursuing many other ideas as well[6]. For example, we are currently exploring new industry standards that can help to stop spam. In particular, our challenge–response systems are expensive for very large legitimate senders. We are exploring standard ways for these senders to become certified as non-spammers.

No single solution will stop spam. New laws will be one part of the solution. Improved filters using better statistical analyses will be another part. Finally, all legitimate senders will be able to bypass any filter mistakes: large senders through certification; smaller senders through low variance computational puzzles. We anticipate a future in which the vast majority of spam is stopped, and all legitimate mail is delivered.

References

1. US Federal Trade Commission (2003) *False Claims in Spam*. Washington DC: US Federal Trade Commission.

2. Fallows, D. (2003) Spam: how it is hurting email and degrading life on the Internet. *Pew Internet and American Life Project*.

3. Sahami, M., Dumais, S., Heckerman, D. and Horvitz, E. (1998) A Bayesian approach to filtering junk e-mail. *American Association for Artificial Intelligence Workshop on Learning for Text Categorization, Madison, July 27th*.

4. Dwork, C. and Naor, M. (1993) Pricing via processing or combatting junk mail. *Lecture Notes in Computer Science*, **740**, 137–147.

5. Goodman, J. and Rounthwaite, R. (2004) SmartProof. Microsoft Research. (Available from `http://www.research.microsoft.com/~joshuago/smartproof.pdf`.)

6. Goodman, J. (2004) Spam technologies and policies. Microsoft Research. (Available from `http://www.research.microsoft.com/~joshuago/spamtech.pdf`.)

Joshua Goodman is a Researcher in the Machine Learning and Applied Statistics Group at Microsoft Research. He has been on loan to Microsoft's Anti-Spam product team since its inception. His previous work was on language modelling (predicting word sequences) and fast algorithms for logistic regression.

David Heckerman is founder and manager of the Machine Learning and Applied Statistics Group at Microsoft Research. Since 1992 he has been a Senior Researcher at Microsoft, where he has created applications including junk mail filters, data mining tools, handwriting recognition for the Tablet PC, troubleshooters in Windows and the Answer Wizard in Office. His work includes Bayesian methods for learning probabilistic graphical models from data. David received his doctorate from Stanford University in 1990 and is a Fellow of the American Association for Artificial Intelligence.