

Email Spam Filtering

ENRIQUE PUERTAS SANZ

*Universidad Europea de Madrid
Villaviciosa de Odón, 28670 Madrid, Spain*

JOSÉ MARÍA GÓMEZ HIDALGO

*Optenet
Las Rozas
28230 Madrid, Spain*

JOSÉ CARLOS CORTIZO PÉREZ

*AINet Solutions
Fuenlabrada 28943, Madrid, Spain*

Abstract

In recent years, email spam has become an increasingly important problem, with a big economic impact in society. In this work, we present the problem of spam, how it affects us, and how we can fight against it. We discuss legal, economic, and technical measures used to stop these unsolicited emails. Among all the technical measures, those based on content analysis have been particularly effective in filtering spam, so we focus on them, explaining how they work in detail. In summary, we explain the structure and the process of different Machine Learning methods used for this task, and how we can make them to be cost sensitive through several methods like threshold optimization, instance weighting, or MetaCost. We also discuss how to evaluate spam filters using basic metrics, TREC metrics, and the receiver operating characteristic convex hull method, that best suits classification problems in which target conditions are not known, as it is the case. We also describe how actual filters are used in practice. We also present different methods used by spammers to attack spam filters and what we can expect to find in the coming years in the battle of spam filters against spammers.

- 1. Introduction 47
 - 1.1. What is Spam? 47
 - 1.2. The Problem of Email Spam 47
 - 1.3. Spam Families 48
 - 1.4. Legal Measures Against Spam 50
- 2. Technical Measures 51
 - 2.1. Primitive Language Analysis or Heuristic Content Filtering 51
 - 2.2. White and Black Listings 51
 - 2.3. Graylisting 52
 - 2.4. Digital Signatures and Reputation Control 53
 - 2.5. Postage 54
 - 2.6. Disposable Addresses 54
 - 2.7. Collaborative Filtering 55
 - 2.8. Honeypotting and Email Traps 55
 - 2.9. Content-Based Filters 56
- 3. Content-Based Spam Filtering 56
 - 3.1. Heuristic Filtering 57
 - 3.2. Learning-Based Filtering 63
 - 3.3. Filtering by Compression 80
 - 3.4. Comparison and Summary 83
- 4. Spam Filters Evaluation 83
 - 4.1. Test Collections 84
 - 4.2. Running Test Procedure 87
 - 4.3. Evaluation Metrics 88
- 5. Spam Filters in Practice 92
 - 5.1. Server Side Versus Client Side Filtering 93
 - 5.2. Quarantines 95
 - 5.3. Proxying and Tagging 96
 - 5.4. Best and Future Practical Spam Filtering 98
- 6. Attacking Spam Filters 98
 - 6.1. Introduction 98
 - 6.2. Indirect Attacks 99
 - 6.3. Direct Attacks 101
- 7. Conclusions and Future Trends 109
- References 109

1. Introduction

1.1 What is Spam?

In literature, we can find several terms for naming unsolicited emails. Junk emails, bulk emails, or unsolicited commercial emails (UCE) are a few of them, but the most common word used for reference is ‘spam.’ It is not clear where the word spam comes from, but many authors state that the term was taken from a Monty Python’s sketch, where a couple go into a restaurant, and the wife tries to get something other than spam. In the background are a bunch of Vikings that sing the praises of spam: ‘spam, spam, spam, spam . . . lovely spam, wonderful spam.’ Pretty soon the only thing you can hear in the skit is the word ‘spam.’ That same idea would happen to the Internet if large-scale inappropriate postings were allowed. You could not pick the real postings out from the spam.

But, what is the difference between spam and legitimate emails? We can consider an email as spam if it has the following features:

- *Unsolicited*: The receiver is not interested in receiving the information.
- *Unknown sender*: The receiver does not know and has no link with the sender.
- *Massive*: The email has been sent to a large number of addresses.

In the next subsections, we describe the most prominent issues regarding spam, including its effects, types, and main measures against it.

1.2 The Problem of Email Spam

The problem of email spam can be quantified in economical terms. Many hours are wasted everyday by workers. It is not just the time they waste reading spam but also the time they spend deleting those messages.

Let us think in a corporate network of about 500 hosts, and each one receiving about 10 spam messages every day. If because of these emails 10 min are wasted we can easily estimate the large number of hours wasted just because of spam. Whether an employee receives dozens or just a few each day, reading and deleting these messages takes time, lowering the work productivity. As an example, the United Nations Conference on Trade and Development estimates the global economic impact of spam could reach \$20 b in lost time and productivity. The California legislature found that spam costs United States organizations alone more than \$10 b in 2004, including lost productivity and the additional equipment, software, and

manpower needed to combat the problem. A repost made by Nucleus Research¹ in 2004 claims that spam will cost US employers \$2K per employee in lost productivity. Nucleus found that unsolicited email reduced employee productivity by a staggering 1.4%. Spam-filtering solutions have been doing little to control this situation, reducing spam levels by only 26% on average, according to some reports.

There are also problems related to the technical problems caused by spam. Quite often spam can be dangerous, containing virus, trojans, or other kind of damaging software, opening security breaches in computers and networks. In fact, it has been demonstrated that virus writers hire spammers to disseminate their so-called malware. Spam has been the main means to perform ‘phishing’ attacks, in which a bank or another organization is supplanted in order to get valid credentials from the user, and steal his banking data leading to fraud.

Also, network and email administrators have to employ substantial time and effort in deploying systems to fight spam. As a final remark, spam is not only dangerous or a waste of time, but also it can be quite disturbing. Receiving unsolicited messages is a privacy violation, and often forces the user to see strongly unwanted material, including pornography. There is no way to quantify this damage in terms of money, but no doubt it is far from negligible.

1.3 Spam Families

In this subsection, we describe some popular spam families or genres, focusing on those we have found most popular or damaging.

1.3.1 *Internet Hoaxes and Chain Letters*

There are a whole host of annoying hoaxes that circulate by email and encourage you to pass them on to other people. Most hoaxes have a similar pattern. These are some common examples to illustrate the language used:

- Warnings about the latest nonexistent virus dressed up with impressive but nonsensical technical language such as ‘nth-complexity binary loops.’
- Emails asking to send emails to a 7-year-old boy dying of cancer, promises that one well-known IT company’s CEO or president will donate money to charity for each email forwarded.
- Messages concerning the Helios Project, about a nonexistent alien being communicating with people on Earth, launched in 2003 and still online. Many people who interacted with Helios argue that Helios is real.

¹ See <http://www.nucleusresearch.com> for more information.

In general, messages that says ‘forward this to everyone you know!’ are usually hoaxes or chain letters. The purpose of these letters is from joking to generating important amounts of network traffic that involves economic losses in ISPs.

1.3.2 Pyramid Schemes

This is a common attempt to get money from people. Pyramid schemes are worded along the lines of ‘send ten dollars to this address and add yourself to the bottom of the list. In six weeks you’ll be a millionaire!’ They do not work (except from the one on the top of the pyramid, of course). They are usually illegal and you will not make any money from them.

1.3.3 Advance Fee Fraud

Advance fee fraud, also known as Nigerian fraud or 419 fraud, is a particularly dangerous spam. It takes the form of an email claiming to be from a businessman or government official, normally in a West African state, who supposedly has millions of dollars obtained from the corrupt regime and would like your help in getting it out of the country. In return for depositing the money in your bank account, you are promised a large chunk of it.

The basic trick is that after you reply and start talking to the fraudsters, they eventually ask you for a large sum of money up front in order to get an even larger sum later. You pay, they disappear, and you lose.

1.3.4 Commercial Spam

This is the most common family of spam messages. They are commercial advertisements trying to sell a product (that usually cannot be bought in a regular store). According to a report made by Sophos about security threats in 2006,² health- and medical-related spam (which primarily covers medication which claims to assist in sexual performance, weight loss, or human growth hormones) remained the most dominant type of spam and rose during the year 2006. In the report we find the top categories in commercial spam:

- Medication/pills – Viagra, Cialis, and other sexual medication.
- Phishing scams – Messages supplanting Internet and banking corporations like Ebay, Paypal, or the Bank of America, in order to get valid credentials and steal users’ money.
- Non-English language – an increasing number of commercial spam is translated or specifically prepared for non-English communities.

² See the full report at <http://www.sophos.com>.

- Software – or how you can get very cheap, as it is OEM (Original Equipment Manufacturer), that is, prepared to be served within a brand new PC at the store.
- Mortgage – a popular category, including not only mortgages but also specially debt grouping.
- Pornography – one of the most successful businesses in the net.
- Stock scams – interestingly, it has been observed that promoting stock corporations via email has had some impact in their results.

The economics behind the spam problem are clear: if users did not buy products marketed through spam, it would not be such a good business. If you are able to send 10 million messages for a 10 dollars product, and you get just one sell among every 10 thousand messages, you will be getting 10 thousand dollars from your spam campaign. Some spammers have been reported earning around five hundred thousand dollars a month, for years.

1.4 Legal Measures Against Spam

Fighting spam requires uniform international laws, as the Internet is a global network and only uniform global legislation can combat spam.

A number of nations have implemented legal measures against spam. The United States of America has both a federal law against spam and a separate law for each state. Something similar can be found in Europe: the European Union has its anti-spam law but most European countries have its own spam law too. There are specially effective or very string antispam laws like those in Australia, Japan, and South Korea. There are also bilateral treaties on spam and Internet fraud, as those between the United States and Mexico or Spain. On the other side, there are also countries without specific regulation about spam so it is an activity that is not considered illegal.

With this scenario, it is very difficult to apply legal measures against spammers. Besides that, anonymity is one of the biggest advantages of spammers. Spammers frequently use false names, addresses, phone numbers, and other contact information to set up ‘disposable’ accounts at various Internet service providers. In some cases, they have used falsified or stolen credit card numbers to pay for these accounts. This allows them to quickly move from one account to the next as each one is discovered and shut down by the host ISPs. While some spammers have been caught (a noticeable case is that of Jeremy Jaynes), there are many spammers that have avoided their capture for years. A trustable spammers’ hall of fame is maintained by The Spamhaus Project, and it is known as the Register Of Known Spam Operations (ROKSO).³

³ The ROKSO can be accessed at <http://www.spamhaus.org/rokso/index.lasso>.

2. Technical Measures

Among all the different techniques used for fighting spam, technical measures have become the most effective. There are several approaches used to filter spam. In the next section, we will comment some of the most popular approaches.

2.1 Primitive Language Analysis or Heuristic Content Filtering

The very first spam filters used primitive language analysis techniques to detect junk email. The idea was to match specific texts or words to email body or sender address. In the mid 1990s when spam was not the problem that it is today, users could filter unsolicited emails by scanning them, searching for phrases or words that were indicative of spam like 'Viagra' or 'Buy now.' Those days spam messages were not as sophisticated as they are today and this very simplistic approach could filter ~80% of the spam.

The first versions of the most important email clients included this technique that it worked quite well for a time, before spammers started to use their tricks to avoid filters. The way they obfuscated messages made this technique ineffective.

Another weakness of this approach was the high false-positive rate: any message containing 'forbidden words' was sent to trash. Most of those words were good for filtering spam, but sometimes they could appear in legitimate emails. This approach is not used nowadays because of the low accuracy and the high error rates it has.

This primitive analysis technique is in fact a form of content analysis, as it makes use of every email content to decide if it is spam or legitimate. We have called this technique heuristic filtering, and it is extensively discussed below.

2.2 White and Black Listings

White and black lists are extremely popular approaches to filter spam email [41]. White lists state which senders' messages are never considered spam, and black lists include those senders that should always be considered spammers.

A white list contains addresses that are supposed to be safe. These addresses can be individual emails, domain names, or IP addresses, and it would filter an individual sender or a group of them. This technique can be used in the server side and/or in the client side, and is usually found as a complement to other more effective approaches.

In server-side white lists, an administrator has to validate the addresses before they go to the trusted list. This can be feasible in a small company or a server with a small number of email accounts, but it can turn into a pain if pretended to be used in large corporate servers with every user having his own white list. This is because the

task of validating each email that is not in the list is a time-consuming job. An extreme use of this technique could be to reject all emails coming from senders that are not in the white list. This could sound very unreasonable, but it is not. It can be used in restricted domains like schools, where you prefer to filter emails from unknown senders but want to keep the children away from potentially harmful content, because spam messages could contain porn or another kind of adult content. In fact, this aggressive antispam technique has been used by some free email service providers as Hotmail, in which a rule can be stated preventing any message coming from any other service to get into their users mailboxes.

White listings can also be used in the client side. In fact, one of the first techniques used to filter spam consisted of using user's address book as a white list, tagging as potential spam all those emails that had in the FROM: field an address that was not in the address book. This technique can be effective for those persons who use email just to communicate with a limited group of contacts like family and friends.

The main problem of white listings is the assumption that trusted contacts do not send junk email and, as we are going to see, this assumption could be erroneous. Many spammers use computers that have been compromised using trojans and viruses for sending spam, sending them to all the contacts of the address book, so we could get a spam message from a known sender if his computer has been infected with a virus. Since these contacts are in the white list, all messages coming from them are flagged as safe.

Black listings, most often known as DNS Blacklists (DNSBL), are used to filter out emails which are sent by known spam addresses or compromised servers. The very first and most popular black list has been the trademarked Realtime Blackhole List (RBL), operated by the Mail Abuse Prevention System. System administrators, using spam detection tools, report IP addresses of machines sending spam and they are stored in a common central list that can be shared by other email filters. Most antispam softwares have some form of access to networked resources of this kind.

Aggressive black listings may block whole domains or ISPs having many false positives. A way to deal with this problem is to have several distributed black listings and contrast sender's information against some of them before blocking an email. Current DNS black lists are dynamic, that is, not only grow with new information, but also expire entries, maintaining fresh reflection of current situation in the address space.

2.3 Graylisting

As a complement to white and black listings, one could use graylistings [94]. The core behind this approach is the assumption that junk email is sent using spam bots, this is specific software made to send thousands of emails in a short time. This software differs from traditional email servers and does not respect email RFC standards. In particular, emails that fail to reach its target are not sent again, as a real system would

do. This is the right feature used by graylistings. When the system receives an email from an unknown sender that is not in a white listing, it creates a tuple sender–receiver. The first time that tuple occurs in the system, the email is rejected so it is bounced back to the sender. A real server will send that email again so the second time the system finds the tuple, the email is flagged as safe and delivered to the recipient.

Graylistings have some limitations and problems. The obvious ones are the delay we could have in getting some legitimate emails when using this approach because we have to wait until the email is sent twice, and the waste of bandwidth produced in the send–reject–resend process.

Other limitations are that this approach will not work when spam is sent from open relays as they are real email servers and the easy way for spammer to work-around graylistings, just adding a new functionality to the software, allowing it to send bounced emails again.

2.4 Digital Signatures and Reputation Control

With the emergence of Public Key Cryptography, and specifically, its application to email coding and signing, most prominently represented by Pretty Good Privacy (PGP) [103] and GNU Privacy Guard (GPG) [64], there exists the possibility of filtering out unsigned messages, and in case they are signed, those sent by untrusted users. PGP allows keeping a nontrivial web/chain of trust between email users, the way that trust is spread over a net of contacts. This way, a user can trust the signer of a message if he or she is trusted by another contact of the email receiver.

The main disadvantage of this approach is that PGP/GPG users are rare, so it is quite risky to consider legitimate email coming only from trusted contacts. However, it is possible to extend this idea to email servers.

As we saw in previous section, many email filters use white listings to store safe senders, usually local addresses and addresses of friends. So if spammers figure out who our friends are, they could forge the FROM: header of the message with that information, avoiding filters because senders that are in white listings are never filtered out. Sender Policy Framework (SPF), DomainKeys, and Sender ID try to prevent forgery by registering IPs of machines used to send email from for every valid email sender in the server [89]. So if someone is sending an email from a particular domain but it does not match the IP address of the sender, you can know the email has been forged. The messages are signed by the public key of the server, which makes its SPF, DomainKeys, or Sender ID record public. As more and more email service providers (specially the free ones, like Yahoo!, Hotmail, or Gmail) are making their IP records public, the approach will be increasingly effective.

Signatures are a basic implementation of a more sophisticated technique, which is reputation control for email senders. When the system receives an email from an unknown sender, the message is scanned and classified as legitimate or spam. If the

email is classified as legitimate, the reputation of the sender is increased, and decreased if classified as spam. The more emails are sent from that address, the more positive or negative the sender is ranked. Once reputation crosses a certain threshold, it can be moved to a white or black list. The approach can be extended to the whole IP space in the net, as current antispam products by IronPort feature, named SenderBase.⁴

2.5 Postage

One of the main reasons of spam success is the low costs of sending spam. Senders do not have to pay for sending email and costs of bandwidth are very low even if sending millions of emails.⁵ Postage is a technique based upon the principle of senders of unsolicited messages demonstrating their goodwill by paying some kind of postage: either a small amount of money paid electronically, a sacrifice of a few seconds of human time at answering a simple question, or some time of computation in the sender machine.

As the email services are based on the Simple Mail Transfer Protocol, economic postage requires a specific architecture over the net, or a dramatic change in the email protocol. Abadi et al. [1] describes a ticket-based client-server architecture to provide postage for avoiding spamming (yet other applications are suitable).

An alternative is to require the sender to answer some kind of question, to prove he is actually a human being. This is the kind of Turing Test [93] that has been implemented in many web-based services, requiring the user to type the hidden word in a picture. These tests are named CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) [2]. This approach can be especially effectively used to avoid *outgoing* spam, that is, preventing the spammers to abuse of free email service providers as Hotmail or Gmail.

A third approach is requiring the sender machine to solve some kind of computationally expensive problem [32], producing a delay and thus, disallowing spammers to send millions of messages per day. This approach is, by far, the less annoying of the postage techniques proposed, and thus, the most popular one.

2.6 Disposable Addresses

Disposable addresses [86] are a technique used to prevent a user to receive spam. It is not a filtering system itself but a way to avoid spammers to find out our address. To harvest email addresses, spammers crawl the web searching for addresses in web

⁴ The level of trust of an IP in SenderBase can be checked at: <http://www.senderbase.org/>.

⁵ Although the case against the famous spammer Jeremy Jaynes has revealed he has been spending more than one hundred thousand dollars per month in high speed connections.

pages, forums, or Usenet groups. If we do not publish our address on the Internet, we can be more or less protected against spam, but the problem is when we want to register in a web page or an Internet service and we have to fill in our email address in a form. Most sites state that they will not use that information for sending spam but we cannot be sure and many times the address goes to a list that is sold to third party companies and used for sending commercial emails. Moreover, these sites can be accessed by hackers with the purpose of collecting valid (and valuable) addresses.

To circumvent this problem, we can use disposable email addresses. Instead of letting the user prepare his own disposable addresses, he can be provided with an automatic system to manage them, like the channels' infrastructure by ATT [50]. The addresses are temporary accounts that the user can use to register in web services. All messages sent to disposable address are redirected to our permanent safe account during a configurable period of time. Once the temporary address is no longer needed, it is deleted so even if that account receives spam, this is not redirected.

2.7 Collaborative Filtering

Collaborative filtering [48] is a distributed approach to filter spam. Instead of having each user to have his own filter, a whole community works together. Using this technique, each user shares his judgments of what is spam and what is not with the other users. Collaborative filtering networks take advantage of the problem of some users that receive spam to build better filters for those that have not yet received those spam messages. When a group of users in the same domain have tagged an email coming from a common sender as spam, the system can use the information in those emails to learn to classify those particular emails so the rest of users in the domain will not receive them.

The weakness of this approach is that what is spam for somebody could be a legitimate content for another. These collaborative spam filters cannot be more accurate as a personal filter in the client side but it is an excellent option for filtering in the server side. Another disadvantage of this approach is that spammers introduce small variations in the messages, disallowing the identification of a new upcoming spam email as a close variation of one received earlier by another user [58].

2.8 Honeypotting and Email Traps

Spammers are known to abuse vulnerable systems like open mail relays and public open proxies. In order to discover spam activities, some administrators have created honeypot programs that simulate being such vulnerable systems. The existence of such fake systems makes more risky for spammers to use open relays

and open proxies for sending spam. Honeypotting is a common technique used for system administrators to detect hacking activities on their servers. They create fake vulnerable servers in order to burst hackers while protecting the real servers. Since the term honeypotting is more appropriate for security environments, the terms ‘email traps’ or ‘spam traps’ can be used instead of referring to these techniques when applied to prevent spam.

Spam traps can be used to collect instances of spam messages on keeping a fresh collection of spammer techniques (and a better training collection in learning-based classifiers), to build and deploy updated filtering rules in heuristic filters, and to detect new spam attacks in advance, avoiding them reach, for example, a corporate network in particular.

2.9 Content-Based Filters

Content-based filters are based on analyzing the content of emails. These filters can be hand-made rules, also known as heuristic filters, or learned using Machine Learning algorithms. Both approaches are widely used these days in spam filters because they can be very accurate when they are correctly tuned up, and they are going to be deeply analyzed in next section.

3. Content-Based Spam Filtering

Among the technical measures to control spam, content-based filtering is one of the most popular ones. Spam filters that analyze the contents of the messages and take decisions on that basis have spread among the Internet users, ranging from individual users at their home personal computers, to big corporate networks. The success of content-based filters is so big that spammers have performed increasingly complex attacks designed to avoid them and to reach the users’ mailbox.

This section covers the most relevant techniques for content-based spam filtering. Heuristic filtering is important for historical reasons, although the most popular modern heuristic filters have some learning component. Learning-based filtering is the main trend in the field; the ability to learn from examples of spam and legitimate messages gives these filters full power to detect spam in a personalized way. Recent TREC [19] competitive evaluations have stressed the importance of a family of learning-based filters, which are those using compression algorithms; they have scored top in terms of effectiveness, and so they deserve a specific section.

3.1 Heuristic Filtering

Since the very first spam messages, users (that were simultaneously their own ‘network administrators’) have coded rules or heuristics to separate spam from their legitimate messages, and avoid reading the first [24]. A content-based heuristic filter is a set of hand-coded rules that analyze the contents of a message and classify it as spam or legitimate. For instance, a rule may look like:

$$\text{if } (P(\text{'Viagra'} \in M) \text{ or } (P(\text{'VIAGRA'} \in M))) \text{ then } \text{class}(M) = \text{spam}$$

This rule means that if any of the words ‘Viagra’ or ‘VIAGRA’ (that are in fact distinct characters strings) occur in a message M , then it should be classified as spam. While first Internet users were often privileged user administrators and used this kind of rules in the context of sophisticated script and command languages, most modern mail user clients allow writing this kind of rules through simple forms. For instance, a Thunderbird straightforward spam filter is shown in Fig. 1. In this example, the users has prepared a filter named ‘spam’ that deletes all messages in which the word ‘**spam**’ occurs in the Subject header.

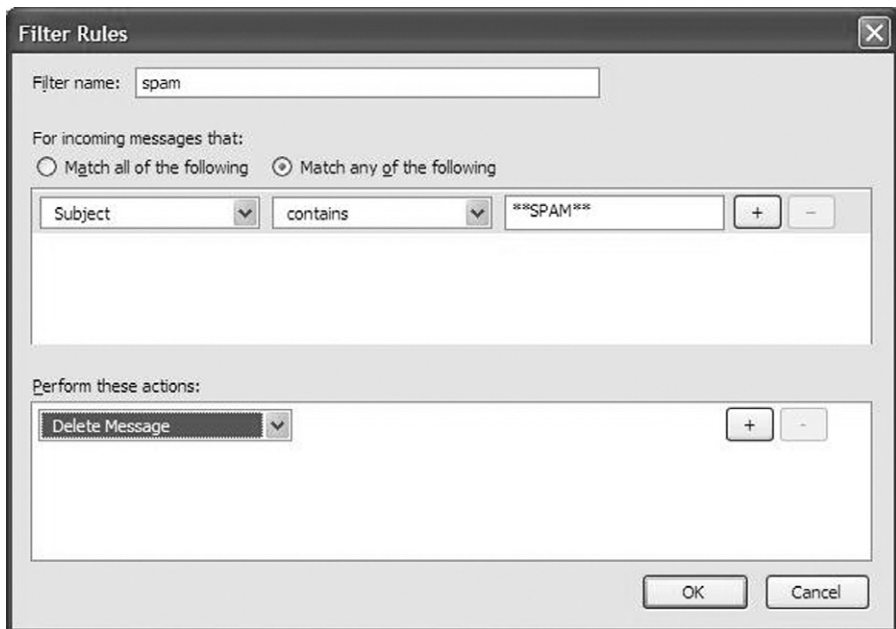


FIG. 1. A simple spam filter coded as a Thunderbird mail client rule. If the word ‘**spam**’ occurs in the Subject of a message, it will be deleted (sent to trash).

However, these filtering utilities are most often used to classify the incoming messages into folders, according to their sender, their topic, or the mail list they belong to. They can also be used in conjunction with a filtering solution out of the mail client, which may tag spam messages (for instance, with the string ‘**spam**’ in the subject, or being more sophisticated, by adding a specific header like, for example, X-mail report, which can include a simple tag or a rather informative output with even a score), that will be later processed by the mail client by applying the suitable filters and performing the desired action. A sensible action is to send the messages tagged as spam to a quarantine folder in order to avoid false positives (legitimate messages classified as spam).

It should be clear that maintaining an effective set of rules can be a rather time-consuming job. Spam messages include offers of pharmacy products, porn advertisements, unwanted loans, stock recommendations, and many other types of messages. Not only their content, but their style is always changing. In fact, it is hard to find a message in which the word ‘Viagra’ occurs without alterations (except for a legitimate one!). In other words, there are quite many technical experts highly committed to make the filter fail: the spammers. This is why spam filtering is considered a problem of ‘adversarial classification’ [25].

Neither a modern single network administrator nor even an advanced user will be writing their own handmade rules to filter spam. Instead, a list of useful rules can be maintained by a community of expert users and administrators, as it has been done in the very popular open-source solution SpamAssassin or in the commercial service Brightmail provided by Symantec Corporation. We discuss these relevant examples in the next subsections, finishing this section with the advantages and disadvantages of this approach.

3.1.1 *The SpamAssassin Filter*

While it is easy to defeat a single network administrator, it is harder to defeat a community. This is the spirit of one of the most spread heuristic filtering solutions: SpamAssassin [87]. This filter has received a number of prizes, and as a matter of example, it had more than 3600 downloads in the 19 months the project was hosted at Sourceforge⁶ (February 2002–September 2003).

SpamAssassin is one of the oldest still-alive filters in the market, and its main feature (for the purpose of our presentation) is its impressive set of rules or heuristics, contributed by tens of administrators and validated by the project

⁶ Sourceforge is the leading hosting server for open-source projects, providing versioning and downloading services, statistics, and more. See: <http://sourceforge.net>.

committee. The current (version 3.2) set of rules (named ‘tests’ in SpamAssassin) has 746 tests.⁷ Some of them are administrative, and a number of them are not truly ‘content-based,’ as they, for example, check the sender address or IP against public white lists. For instance, the test named ‘RCVD_IN_DNSWL_HI’ checks if the sender is listed in the DNS Whitelist.⁸ Of course, this is a white listing mechanism, and it makes nearly no analysis of the very message content. On the other side, the rule named ‘FS_LOW_RATES’ tests if the Subject field contains the words ‘low rates,’ which is very popular in spam messages dealing with loans or mortgages. Many SpamAssassin tests address typing variability by using quite sophisticated regular expressions. We show a list of additional examples in the Fig. 2, as they are presented in the project web page.

A typical SpamAssassin content matching rule has the structure shown in the next example:

```
body DEMONSTRATION_RULE /test/
score DEMONSTRATION_RULE 0.1
describe DEMONSTRATION_RULE This is a simple test rule
```

The rule starts with a line that describes the test to be performed, it goes on with line presenting the score, and it has a final line for the rule description. The sample rule name is ‘DEMONSTRATION_RULE,’ and it checks the (case sensitive) occurrence of the word ‘test’ in the body section of an incoming email message. If the condition is

AREA TESTED	LOCALE	DESCRIPTION OF TEST	TEST NAME	DEFAULT SCORES (local, net, with bayes, with bayes+net)
header		Envelope sender listed in bl.open-whois.org.	DNS_FROM_OPENWHOIS	0 2.431 0 1.130
body		Provision for income taxes	DOS_PROVISION4	1.5
body		Report of financial income	DOS_REPORT_FIN_INC	0.5
body		Pump and dump stock spam	DOS_STOCK_CDYV_GENERIC	2.5
uri		Found an asterisk in a URI	DOS_URI_ASTERISK	1
header		Subject =~ /\bhoodia\b/i	DRUGS_HDIA	2.529 2.501 2.483 2.097
body		Add / Gain inches	FB_ADD_INCHES	2.999 2.999 2.620 2.131
body		It's almost sex, but not!	FB_ALMOST_SEX	3.099 3.096 2.841 2.110

FIG. 2. A sample of SpamAssassin test or filtering rules. The area tested may be the header, the body, etc. and each test is provided with one or more scores that can be used to set a suitable threshold and vary the filter sensitivity.

⁷ The lists of tests used by SpamAssassin are available at: <http://spamassassin.apache.org/tests.html>.

⁸ The DNS Whitelist is available at: <http://www.dnswl.org/>.

satisfied, that is, the word occurs, then the score 0.1 is added to the message global score. The score of the message may be incremented by other rules, and the message will be tagged as spam if the global score exceeds a manually or automatically set threshold. Of course, the higher the score of a rule, the more it contributes to the decision of tagging a message as spam.

The tests performed in the rules can address all the parts in a message, and request preprocessing or not. For instance, if the rule starts with ‘header,’ only the headers will be tested:

```
header DEMONSTRATION_SUBJECT Subject =~ /test/
```

In fact, the symbols ‘=~’ preceding the test, along with the word ‘Subject,’ mean that only the subject header will be tested. This case, the subject field name is case insensitive.

The tests performed allow complex expressions written in the Perl Regular Expressions (Regex) Syntax. A slightly more complex example may be:

```
header DEMONSTRATION_SUBJECT Subject =~ /\btest\b/i
```

In this example, the expression ‘/\btest\b/i’ means that the word ‘test’ will be searched as a single word (and not as a part of others, like ‘testing’), because it starts and finishes with the word-break mark ‘\b,’ and the test will be case insensitive because of the finishing mark ‘/i.’ Of course, regular expressions may be much more complex, but covering them in detail is beyond the scope of this chapter. We suggest [54] for the interested reader.

Manually assigning scores to the rules is not a very good idea, as the rule coder must have a precise and global idea of all the scores in all the rest of the rules. Instead, an automated method is required, which should be able to look at all the scores and a set of testing messages, and compute the scores that minimize the error of the filter. In versions 2.x, the scores of the rules have been assigned using a Genetic Algorithm, while in (current) versions 3.x, the scores are assigned using a neural network trained with error back propagation (a perceptron). Both systems attempt to optimize the effectiveness of the rules that are run in terms of minimizing the number of false positives and false negatives, and they are presented in [87] and [91], respectively.

The scores are optimized on a set of real examples contributed by volunteers. The SpamAssassin group has in fact released a corpus of spam messages, the so-named SpamAssassin Public Corpus.⁹ This corpus includes 6047 messages, with ~31% spam ratio. As it has been extensively used for the evaluation of content-based spam filter, we leave a more detailed description of it for Section 4.

⁹ The SpamAssassin Public Corpus is available at: <http://spamassassin.apache.org/publiccorpus/>.

3.1.2 The Symantec Brightmail Solution

Some years ago, Brightmail emerged as an antispam solution provider based on an original business model in the antispam market. Instead of providing a final software application with filtering capabilities, it focused more on the service, and took the operation model from antivirus corporations: analysts working 24 h a day on new attacks, and frequent delivering of new protection rules. The model succeeded, and on June 21st, 2004, Symantec Corporation acquired Brightmail Incorporated, with its solution and its customers. Nowadays, Symantec claims that Brightmail Anti-spam protects more than 300 million users, and filters over 15% of the worldwide emails.

The Brightmail Anti-spam solution works at the clients' gateway, scanning incoming messages to the corporation, and deciding if they are spam or not. The decision is taken on the basis of a set of filtering rules provided by the experts working in the Symantec operations center, named BLOC (Brightmail Logistics Operations Center). The operational structure of the solution is shown in Fig. 3. In this figure, circles are used to denote the next processing steps:

1. The Probe Network (TM) is a network of fake email boxes ('spam traps' or 'honeypots') that have been seeded with the only purpose of receiving spam. These email addresses can be collected only by automatic means, as spammers

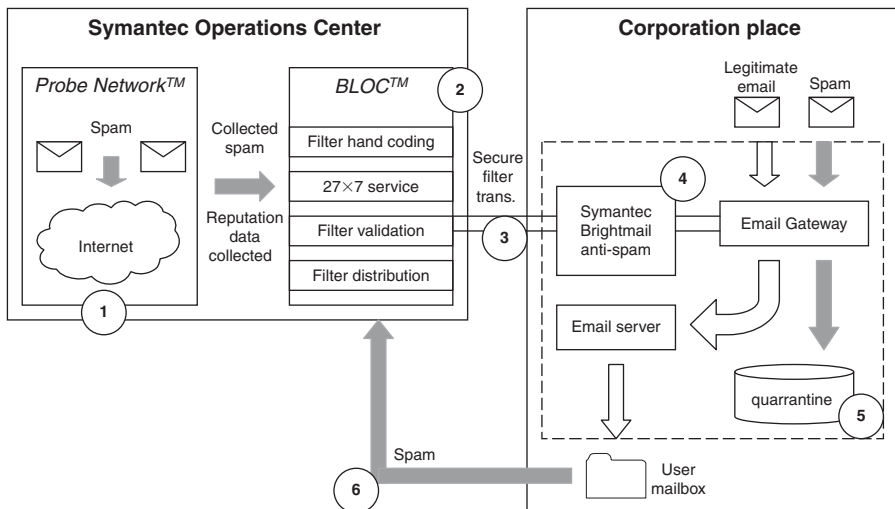


FIG. 3. Operational structure of the Symantec Brightmail Anti-spam solution. The numbers in circles denote processes described.

do, and in consequence, they can receive only spam emails. The spam collected is sent to the BLOC.

2. At the BLOC, groups of experts and, more recently, content-based analysis automatic tools build, validate, and distribute antispam filters to Symantec corporate customers.
3. Every 10 min, the Symantec software downloads the most recent version of the filters from the BLOC, in order to keep them as updated as possible.
4. The software filters the incoming messages using the Symantec and the user-customized filters.
5. The email administrator determines the most suitable administration mode for the filtered email. Most often, the (detected as) spam email is kept into a quarantine where users can check if the filter has mistakenly classified a legitimate message as spam (a false positive).
6. The users can report undetected spam to Symantec for further analysis.

The processes at the BLOC were once manual, but the ever-increasing number of spam attacks has progressively made impossible to approach filter building as a hand-made task. In the recent times, spam experts in the BLOC have actually switched their role to filter adapters and tuners, as the filters are being produced by using the automatic, learning-based tools described in the next section.

3.1.3 Problems in Heuristic Filtering

Heuristic content-based filtering has clear advantages over other kinds of filtering, especially those based on black and white listing. The most remarkable one is that it filters not only on the 'From' header, but also it can make use of the entire message, and in consequence, to make a more informed decision. Furthermore, it offers a lot of control on the message information that is scanned, as the filter programmer decides which areas to scan, and what to seek.

However, heuristic filtering has two noticeable drawbacks. First, writing rules is not an easy task. Or it has to be left on the hands of an experienced email administrator, or it has to be simplified via the forms in commercial mail clients as described above. The first case usually involves some programming, probably including a bit of regular expression definition, which is hard and error-prone. The second one implies a sacrifice of flexibility to gain simplicity.

The second drawback is that, even being the rules written by a community of advanced users or administrators, the number of spammers is bigger, and moreover, they have a strong economic motivation to design new methods to avoid detection. In this arms race, the spammers will be always having the winning hand if the work of administrators is not supported with automatic (learning-based) tools as those we describe in the next section.

3.2 Learning-Based Filtering

During the past 9 years, a new paradigm of content-based spam filtering has emerged. Bayesian filters, or more in general, learning-based filters, have the ability to learn from the email flow and to improve their performance over time, as they can adapt themselves to the actual spam and legitimate email a particular user receives. Their impressive success is demonstrated by the deep impact they have had on spam email, as the spammers have to costly change their techniques in order to avoid them. Learning-based filters are the current state of the art of email filtering, and the main issue in this chapter.

3.2.1 *Spam Filtering as Text Categorization*

Spam filtering is an instance of a more general text classification task named Text Categorization [85]. Text Categorization is the assignment of text documents to a set of predefined classes. It is important to note that the classes are preexistent, instead of being generated on the fly (what corresponds to the task of Text Clustering). The main application of text categorization is the assignment of subject classes to text documents. Subject classes can be web directory categories (like in Yahoo!¹⁰), thematic descriptors in libraries (like the Library of Congress Subject Headings¹¹ or, in particular domains, the Medical Subject Headings¹² by the National Library of Medicine, or the Association for Computing Machinery ACM's Computing Classification System descriptors used in the ACM Digital Library itself), personal email folders, etc. The documents may be Web sites or pages, books, scientific articles, news items, email messages, etc.

Text Categorization can be done manually or automatically. The first method is the one used in libraries, where expert catalogers scan new books and journals in order to get them indexed according to the classification system used. For instance, the National Library of Medicine employs around 95 full-time cataloguers in order to index the scientific and news articles distributed via MEDLINE.¹³ Obviously this is a time- and money-consuming task and the number of publications is always increasing.

The increasing availability of tagged data has allowed the application of Machine Learning methods to the task. Instead of hand classifying the documents, or manually building a classification system (as what we have named a heuristic filter

¹⁰ Available at: <http://www.yahoo.com/>.

¹¹ Available at: <http://www.loc.gov/cds/lcsh.html>.

¹² Available at: <http://www.nlm.nih.gov/mesh/>.

¹³ James Marcetich, Head of the Cataloguing Section of the National Library of Medicine, in personal communication (July 18, 2001).

above), it is possible to automatically build a classifier by using a Machine Learning algorithm on a collection of hand-classified documents suitably represented. The administrator or the expert does not have to write the filter, but let the algorithm learn the document properties that make them suitable for each class. This way, the traditional expert system ‘knowledge acquisition bottleneck’ is alleviated, as the expert can keep on doing what he or she does best (that is, in fact, classifying), and the system will be learning from his decisions.

The Machine Learning approach has achieved considerable success in a number of tasks, and in particular, in spam filtering. In words by Sebastiani:

(Automated Text Categorization) has reached effectiveness levels comparable to those of trained professionals. The effectiveness of manual Text Categorization is not 100% anyway (...) and, more importantly, it is unlikely to be improved substantially by the progress of research. The levels of effectiveness of automated TC are instead growing at a steady pace, and even if they will likely reach a plateau well below the 100% level, this plateau will probably be higher than the effectiveness levels of manual Text Categorization. [85] (p. 47)

Spam filtering can be considered an instance of (Automated) Text Categorization, in which the documents to classify are the user email messages, and the classes are spam and legitimate email. It may be considered easy, as it is a single-class problem, instead of the many classes that are usually considered in a thematic TC task.¹⁴ However, it shows special properties that makes it a very difficult task:

1. Both the spam and the complementary class (legitimate email) are not thematic, that is, they can contain messages dealing with several topics or themes. For instance, as of 1999, a 37% of the spam email was ‘get rich quick’ letters, a 25% was pornographic advertisements, and an 18% were software offers. The rest of the spam included Web site promos, investment offers, (fake) health products, contests, holidays, and others. Moreover, some of the spam types can overlap with legitimate messages, both commercial and coming from distribution lists. While the percentages have certainly changed (health and investment offers are now most popular), this demonstrates that current TC systems that rely on words and features for classification may have important problems because the spam class is very fragmented.
2. Spam has an always changing and often skewed distribution. For instance, according to the email security corporation MessageLabs [66], spam has gone from 76.1% of the email sent in the first quarter of 2005, to 56.9% in the first quarter of 2006, and back to 73.5% in the last quarter of 2006. On one side,

¹⁴ The Library of Congress Subject Headings 2007 edition has over 280000 total headings and references.

Machine Learning classifiers expect the same class distribution they learnt from; any variation of the distribution may affect the classifier performance. On the other, skewed distributions like 90% spam (reached in 2004) may make a learning algorithm to produce a trivial acceptor, that is, a classifier that always classifies a message as spam. This is due to the fact that Machine Learning algorithms try to minimize the error or maximize the accuracy, and the trivial acceptor is then 90% accurate. And even worse, the spam rate can vary from place to place, from company to company, and from person to person; in that situation, is very difficult to build a fit-them-all classifier.

3. Like many other classification tasks, spam filtering has imbalanced misclassification costs. In other words, the kinds of mistakes the filter makes are significant. No user will be happy with a filter that catches 99% of spam but that deletes a legitimate message once-a-day. This is because false positives (legitimate messages classified as spam) are far more costly than false negatives (spam messages classified as legitimate, and thus, reaching the users' inbox). But again, it is not clear which proportion is right: a user may accept a filter that makes a false positive per 100 false negatives or per 1,000, etc. It depends on the user's taste, the amount of spam he or she receives, the place where he or she lives, the kind of email account (work or personal), etc.
4. Perhaps the most difficult issue with spam classification is that it is an instance of adversarial classification [25]. An adversarial classification task is one in which there exists an adversary that modifies the data arriving at the classifier in order to make it fail. Spam filtering is perhaps the most representative instance of adversarial classification, among many others like computer intrusion detection, fraud detection, counter-terrorism, or a much related one: web spam detection. In this latter task, the system must detect which webmasters manipulate pages and links to inflate their rankings, after reverse engineering the ranking algorithm. The term spam, although coming from the email arena, is so spread that it is being used for many other fraud problems: mobile spam, blog spam, 'spim' (spam over Instant Messaging), 'spit' (spam over Internet Telephony), etc. Regarding spam email filtering, standard classifiers like Naïve Bayes were initially successful [79], but spammers soon learned to fool them by inserting 'nonspam' words into emails, breaking up 'spam' ones like 'Viagra' with spurious punctuation, etc. Once spam filters were modified to detect these tricks, spammers started using new ones [34].

In our opinion, these issues make spam filtering a very unusual instance of Automated Text Categorization. Being said this, we must note that the standard structure of an Automated Text Categorization system is suited to the problem of spam filtering, and so we will discuss this structure in the next subsections.

3.2.2 Structure of Processing

The structure of analysis and learning in modern content-based spam filters that make use of Machine Learning techniques, is presented in Fig. 4. In this figure,¹⁵ we represent processes or functions as rounded boxes, and information items as plain boxes. The analysis, learning, and retraining (with feedback information) of the classifier are time- and memory-consuming processes, intended to be performed offline and periodically. The analysis and filtering of new messages must be a fast process, to be performed online, as soon as the message arrives at the system. We describe these processes in detail below.

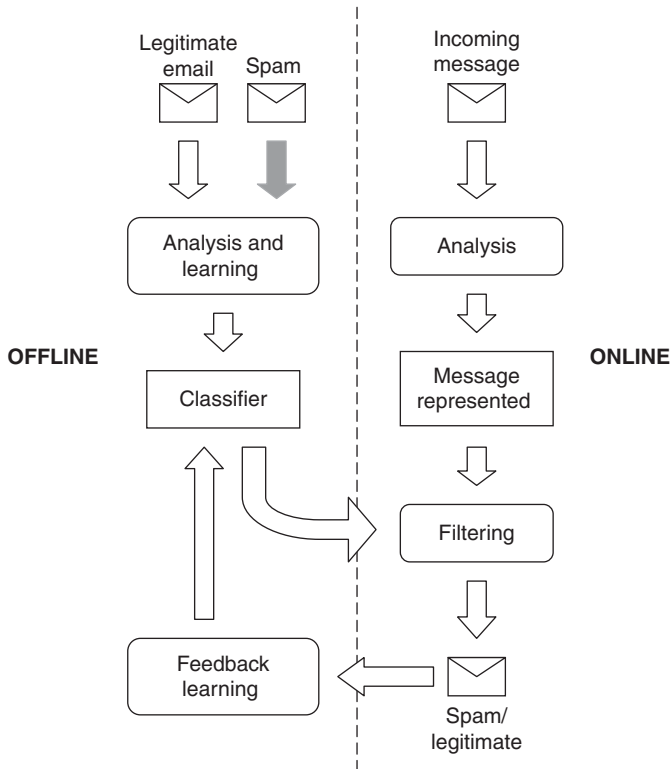


FIG. 4. Processing structure of a content-based spam filter that uses Machine Learning algorithms.

¹⁵ That figure is a remake of that by Belkin and Croft for text retrieval in [7].

The first step in a learning-based filter is getting a collection of spam and legitimate messages and training a classifier on it. Of course, the collection of messages (the training collection) must represent the operating conditions of the filter as accurately as possible. Machine Learning classifiers often perform poorly when they have been trained on noisy, inaccurate, and insufficient data. There are some publicly available spam/legitimate email collections, discussed in [Section 4](#) because they are most often used as test collections.

The first process in the filter is learning a classifier on the training messages, named instances or examples. This process involves analyzing the messages in order to get a suitable representation for learning from them. In this process, the messages are often represented as attribute-value vectors, in which attributes are word tokens in the messages, and values are, for example, binary (the word token occurs on the message or not). Next, a Machine Learning algorithm is fed with the represented examples, and it produces a classifier, that is a model of the messages or a function able to classify new messages if they follow the suitable representation. Message representation and classifier learning are the key processes in a learning-based filter, and so they are discussed in detail in the next subsections.

Once the classifier has been trained, it is ready to filter new messages. As they arrive, they are processed in order to represent them according to the format used in the training messages. That typically involves, for instance, ignoring new words in the messages, as classification is made on the basis of known words in the training messages. The classifier receives the represented message and classifies it as spam or legitimate (probably with a probability or a score), and tagged accordingly (or routed to the quarantine, the user mailbox, or whatever).

The main strength of Machine Learning-based spam filters is their ability to learn from user relevance judgments, adapting the filter model to the actual email received by the user. When the filter commits a mistake (or depending on the learning mode, after every message), the correct output is submitted to the filter, which stores it for further re-training. This ability is a very noticeable strength, because if every user receives different email, every filter is different (in terms of stored data and model learned), and it is very complex to prepare attacks able to avoid the filters of all users simultaneously. As spammers benefit relies on the number of messages read, they are forced to prepare very sophisticated attacks able to break different vendor filters with different learned models. As we discuss in [Section 6](#), they sometimes succeed using increasingly complex techniques.

3.2.3 Feature Engineering

Feature engineering is the process of deciding, given a set of training instances, which properties will be considered for learning from them. The properties are the features or attributes, and they can take different values; this way, every training

instance (a legitimate or email message) is mapped into a vector in a multidimensional space, in which the dimensions are the features. As many Machine Learning are very slow or just unable to learn in highly dimensional spaces, it is often required to reduce the number of features used in the representation, performing attribute selection (determining a suitable subset of the original attributes) or attribute extraction (mapping the original set of features into a new, reduced set). These tasks are also a part of the feature engineering process.

3.2.3.1 Tokens and Weights. In Text Categorization and spam filtering, the most often used features are the sequences of characters or strings that minimally convey some kind of meaning in a text, that is, the words [39, 85]. More generally, we speak of breaking a text into tokens, a process named tokenization. In fact, this just follows the traditional Information Retrieval Vector Space Model by Salton [81]. This model specifies that, for the purpose of retrieval, texts can be represented as term-weight vectors, in which the terms are (processed) words (our attributes), and weights are numeric values representing the importance of every word in every document.

First, learning-based filters have taken relatively simple decisions in this sense, following what was the state of the art on thematic text categorization. The simplest definition of features is words, being a word any sequence of alphabetic characters, and considering any other symbol as a separator or blank. This is the approach followed in the seminal work in this field, by Sahami et al. [79]. This work has been improved by Androutsopoulos et al. [4–6], in which they make use of a lemmatizer (or stemmer), to map words into their root, and a stoplist (a list of frequent words that should be ignored as they bring more noise than meaning to thematic retrieval: pronouns, prepositions, etc.). The lemmatizer used is Morph, included in the text analysis package GATE,¹⁶ and the stoplist includes the 100 most frequent words in the British National Corpus.¹⁷

In the previous work, the features are binary, that is, the value is one if the token occurs in the message, and zero otherwise. There are several more possible definitions of the weights or values, traditionally coming in the Information Retrieval field. For instance, using the same kind of tokens or features, the authors of [39] and [40] make use of TF.IDF weights. Its definition is the following one:

$$w_{ij} = tf_{ij} \times \log_2 \left(\frac{N}{df_i} \right)$$

¹⁶ The GATE package is available at: <http://gate.ac.uk/>.

¹⁷ The British National Corpus statistics are available at: <http://www.natcorp.ox.ac.uk/>.

tf_{ij} being the number of times that the i -th token occurs in the j -th message, N the number of messages, and df_i the number of messages in which the i -th token occurs. The TF (Term Frequency) part of the weight represents the importance of the token or term in the current document or messages, while the second part IDF (Inverse Document Frequency) gives an *ad hoc* idea of the importance of the token in the entire document collection. TF weights are also possible.

Even relatively straightforward decisions like lowercasing all words, can strongly affect the performance of a filter. The second generation of learning filters has been much influenced by Graham's work [46, 47], who took advantage of the increasing power and speed of computers to ignore most preprocessing and simplifying decisions taken before. Graham makes use of a more complicated definition of a token:

1. Alphanumeric characters, dashes, apostrophes, exclamation points, and dollar signs are part of tokens, and everything else is a token separator.
2. Tokens that are all digits are ignored, along with HTML comments, not even considering them as token separators.
3. Case is preserved, and there is neither stemming nor stoplisting.
4. Periods and commas are constituents if they occur between two digits. This allows getting IP addresses and prices intact.
5. Price ranges like \$20–\$25 are mapped to two tokens, \$20 and \$25.
6. Tokens that occur within the To, From, Subject, and Return-Path lines, or within URLs, get marked accordingly. For example, 'foo' in the Subject line becomes 'Subject*foo.' (The asterisk could be any character you do not allow as a constituent.)

Graham obtained very good results on his personal email by using this token definition and an *ad hoc* version of a Bayesian learner. This definition has inspired other more sophisticated works in the field, but has also led spammers to focus on tokenization as one of the main vulnerabilities of learning-based filters. The current trend is just the opposite: making nearly no analysis of the text, considering any white-space separated string as a token, and letting the system learn from a really big number of messages (tens of thousands instead of thousands). Even more, HTML is not decoded, and tokens may include HTML tags, attributes, and values.

3.2.3.2 Multi-word Features. Some researchers have investigated features spanning over two or more tokens, seeking for 'get rich,' 'free sex,' or 'OEM software' patterns. Using statistical word phrases has not resulted into very good results in Information Retrieval [82], leading to even decreases in effectiveness. However, they have been quite successful in spam filtering. Two important works in this line are the ones by Zdziarski [102] and by Yerazunis [100, 101].

Zdziarski has first used case-sensitive words in his filter Dspam, and latter added what he has called ‘chained tokens.’ These tokens are sequences of two adjacent words, and follow the additional rules:

- There are no chains between the message header and the message body.
- In the message header, there are no chains between individual headers.
- Words can be combined with nonword tokens.

Chained tokens are not a replacement for individual tokens, but rather a complement to be used in conjunction with them for better analysis. For example, if we are analyzing an email with the phrase ‘CALL NOW, IT’s FREE!’, there are four tokens created under standard analysis (‘CALL,’ ‘NOW,’ ‘IT’s,’ and ‘FREE!’) and three more chained tokens: ‘CALL NOW,’ ‘NOW IT’s,’ ‘IT’s FREE!’. Chained tokens are traditionally named word bigrams in the fields of Language Modeling and Information Retrieval.

In Table I, we can see how chained tokens may lead to better (more accurate) statistics. In this table, we show the probability of spam given the occurrence of a word, which is the conditional probability usually estimated with the following formula¹⁸:

$$P(\text{spam}|w) \approx \frac{N(\text{spam}, w)}{N(w)}$$

where $N(\text{spam}, w)$ is the number of times that w occurs in spam messages, and $N(w)$ is the number of times the word w occurs. Counting can be done also *per message*: the number of spam messages in which w occurs, and the number of messages in which w occurs.

In the table, we can see that the words ‘FONT’ and ‘face’ have probabilities next to 0.5, meaning that they neither support spam nor legitimate email. However, the probability of the bigram is around 0.2, what represents a strong support to the legitimate class. This is due to the fact that spammers and legitimate users use

TABLE I
SOME EXAMPLES OF CHAINED TOKENS ACCORDING TO [102]

Token words	$P(\text{spam} w_1)$	$P(\text{spam} w_2)$	$P(\text{spam} w_1 * w_2)$
$w_1=\text{FONT}, w_2=\text{face}$	0.457338	0.550659	0.208403
$w_1=\text{color}, w_2=\#000000$	0.328253	0.579449	0.968415
$w_1=\text{that}, w_2=\text{sent}$	0.423327	0.404286	0.010099

¹⁸ Please note that this is the Maximum Likelihood Estimator.

different patterns of HTML code. While the firsts use the codes *ad hoc*, the seconds generate HTML messages with popular email clients like Microsoft Outlook or Mozilla Thunderbird, that always use the same (more legitimate) patterns, like putting the face attribute of the font next to the FONT HTML tag. We can also see how being ‘color’ and ‘#000000’ quite neutral, the pattern ‘color=#000000’ (the symbol ‘=’ is a separator) is extremely guilty. Zdziarski experiments demonstrate noticeable decreases in the error rates and especially in false positives, when using chained tokens plus usual tokens.

Yerazunis follows a slightly more sophisticated approach in [100] and [101]. Given that spammers had already begun to fool learning-based filters by disguising spam-like expressions with intermediate symbols, he proposed to enrich the feature space with bigrams obtained by combining tokens in a sliding 5-words window over the training texts. He has called this Sparse Binary Polynomial Hash (SBPH), and it is implemented in his CRM114 Discriminator filter. In a window, all pairs of sorted words with the second word being the final one are built. For instance, given the phrase/window ‘You can get porn free,’ the following four bigrams are generated: ‘You free,’ ‘can free,’ ‘get free,’ ‘porn free.’ With this feature and what the author calls the Bayesian Chain Rule (a simple application of the Bayes Theorem), impressive results have been obtained on his personal email, claiming that the 99.9% (of accuracy) plateau has been achieved.

3.2.3.3 Feature Selection and Extraction. Dimensionality reduction is a required step because it improves efficiency and reduces overfitting. Many algorithms perform very poorly when they work with a large amount of attributes (exceptions are k Nearest Neighbors or Support Vector Machines), so a process to reduce the number of elements used to represent documents is needed. There are mainly two ways to accomplish this task: feature selection and feature extraction.

Feature selection tries to obtain a subset of terms with the same or even greater predictive power than the original set of terms. For selecting the best terms, we have to use a function that selects and ranks terms according to how good they are. This function measures the quality of the attributes.

In the literature, terms are often selected with respect to their information gain (IG) scores [6, 79, 80], and sometimes according to *ad hoc* metrics [42, 71]. Information gain can be described as:

$$IG(X, C) = \sum_{x=0,1; c=s,l} P(X = x, C = c) \log_2 \frac{P(X = x, C = c)}{P(X = x) \times P(C = c)}$$

s being the spam class and l the legitimate email class in the above equation. Interestingly, IG is one of the best selection metrics [78]. Other quality metrics are

Mutual Information [35, 59], χ^2 [13, 99], Document Frequency [85, 99], or Relevancy Score [95].

Feature extraction is a technique that aims to generate an artificial set of terms different and smaller than the original one. Techniques used for feature extraction in Automated Text Categorization are Term Clustering and Latent Semantic Indexing.

Term Clustering creates groups of terms that are semantically related. In particular, cluster group words then can be synonyms (like thesaurus classes) or just in the same semantic field (like ‘pitcher,’ ‘ball,’ ‘homerun,’ and ‘baseball’). Term Clustering, as far as we know, has not been used in the context of spam filtering.

Latent Semantic Indexing [27] tries to alleviate the problem produced by polysemy and synonymy when indexing documents. It compresses index vectors creating a space with a lower dimensionality by combining original vectors using patterns of terms that appear together. This algebraic technique has been applied to spam filtering by Gee and Cook with moderate success [37].

3.2.4 Learning Algorithms

One of the most important parts in a document classification system is the learning algorithm. Given an ideally perfect classification function Φ that assigns each message a T/F value,¹⁹ learning algorithms have the goal to build a function $\bar{\Phi}$ that approximates the function Φ . The approximation function is usually named a classifier, and it often takes the form of model of the data it has been trained in. The most accurate the approximation is, the better the filter will perform.

A wide variety of learning algorithms families have been applied to spam classification, including the probabilistic Naïve Bayes [4, 6, 42, 71, 75, 79, 80], rule learners like Ripper [34, 71, 75], Instance Based k-Nearest Neighbors (kNN) [6, 42], Decision Trees like C4.5 [14, 34], linear Support Vector Machines (SVM) [30], classifiers committees like stacking [80] and Boosting [14], and Cost-Sensitive learning [39]. By far, the most often applied learner is the probability-based classifier Naive Bayes. In the next sections, we will describe the most important learning families.

To illustrate some of the algorithms that we are going to describe in following sections, we are going to use the public corpus SpamBase.²⁰ The SpamBase collection contains 4,601 messages, being 1,813 (39.4%) spam. This collection has been preprocessed, and the messages are not available in raw form (to avoid privacy problems). Each message is described in terms of 57 attributes, being the first 48

¹⁹ Being T equivalent to spam; that is, spam is the “positive” class because it is the class to be detected.

²⁰ This collection can be accessed at: <ftp://ftp.ics.uci.edu/pub/machine-learning-databases/spambase/>.

continuous real [0,100] attributes of type word_freq_WORD (percentage of words in the email that match WORD). A word is a sequence of alphanumeric characters, and the words have been selected as the most unbalanced ones in the collection. The last 9 attributes represent the frequency of special characters like '\$' and capital letters.

In order to keep examples as simple as possible, we have omitted the nonword features and selected the five attributes with higher Information Gain scores. We have also binarized the attributes, ignoring the percentage values. The word attributes we use are 'remove,' 'your,' 'free,' 'money,' and 'hp.' The first four words correspond to spammy words, as they occur mainly in spam messages (within expressions like 'click here to remove your email from this list,' 'get free porn,' or 'save money'), and the fifth word is a clue of legitimacy, as it is the acronym of the HP corporation in which the email message donors work.

3.2.4.1 Probabilistic Approaches. Probabilistic filters are historically the first filters and have been frequently used in recent years [46, 79]. This approach is mostly used in spam filters because of its simplicity and the very good results it can achieve.

This kind of classifiers is based on the Bayes Theorem [61], computing the probability for a document d to belong to a category c_k as:

$$P(c_k|d) = \frac{P(c_k) \cdot P(d|c_k)}{P(d)}$$

This probability can be used to make the decision about whether a document should belong to the category. In order to compute this probability, estimations about the documents in the training set are made. When computing probabilities for spam classification, we can obviate the denominator because we have only two classes (spam and legitimate), and one document cannot be classified in more than one of them, so denominator is the same for every k . $P(c_k)$ can be estimated as the number of documents in the training set belonging to the category, divided by the total number of documents.

Estimating $P(d|c_k)$ is a bit more complicated because we need in the training set some documents identical to the one we want to classify. When using Bayesian learners, it is very frequent to find the assumption that terms in a document are independent and the order they appear in the document is irrelevant. When this happens, the learner is called 'Naïve Bayes' learner [61, 65]. This way probability can be computed in the following way:

$$P(d|c_k) = \prod_{i=1}^T P(t_i|c_k)$$

T being the number of terms considered in the documents representation and t_i the i -th term (or feature) in the representation.

It is obvious that this assumption about term independency is not found in a real domain, but it helps to compute probabilities and rarely affects accuracy [28]. That is the reason why this is the approach used in most works.

The most popular version of a Naïve Bayes classifier is that by Paul Graham [46, 47]. Apart from using an *ad hoc* formula for computing terms probabilities, it makes use of only the 15 most extreme tokens, defined as those occurring in the message that have a probability far from the half point (0.5). This approach leads to more extreme probabilities, and has been proven more effective than using the whole set of terms occurring or not in a message. So strong is the influence of this work in the literature that learning-based filters have been quite often named Bayesian Filters despite using radically different learning algorithms.

3.2.4.2 Decision Trees. One of the biggest problems of probabilistic approaches is that results are not easy to understand for human beings, speaking in terms of legibility. In the Machine Learning field, there are some families of learners that are symbolic algorithms, with results that are easier to understand for people. One of those is the family of Decision Tree learners.

A Decision Tree is a finite tree with branches annotated with tests, and leaves being categories. Tests are usually Boolean expressions about term weights in the document. When classifying a document, we move from top to down in the tree, starting in the root and selecting conditions in branches that are evaluated as true. Evaluations are repeated until a leaf is reached, assigning the document to the category that has been used to annotate the leaf.

There are many algorithms used for computing the learning tree. The most important ones are ID3 [35], C4.5 [18, 56, 61], and C5 [62].

One of the simplest ways to induce a Decision Tree from a training set of already classified documents is:

1. Verify if all documents belong to the same category. Otherwise, continue.
2. Select a term t_i from the document representation and, for every feasible r weight values w_{ir} (i.e., 0 or 1), build a branch with a Boolean test $t_i = w_{ir}$, and a node grouping all documents that satisfy the test.
3. For each document group, go to 1 and repeat the process in a recursive way.

The process ends in each node when all grouped documents in it belong to the same category. When this happens, the node is annotated with the category name.

A critical aspect in this approach is how terms are selected. We can usually find functions that measure the quality of the terms according to how good they are separating the set of documents, using Information Gain, or Entropy metrics. This is basically the algorithm used by ID3 system [76]. This algorithm has been greatly improved using better test selection techniques, and tree pruning algorithms. C4.5 can be considered a state of the art in Decision Tree induction. In Fig. 5, we show a portion of the tree learned by C4.5 on our variation of the SpamBase collection. The figure shows how a message that does not contain the words ‘remove,’ ‘money,’ and ‘free’ is classified as legitimate (often called ham), and if does not contain ‘remove,’ ‘money,’ and ‘hp’ but ‘free,’ it is classified as spam. The triangles represent other parts of the tree, omitted for the sake of readability.

3.2.4.3 Rule Learners. Rules of the type ‘if-then’ are the base of one of the concept description languages most popular in Machine Learning field. On one hand, they allow one to present the knowledge extracted using learning algorithms in an easy to understand way. On the other hand, they allow the experts to examine and validate that knowledge, and combine it with other known facts in the domain.

Rule learners algorithms build this kind of conditional rules, with a logic condition on the left part of it, the premise, and the class name as the consequent, on the

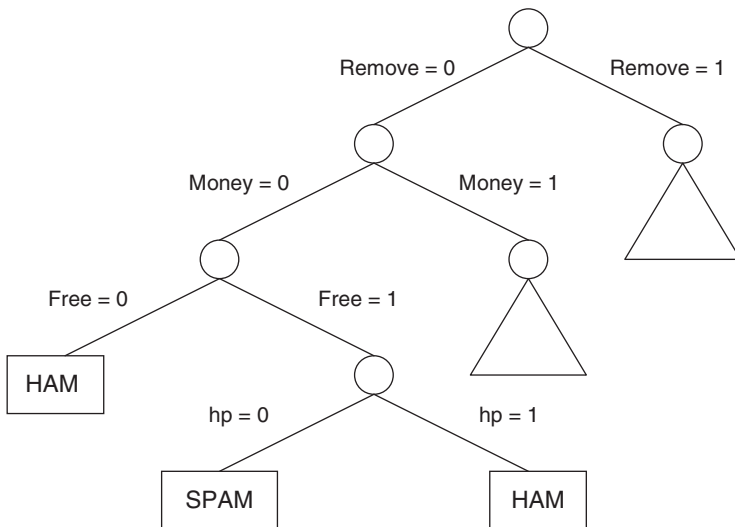


FIG. 5. Partial Decision Tree generated by C4.5 using the SpamBase corpus.

right part. The premise is usually built as a Boolean expression using weights of terms that appear in the document representation. For binary weights, conditions can be simplified to rules that look for if certain combination of terms appears or not in the document.

Actually there are several techniques used to induce rules. One of the most popular ones is the algorithm proposed in [68]. It consists of:

1. Iterate building one rule on each step, having the maximum classify accuracy over any subset of documents.
2. Delete those documents that are correctly classified by the rule generated in the previous step.
3. Repeat until the set of pending documents is empty.

As in other learners, the criteria for selecting the best terms to build rules in step 1 can be quality metrics like Entropy or Information Gain. Probably the most popular and effective rule learner is Ripper, applied to spam filtering in [34, 71, 75]. If the algorithm is applied to our variation of the SpamBase collection, we get the next four rules:

```
(remove = 1) => SPAM
((free = 1) and (hp = 0)) => SPAM
((hp = 0) and (money = 1)) => SPAM
() => HAM
```

The rules have been designed to be applied sequentially. For instance, the second rule is fired by a message that has not fired the first rule (and in consequence, does not contain the word ‘remove’), and that contains ‘free’ but not ‘hp.’ As it can be seen, the fourth rule is a default one that covers all the instances that are not covered by the previous rules.

3.2.4.4 Support Vector Machines. Support Vector Machines (SVM) have been recently introduced in Automatic Classification [55, 56], but they have become very popular rather quickly because of the very good results obtained with these algorithms especially in spam classification [22, 30, 39].

SVMs are an algebraic method, in which maximum margin hyperplanes are built in order to attempt to separate training instances, using, for example, Platt’s sequential minimal optimization algorithm (SMO) with polynomial kernels [72].

Training documents do not need to be linearly separable. Thus, the main method is based on calculation of an arbitrary hyperplane for separation. However, the simplest form of hyperplane is a plane, that is, a linear function of the attributes. Fast to learn, impressively effective in Text Categorization in general, and in spam

classification in particular, SVMs represent one of the leading edges in learning-based spam filters.

The linear function that can be obtained when using the SMO algorithm on our version of the SpamBase collection is:

$$f(m) = (-1.9999 * w_{\text{remove}}) + (-0.0001 * w_{\text{your}}) + (-1.9992 * w_{\text{free}}) \\ + (-1.9992 * w_{\text{money}}) + (2.0006 * w_{\text{hp}}) + 0.9993$$

This function means that given a message m , in which the weights of the words are represented by ‘w_word,’ being 0 or 1, the message is classified as legitimate if replacing the weights in the functions leads to a positive number. So, negative factors like -1.9 (for ‘remove’) are spammy, and positive factors like 2.0 (for ‘hp’) are legitimate. Note that there is an independent term (0.9993) that makes a message without any of the considered words being classified as legitimate.

3.2.4.5 *k*-Nearest Neighbors. Previous learning algorithms were based on building models about the categories used for classification. An alternative approach consists of storing training documents once they have been preprocessed and represented, and when a new instance has to be classified, it is compared to stored documents and assigned to the more appropriate category according to the similarity of the message to those in each category.

This strategy does not build an explicit model of categories, but it generates a classifier known as ‘instance based,’ ‘memory based,’ or ‘lazy’ [68]. The most popular one is kNN [99]. The k parameter represents the number of neighbors used for classification. This algorithm does not have a training step and the way it works is very simple:

1. Get the k more similar documents in the training set.
2. Select the most often category in those k documents.

Obviously, a very important part of this algorithm is the function that computes similarity between documents. The most common formula to obtain the distance between two documents is the ‘cosine distance’ [82], which is the cosine of the angle between the vectors representing the messages that are being compared. This formula is very effective, as it normalizes the length of the documents or messages.

In Fig. 6, we show a geometric representation of the operation of kNN. The document to be classified is represented by the letter D, and instances in the positive class (spam) are represented as X, while messages in the negative class are represented as O. In the left pane, we show the case of a linearly separable space. In that case, a linear classifier and a kNN classifier would give the same outcome (spam in this case). In the right pane, we can see a more mixed space, where kNN can show its

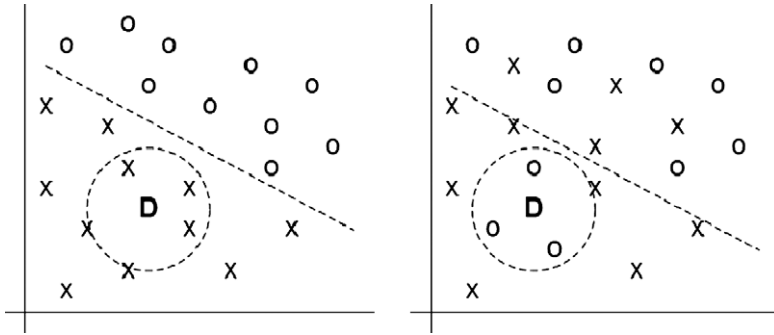


FIG. 6. Geometric representation of k-Nearest Neighbors (kNN) classifier. For making figure simpler, Euclidean distance has been used instead of ‘cosine distance.’

full power by selecting the locally most popular class (legitimate), instead of the one a linear classifier would learn (spam).

3.2.4.6 Classifier Committees. Another approach consists of applying different models to the same data, combining them to get better results. Bagging, boosting, and stacking are some of the techniques used to combine different learners.

The concept of bagging (voting for classification, averaging for regression-type problems with continuous dependent variables of interest) combines the predicted classifications (prediction) from multiple models, or from the same type of model for different learning data. Note that some weighted combination of predictions (weighted vote, weighted average) is also possible, and commonly used. A sophisticated (Machine Learning) algorithm for generating weights for weighted prediction or voting is the boosting procedure. The concept of boosting (applied to spam detection in [14]) is used to generate multiple models or classifiers (for prediction or classification), and to derive weights to combine the predictions from those models into a single prediction or predicted classification.

A simple algorithm for boosting works like this: Start by applying some method to the learning data, where each observation is assigned an equal weight. Compute the predicted classifications, and apply weights to the observations in the learning sample that are inversely proportional to the accuracy of the classification. In other words, assign greater weight to those observations that were difficult to classify (where the misclassification rate was high), and lower weights to those that were easy to classify (where the misclassification rate was low). Then apply the classifier again to the weighted data (or with different misclassification costs), and

continue with the next iteration (application of the analysis method for classification to the re-weighted data).

If we apply boosting to the C4.5 learner, with 10 iterations, we obtain 10 decision trees with weights, which are applied to an incoming message. The first tree is that of Fig. 5, with weight 1.88, and the last tree has got only a test on the word 'hp': if it does not occur in the message, it is classified as spam, and as legitimate otherwise. The weight of this last tree is only 0.05.

The concept of stacking (short for Stacked Generalization) [80] is used to combine the predictions from multiple models. It is particularly useful when the types of models included in the project are very different. Experience has shown that combining the predictions from multiple methods often yields more accurate predictions than can be derived from any one method [97]. In stacking, the predictions from different classifiers are used as input into a meta-learner, which attempts to combine the predictions to create a final best predicted classification.

3.2.4.7 Cost-Sensitive Learning. When talking about spam filtering, we have to take into account that costs of misclassifications are not balanced in real life, as the penalty of a false positive (a legitimate message classified as spam) is much higher than the one of a false negative (a spam message classified as legitimate). This is due to the risk of missing important valid messages (like those from the users' boss!) because messages considered spam can be immediately purged or, in a more conservative scenario, conserved in a quarantine that the user rarely screens. The algorithms commented above assume balanced misclassification costs by default, and it is wise to use techniques to make those algorithms cost-sensitive, in order to build more realistic filters [39].

Thresholding is one of the methods used for making algorithms cost-sensitive. Once a numeric-prediction classifier has been produced using a set of pre-classified instances (the training set), one can compute a numeric threshold that optimizes cost on another set of pre-classified instances (the validation set). When new instances are to be classified, the numeric threshold for each of them determines if the instances are classified as positive (spam) or negative (legitimate). The cost is computed in terms of a cost matrix that typically assigns 0 cost to the hits, a positive cost to false negatives, and a much bigger cost to false positives. This way, instead of optimizing the error or the accuracy, the classifier optimizes the cost.

The weighting method consists of re-weighting training instances according to the total cost assigned to each class. This method is equivalent to stratification by over-sampling as described in [29]. The main idea is to replicate instances of the most costly class, to force the Machine Learning algorithm to correctly classify that class

instances. Another effective cost-sensitive meta-learner is the MetaCost method [29], based on building an ensemble of classifiers using the bagging method, relabeling training instances according to cost distributions and the ensemble outcomes, and finally training a classifier on the modified training collection.

In [39], the experiments with a number of Machine Learning algorithms and the three previous cost-sensitive schemas have shown that the combination of weighting and SVM is the most effective one.

3.3 Filtering by Compression

Compression has recently emerged as a new paradigm for Text Classification in general [90], and for spam filtering in particular [11]. Compression demonstrates high performance in Text Categorization problems in which classification depends on nonword features of a document, such as punctuation, word stems, and features spanning more than one word, like dialect identification and authorship attribution. In the case of spam filtering, they have emerged as the top performers in competitive evaluations like TREC [19].

An important problem of Machine Learning algorithms is the dependence of the results obtained with respect to their parameter settings. In simple words, a big number of parameters can make it hard to find the optimal combination of them, that is, the one that leads to the most general and effective patterns. Keogh et al. discuss the need of a parameter-free algorithm:

Data mining algorithms should have as few parameters as possible, ideally none. A parameter-free algorithm prevents us from imposing our prejudices and presumptions on the problem at hand, and let the data itself speak to us. [57] (p. 206)

Keogh presents data compression as a Data Mining paradigm that realizes this vision. Data compression can be used as an effective Machine Learning algorithm, especially on text classification tasks. The basic idea of using compression in a text classification task is to assign a text item to the class that best compresses it. This can be straightforwardly achieved by using any state-of-the-art compression algorithm and a command line process. As a result, the classification algorithm (the compression algorithm plus a decision rule) is easy to code, greatly efficient, and it does not need any preprocessing of the input texts. In other words, there is no need to represent it as a feature vector, avoiding one of the most difficult and challenging tasks, that is, text representation. As a side effect, this makes especially hard to reverse engineer the classification process, leading to more effective and stronger spam detection systems.

The rule of classifying a message into the class that best compresses it is a straightforward application of the Minimum Description Length Principle (MDL) [11] that favors the most compact (short) explanation of the data. The recent works by Sculley and Brodley [84] and by Bratko et al. [11] formalize this intuition in a different way; we will follow partially the work by Sculley and Brodley.

Let $C(\cdot)$ be a compression algorithm. A compression algorithm is a function that transforms strings into (shorter) strings.²¹ A compression algorithm usually generates a (possibly implicit) model. Let $C(X|Y)$ also be the compression of the string Y using the model generated by compressing the string X . We denote by $|S|$ the length of a string S , typically measured as a number of bits, and by XY the string Y appended to the string X .

The MDL principle states that, given a class A of text instances, a new text X should be assigned to A if it compresses X better than A^C . If we interpret the class A as a sequence of texts (and so it is A^C), the decision rule may be:

$$\text{class}(X) = \arg \min_{c=A, A^C} \{|C(c|X)|\}$$

This formula is one possible decision rule for transforming a compression algorithm into a classifier. The decision rule is based on the ‘approximate’ distance²² $|C(A|X)|$. Sculley and Brodley [84] review a number of metrics and measures that are beyond the scope of this presentation.

The length of the text X compressed with the model obtained from a class A can be approached by compressing AX :

$$|C(A|X)| \approx |C(AX)| - |C(A)|$$

This way, any standard compressor can be used to predict the class of a text X , given the classes A and A^C .

Following [11], there are two basic kinds of compressors: two part and adaptive coders. The first class of compressors first trains a model over the data to encode, and then encode the data. These require two passes over the data. These kinds of encoders append the data to the model, and the decoder reads the model and then decodes the data. The most classical example of this kind of compressors is a double pass Huffman coder, which accumulates the statistics for the observed symbols, builds a statistically optimal tree using a greedy algorithm, and builds a file with the tree and the encoded data.

²¹ This presentation can be made in terms of sequences of bits instead of strings as sequences of characters.

²² This is not a distance in the proper sense, as it does not satisfy all the formal requirements of a distance.

Adaptive compressors instead start with an empty model (e.g., a uniform distribution over all the symbols), and update it as they are encoding the data. The decoder repeats the process, building its own version of the model as the decoding progresses. Adaptive coders require a single pass over the data, so they are more efficient in terms of time. They have also reached the quality of two-part compressors in terms of compressing ratio, and more interestingly for our purposes, they make the previous approximation an equality. Examples of adaptive compressors include all of those used in our work, which we describe below. For instance, the Dynamic Markov Compression (DMC), the Prediction by Partial Matching (PPM), and the family of Lempel-Ziv (LZ) algorithms are all adaptive methods (see [11] and [84]).

Let us get back to the parameter issue. Most compression algorithms do not require any preprocessing of the input data. This clearly avoids the steps of feature representation and selection usually taken when building a learning-based classifier. Also, most often compressors have a relatively small number of parameters, approaching the vision of a parameter-free or parameter-light Data Mining. However, a detailed analysis performed by Sculley and Brodley in [84] brings light to this point, as they may depend on explicit parameters in compression algorithms, the notion of distance used, and the implicit feature space defined by each algorithm. On the other hand, it is extremely easy to build a compression-based classifier by using the rules above and a standard out-of-the-shelf compressor, and they have proven to be effective on a number of tasks, and top-performer in spam filtering. In words by Cormack (one of the organizers of the reputed TREC spam Track competition) and Bratko:

“At TREC 2005, arguably the best-performing system was based on adaptive data compression methods”, and “one should not conclude, for example, that SVM and LR (Logistic Regression) are inferior for on-line filtering. One may conclude, on the other hand, that DMC and PPM set a new standard to beat on the most realistic corpus and test available at this time.” [21]

While other algorithms have been used in text classification, it appears that only DMC and PPM have been used in spam filtering. The Dynamic Markov Compression [11] algorithm models an information source with a finite state machine (FSM). It constructs two variable-order Markov models (one for spam and one for legitimate email), and classifies a message according to which of the models predicts it best. The Partial Prediction Matching algorithm is a back-off smoothing technique for finite-order Markov models, similar to back-off models used in natural language processing, and has set the standard for lossless text compression since its introduction over two decades ago according to [17]. The best of both is DMC, but PPM is better known and very competitive with it (and both superior to Machine Learning approaches at spam filtering). There are some efforts that also work at the character level (in combination with some *ad hoc* scoring function), but they are different

from compression as this has been principle designed to model sequential data. In particular, the IBM's Chung-Kwei system [77] uses pattern matching techniques originally developed for DNA sequences, and the Pampapathi et al. [70] have proposed a filtering technique based on the suffix tree data structure. This reflects the trend of minimizing tokenization complexity in order to avoid one of the most relevant vulnerabilities of learning-based filters.

3.4 Comparison and Summary

Heuristic filtering relying on the manual effort of communities of experts has been quickly beaten by spammers, as they have stronger (economic) motivation and time. But automated with content-based methods, a new episode in the war against spam is being written. The impact of and success of content-based spam filtering using Machine Learning and compression is significant, as the spammers have invested much effort to avoid this kind of filtering, as we review below. The main strength of Machine Learning is that it makes the filter adapted to the actual users' email. As every user is unique, every filter is different, and it is quite hard to avoid all vendors' and users' filters simultaneously.

The processing structure of learning-based filters includes a tokenization step in which the system identifies individual features (typically strings of character, often meaningful words) on which the system relies to learn and classify. This step has been recognized as the major vulnerability of learning-based filtering, and the subject of most spammers' attacks, in the form of tokenization attacks and image spam. The answer from the research community has been fast and effective, focusing on character-level modeling techniques based on compression, that detect *implicit* patterns that not even spammers know they are using! Compression-based methods have proven top effective in competitive evaluations, and in consequence, that can be presented as the current (successful) trend in content-based spam filtering.

4. Spam Filters Evaluation

Classifier system evaluation is a critical point: no progress may be expected if there is no way to assess it. Standard metrics, collections, and procedures are required, which allow cross-comparison of research works. The Machine Learning community has well-established evaluation methods, but these have been adapted and improved when facing what is probably the main difficulty in spam filtering: the problem of asymmetric misclassification costs. The fact that a false positive

(a legitimate message classified as spam) is much more damaging than a false negative (a spam classified as legitimate) implies that evaluation metrics must attribute bigger weights to worse errors, but ... which weights? We analyze this point in [Section 4.3](#).

Scientific evaluations have well-defined procedures, consolidated metrics, and public test collections that make cross-comparison of results relatively easy. The basis of scientific experiments is that they have to be reproducible. There are a number of industrial evaluations, performed by the filter vendors themselves and typically presented in their white papers, and more interestingly, those performed by specialized computer magazines. For instance, in [53], the antispam appliances BorderWare MXtreme MX-200 and Proofpoint P800 Message Protection Appliance are compared using an *ad hoc* list of criteria, including Manageability, Performance, Ease of Use, Setup, and Value. Or in [3], 10 systems are again tested according to a different set of criteria, allowing no possible comparison. The limitations of industrial evaluations include self-defined criteria, private test collections, and self-defined performance metrics. In consequence, we focus on scientific evaluations in this chapter.

The main issues in the evaluation of spam filters are the test collections, the running procedure, and the evaluation metrics. We discuss these issues in the next subsections, with special attention to what we consider a real and accurate standard in current spam filter evaluation, the TREC spam Track competition.

4.1 Test Collections

A test collection is a set of manually classified messages that are sent to a classifier in order to measure its effectiveness, in terms of hits and mistakes. It is important that test collections are publicly available, because it allows the comparison of approaches and the improvement of the technology. On the other hand, message collections may include private email, so privacy protection is an issue. There are a number of works in which the test collections employed are kept private, as they are personal emails from the author or are donated to them with the condition of being kept private, like in early works [30], [75], and [79], or even in more recent works like [36], [47], and [69]. The privacy problem can be solved in different ways:

- Serving a processed version of the messages that does not allow rebuilding them. This approach has been followed in the SpamBase, PU1, and the 2006 ECML-PKDD Discovery Challenge public collections.
- Building the collection using only messages from public sources. This is the approach in the Lingspam and SpamAssassin Public Corpus.

- Keeping the collection private in the hands of a reputable institution that performs the testing on behalf of the researchers. The TREC spam Track competition is such an institution, and makes some test collections public, and keeps some others private.

In the next paragraphs, we present the test collections that have been publicly available, solving the privacy problem:

- SpamBase²³ is an email message collection containing 4,601 messages, being 1,813 (39%) marked as spam. The collection comes in preprocessed (not raw) form, and its instances have been represented as 58-dimensional vectors. The first 48 features are words extracted from the original messages, without stop list or stemming, and selected as the most unbalanced words for the UCE class. The next 6 features are the percentage of occurrences of the special characters ‘;,’ ‘(,’ ‘[,’ ‘!,’ ‘\$,’ and ‘#.’ The following 3 features represent different measures of occurrences of capital letters in the text of the messages. Finally, the last feature is the class label. This collection has been used in, for example, [34], and its main problem is that it is preprocessed, and in consequence, it is not possible to define and test other features apart from those already included.
- The PU1 corpus,²⁴ presented in [6], consists of 1,099 messages, being 481 (43%) spam and 618 legitimate. Been received by Ion Androutsopoulos, it has been processed by removing attachments and HTML tags. To respect privacy issues, in the publicly available version of PU1, fields other than ‘Subject:’ have been removed, and each token (word, number, punctuation symbol, etc.) in the bodies or subjects of the messages was replaced by a unique number, the same number throughout all the messages. This hashing ‘encryption’ mechanism makes impossible to perform experiments with other tokenization techniques and features apart from those included by the authors. It has been used in, for example, [14] and [52].
- The ECML-PKDD 2006 Discovery Challenge collection²⁵ has been collected by the challenge organizers in order to test how to improve spam classification using untagged data [9]. It is available in a processed form: strings that occur fewer than four times in the corpus are eliminated, and each message is represented by a vector indicating the number of occurrences of each feature in the message. The same comments to PU1 are applicable to this collection.

²³ This collection has been described above, but some information is repeated for the sake of comparison.

²⁴ Available at: <http://www.aueb.gr/users/ion/data/PU123ACorpora.tar.gz>.

²⁵ Available at: <http://www.ecmlpkdd2006.org/challenge.html>.

- The Lingspam test collection,²⁶ presented in [4] and used in many studies (including [14], [39], and very recent ones like [21]), has been built by mixing spam messages with messages extracted from spam-free public archives of mailing lists. In particular, the legitimate messages have been extracted from the Linguist list, a moderated (hence, spam-free) list about the profession and science of linguistics. The number of legitimate messages is 2,412, and the number of spam messages is 481 (16%). The Linguist messages are, of course, more topic-specific than most users' incoming email. They are less standardized, and for instance, they contain job postings, software availability announcements, and even flame-like responses. In consequence, the conclusions obtained from experiments performed on it are limited.
- The SpamAssassin Public Corpus²⁷ has been collected by Justin Mason (a SpamAssassin developer) with the public contributions of many others, and consists of 6,047 messages, being 1,897 (31%) spam. The legitimate messages (named 'ham' in this collection) have been further divided into easy and hard (the ones that make use of rich HTML, colored text, spam-like words, etc.). As it is relatively big, realistic, and public, it has become the standard in spam filter evaluation. Several works make use of it (including [11], [21], [22], and [67]), and it has been routinely used as a benchmark in the TREC spam Track [19].

Apart from these collections, the TREC spam Track features several public and private test collections, like the TREC Public Corpus – trec05p-1, and the Mr. X, S.B., and T.M. Private Corpora. For instance, the S.B. corpus consists of 7,006 messages (89% ham, 11% spam) received by an individual in 2005. The majority of all ham messages stems from four mailing lists (23%, 10%, 9%, and 6% of all ham messages) and private messages received from three frequent correspondents (7%, 3%, and 2%, respectively), while the vast majority of the spam messages (80%) are traditional spam: viruses, phishing, pornography, and Viagra ads.

Most TREC collections have two very singular and interesting properties:

1. Messages are chronologically sorted, allowing testing the effect of incremental learning, what we mention below as online testing.
2. They have build by using an incremental procedure [20], in which messages are tagged using several antispam filters, and the classification is reviewed by hand when a filter disagrees.

TREC collections are also very big in comparison with the previously described ones, letting the researchers arriving at more trustable conclusions.

²⁶ Available at: http://www.aueb.gr/users/ion/data/lingspam_public.tar.gz.

²⁷ Available at: <http://spamassassin.apache.org/publiccorpus/>.

In other words, TREC spam Track has set the evaluation standard for antis spam learning-based filters.

4.2 Running Test Procedure

The most frequent evaluation procedure for a spam filter is batch evaluation. The spam filter is trained on a set of messages, and applied to a different set of test messages. The test messages are labeled, and it is possible to compare the judgment of the filter and of the expert, computing hits and mistakes. It is essential that the test collection is similar to, but disjoint of, the training one, and that it reflects operational settings as close as possible. This is the test procedure employed in most evaluations using the SpamBase, PU1, and Linspam test collections.

A refinement of this evaluation is to perform N -fold cross-validation. Instead of using a separate test collection, portions of the labeled collection are sometimes used as training and as test sets. In short, the labeled collection is randomly divided into N sets or folds (preserving the class distribution), and N tests are run, using $N-1$ folds as training set, and the remaining one as test set. The results are averaged over the N runs, leading to more statistically valid figures, as the experiment does not depend on unpredictable features of the data (all of them are used for training and testing). This procedure has been sometimes followed in spam filtering evaluation, like in [39].

A major criticism to this approach is that the usual operation of spam filters allows them to learn from the mistakes they made (if the user reports them, of course). The batch evaluation does not allow the filters to learn as it classifies, and ignores chronological ordering if available. The TREC organizers have instead approached filter testing as an on-line learning task in which messages are presented to the filter, one at a time, in chronological order. For each message, the filter predicts its class (spam or legitimate) by computing a score S which is compared to a fixed but arbitrary threshold T . Immediately after the prediction, the true class is presented to the filter so that it might use this information in future predictions. This evaluation procedure is supported with a specific set of scripts, requiring a filter to implement the next command-line functions:

- *Initialize* – creates any files or servers necessary for the operation of the filter.
- *Classify message* – returns ham/spam classification and spamminess score for message.
- *Train ham message* – informs filter of correct (ham or legitimate) classification for previously classified message.
- *Train spam message* – informs filter of correct (spam) classification for previously classified message.
- *Finalize* – removes any files or servers created by the filter.

This is the standard procedure used in TREC. An open question is if both methods are equivalent, in term of the results obtained in public works. Cormack and Bratko have made in [21] a systematic comparison of a number of top-performing algorithms following both procedures, and arriving at the conclusion that the current leaders are compression methods, and that the online procedure is more suitable because it is closer to operational settings.

4.3 Evaluation Metrics

We have divided the metrics used in spam filtering test studies into three groups: the basic metrics employed in the initial works, the ROCCH method as a quite advanced one that addresses prior errors, and the TREC metrics as the current standard.

4.3.1 Basic Metrics

The effectiveness of spam filtering systems is measured in terms of the number of correct and incorrect decisions. Let us suppose that the filter classifies a given number of messages. We can summarize the relationship between the system classifications and the correct judgments in a confusion matrix, like that shown in Table II. Each entry in the table specifies the number of documents with the specified outcome. For the problem of filtering spam, we take spam as the positive class (+) and legitimate as the negative class (−). In this table, the key ‘tp’ means ‘number of true-positive decisions’ and ‘tn,’ ‘fp,’ and ‘fn’ refer to the number of ‘true-negative,’ ‘false-positive,’ and ‘false-negative’ decisions, respectively.

Most traditional TC evaluation metrics can be defined in terms of the entries of the confusion matrix. F_1 [85] is a measure that gives equal importance to recall and precision. Recall is defined as the proportion of class members assigned to a category by a classifier. Precision is defined as the proportion of correctly assigned

TABLE II
A SET OF N CLASSIFICATION DECISIONS REPRESENTED
AS A CONFUSION MATRIX

	+	−
+	tp	fp
−	fn	tn

documents to a category. Given a confusion matrix like the one shown in the table, recall (R), precision (P), and F_1 are computed using the following formulas:

$$R = \frac{tp}{tp + fn}$$

$$P = \frac{tp}{tp + fp}$$

$$F_1 = \frac{2RP}{R + P}$$

Recall and precision metrics have been used in some of the works in spam filtering (e.g., [4–6, 42, 80]). Other works make use of standard ML metrics, like accuracy and error [71, 75]. Recall that not all kinds of classification mistakes have the same importance for a final user. Intuitively, the error of classifying a legitimate message as spam (a false positive) is far more dangerous than classifying a spam message as legitimate (a false negative). This observation can be re-expressed as the cost of a false positive is greater than the cost of a false negative in the context of spam classification. Misclassification costs are usually represented as a cost matrix in which the entry $C(A,B)$ means the cost of taking a A decision when the correct decision is B , that is the cost of A given B ($\text{cost}(A|B)$). For instance, $C(+,-)$ is the cost of a false-positive decision (classifying legitimate email as spam) and $C(-,+)$ is the cost of a false-negative decision.

The situation of unequal misclassification costs has been observed in many other ML domains, like fraud and oil spills detection [74]. The metric used for evaluating classification systems must reflect the asymmetry of misclassification costs. In the area of spam filtering, several cost-sensitive metrics have been defined, including weighted accuracy (WA), weighted error (WE), and total cost ratio (TCR) (see e.g., [5]). Given a cost matrix, the cost ratio (CR) is defined as the cost of a false positive over the cost of a false negative. Given the confusion matrix for a classifier, the WA, WE, and TCR for the classifier are defined as:

$$WA = \frac{CR \times tn + tp}{CR(tn + fp) + (tp + fn)}$$

$$WE = \frac{CR \times fp + fn}{CR(tn + fp) + (tp + fn)}$$

$$TCR = \frac{tn + fp}{CR \times fp + fn}$$

The WA and WE metrics are versions of the standard accuracy and error measures that penalize those mistakes that are not preferred. Taking the trivial rejecter that classifies every message as legitimate (equivalent to not using a filter) as a baseline, the TCR of a classifier represents to what extent is a classifier better than it. These metrics are less standard than others used in cost-sensitive classification, as Expected Cost, but to some extent they are equivalent. These metrics have been calculated for a variety of classifiers, in three scenarios corresponding to three CR values (1, 9, and 999) [4–6, 42, 80].

The main problem presented in the literature on spam cost-sensitive categorization is that the CR used does not correspond to real world conditions, which are unknown and may be highly variable. There is no evidence that a false positive is neither 9 nor 999 times worse than the opposite mistake. As class distributions, CR values may vary from user to user, from corporation to corporation, and from ISP to ISP. The evaluation methodology must take this fact into account. Fortunately, there are methods that allow evaluating classifiers effectiveness when target (class distribution and CR) conditions are not known, as in spam filtering. In the next subsection, we introduce the ROCCH method for spam filtering.

4.3.2 The ROCCH Method

The receiver operating characteristics (ROC) analysis is a method for evaluating and comparing a classifiers performance. It has been extensively used in signal detection, and introduced and extended by Provost and Fawcett in the Machine Learning community (see e.g., [74]). In ROC analysis, instead of a single value of accuracy, a pair of values is recorded for different class and cost conditions a classifier is learned. The values recorded are the false-positive (FP) rate and the true-positive (TP) rate, defined in terms of the confusion matrix as:

$$FP = \frac{fp}{fp + tn}$$

$$TP = \frac{tp}{tp + fn}$$

The TP rate is equivalent to the recall of the positive class, while the FP rate is equivalent to 1 less than the recall of the negative class. Each (FP,TP) pair is plotted as a point in the ROC space. Most ML algorithms produce different classifiers in different class and cost conditions. For these algorithms, the conditions are varied to obtain a ROC curve. We will discuss how to get ROC curves by using methods for making ML algorithms cost-sensitive.

One point on a ROC diagram dominates another if it is above and to the left, that is, has a higher TP and a lower FP. Dominance implies superior performance for a variety of common performance measures, including expected cost (and then WA and WE), recall, and others. Given a set of ROC curves for several ML algorithms, the one which is closer to the left upper corner of the ROC space represents the best algorithm.

Dominance is rarely got when comparing ROC curves. Instead, it is possible to compute a range of conditions in which one ML algorithm will produce at least better results than the other algorithms. This is done through the ROC convex hull (ROCCH) method, first presented in [74]. Concisely, given a set of (FP,TP) points, that do not lie on the upper convex hull, corresponds to suboptimal classifiers for any class and cost conditions. In consequence, given a ROC curve, only its upper convex hull can be optimal, and the rest of its points can be discarded. Also, for a set of ROC curves, only the fraction of each one that lies on the upper convex hull of them is retained, leading to a slope range in which the ML algorithm corresponding to the curve produces best performance classifiers. An example of ROC curves taken from [39] is presented in Fig. 7. As it can be seen, there is no single dominator.

The ROC analysis allows a visual comparison of the performance of a set of ML algorithms, regardless of the class and cost conditions. This way, the decision of which is the best classifier or ML algorithm can be delayed until target (real world) conditions are known, and valuable information can be obtained at the same time. In the most advantageous case, one algorithm is dominant over the entire slope range.

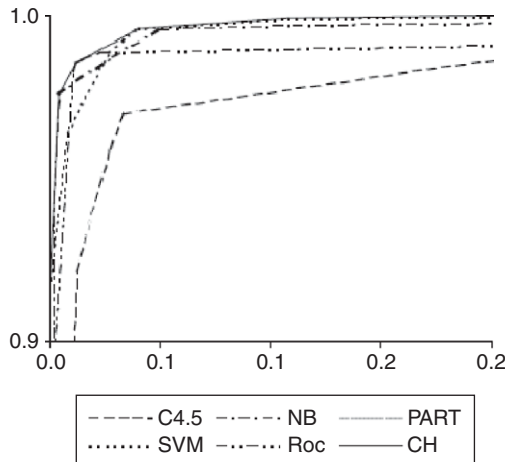


FIG. 7. A ROC curve example.

Usually, several ML algorithms will lead to classifiers that are optimal (among those tested) for different slope ranges, corresponding to different class and cost conditions.

Operatively, the ROCCH method consists of the following steps:

1. For each ML algorithm, obtain a ROC curve and plot it (or only its convex hull) on the ROC space.
2. Find the convex hull of the set of ROC curves previously plotted.
3. Find the range of slopes for which each ROC curve lies on the convex hull.
4. In case the target conditions are known, compute the corresponding slope value and output the best algorithm. In other case, output all ranges and best local algorithms or classifiers.

We have made use of the ROCCH method for evaluating a variety of ML algorithms for the problem of spam filtering in [39]. This is the very first time that ROC curves have been used in spam filtering testing, but they have become a standard in TREC evaluations.

4.3.3 TREC Metrics

TREC Spam Track metrics [19] are considered also a standard in terms of spam filtering evaluation. The main improvement over the ROC method discussed above is their adaptation to the online procedure evaluation.

As online evaluation allows a filter to learn from immediately classified errors, its (TP,FP) rate is always changing and the values improving with time. The ROC graph is transformed into a single numeric figure, by computing the Area Under the ROC curve (AUC). As the evaluated filters are extremely effective, it is better to report the inverse $1 - \text{AUC}$ value, which is computed over time as the learning-testing online process is active. This way, it is possible to get an idea of how quickly the systems learn, and at which levels of error the performance arrives a plateau. In Fig. 8, a ROC Learning Curve graph is also shown, taken from [19] for Mr. X text collection. It is easy to see how filters start committing a relatively high number of mistakes at the beginning of the ROC Learning Curve, and they improve their result finally achieving an average performance level.

5. Spam Filters in Practice

Implementing effective and efficient spam filters is a nontrivial work. Depending on concrete needs, it should be implemented in a certain way and the management of spam messages can vary depending on daily amount of messages, the estimated impact of possible false positives, and other peculiarities of users and companies.

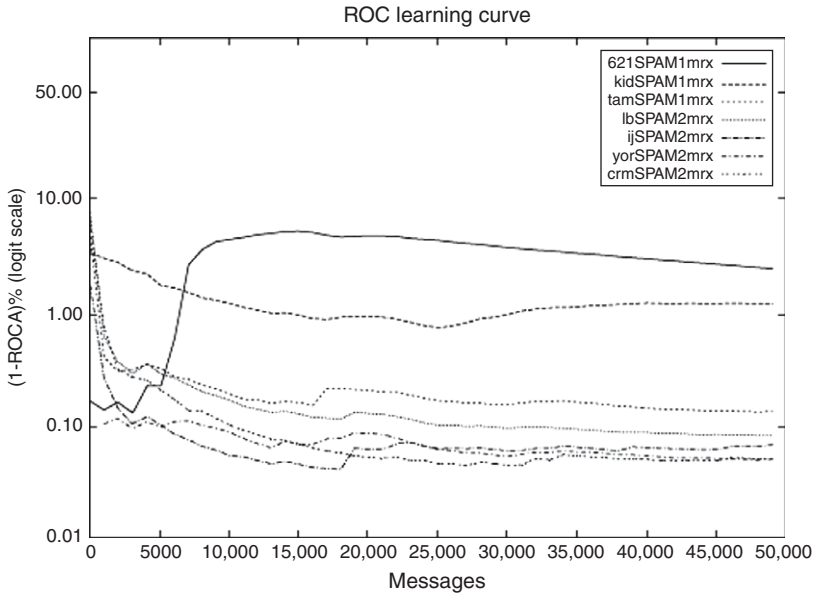


FIG. 8. A ROC learning curve from TREC.

Figure 9 shows a diagram of the mail transportation process, where spam filters can be implemented at any location in this process, although the most common and logical ones are in the receiver's SMTP Server/Mail Delivery Agent (server side) or in the receiver's email client (client side). There are also some enterprise solutions that perform the spam filtering in external servers, which is not reflected in this diagram as not being usual at the current moment.

There are important questions that we cannot obviate and we need to test carefully for their response before implementing or implanting a spam filter solution. Should we implement our spam filter in the client side or in the server side? Should we use any kind of collaborative filtering? Should the server delete the spam messages or the user should decide what to do with those undesirable messages?

5.1 Server Side Versus Client Side Filtering

The first important choice when implementing or integrating a spam filter in our IT infrastructure is the location of the filter: in the client side or in the server side. Each location has its own benefits and disadvantages and those must be measured to choose the right location depending on the user/corporation concrete needs.

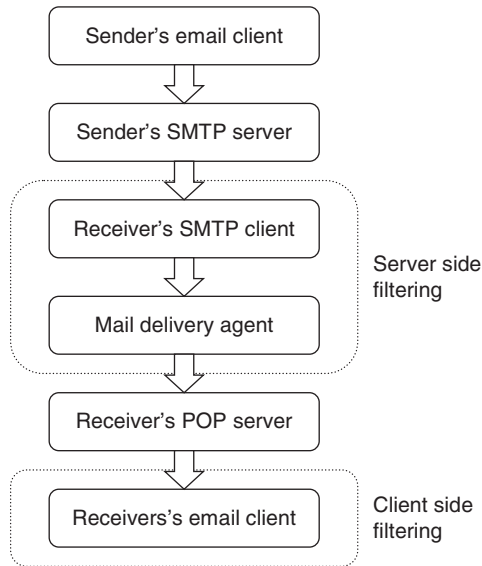


FIG. 9. Email transport process and typical location of server and client side filters. Filters can, virtually, be implemented in any step of the process.

The main features of server side spam filters are:

- They reduce the network load, as the server does not send the mails categorized as spam, which can be the biggest part of the received messages.
- They also reduce the computational load on the client side, as the mail client does not need to check each received message. This can be very helpful when most clients have a low load capacity (handhelds, mobile phones, etc.).
- They allow certain kinds of collaborative filtering or a better integration of different antispam techniques. For example, when detecting spam messages received in different users' account from the same IP, this can be added to a black list preventing other users to receive messages from the same spammer.

On the other side, client side spam filters:

- Allow a more personalized detection of management of the messages.
- Integrate the knowledge from several email accounts belonging to the same user, preventing the user to receive the same spam in different email accounts.
- Reduce the need of dedicated mail servers as the computation is distributed among all the users' computers.

As corporations and other organizations grow, their communications also need to grow and they increasingly rely on dedicated servers and appliances that are responsible for email services, including the email server, and the full email security suite of services, including the antivirus and antispam solutions. Appliances and dedicated servers are the choice for medium to big corporations, while small organizations and individual users can group the whole Internet services in one machine. In big corporations, each Internet-related service has its own machine, as the computational requirements deserve special equipment, and it is a nice idea to distribute the services in order to minimize risks.

5.2 Quarantines

Using statistical techniques is really simple to detect almost all the received spam but a really difficult, almost impossible, point is to have a 0 false-positive ratio. As common bureaucratic processes and communication are delegated to be conducted via email, false positives are being converted into a really annoying question; losing only one important mail can have important economic consequences or, even, produce a delay in some important process.

On the other hand as each user is different, what should be done with a spam message should be left to the user's choice. A clocks lover user would conserve Rolex-related spam. A key point here is that spam is sent because the messages sent are of interest to a certain set of users.

These points are difficult to solve with server spam filter solution as the spam filter is not as personalized as a spam filter implemented on the client side. Quarantine is a strategy developed to face these points. The messages detected as spam by the spam filter are stored in the server for a short period of time and the server mails a Quarantine Digest to the user reporting all the messages under quarantine. The user is given the choice of preserving those messages or deleting them.

Quarantine is a helpful technique that allows to:

- Reduce the disk space and resources the spam is using on the mail servers.
- Reduce the user's level of frustration when they receive spam.
- Keeps spam from getting into mailing lists.
- Prevent auto replies (vacation, out of office, etc.) from going back to the spammers.

In Fig. 10, we show an example of quarantine, in the Trend Micro InterScan Messaging Security Suite, a solution that includes antivirus and antispam features, and that is designed for serve side filtering. The quarantine is accessed by actual



Spam Quarantine			Log Off 
Approved Senders [15] (Of Max 50 addresses)		Display: 15 per page 	
<input type="button" value="Delete"/> <input type="button" value="Not Spam"/>		1 - 15 of 33    	
<input type="checkbox"/> Sender	Subject	Received	
<input type="checkbox"/> baller_34_21@starproperty.co.nz	Requested info	14/10/07 9:23:01	
<input type="checkbox"/> raquel@influenza.etc.br	20th October NEURÓTICA:BIOII @ Intermediae Matadero, Madrid	13/10/07 22:00:06	
<input type="checkbox"/> raquel@influenza.etc.br	20th October NEURÓTICA:BIOII @ Intermediae Matadero, Madrid	13/10/07 22:00:06	
<input type="checkbox"/> a-aimew@abacus.cz	Let's chat	13/10/07 20:48:17	
<input type="checkbox"/> a-aimew@abacus.cz	Let's chat	13/10/07 20:48:17	
<input type="checkbox"/> wkxnayn@book-keepingnetwork.com.au	Download cheap softwares in a matter of seconds	13/10/07 12:26:14	
<input type="checkbox"/> wkxnayn@book-keepingnetwork.com.au	Download cheap softwares in a matter of seconds	13/10/07 12:26:14	
<input type="checkbox"/> herschel.danby@we-ga.dk	Get ready for sex in 15 min	13/10/07 9:28:11	

FIG. 10. An example of list of messages in the Trend Micro InterScan Messaging Security Suite.

users through a web application, where they log and screen the messages that the filter has considered spam. This quarantine in particular features the access to a white list of approved senders, a white list where the user can type in patterns that make the filter ignore messages, like those coming from the users' organization.

5.3 Proxying and Tagging

There are a number of email clients that currently implement their own content-based filters, most often based on the Graham's Bayesian algorithm. For instance, the popular open-source Mozilla Thunderbird client includes a filter that is able to learn from the users' actual email, leading to better personalization and increased effectiveness.

However, there are a number of email clients that do not feature an antispam filter at all. Although there are a number of extensions or plugins for popular email clients (like the ThunderBayes extension for Thunderbird, or the SpamBayes Microsoft

Outlook plugin – both including the SpamBayes filter controls into the email clients), the user may wish to keep the antispam software out of the email client, in order to change any of them if a better product is found. There are a number of spam filters that run as POP/SMTP/Imap proxy servers. These products download the email on behalf of the user, analyze it deciding if it is spam or legitimate, and tag it accordingly. Example of these products are POPFile (that features general email classification, apart from spam filtering), SpamBayes (proxy version), K9, or SAwin32 (Fig. 11).

As an example, the program SAwin32 is a POP proxy that includes a fully functional port of SpamAssassin for Microsoft Windows PCs. The proxy is configured to access the POP email server from where the user downloads his email and to check it using the SpamAssassin list, rule, and Bayesian filters. If found to be spam, a message is tagged with a configurable string in the subject (e.g., the default is ‘*** SPAM ***’) and the message is forwarded to the user with an explanation in the body and the original message as an attachment. The explanation presents a digest of the message, and describes the tests that have been fired by the message, the spam

```
X-Mozilla-Status: 0001
X-Mozilla-Status2: 10000000
X-SAproxy-Timeout: 0
Return-Path: <frederick@escortcorp.com>
Received: from localhost by mercury with SpamAssassin (version 3.2.3); Wed, 10 Oct 2007 06:53:56 +0200
Message-Id: <000901c80ae7$02f2d627$9246d393@gwinfag>
X-Spam-Flag: YES
X-Spam-Checker-Version: SpamAssassin 3.2.3 (2007-08-08) on mercury
X-Spam-Level: *****
X-Spam-Status: Yes, score=16.6 required=6.3 tests=DRUG5_DIET,FB_GET_MEDS, FB_GVR,FH_HELO_EQ_D_D_D_D,HELO_DYNAMIC_IPAI
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="-----_470C5AE4.7E980000"

SAproxy believes that this mail is spam. The original message
has been attached intact in RFC 822 format.

Content preview: Good afternoon Bro. We want to inform you that now you have
astonishing chance to solve your ED troubles. Huge variety of high quality
meds at the lowest prices. Save your time and money purchase medical products
only in our shop. C_1*!_C_K he|re. Good afternoon Bro. We want to inform
you that now you have astonishing chance to solve your ED troubles. Huge variety
of high quality meds at the lowest prices. Save your time and money purchase
medical products only in our shop. C_1*!_C_K he|re. [...]

Content analysis details: (16.6 points, 6.3 required)

2.4 FH_HELO_EQ_D_D_D_D Helo is d-d-d-d
4.4 HELO_DYNAMIC_IPADDR Relay HELO'd using suspicious hostname (IP addr
1)
2.6 TVD_QUAL_MEDS BODY: TVD_QUAL_MEDS
0.5 FB_GVR BODY: Looks like generic viagra
3.6 FB_GET_MEDS BODY: Looks like trving to sell meds
```

FIG. 11. An example of message scanned by the SAwin32 proxy, implementing the full set of tests and techniques of SpamAssassin.

score of the message, and other administrative data. The proxy also adds some nonstandard headers, beginning with 'X-,'²⁸ like 'X-Spam-Status,' that includes the spam value of the message ('Yes' in this case), and the tests and scores obtained.

The user can then configure a filter in his email client, sending to a spam box (a local quarantine) all the messages that arrive tagged as spam. The user can also suppress the additional tag and rely on the spam filter headers.

Also, most proxy-type filters also include the possibility of not downloading the messages tagged as spam to the email client, and keeping a separate quarantine in the proxy program, many often accessed as a local web application.

The tagging approach can also work at the organization level. The general email server tags the messages for all users, but each one prepares a local filter based on the server-side tags if that is needed.

5.4 Best and Future Practical Spam Filtering

As stated in [27], 'the best results in spam filtering could be achieved by a combination of methods: Black lists stop known spammers, graylists eliminate spam sent by spam bots, decoy email boxes alert to new spam attacks, and Bayesian filters detect spam and virus attack emails right from the start, even before the black list is updated.' False positives are still a problem due to the important effects that loosing an important mail can produce, along with the bouncing emails created by viruses [8], an important new kind of spam.

Probably, in the future, all these techniques will be mixed with economic and legal antispam measures like computing time-based systems,²⁹ money-based systems [53], strong antispam laws [15], and other high-impact social measures [80]. Each day, the filters apply more and more measures to detect spam due to its high economic impact.

6. Attacking Spam Filters

6.1 Introduction

As the volume of bulk spam email increases, it becomes more and more important to apply techniques that alleviate the cost that spam implies. Spam filters evolve to better recognize spam and that has forced the spammers to find

²⁸ Non-standard headers in emails begin with 'X-.' In particular, our example includes some Mozilla Thunderbird headers, like 'X-Mozilla-Status.'

²⁹ See, for instance, the HashCash service: <http://www.hashcash.org>.

new ways to avoid the detection of their spam, ensuring the delivery of their messages. For example, as statistical spam filters began to learn that words like ‘Viagra’ mostly occur in spam, spammers began to obfuscate them with spaces and other symbols in order to transform spam-related words in others like ‘V-i-a-g-r-a’ that, while conserving all the meaning for a human being, are hardly detected by software programs.

We refer to all the techniques that try to mislead the spam filters as attacks, and the spammers employing all these methods as attackers, since their goal is to mislead the normal behavior of the spam filter, allowing a spam message to be delivered as a normal message. A good spam filter would be most robust to past, present, and future attacks, but most empirical evaluations of spam filters ignore this because the spammers’ behavior is unpredictable and then, the real effectiveness of a filter cannot be known until its final release.

The attacker’s point of view is of vital importance when dealing with spam filters because it gives full knowledge about the possible attacks to spam filters. This perspective also gives a schema of the way of thinking of spammers, which allows predicting possible attacks and to detect tendencies that can help to construct a robust and safe filter. Following this trend, there exist some attempts to compile spammers’ attacks as the Graham Cumming’s *Spammer’s Compendium* [42].

Usually, a spam filter is part of a corporate complex IT architecture, and attackers are capable to deal with all the parts of that architecture, exploiting all the possible weak points. Attending to this, there exist direct attacks that try to exploit the spam filter vulnerabilities and indirect attacks that try to exploit other weak points of the infrastructure. Indirect attacks’ relevance has been growing since 2004 as spammers shifted their attacks away from content and focuses more on the SMTP connection point [75].

Spammers usually make use of other security-related problems like virus and trojans in order to increase their infrastructure. Nowadays, a spammer uses trojan programs to control a lot of zombie-machines from which he is able to attack many sites while not being easily located as he is far from the origin of the attacks. That makes it more and more difficult to counteract the spammers’ attacks and increases the side effects of the attacks.

6.2 Indirect Attacks

Mail servers automatically send mails when certain delivery problems, such as a mail over-quota problem, occur. These automatically sent emails are called bounce messages. These bounce messages can be seen, in a way, as auto replies (like the out of office auto replies) but are not sent by human decisions, and in fact are sent

automatically by the mail server. All these auto replies are discussed in the RFC3834³⁰ that points out that it must be sent to the Return-Path established in the received email that has caused this auto reply. The return message must be sent without Return-Path in order to avoid an infinite loop of auto replies.

From these bounce messages, there exist two important ones: the NDRs (Non-Delivery Reports), which are a basic function of SMTP and inform that a certain message could not be delivered and the DSNs (Delivery Status Notifications) that can be explicitly required by means of the ESMTP (SMTP Service Extension) protocol. The NDRs implement part of the SMTP protocol that appears on the RFC2821³¹: ‘If an SMTP server has accepted the task of relaying the mail and later finds that the destination is incorrect or that the mail cannot be delivered for some other reason, then it must construct an “undeliverable mail” notification message and send it to the originator of the undeliverable mail (as indicated by the reverse-path).’

The main goal of a spammer is to achieve the correct delivery of a certain nondesired mail. To achieve this goal, some time is needed to achieve a previous target. Indirect attacks are those that use characteristics outside the spam filter in order to achieve a previous target (like obtain valid email addresses) or that camouflages a certain spam mail as being a mail server notification. These indirect attacks use the bounce messages that the mail servers send in order to achieve their goals. Three typical indirect attacks are:

1. NDR (Non-Delivery Report) Attack
2. Reverse NDR Attack
3. Directory Harvest Attack

The NDR Attack consists of camouflaging the spam in an email that appears to be a NDR in order to confuse the user who can believe that he sent the initial mail that could not be delivered. The curiosity of the user may drive him to open the mail and the attachment where the spam resides. The NDR Attacks have two weak points:

1. Make intensive use of the attacker’s mail server trying to send thousands or millions of NDR like messages that many times are not even opened by the recipient.
2. If the receiver’s mail server uses a black list where the attacker’s mail server’s IP is listed, the false NDR would not ever reach their destination. Then, all the efforts made by the spammer would not be useful at all.

³⁰ Available at: <http://www.rfc-editor.org/rfc/rfc3834.txt>.

³¹ Available at: <http://www.ietf.org/rfc/rfc2821.txt>.

To defeat these two weak points, the Reverse NDR Attack was devised. In a Reverse NDR Attack, the intended target's email is used as the sender, rather than the recipient. The recipient is a fictitious email address that uses the domain name for the target's company (for instance, 'example.com'), such as noexist@example.com. The mail server of Example Company cannot deliver the message and sends an NDR mail back to the sender (which is the target email). This return mail carries the NDR and the original spam message attached and the target can read the NDR and the included spam thinking they may have sent the email. As can be seen in this procedure, a reliable mail server that is not in any black list and cannot be easily filtered sends the NDR mail.

Another attack that exploits the bounce messages is the DHA (Directory Harvest Attack) which is a technique used by spammers attempting to find valid email addresses at a certain domain. The success of a Directory Harvest Attack relies on the recipient email server rejecting email sent to invalid recipient email addresses during the Simple Mail Protocol (SMTP) session. Any addresses to which email is accepted are considered valid and are added to the spammer's list. There are two main techniques for generating the addresses that a DHA will target. In the first one, the spammer creates a list of all possible combinations of letters and numbers up to a maximum length and then appends the domain name. This is a standard brute force attack and implies a lot of workload in both servers. The other technique is a standard dictionary attack and is based on the creation of a list combining common first names, surnames, and initials. This second technique usually works well in company domains where the employees email addresses are usually created using their real names and surnames and not nicknames like in free mail services like Hotmail or Gmail.

6.3 Direct Attacks

The first generation of spam filters used rules to recognize specific spam features (like the presence of the word 'Viagra') [46]. Nowadays, as spam evolves quickly, it is impossible to update the rules as fast as new spam variations are created, and a new generation of more adaptable, learning-based spam filters has been created [46].

Direct attacks are attacks to the heart of those statistical spam filters and try to transform a given spam message into a stealthy one. The effectiveness of the attacks relies heavily on the filter type, configuration, and the previous training (mails received and set by the user as spam). One of the simplest attacks is called *picospam* and consists of appending random words to a short spam message, trying that those random words would be recognized as 'good words' by the spam filter. This attack is

very simple and was previously seen to be ineffective [63] but shows the general spirit of a direct attack to a spam filter.

There are many spammer techniques [25, 91], which can be grouped into four main categories [96]:

- *Tokenization*: The attacks using tokenization work against the feature selection used by the filter to extract the main features from the messages. Examples of tokenization attacks include splitting up words with spaces, dashes, and asterisks, or using HTML, JavaScript, or CSS tricks.
- *Obfuscation*: With this kind of attacks, the message's contents are obscured from the filter using different kinds of encodings, including HTML entity or URL encoding, letter substitution, Base64 printable encoding, and others.
- *Statistical*: These methods try to skew the message's statistics by adding more good tokens or using fewer bad ones. There exist some variations of this kind of attacks depending on the methods used to select the used words. Weak statistical or passive attacks, that use random words and strong statistical or active attacks, which carefully select the words that are needed to mislead the filter by means of some kind of feedback. Strong statistical attacks are more refined versions of weak attacks, being more difficult to develop and their practical applicability can be questioned.
- *Hiding the text*: Some attacks try to avoid the use of words and inserts the spam messages as images, Flash, RTF, or in other file format; some other attacks insert a link to a web page where the real spam message resides. The goal is that the user could see the message but the filter could not extract any relevant feature.

We discuss instances of these attacks in the next sections.

6.3.1 Tokenization Attacks

The statistical spam filters need a previous tokenization stage where the original message is transformed into a set of features describing the main characteristics of the email. A typical tokenization would count the occurrence of the words appearing in the email and would decompose the words by locating the spaces and other punctuation signals that separate the words. Tokenization attacks are conceived to attack this part of the filter, trying to avoid the correct recognition of spammy words by means of inserting spaces or other typical word delimiters inside the words. A typical example for avoiding the recognition of the word 'VIAGRA' would be separating the letters in this way 'V-I.A:G-R_A.' The user would easily recognize the word 'VIAGRA' but many filters would tokenize the word into multiple letters that do not represent the real content of the email.

6.3.1.1 *Hypertextus Interruptus.* The tokenization attack is based on the idea of splitting the words using HTML comments, pairs of zero width tags, or bogus tags. As the mail client renders the HTML, the user would see the message as if not containing any tag or comment, but the spam filters usually separates the words according to the presence of certain tags. Some examples trying to avoid the detection of the word ‘VIAGRA’:

- VIA<!-- -garbage- --> GRA
- VI</n>AGRA
- VIAG<xyz>R<xyz>A
- V<comment>xyz</comment>IAGRA
- VIAGRA

A typical tokenizer would decompose each of the last lines into a different set of words:

- VIA, GRA
- VI, AGRA
- VIAG, R, A
- V, IAGRA
- VIAGR, A

This makes more difficult to learn to distinguish the VIAGRA-related spam as each of the spam messages received contains a different set of features. The only way to face this kind of attacks is to parse carefully the messages trying to avoid HTML comments or extracting the features from the output produced by an HTML rendering engine.

6.3.1.2 *Slice and Dice.* Slice and Dice means to break a certain body of information down into smaller parts and then examine it from different viewpoints. Applied to spam attacks, slice and dice consists of dividing a spam message into text columns and then rewrite the message putting each text column in a column inside an HTML table. Applying the typical VIAGRA example, we could render it using a table as follows:

```
<table><tr><td>V</td><td>I</td><td>A</td><td>G</td>
<td>R</td><td>A</td></tr></table>
```

The user would see only VIAGRA but the tokenizer would extract one feature by each different letter in the message.

6.3.1.3 Lost in Space. This is the most basic tokenization attack, and consists of adding spaces or other characters between the letters that composes a word in order to make them unrecognizable to word parsers. ‘V*I*A*G*R*A,’ ‘V I A G R A,’ and ‘V.I.A.G.R.A’ are typical examples applying this simple and common technique. Some spam filters recognize the words by merging nonsense words separated by common characters and studying if they compose a word that could be considered as a spam feature.

6.3.2 Obfuscation Attacks

In obfuscation attacks, the message’s contents are obscured to the filter using different encodings or misdirection like letter substitution, HTML entities, etc. The way these attacks affect the spam filter is very similar to the way tokenization attacks affect. The features extracted from an obfuscation attack do not correspond with the standard features of the spam previously received. Obfuscating a word like ‘VIAGRA’ can be done in many ways:

- ‘VIAGRA’
- ‘VI4GR4’
- ‘VÍAGRÀ’

All the previous ways to obfuscate the word ‘VIAGRA’ produce a different feature that would be used by the spam filter to distinguish or learn how to distinguish a spam message related to the VIAGRA.

A prominent form of obfuscation is the utilization of *leetspeak*. Leetspeak or leet (usually written as l33t or l337) is an argot used primarily on the Internet, but becoming very common in many online video games due to the excellent reception of this argot from the youngsters who use to obfuscate their mails or SMS trying to avoid their parents to understand what they write to other friends. The leet speech uses various combinations of alphanumeric characters to replace proper letters. Typical replacements are ‘4’ for ‘A,’ ‘8’ or ‘13’ for ‘B,’ ‘(’ for ‘C,’ ‘)’ or ‘l’ for ‘D,’ ‘3’ for ‘E,’ ‘ph’ for ‘F,’ ‘6’ for ‘G,’ ‘#’ for ‘H,’ ‘1’ or ‘!’ for ‘I,’ etc. Using these replacements, ‘VIAGRA’ could be written as ‘V14GR4’ or ‘V!4G2A,’ which can be understood by a human being but would be intelligible by a spam filter.

Foreign accent is an attack very similar to the leetspeak, but do not replace letters with alphanumeric characters, it uses accented letters to substitute vocals or even characters like ‘ç’ to substitute ‘c’ due to their similarity. ‘VIAGRA’ could be rewritten in huge set of ways like ‘VÍÁGRÁ,’ ‘VÌÀGRÀ,’ ‘VÏÄGRÄ,’ etc.

The simplest way to affront these attacks is to undo these replacements, which can be very simple in the foreign accent attacks because there exists a univocal

correspondence between an accented letter and the unaccented letter. But the leet-speak is more difficult to translate as when a certain number or character is found, it would be needed to study whether the alphanumeric is representing a certain letter or it must continue being an alphanumeric.

6.3.3 *Statistical Attacks*

While tokenization and obfuscation attacks are more related with the preprocessing stage of a spam filter, the statistical attacks have the main goal of attacking the heart of the statistical filter. Statistical attacks, more often called Bayesian poisoning [27, 37, 88] as most of the classifiers used to detect spam are Bayesian or Good Word Attacks [63], are based on adding random, or even carefully selected, words that are unlikely to appear in spam messages and are supposed to cause the spam filter to believe the message is not a spam (a statistical type II error). The statistical attacks have a secondary effect, a higher false-positive rate (statistical I error) because when the user trains their spam filter with spam messages containing normal words, the filter learns that these normal words are a good indication of spam.

Statistical attacks are very similar, and what most vary among them is the way the words are added into the normal message and the way the words are selected. According to the word selection, there exist active attacks and passive attacks. According to the way the words are inserted in the email, there exist some variations like Invisible Ink and MIME-based attacks. Attacks can combine approaches, and for each possible variation of including the words in the message, the attack can be active or passive.

6.3.3.1 *Passive Attacks.* In passive attacks, the attacker constructs the word list to be used as the good words to be added in the spam messages without any feedback from the spam filter. Calburn [16] explains this process as ‘The automata will just keep selecting random words from the legit dictionary . . . When it reaches a Bayesian filtering system, [the filtering system] looks at these legitimate words and the probability that these words are associated with a spam message is really low. And the program will classify this as legitimate mail.’

The simplest passive attack consists of selecting a random set of words that would be added to all the spam messages sent by the attacker. If the same words are added to all the spam messages sent, that set of words would finish being considered as a good indicative of a spam message and the attack would convert into unproductive. Another simple yet more effective attack consists of a random selection of words per each spam mail sent or for each certain number of spam messages sent. Wittel [96] shows that the addition of random words was ineffective against the filter CRM-114 but effective against SpamBayes.

A smarter attack can be achieved by selecting common or hammy words instead of performing a random selection. Wittel [96] shows that attacks using common words are more effective against SpamBayes even when adding fewer words than when the word selection was randomizing. Instead, the work in [88] shows that by adding common words, the filter's precision decreases from 84% to 67% and from 94% to 84% and proposes to ignore common words when performing the spam classification to avoid this performance falling.

6.3.3.2 Active Attacks. In active attacks, the attacker is allowed to receive some kind of feedback to know whether the spam filter labels a message as spam or not. Graham-Cumming [43] presents a simple way of getting this feedback by including a unique web bug at each message sent. A web bug is a graphic on a web page or inside an email that is designed to monitor who is reading a certain web or mail by counting the hits or accesses to this graphic. Having one web bug per word or per each word set allows the spammer to control what are the words that makes a spam message to look like a ham message to a certain spam filter.

Lowd [63] shows that passive attacks adding random words to spam messages is ineffective as a form of attacks and also demonstrates that adding hammy words was very effective against naïve Bayesian filters. Lowd [63] also shows in detail two active attacks that are very effective against most typical spam filters.

The best way of preventing active attacks is to close any door that allows the spammer to receive any feedback from our system such as nondelivery reports, SMTP level errors, or web bugs.

6.3.3.3 Invisible Ink. This statistical attack consists of the addition of some real random words in the message but not letting the user to see those words. There are some variants of doing this:

- Add the random words before the HTML.
- Add an email header packer with the random words.
- Write a colored text on a background of the same color.

For avoiding this spammer practice, spam filters should work only with the data the user can see, avoiding headers and colored text over the same color background (Fig. 12).

6.3.4 Hidden Text Attacks

These kind of attacks try to show the user a certain spam message but avoiding the spam filter to capture any feature from the content. The important point here is to place the spam message inside an image or other format, as RTF or PDF, that the



FIG. 12. An example of image-based spam.

email client will show to the user embedded into the message. We describe several kinds of attacks that try to avoid using text, or disguise it in formats hard to process effectively and efficiently.

6.3.4.1 MIME Encoding. MIME (Multipurpose Internet Mail Extensions) is an Internet standard that allows the email to support plain text, non-text attachments, multi-part bodies, and non-ASCII header information. In this attack, the spammer sends a MIME document with the spam message in the HTML section and any normal text in the plain text section, which makes more difficult the work of the spam filter.

6.3.4.2 Script Hides the Contents. Most email clients parse the HTML to extract the features and avoid the information inside SCRIPT tags as they usually contain no important information when we are extracting features to detect spam contents. Recently, some attackers have made use of script languages to change the contents of the message on mouse-over event or when the email is opened. Using this technique, the spam filter would read a normal mail that would be converted into a spam mail when opened by the email client.

6.3.4.3 Image-Based Spam. The first image-based spam was reported by Graham-Cumming in 2003 and was very simple; it included only an image with the spam text inside. At that time images were loaded from Web sites using simple HTML image tags, but as email clients started to avoid the remote image load, spammers started to send the images as MIME attachments. First attempts to detect this kind of spam made the developers to use OCR (Optical Character Recognition) techniques to pull words out of the images and use them as features to classify the message as spam or ham.

But OCR is computationally expensive; its accuracy leaves much to be desired and can be easily misled by adding noise to the images or even obfuscating the contents with the same techniques used to obfuscate the text content in text spam. According to IronPort statistics [49], 1% of spam was image based in June 2005 and one year later, it had risen to 16%. Cosoi [23] shows that the evolution of image-based spam was from 5% by March 2006 to almost 40% at the end of 2006. Samosseiko [83] asserts that more than the 40% of the spam seen in SophosLabs at the end of 2006 is image-based spam. All these numbers show the real importance of image-based spam in the present spam's world (Fig. 13).

For a more detailed review of actual image-based techniques, Cumming [45] shows the evolution of image spam along the last years, from single images containing the whole spam, combinations (more or less complicated) of different images to show the spam to the user, integrating images with tokenization and obfuscation attacks, using strange fonts, adding random pixels to avoid OCR, hashing, etc.

Interestingly, a number of researchers have devised approaches to deal with specific form of image spam. In particular, Biggio and others [10] have proposed to detect spam email by using the fact that spammers try to add noise to images in order to avoid OCRs, what it is called 'obscuring' by the authors. Also, and in a more general approach, Byun and others make use of a suite of synthesized attributes of

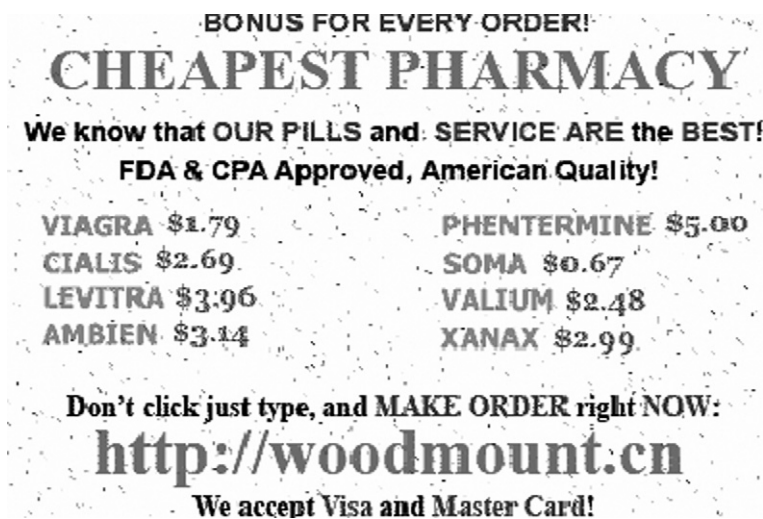


FIG. 13. An example of image-based spam with noise in order to avoid text recognition using OCR software.

spam images (color moment, color heterogeneity, conspicuousness, and self-similarity) in order to characterize a variety of image spam types [12]. These are promising lines of research, and combined with other techniques offer the possibility of high accuracy filtering.

6.3.4.4 Spam Using Other Formats. As spam filter developers increase the precision of their software, spammers develop new variation of their attacks. One simple variation of image-based attacks consists of using a compressed PDF instead of an image to place the spam content trying to avoid the OCR scanning as PDF do not use to contain spam. Another attack consists of embedding RTF files, containing the spam message, which are sniffed by Microsoft email clients.

7. Conclusions and Future Trends

Spam is an ever growing menace that can be very harmful. Its effects could be very similar to those produced by a Denial of Service Attack (DoS). Political, economical, legal, and technical measures are not enough to end the problem, and only a combination of all of them can lower the harm produced by it.

Among all those approaches, content-based filters have been the best solution, having big impact in spammers that have had to search new ways to pass those filters.

Luckily, systems based on Machine Learning algorithms allow the system to learn adapt to new treats, reacting to countermeasures used by spammers.

Recent competitions in spam filtering have shown that actual systems can filter out most of the spam, and new approaches like those based on compression can achieve high accuracy ratios. Spammers have designed new and refined attacks that hit one of the critical steps in every learning method: the tokenization process, but compression-based filters have been very resistant to this kind of attack.

REFERENCES

- [1] Abadi M., Birrell A., Burrows M., Dabek F., and Wobber T., December 2003. Bankable postage for network services. In *Proceedings of the 8th Asian Computing Science Conference*, Mumbai, India.
- [2] Ahn L. V., Blum M., and Langford J., February 2004. How lazy cryptographers do AI. *Communications of the ACM*.
- [3] Anderson R., 2004. Taking a bit out of spam. *Network Computing*, Magazine Article, May 13.
- [4] Androutsopoulos I., Koutsias J., Chandrinos K. V., Paliouras G., and Spyropoulos C. D., 2000. An evaluation of Naive Bayesian anti-spam filtering. In *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML)*, pp. 9–17. Barcelona, Spain.

- [5] Androutsopoulos I., Paliouras G., Karkaletsis V., Sakkis G., Spyropoulos C. D., and Stamatiopoulos P., 2000. Learning to filter spam e-mail: A comparison of a naive Bayesian and a memory-based approach. In *Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pp. 1–13. Lyon, France.
- [6] Androutsopoulos I., Koutsias J., Chandrinou K. V., and Spyropoulos C. D., 2000. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with encrypted personal e-mail messages. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 160–167. Athens, Greece, ACM Press, New York, US.
- [7] Belkin N. J., and Croft W. B., 1992. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, **35**(12): 29–38.
- [8] Bell S., 2003. Filters causing rash of false positives: TelstraClear's new virus and spam screening service gets mixed reviews. <http://computerworld.co.nz/news.nsf/news/CC256CED0016AD1ECC-256DAC000D90D4?Opendocument>.
- [9] Bickel S., September 2006. ECML/PKDD discovery challenge 2006 overview. In *Proceedings of the Discovery Challenge Workshop, 17th European Conference on Machine Learning (ECML) and 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, Berlin, Germany.
- [10] Biggio B., Giorgio Fumera, Ignazio Pillai, and Fabio Roli, August 23, 2007. Image spam filtering by content obscuring detection. In *Proceedings of the Fourth Conference on Email and Anti-Spam (CEAS 2007)*, pp. 2–3. Microsoft Research Silicon Valley, Mountain View, California.
- [11] Bratko A., Cormack G. V., Filipic B., Lynam T. R., and Zupan B., Dec 2006. Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, **7**: 2699–2720.
- [12] Byun B., Lee C.-H., Webb S., and Calton P., August 2–3, 2007. A discriminative classifier learning approach to image modeling and spam image identification. In *Proceedings of the Fourth Conference on Email and Anti-Spam (CEAS 2007)*, Microsoft Research Silicon Valley, Mountain View, California.
- [13] Caropreso M. F., Matwin S., and Sebastiani F., 2001. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In *Text Databases and Document Management: Theory and Practice*, A. G. Chin, editor, pp. 78–102. Idea Group Publishing, Hershey, US.
- [14] Carreras X., and Márquez L., 2001. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*.
- [15] Caruso J., December 2003. Anti-spam law just a start, panel says. *Network World*, <http://www.networkworld.com/news/2003/1218panel.html>.
- [16] Claburn T., 2005. Constant struggle: How spammers keep ahead of technology, Message Pipeline. <http://www.informationweek.com/software/messaging/57702892>.
- [17] Cleary J. G., and Teahan W. J., 1997. Unbounded length contexts for PPM. *The Computer Journal*, **40**(2/3): 67–75.
- [18] Cohen W. W., and Hirsh H., 1998. Joins that generalize: Text classification using WHIRL. In *Proceedings of KDD-98, 4th International Conference on Knowledge Discovery and Data Mining*, pp. 169–173. New York, NY.
- [19] Cormack G. V., and Lynam T. R., 2005. TREC 2005 spam track overview. In *Proc. TREC 2005 – the Fourteenth Text REtrieval Conference*, Gaithersburg.
- [20] Cormack G. V., and Lynam T. R., July 2005. Spam corpus creation for TREC. In *Proc. CEAS 2005 – The Second Conference on Email and Anti-spam*, Palo Alto.

- [21] Cormack G. V., and Bratko A., July 2006. Batch and on-line spam filter evaluation. In *CEAS 2006 – Third Conference on Email and Anti-spam*, Mountain View.
- [22] Cormack G., Gómez Hidalgo J. M., and Puertas Sanz E., November 6-9, 2007. Spam filtering for short messages. In *ACM Sixteenth Conference on Information and Knowledge Management (CIKM 2007)*, Lisboa, Portugal.
- [23] Cosoi C. A., December 2006. The medium or the message? Dealing with image spam. *Virus Bulletin*, <http://www.virusbtn.com/spambulletin/archive/2006/12/sb200612-image-spam.dkb>.
- [24] Cranor L. F., and LaMacchia B. A., 1998. Spam! *Communications of the ACM*, **41**(8): 74–83.
- [25] Dalvi N., Domingos P., Sanghai M. S., and Verma D., 2004. Adversarial classification. In *Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining*, pp. 99–108. ACM Press, Seattle, WA.
- [26] Dantin U., and Paynter J., 2005. Spam in email inboxes. In *18th Annual Conference of the National Advisory Committee on Computing Qualifications*, Tauranga, New Zealand.
- [27] Deerwester S., Dumais S. T., Furnas G. W., Landauer T. K., and Harshman R., 1990. Indexing by latent semantic indexing. *Journal of the American Society for Information Science*, **41**(6): 391–407.
- [28] Domingos P., and Pazzani M. J., 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, **29**(2–3): 103–130.
- [29] Domingos P., 1999. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pp. 155–164. San Diego, CA, ACM Press.
- [30] Drucker H., Vapnik V., and Wu D., 1999. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, **10**(5): 1048–1054.
- [31] Dumais S. T., Platt J., Heckerman D., and Sahami M., 1998. Inductive learning algorithms and representations for text categorization. In *Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management*, G. Gardarin, J. C. French, N. Pissinou, K. Makki, and L. Bouganim, eds. pp. 148–155. ACM Press, New York, US, Bethesda, US.
- [32] Dwork C., Goldberg A., and Naor M., August 2003. On memory-bound functions for fighting spam. In *Proceedings of the 23rd Annual International Cryptology Conference (CRYPTO 2003)*.
- [33] Eckelberry A., 2006. What is the effect of Bayesian poisoning? *Security Pro Portal*, <http://www.netscape.com/viewstory/2006/08/21/what-is-the-effect-of-bayesian-poisoning/>.
- [34] Fawcett T., 2003. “In vivo” spam filtering: A challenge problem for KDD. *SIGKDD Explorations*, **5** (2): 140–148.
- [35] Fuhr N., Hartmann S., Knorz G., Lustig G., Schwantner M., and Tzeras K., 1991. AIR/X—a rule-based multistage indexing system for large subject fields. In *Proceedings of RIAO-91, 3rd International Conference “Recherche d’Information Assistée par Ordinateur,”* pp. 606–623. Barcelona, Spain.
- [36] García F. D., Hoepman J.-H., and van Nieuwenhuizen J., 2004. Spam filter analysis. In *Security and Protection in Information Processing Systems, IFIP TC11 19th International Information Security Conference (SEC2004)*, Y. Deswarte, F. Cuppens, S. Jajodia, and L. Wang, eds. pp. 395–410. Toulouse, France.
- [37] Gee K., and Cook D. J., 2003. Using latent semantic indexing to filter spam. *ACM Symposium on Applied Computing*, Data Mining Track.
- [38] Gómez-Hidalgo J. M., Maña-López M., and Puertas-Sanz E., 2000. Combining text and heuristics for cost-sensitive spam filtering. In *Proceedings of the Fourth Computational Natural Language Learning Workshop, CoNLL-2000*, Association for Computational Linguistics, Lisbon, Portugal.
- [39] Gómez-Hidalgo J. M., 2002. Evaluating cost-sensitive unsolicited bulk email categorization. In *Proceedings of SAC-02, 17th ACM Symposium on Applied Computing*, pp. 615–620. Madrid, ES.

- [40] Gómez-Hidalgo J. M., Maña-López M., and Puertas-Sanz E., 2002. Evaluating cost-sensitive unsolicited bulk email categorization. In *Proceedings of JADT-02, 6th International Conference on the Statistical Analysis of Textual Data*, Madrid, ES.
- [41] Goodman J., 2004. IP addresses in email clients. In *Proceedings of The First Conference on Email and Anti-Spam*.
- [42] Graham-Cumming J., 2003. The Spammer's Compendium. In *MIT Spam Conference*.
- [43] Graham-Cumming J., 2004. How to beat an adaptive spam filter. In *MIT Spam Conference*.
- [44] Graham-Cumming J., February 2006. Does Bayesian poisoning exist? *Virus Bulletin*.
- [45] Graham-Cumming J., November 2006. The rise and rise of image-based spam. *Virus Bulletin*.
- [46] Graham P., 2002. A plan for spam. Reprinted in Paul Graham, *Hackers and Painters, Big Ideas from the Computer Age*, O'Really (2004). Available: <http://www.paulgraham.com/spam.html>.
- [47] Graham P., January 2003. Better Bayesian filtering. In *Proceedings of the 2003 Spam Conference*. Available: <http://www.paulgraham.com/better.html>.
- [48] Gray A., and Haahr M., 2004. Personalised, collaborative spam filtering. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*.
- [49] Hahn J., 2006. Image-based spam makes a comeback. *Web Trends*, http://www.dmconfidential.com/blogs/column/Web_Trends/916/.
- [50] Hall R. J., March 1998. How to avoid unwanted email. *Communications of the ACM*.
- [51] Hird S., 2002. Technical solutions for controlling spam. In *Proceedings of AUUG2002, Melbourne*.
- [52] Hovold J., 2005. Naive Bayes spam filtering using word-position-based attributes. In *Proceedings of the Second Conference on Email and Anti-spam, CEAS*, Stanford University.
- [53] InfoWorld Test Center. Strong spam combatants: Brute anti-spam force takes on false-positive savvy. Issue 22 May 31, 2004.
- [54] Jeffrey E., and Friedl F., August 2006. Mastering Regular Expressions. 3rd edn. O'Really.
- [55] Joachims T., 1998. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pp. 137–142. Chemnitz, Germany.
- [56] Joachims T., 1999. Transductive inference for text classification using support vector machines. In *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pp. 200–209. Bled, Slovenia.
- [57] Keogh E., Lonardi S., and Ratanamahatana C. A., 2004. Towards parameter-free data mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 206–215. Seattle, WA, USA, August 22–25, 2004). KDD '04. ACM, New York, NY.
- [58] Kolcz A., Chowdhury A., and Alspector J., 2004. The impact of feature selection on signature-driven spam detection. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*.
- [59] Larkey L. S., and Croft W. B., 1996. Combining classifiers in text categorization. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, eds. pp. 289–297. ACM Press, New York, US, Zurich, CH.
- [60] Lewis D. D., and Gale W. A., 1994. A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, W. B. Croft and C. J. van Rijsbergen, eds. pp. 3–12. Springer Verlag, Heidelberg, DE, Dublin, IE.
- [61] Lewis D. D., 1998. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, C. Nédellec and

- C. Rouveirol, eds. pp. 4–15. Springer Verlag, Heidelberg, DE, Chemnitz, DE. Lecture Notes in Computer Science, 1398.
- [62] Li Y. H., and Jain A. K., 1998. Classification of text documents. *The Computer Journal*, **41**(8): 537–546.
 - [63] Lowd D., and Meek C., 2005. Adversarial Learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ACM Press, Chicago, IL.
 - [64] Lucas M. W., 2006. PGP & GPG: email for the Practical Paranoid. No Starch Press.
 - [65] McCallum A., and Nigam K., 1998. A comparison of event models for Naive Bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*.
 - [66] MessageLabs, 2006. MessageLabs Intelligence: 2006 Annual Security Report. Available: http://www.messagelabs.com/mlireport/2006_annual_security_report_5.pdf.
 - [67] Meyer T. A., and Whateley B., 2004. Spambayes: Effective open-source, bayesian based, email classification system. In *Proceedings of the First Conference on Email and Anti-spam (CEAS)*.
 - [68] Mitchell T. M., 1996. Machine learning. McGraw Hill, New York, US.
 - [69] O'Brien C., and Vogel C., September 2003. Spam filters: Bayes vs. chi-squared; letters vs. words. In *Proceedings of the International Symposium on Information and Communication Technologies*.
 - [70] Pampapathi R., Mirkin B., and Levene M., 2006. A suffix tree approach to anti-spam email filtering. *Mach. Learn.*, **65**(1): 309–338.
 - [71] Pantel P., and Lin D., 1998. Spamcop: A spam classification and organization program. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin. AAAI Technical Report WS-98-05.
 - [72] Platt J., 1998. Fast training of support vector machines using sequential minimal optimization. B. Schölkopf, C. Burges, and A. Smola, eds. *Advances in Kernel Methods – Support Vector Learning*.
 - [73] Postini White Paper, 2004. Why content filter is no longer enough: Fighting the battle against spam before it can reach your network. Postini Pre-emptive email protection.
 - [74] Provost F., and Fawcett T., 1997. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*.
 - [75] Provost J., 1999. Naive-bayes vs. rule-learning in classification of email. Technical report Department of Computer Sciences at the University of Texas at Austin.
 - [76] Quinlan R., 1986. Induction of decision trees. *Machine Learning*, **1**(1): 81–106.
 - [77] Rigoutsos I., and Huynh T., 2004. Chung-kwei: A pattern-discovery-based system for the automatic identification of unsolicited e-mail messages (spam). In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*.
 - [78] Saarinen J., 2003. Spammer ducks for cover as details published on the web, NZHerald. http://www.nzherald.co.nz/section/1/story.cfm?c_id=1&objectid=3518682.
 - [79] Sahami M., Dumais S., Heckerman D., and Horvitz E., 1998. A Bayesian approach to filtering junk e-mail. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, AAAI Press, Madison, WI.
 - [80] Sakis G., Androutsopoulos I., Paliouras G., Karkaletsis V., Spyropoulos C. D., and Stamatopoulos P., 2001. Stacking classifiers for anti-spam filtering of e-mail. In *Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing*, Pittsburgh, US, Association for Computational Linguistics, Morristown, US.
 - [81] Salton G., 1981. A blueprint for automatic indexing. *SIGIR Forum*, **16**(2): 22–38.
 - [82] Salton G., and McGill M. J., 1983. Introduction to Modern Information Retrieval. McGraw Hill, New York, US.

- [83] Samosseiko D., and Thomas R., 2006. The game goes on: An analysis of modern spam techniques. In *Proceedings of the 16th Virus Bulletin International Conference*.
- [84] Sculley D., and Brodley C. E., 2006. Compression and machine learning: a new perspective on feature space vectors. In *Data Compression Conference (DCC'06)*, pp. 332–341.
- [85] Sebastiani F., 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, **34**(1): 1–47.
- [86] Seigneur J.-M., and Jensen C. D., 2004. Privacy recovery with disposable email addresses. *IEEE Security and Privacy*, **1**(6): 35–39.
- [87] Sergeant M., 2003. Internet-level spam detection and SpamAssassin 2.50. In *Spam Conference*.
- [88] Stern H., Mason J., and Shepherd M., A linguistics-based attack on personalized statistical e-mail classifiers. Technical report CS-2004-06, Faculty of Computer Science, Dalhousie University, Canada, March 25, 2004.
- [89] Taylor B., 2006. Sender reputation in a large webmail service. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS)*, Mountain View, California.
- [90] Teahan W. J., and Harper D. J., 2003. “Using compression based language models for text categorization”. In *Language Modeling for Information Retrieval*, W. B. Croft and J. Laferty, eds. The Kluwer International Series on Information Retrieval, Kluwer Academic Publishers.
- [91] Theo V. D., 2004. New and upcoming features in SpamAssassin v3, ApacheCon.
- [92] Thomas R., and Samosseiko D., October 2006. The game goes on: An analysis of modern spam techniques. In *Virus Bulletin Conference*.
- [93] Turing A. M., 1950. Computing machinery and intelligence. *Mind*, **59**: 433–460.
- [94] Watson B., 2004. Beyond identity: Addressing problems that persist in an electronic mail system with reliable sender identification. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA.
- [95] Wiener E. D., Pedersen J. O., and Weigend A. S., 1995. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pp. 317–332. Las Vegas, US.
- [96] Wittel G. L., and Wu F., 2004. On attacking statistical spam filters. In *Proceedings of the Conference on Email and Anti-spam (CEAS)*.
- [97] Witten I. H., and Frank E., 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, Los Altos, US.
- [98] Yang Y., and Chute C. G., 1994. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, **12**(3): 252–277.
- [99] Yang Y., and Pedersen J. O., 1997. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*. D. H. Fisher, editor.
- [100] Yerazunis B., 2003. Sparse binary polynomial hash message filtering and the CRM114 discriminator. In *Proceedings of the Spam Conference*.
- [101] Yerazunis B., 2004. The plateau at 99.9. In *Proceedings of the Spam Conference*.
- [102] Zdziarski J., 2004. Advanced language classification using chained tokens. In *Proceedings of the Spam Conference*.
- [103] Zimmermann P. R., 1995. *The Official PGP User’s Guide*. MIT Press.