Crossroads
The ACM Student Magazine

Learning How to Tell Ham from Spam

by ***George Sakkis***

Introduction

Unsolicited Commercial Email (UCE), commonly known as spam, has evolved from a mere nuisance to a multi-billion dollar problem. The near zero cost of acquiring huge lists of email addresses and flooding them with advertising messages has affected individual users, small companies, and large institutions alike. Spam is forcing users to wade through their mailbox to find the relatively few emails specifically addressed to them, colloquially referred to as "ham." Spam also wastes bandwidth, intermediate storage space, CPU time of Internet Service Providers (ISPs), and is often insulting and unsuitable (e.g. having pornographic content), especially to minors. Indicative of spam's extent is the first Conference on Email and Anti-Spam (CEAS) that recently took place in Mountain View, CA, following several industrial conferences on the topic. The conference attracted over two hundred academics and practitioners, most of them actively working on the problem from different perspectives, such as machine learning, security, and law.

Various counter-measures to spam have been proposed, ranging from regulatory to technical. So far, the legislative measures adopted in the United States and Europe have not brought the desired results. On the other hand, **anti-spam filters**, software tools that try to distinguish spam from legitimate mails automatically, have been far more successful. This article explores learning-based anti-spam filtering. Before describing what a learning-based filter is and how one works, traditional filtering methods will be reviewed to illustrate the convenience of learning-based filters.

Traditional Spam Filtering

The simplest filtering techniques are **black-listing** and **white-listing**. A black-list is a list of email addresses and domains that are considered to be used by spammers. All incoming messages from an address or domain that appears in the black-list are automatically considered spam without further processing. Similarly, a white-list is a list of trusted correspondents and domains so any incoming message from the list is automatically accepted. Most email client programs allow manual updates while some also connect to one or more **Real-time Black-Lists (RBL)** such as the Spamhaus block list for automatic updates [12].

Another popular way to filter spam is through handcrafted keyword-based rules. These rules attempt to identify spam by looking for specific words or phrases in the message's subject or body (e.g., "win cash") or other suspicious patterns in the header fields (e.g. an empty "To:" field). Modern email clients make such rules easy to write and share, even for novice users.

Although white-lists, black-lists, and handcrafted rules can filter out a great deal of spam, they are far from being the perfect solution. White-lists are ideal if email is used for closed-world communication. However, this negates one of the primary features of email, the ease of communication among people that do not know each other in advance, and is certainly not applicable for activities such as e-commerce. On the black-list side, spammers typically use fake sender addresses; therefore, static black-lists are of no practical value. Real-time black-lists of IP numbers that have or could be compromised (e.g., open SMTP relays) are more effective, but are still inadequate as new hosts are constantly exploited by spammers. Moreover, black-lists can victimize innocent users or even entire domains when their addresses or IP numbers are taken over and used illegally by spammers.

Handcrafted rules have their own deficiencies as well. Firstly, rules that are publicly available across users (e.g., published on an online database) can be exploited by spammers, who can always adjust their messages to avoid triggering the rules. Devising private rules that catch as much spam as possible while letting through all legitimate messages is a tedious and error-prone task. Even worse, the characteristics of spam change over time and most users cannot afford to constantly update their rules to adapt to the spammers' new tricks.

Why A Learning Spam Filter?

In contrast to the approaches above, a **learning-based** (or **content-based**) **spam filter** is a program that tries to *learn* what is considered spam by a given user. The concept of spam is not well-defined, and although a large percentage of spam is commercial in nature, this is not always the case (e.g., hoax and chain letters). Furthermore, not every message with commercial content is spam. Thus, one of the strong points of learning-based filters is that they are adapted to each individual user or organization.

A second advantage of learning-based filters is that they are in general easier to configure and use than handcrafted rule filters. The learning and training process is fairly unobtrusive. Initially, the user has only to feed the filter with two separate sample collections of incoming messages, one for legitimate and one for spam; then the filter can be retrained on misclassified emails as the characteristics of spam change over time.

Most importantly, the accuracy of learning-based filters is astonishing. Catching 95% of incoming spam is not unusual, and some state-of-the-art-filters, such as SpamBayes [11] and CRM114 [2] boast, according to user reports, a 99% or higher spam blocking rate with almost no blocked legitimate emails.

Detecting Spam Using Probabilities

Learning filters bring up many questions: How is it possible for a program to learn the difference between ham from spam? What exactly are the input, the output, and the parameters of the learning process? What does "learning" really mean for a program, and how does it relate to human or animal learning? In fact, these are some of the key questions of **Machine Learning** [7], a multidisciplinary field that brings together areas as diverse as theory of algorithms, artificial intelligence, cognitive psychology, and operations research with applications ranging from computer games and fraud detection systems to medical assistants and auto-driven vehicles. What will be outlined here is a particular type of machine learning, the engine behind practically all current learning-based spam filters.

The core principle behind most contemporary learning-based filters is **Bayes' theorem** (or **Bayes' rule**), named after the Reverend Thomas Bayes who first expressed it almost three centuries ago. It lead to the birth of Bayesian inferencing in the late 20th century, with far-reaching impacts on many scientific fields, from applied mathematics to sociology and philosophy. The general form of Bayes' theorem gives the conditional probability distribution of a random variable *A* given a second random variable *B* in terms of the conditional probability distribution of *B* given *A* and the marginal (also called prior) probability distributions of A and B alone:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} \qquad (1)$$

In the context of spam filtering, Bayes' theorem takes a more intuitive form:

$$P(class \mid message) = \frac{P(message \mid class) \cdot P(class)}{P(message)}, class \in \{spam, ham\} \qquad (2)$$

Each term in the theorem has a conventional name:

- The term *P(class | message)* is called the **posterior probability** of a message belonging to a specific category (spam or ham), a quantity that has to be computed eventually, since the filter immediately classifies every incoming message. The greater *P(spam | message)* is, the greater the confidence that this message is spam.
- The term *P(class)* is called the **prior probability** of a message belonging to a given class. It follows that *P(spam) = 1 - P(ham)*. The term is "prior" in the sense that it is independent of the message and can therefore be computed before the message is seen. Informally, it represents the background knowledge on the possibility of receiving a spam email.
- The term *P(message | class)* is called the **likelihood** of seeing this message if its class is known. Conceptually, *P(message | spam)* is the probability of this message being chosen among all conceivable spam emails.
- Finally, the term *P(message)* is the prior for the given document, and acts as the **normalizing constant** in the equation so that the left hand side is actually a probability.

Less formally, the theorem argues the "spam-miness" of a specific message depend on only two factors: the probability of *any* message to be spam, and how likely it is to find a spam message identical to this one.

To apply the Bayes' rule, the quantities in the right hand side of equation (2) have to be estimated. Fortunately, the normalizing constant *P(message)* does not have to be computed directly because it can be expressed as *P(message | spam) * P(spam) + P (message | ham) * (1 - P(spam))*. Moreover, it does not have to be computed at all, since what actually matters is the ratio of the two posteriors for spam and ham. This leaves only the likelihoods *P(message | spam)* and *P(message | ham)* to be estimated. But first, it is necessary to have a concrete representation of email messages.

Representing Email Messages

Current machine learning algorithms require training examples to be represented in some well-defined way. The prevalent representation is the **vector-space model** [8], which consists of a single vector containing a fixed number of **features** (also known as **attributes**). Each feature defines a set of possible values that can be either numeric or categorical, such as {"male", "female"}, {"up", "down", "left", "right"}, etc.

The selection of an appropriate set of features to represent email messages as vectors is crucial. Clearly, the most important information in an email is its textual content, mainly the subject and the body of the message. Following the literature on using machine learning for text classification [9], most spam filters rely on the so-called **bag-of-words** representation, which represents a document by the multiset ("bag") of words in it.

As a simple example, consider the sentences "Click now and win big" and "Click here to win." If the list of features consists of any word that appears at least once, say ["and","big","Click","here","now","to","win"], then the two sentences are represented by the vectors [1,1,1,0,1,0,1] and [0,0,1,1,0,1,1], respectively. It becomes evident from this example that the order of the words in the document is ignored in the bag-of-words representation. In fact, many implementations make a further simplification by using a set-of-words representation in which the number of occurrences of each word in the document is also disregarded. Interestingly, in practice, the loss of information about the order and the frequency of words is usually insignificant for many learning tasks [4].

Of course, an email message is not just free text; it also contains other structured information, such as the date it was sent and the (alleged) list of server IPs the email has passed through before reaching the recipient. Other non-textual features that are sometimes incorporated in email representations include the percentage of punctuation characters in the message, the frequency of hyperlinks, HTML tags and attachments in it, and other attributes that tend to be different between spam and legitimate emails. Finally, more sophisticated representations go beyond the simple single-word features and encompass multi-word features, such as *n*-grams or sliding windows of permutated words [14].

A Naïve Assumption of Remarkable Strength

Given a vector-space representation of email messages from a training sample, it is possible to estimate the missing likelihoods *P(message_vector | spam)* and *P (message_vector | ham)* as vector frequencies. Still, the probability of a particular vector is hard to estimate directly since the space of all possible vectors is exponentially large compared to the number of features and only a relatively small training corpus is available. Most vectors appear no more than once in the sample, and usually not even once, so approximating their probabilities with the observed frequencies would introduce a large bias.

To illustrate the problem, assume a set-of-words representation using 200 single-word binary features, a relatively modest vector size for learning-based filtering. There are 2^{200} different vectors that can be represented in this space. A training set is many orders of magnitude smaller, typically comprising of hundreds or thousands of email messages. When an incoming email arrives, chances are that its vector is not identical to any of the spam vectors seen before. Hence, a direct estimate of the likelihood that the new message is spam would be zero. This is clearly a biased estimate; the real spam likelihood of this vector is very small, yet non-zero.

Here an assumption that overcomes this obstacle comes into play: every feature in the representation is **conditionally independent** from any other feature, given the message's class. For instance, if both "free" and "cash" are single-word features in the vector, then according to the assumption, the presence of the word "cash" in a spam message is independent of whether the word "free" is in the message too. Under the conditional independence hypothesis, the likelihood can be computed as the product of individual feature conditional probabilities:

$$P(message \mid class) = \prod P(feature_i \mid class) \qquad (3)$$

Continuing the example, there are two steps in computing the likelihood of a 200-dimensional vector being spam:

1. Estimate the likelihood $P(feature_i \mid spam)$ of each of the 200 feature values being spam as a frequency in the sample spam population. In contrast to the extremely small vector frequencies, the individual feature frequencies *can* be estimated reliably from a moderately large set of training data. For instance, if the new message contains the word-feature "cash" appears in 60 spam messages out of 3000 spam messages overall, $P(cash = 1 \mid spam)$ would be 0.02.
2. Estimate the likelihood of the message being spam as the product of all probabilities computed in step (1).

Most implementations perform the first step at training time and store the individual estimates in a hash table. Then at classification time, the appropriate probabilities are retrieved from the table and “chained” together to provide the vector's likelihood estimate. Storing the logarithm of each individual likelihood $log(P(feature_i \mid spam))$ during training reduces the vector likelihood computation to a simple summation, saving computational resources.

The learning algorithm that uses Bayes' theorem under the assumption of conditional independence is known as the **Naïve Bayes algorithm** because theoretically, it is flawed.Yet a large number of empirical studies have found Naïve Bayes to be surprisingly effective in many domains [3], a fortunate event which is evidenced also in the case of Bayesian spam filters. This was the the algorithm outlined by Paul Graham [5] in one of the first articles that prompted the development of learning-based filters

To Dump, Or Not To Dump: That Is The Question

The previous sections outlined the procedure and the reasoning of estimating the posterior probabilities according to the Naïve Bayes algorithm. There is still one remaining issue: how to use these numbers to make a binary classification

The straightforward way is classify the message as spam if the posterior spam probability exceeds a predefined threshold. Naturally, the question that follows is how to set the threshold. It turns out that the answer is not trivial since spam filtering is a **cost-sensitive** problem so the cost of accidentally blocking a legitimate message ("false positive") can be different, typically higher, than letting a spam message pass the filter ("false negative"). For instance, if it is crucial that no legitimate message is ever blocked, the threshold should be set very close to 1.0. On the other hand, if both kinds of errors are deemed equally important, the most reasonable value is 0.5.

The subjectivity of error costs points out an aspect of spam filtering not mentioned so far: the distinction between **categorization** and **policy**. The role of a spam filter, as illustrated in this article, is to classify messages as spam or legitimate. However, there is still a second issue: what to do with the flagged email messages. In the face of misclassifications, automatic discharge of messages without the user's consent is rarely a choice. Other less risky policies that are used more often include moving the flagged messages to a separate "bulk" folder, decreasing their priority, or bouncing them back to the sender (often done in a challenge/response fashion [1]). However, wading through potential spam messages to prevent false positives is also self-defeating by erasing the time saved using the filter. Ultimately, just like determining the spam threshold, a particular policy should also be carefully chosen based on the situation of each individual user.

Conclusions

Bayesian anti-spam filtering is an approach that combines superior accuracy, robustness, and personalization with a minimal need for user intervention. While remarkably successful so far, Bayesian filters may face new challenges in the near future. There are signs that spammers have become aware of the threat that content-based filters pose to them and have already began to design messages that circumvent the filters. Elaborate text preprocessing for extracting useful features and regular filter re-training are expected to be even more significant than today, as the arms race between spammers and filter developers escalates [6].

The combination of many diverse anti-spam measures is also expected to become crucial. SpamAssassin [10] is a good example of a versatile filter that combines black-lists, white-lists, handcrafted rules for email header and text analysis, a Bayesian classifier and support for distributed hash databases (e.g. Vipul's Razor [13]). A genetic algorithm may also assign weights to each technique for calculation of the overall likelihood of spam, resulting in a robust personalized filter that spammers cannot fool by simply exploiting a single technique.

That is not to say that learning-based spam filtering, or spam filtering in general, is the absolute solution to the spam plague. An argument against spam filtering is that by the time it takes place, it is already too late. The spam message directed to the user has already left from a most likely compromised host, wasted bandwidth and storage resources in all intermediate hosts and finally reached the filter at the recipient's email server or client. There is an active literature on how to stop spam at the root such as preventing address harvesting or breaking down altogether the business model of spam through economic or legislative measures. As of now however, spam filtering, in particular learning-based, is the most viable approach to tell ham from spam.

Acknowledgements

I would like to thank **George Paliouras**, **Ion Androutsopoulos** and the **Software and Knowledge Engineering Laboratory** in the National Center for Scientific Research "Demokritos" in Athens, Greece, for giving me the chance to gain invaluable knowledge and research experience in the theory and applications of Machine Learning.

References

1
Challenge-response Test. In *Wikipedia: The Free Encyclopedia.* <**http://en.wikipedia.org/wiki/Challenge-response_test**>.

2
CRM114, The Controllable Regex Mutilator. <**http://crm114.sourceforge.net**>.

3
Domingos, P. and Pazzani, M. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. In *Proceedings of the 13th International Conference on Machine Learning*, Bari, Italy, 1996, pp. 105-112.

4
Dumais, S., Platt, J., Heckerman, D., and Sahami, M. Inductive Learning Algorithms and Representations for Text Categorization. In *Proceedings of ACM-CIKM98*, Bethesda, Maryland, 1998, pp. 148-155.

5
Graham, P. A Plan for Spam. August 2002. <**http://www.paulgraham.com/spam.html**>.

6
Michelakis, E., Androutsopoulos, I., Paliouras, G., Sakkis, G., and

Stamatopoulos, P. Filtron: A Learning-Based Anti-Spam Filter. In *CEAS 2004*, July 30-31, 2004, Mountain View, California. <**http://www.ceas.cc/papers-2004/142.pdf**>.

7
Mitchell., T. *Machine Learning.* McGraw-Hill, 1997.

8
Salton, G., and McGill, M. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

9
Sebastiani, F. Machine Learning in Automated Text Categorization. In *ACM Computing Surveys* 34(1), 2002, pp. 1-47.

10
The SpamAssassin Project. <**http://spamassassin.apache.org/**>.

11
The SpamBayes Project. <**http://spambayes.sourceforge.net**>.

12
The Spamhaus Block List. <**http://www.spamhaus.org/sbl**>.

13
The Vipuls's Razor. <**http://razor.sourceforge.net/**>.

14
Yerazunis, W. The Spam-Filtering Accuracy Plateau at 99.9% Accuracy and How to Get Past It. 2004 MIT Spam Conference, Jan. 18, 2004, Cambridge, Massachusetts. <**http://crm114.sourceforge.net/Plateau_Paper.html**>.

Biography

George Sakkis (gsakkis@rutgers.edu) is a Master's student in Computer Science at Rutgers, the State University of New Jersey. His research interests focus mainly on Machine Learning, Data Mining, and Sequential Decision Making. He has work experience in spam filtering, web page classification, information extraction, and focused web crawling. He received a BS in Informatics and Telecommunications from the University of Athens, Greece in 2001.