Foundations and Trends® in
Information Retrieval
Vol. 1, No. 4 (2006) 335–455

DOI: 10.1561/1500000006

now
the essence of knowledge

Email Spam Filtering: A Systematic Review

Gordon V. Cormack

David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada, gvcormac@uwaterloo.ca

Abstract

Spam is information crafted to be delivered to a large number of recipients, in spite of their wishes. A spam filter is an automated tool to recognize spam so as to prevent its delivery. The purposes of spam and spam filters are diametrically opposed: spam is effective if it evades filters, while a filter is effective if it recognizes spam. The circular nature of these definitions, along with their appeal to the intent of sender and recipient make them difficult to formalize. A typical email user has a working definition no more formal than "I know it when I see it." Yet, current spam filters are remarkably effective, more effective than might be expected given the level of uncertainty and debate over a formal definition of spam, more effective than might be expected given the state-of-the-art information retrieval and machine learning methods for seemingly similar problems. But are they effective enough? Which are better? How might they be improved? Will their effectiveness be compromised by more cleverly crafted spam?

We survey current and proposed spam filtering techniques with particular emphasis on how well they work. Our primary focus is spam filtering in email; Similarities and differences with spam filtering in other communication and storage media — such as instant messaging

and the Web — are addressed peripherally. In doing so we examine the definition of spam, the user's information requirements and the role of the spam filter as one component of a large and complex information universe. Well-known methods are detailed sufficiently to make the exposition self-contained, however, the focus is on considerations unique to spam. Comparisons, wherever possible, use common evaluation measures, and control for differences in experimental setup. Such comparisons are not easy, as benchmarks, measures, and methods for evaluating spam filters are still evolving. We survey these efforts, their results and their limitations. In spite of recent advances in evaluation methodology, many uncertainties (including widely held but unsubstantiated beliefs) remain as to the effectiveness of spam filtering techniques and as to the validity of spam filter evaluation methods. We outline several uncertainties and propose experimental methods to address them.

1

Introduction

The Spam Track at the Text Retrieval Conference (TREC) defines email spam as

> "Unsolicited, unwanted email that was sent indiscriminately, directly or indirectly, by a sender having no current relationship with the recipient." [40]

Although much of the history of spam is folklore, it is apparent that spam was prevalent in instant messaging (Internet Relay Chat, or IRC) and bulletin boards (Usenet, commonly dubbed *newsgroups*) prior to the widespread use of email. Spam countermeasures are as old as spam, having progressed from *ad hoc* intervention by administrators through simple hand-crafted rules through automatic methods based on techniques from information retrieval and machine learning, as well as new methods specific to spam. Spam has evolved so as to defeat countermeasures; countermeasures have evolved so as to thwart evasion.

We generalize the TREC definition of spam to capture the essential adversarial nature of spam and spam abatement.

> **Spam**: *unwanted communication intended to be delivered to an indiscriminate target, directly or indirectly, notwithstanding measures to prevent its delivery.*
> **Spam filter**: *an automated technique to identify spam for the purpose of preventing its delivery.*

Applying these definitions requires the adjudication of subjective terms like *intent* and *purpose.* Furthermore, any evaluation of spam filtering techniques must consider their performance within the context of how well they fulfill their intended purpose while avoiding undesirable consequences. It is tempting to conclude that scientific spam filter evaluation is therefore impossible, and that the definition of spam or the choice of one filter over another is merely a matter of taste. Or to conclude that the subjective aspects can be "defined away" thus reducing spam filter evaluation to a simple mechanical process. We believe that both conclusions are specious, and that sound quantitative evaluation can and must be applied to the problem of spam filtering.

While this survey confines itself to email spam, we note that the definitions above apply to any number of communication media, including text and voice messages [31, 45, 84], social networks [206], and blog comments [37, 123]. It applies also to web spam, which uses a search engine as its delivery mechanism [187, 188].

1.1 The Purpose of Spam

The motivation behind spam is to have information delivered to the recipient that contains a *payload* such as advertising for a (likely worthless, illegal, or non-existent) product, bait for a fraud scheme, promotion of a cause, or computer malware designed to hijack the recipient's computer. Because it is so cheap to send information, only a very small fraction of targeted recipients — perhaps one in ten thousand or fewer — need to receive and respond to the payload for spam to be profitable to its sender [117].

A decade ago (circa 1997), the mechanism, payload, and purpose of spam were quite transparent. The majority of spam was sent by "cottage industry" spammers who merely abused social norms to promote

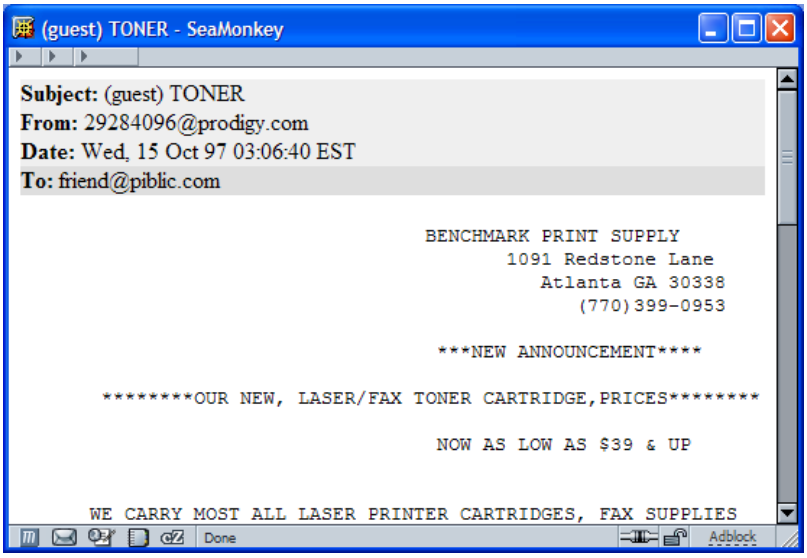

Fig. 1.1 Marketing spam.

their wares (Figure 1.1). Fraud bait consisted of clumsily written "Nigerian scams" (Figure 1.2) imploring one to send bank transit information so as to receive several MILLION DOLLARS from an aide to some recently deposed leader. Cause promotion took the form of obvious chain letters (Figure 1.3), while computer viruses were transmitted as attached executable files (Figure 1.4). Yet enough people received and responded to these messages to make them lucrative, while their volume expanded to become a substantial inconvenience even to those not gullible enough to respond.

At the same time, spamming has become more specialized and sophisticated, with better hidden payloads and more nefarious purposes. Today, cottage industry spam has been overwhelmed by spam sent in support of organized criminal activity, ranging from traffic in illegal goods and services through stock market fraud, wire fraud, identity theft, and computer hijacking [140, 178]. Computer viruses are no longer the work of simple vandals, they are crafted to hijack computers so as to aid in identity theft and, of course, the perpetration of more spam!

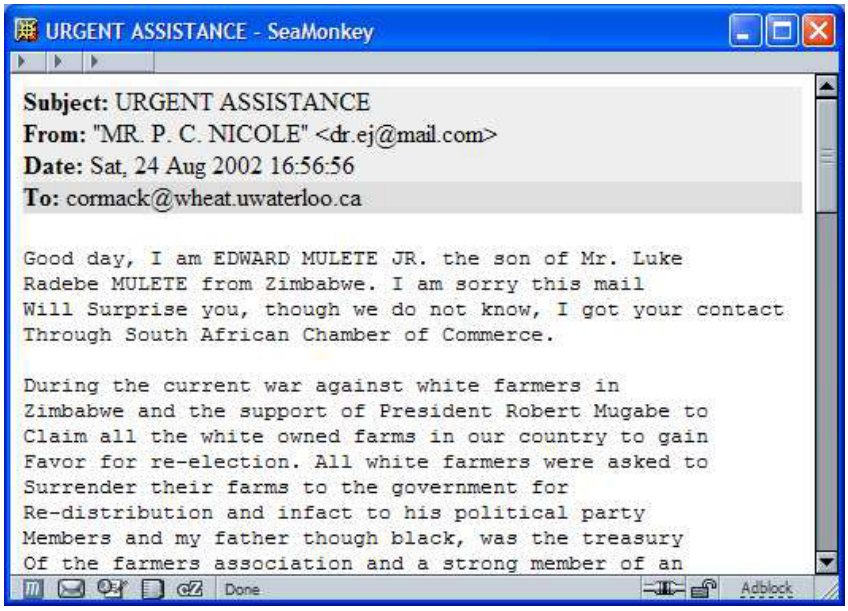

Fig. 1.2 Nigerian spam.

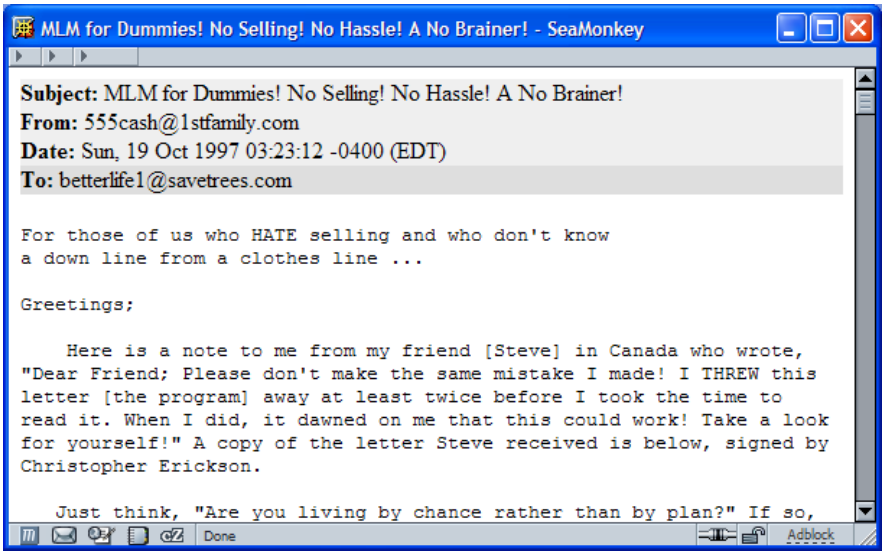

Fig. 1.3 Chain letter spam.

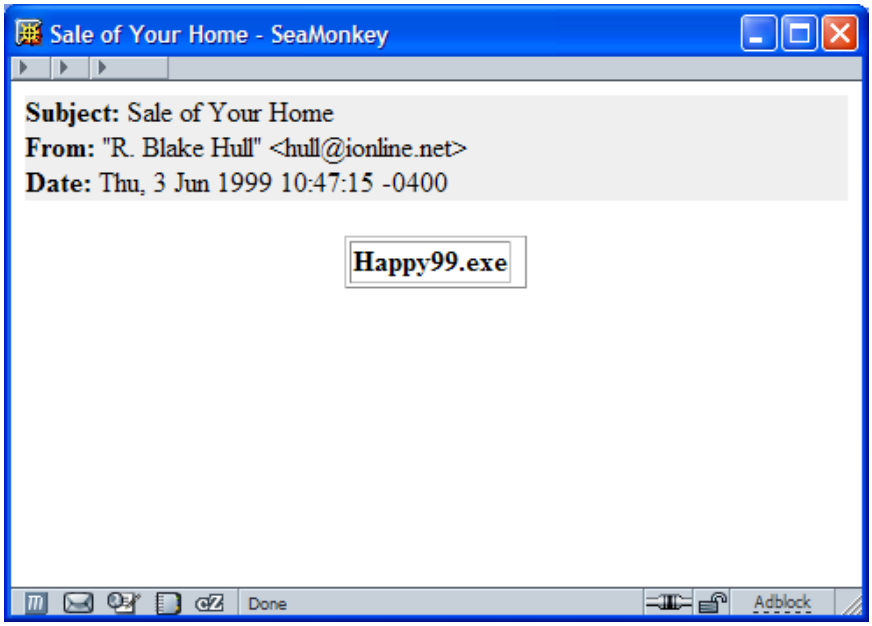

Fig. 1.4 Virus spam.

Spam, to meet its purpose, must necessarily have a payload which is delivered and acted upon[1] in the intended manner. Spam abatement techniques are effective to the extent that they prevent delivery, prevent action, or substitute some other action that acts as a disincentive.[2] Spam filters, by identifying spam, may be used in support of any of these techniques. At the same time, the necessary existence of a payload may aid the filter in its purpose of identifying spam.

1.2 Spam Characteristics

Spam in all media commonly share a number of characteristics that derive from our definition and discussion of the purpose of spam.

[1] The target need not be a person; a computer may receive and act upon the spam, serving its purpose just as well.
[2] Such as arresting the spammer.

1.2.1 Unwanted

It seems obvious that spam messages are unwanted, at least by the vast majority of recipients. Yet some people respond positively to spam, as evidenced by the fact that spam campaigns work [71]. Some of these individuals no doubt come to regret having responded, thus calling into question whether they indeed wanted to receive the spam in the first place. Some messages — such as those trafficking in illegal goods and services — may be wanted by specific individuals, but classed as unwanted by society at large. For most messages there is broad consensus as to whether or not the message is wanted, for a substantial minority (perhaps as high as 3% [168, 199]) there is significant disagreement and therefore some doubt as to whether the message is spam or not.

1.2.2 Indiscriminate

Spam is transmitted outside of any reasonable relationship[3] or prospective relationship between the sender and the receiver. In general, it is more cost effective for the spammer to send more spam than to be selective as to its target. An unwanted message targeting a specific individual, even if it promotes dubious products or causes or contains fraud bait or a virus, does not meet our definition of spam.

A message that is automatically or semi-automatically tailored to its target is nonetheless indiscriminate. For example, a spammer may harvest the name of the person owning a particular email address and include that name in the salutation of the message. Or a spammer may do more sophisticated data mining and sign the message with the name and email address of a colleague or collaborator, and may include in the text subjects of interest to the target. The purpose of such tailoring is, of course, to disguise the indiscriminate targeting of the message.

1.2.3 Disingenuous

Because spam is unwanted and indiscriminate, it must disguise itself to optimize the chance that its payload will be delivered and acted

[3] We have dropped the term *unsolicited* used in TREC and earlier definitions of spam, because not all unsolicited email is spam, and that which is captured by our notion of *indiscriminate*. Solicited email, on the other hand, is clearly not indiscriminate.

upon. The possible methods of disguise are practically unlimited and cannot be enumerated in this introduction (cf. [27, 67, 75]). Some of the most straightforward approaches are to use plausible subject and sender data, as well as subject material that appears to be legitimate. It is common, for example, to receive a message that appears to be a comment from a colleague pertaining to a recent news headline. Even messages with random names; for example a wire transfer from John to Judy, will appear legitimate to some fraction of its recipients. Messages purporting to contain the latest security patch from Microsoft will similarly be mistaken for legitimate by some fraction of recipients.

Spam must also disguise itself to appear legitimate to spam filters. Word misspelling or obfuscation, embedding messages in noisy images, and sending messages from newly hijacked computers, are spam characteristics designed to fool spam filters. Yet humans — or filters employing different techniques — can often spot these characteristics as unique to spam.

1.2.4 Payload Bearing

The payload of a spam message may be obvious or hidden; in either case spam abatement may be enhanced by identifying the payload and the mechanism by which actions triggered by it profit the spammer. Obvious payloads include product names, political mantras, web links, telephone numbers, and the like. These may be in plain text, or they may be obfuscated so as to be readable by the human but appear benign to the computer. Or they may be obfuscated so as to appear benign to the human but trigger some malicious computer action.

The payload might consist of an obscure word or phrase like "gouranga" or "platypus race" in the hope that the recipient will be curious and perform a web search and be delivered to the spammer's web page or, more likely, a paid advertisement for the spammer's web page. Another form of indirect payload delivery is *backscatter*: The spam message is sent to a non-existent user on a real mail server, with the (forged) return address of a real user. The mail server sends an "unable to deliver" message to the (forged) return address, attaching and thus delivering the spam payload. In this scenario we consider

both the original message (to the non-existent user) and the "unable to deliver" message to be spam, even though the latter is transmitted by a legitimate sender.

The payload might be the message itself. The mere fact that the message is not rejected by the mail server may provide information to the spammer as to the validity of the recipient's address and the nature of any deployed spam filter. Or if the filter employs a machine learning technique, the message may be designed to *poison* the filter [70, 72, 191], compromising its ability to detect future spam messages.

1.3 Spam Consequences

The transmission of spam — whether or not its payload is delivered and acted upon — has several negative consequences.

1.3.1 Direct Consequences

Spam provides an unregulated communication channel which can be used to defraud targets outright, to sell shoddy goods, to install viruses, and so on. These consequences are largely, but not exclusively, borne by the victims. For example, the victim's computer may be used in further spamming or to launch a cyber attack. Similarly, the victim's identity may be stolen and used in criminal activity against other targets.

1.3.2 Network Resource Consumption

The vast majority of email traffic today is spam. This traffic consumes bandwidth and storage, increasing the risk of untimely delivery or outright loss of messages. For example, during the Sobig virus outbreak of 2003, the author's spam filter correctly identified the infected messages as spam and placed them in a quarantine folder. However, the total volume of such messages exceeded 5 GB per day, quickly exhausting all available disk space resulting in non-delivery of legitimate messages.

1.3.3 Human Resource Consumption

It is an unpleasant experience and a waste of time to sort through an inbox full of spam. This process necessarily interferes with the

timeliness of email because the recipient is otherwise occupied sorting through spam. Furthermore, the frequent arrival of spam may preclude the use of email arrival alerts, imposing a regimen of batch rather than on-arrival email reading, further compromising timeliness.

Over and above the wasted time of routinely sifting through spam, some spam messages may consume extraordinary time and resources if they appear legitimate and cannot be dismissed based on the summary information presented by the mail reader's user interface. More importantly, legitimate email messages may be overlooked or dismissed as spam, with the consequence that the message is missed.

A spam filter may mitigate any or all of the problems associated with human resource consumption, potentially reducing effort while also enhancing timeliness and diminishing the chance of failing to read a legitimate message.

1.3.4 Lost Email

Sections 1.3.2 and 1.3.3 illustrate situations in which spam may cause legitimate email to be lost or overlooked. Spam abatement techniques may, of course, also cause legitimate email to be lost. More generally, spam brings the use of email into disrepute and therefore discourages its use. Users may refuse to divulge their email addresses or may obfuscate them in ways that inhibit the use of email as a medium to contact them.

In evaluating the consequences of email loss (or potential loss), one must consider the probability of loss, the importance and time criticality of the information, and the likelihood of receiving the information, or noticing its absence, via another medium. These consequences vary from message to message, and must be considered carefully in evaluating the effectiveness of any approach to spam abatement, including human sorting.

1.4 The Spam Ecosystem

Spam and spam filters are components of a complex interdependent system of social and technical structures. Many spam abatement proposals seek to alter the balance within the system so as to render

spam impractical or unprofitable. Two anonymous whimsical articles [61, 1] illustrate the difficulties that arise with naive efforts to find the Final Ultimate Solution to the Spam Problem (FUSSP). Crocker [43] details the social issues and challenges in effecting infrastructure-based solutions such as protocol changes and sender authentication. Legislation, prosecution and civil suits have been directed at spammers [101, 124], however, the international and underground nature of many spam operations makes them difficult to target. Spammers and legitimate advertisers have taken action against spam abatement outfits [119]. Vigilante actions have been initiated against spammers, and spammers have reacted in kind with sabotage and extortion [103]. Economic and technical measures have been proposed to undermine the profitability of spam [89, 138].

A detailed critique of system-wide approaches to spam abatement is beyond the scope of this survey, however, it is apparent that no FUSSP has yet been found nor, we daresay, is likely to be found in the near future. And even if the email spam problem were to be solved, it is not obvious that the solution would apply to spam in other media. The general problem of adversarial information filtering [44] — of which spam filtering is the prime example — is likely to be of interest for some time to come.

We confine our attention to this particular problem — identifying spam — while taking note of the fact that the deployment of spam filters will affect the spam ecosystem, depending on the nature of their deployment. The most obvious impact of spam filtering is the emergence of technical countermeasures in spam; it is commonly held that filtering methods become obsolete as quickly as they are invented. Legal retaliation is also a possibility: Spammers or advertisers or recipients may sue for damages due to the non-delivery of messages. Spam filtering is itself a big business, a tremendous amount of money rests on our perception of which spam methods work best, so the self-interest of vendors may be at odds with objective evaluation. And filter market share will itself influence the design of spam.

In general, we shall consider the marginal or incremental effects of spam filter deployment, and mention in passing its potential role in revolutionary change.

1.5 Spam Filter Inputs and Outputs

We have defined a spam filter to be an automated technique to identify spam. A spam filter with perfect knowledge might base its decision on the content of the message, characteristics of the sender and the target, knowledge as to whether the target or others consider similar messages to be spam, or the sender to be a spammer, and so on. But perfect knowledge does not exist and it is therefore necessary to constrain the filter to use well defined information sources such as the content of the message itself, hand-crafted rules either embedded in the filter or acquired from an external source, or statistical information derived from feedback to the filter or from external repositories compiled by third parties.

The desired result from a spam filter is some indication of whether or not a message is spam. The simplest result is a binary categorization — spam or non-spam — which may be acted upon in various ways by the user or by the system. We call a filter that returns such a binary categorization a *hard classifier*. More commonly, the filter is required to give some indication of how likely it considers the message to be spam, either on a continuous scale (e.g., $1 =$ *sure spam*; $0 =$ *sure non-spam*) or on an ordinal categorical scale (e.g., *sure spam*, *likely spam*, *unsure*, *likely non-spam*, *sure non-spam*). We call such a filter a *soft classifier*. Many filters are internally soft classifiers, but compare the soft classification result to a sensitivity threshold t yielding a hard classifier. Users may be able to adjust this sensitivity threshold according to the relative importance they ascribe to correctly classifying spam vs. correctly classifying non-spam (see Section 1.7).

A filter may also be called upon to justify its decision; for example, by highlighting the features upon which it bases is classification. The filter may also classify messages into different *genres* of spam and good mail (cf. [42]). For example, spam might be advertising, phishing or a Nigerian scam, while good email might be a personal correspondence, a news digest or advertising. These genres may be important in justifying the spam/non-spam classification of a message, as well in assessing its impact (e.g., does the user really care much about the distinction between spam and non-spam advertising?).

1.5.1 Typical Email Spam Filter Deployment

Figure 1.5 outlines the typical use of an email spam filter from the perspective of a single user. Incoming messages are processed by the filter one at a time and classified as ham (a widely used colloquial term for non-spam) or spam. Ham is directed to the user's inbox which is read regularly. Spam is directed to a quarantine file which is irregularly (or

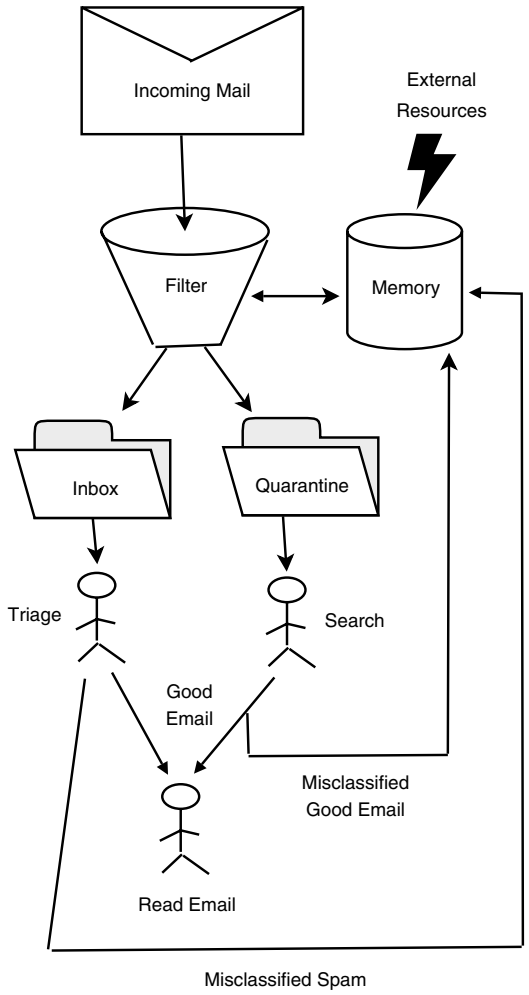

Fig. 1.5 Spam filter usage.

never) read but may be searched in an attempt to find ham messages which the filter has misclassified. If the user discovers filter errors — either spam in the inbox or ham in the quarantine — he or she may report these errors to the filter, particularly if doing so is easy and he or she feels that doing so will improve filter performance. In classifying a message, the filter employs the content of the message, its built-in knowledge and algorithms, and also, perhaps, its memory of previous messages, feedback from the user, and external resources such as blacklists [133] or reports from other users, spam filters, or mail servers. The filter may run on the user's computer, or may run on a server where it performs the same service for many users.

1.5.2 Alternative Deployment Scenarios

The filter diagrammed in Figure 1.5 is on-line in that it processes one message at a time, classifying each in turn before examining the next. Furthermore, it is passive in that it makes use only of information at hand when the message is examined. Variants of this deployment are possible, only some of which have been systematically investigated:

- Batch filtering, in which several messages are presented to the filter at once for classification. This method of deployment is atypical in that delivery of messages must necessarily be delayed to form a batch. Nevertheless, it is conceivable that filters could make use of information contained in the batch to classify its members more accurately than on-line.
- Batch training, in which messages may be classified on-line, but the classifier's memory is updated only periodically. Batch training is common for classifiers that involve much computation, or human intervention, in harnessing new information about spam.
- Just-in-time filtering, in which the classification of messages is driven by client demand. In this deployment a filter would defer classification until the client opened his or her mail client, sorting the messages in real-time into inbox and quarantine.

- Deferred or tentative classification, in which the classification of messages by the filter is uncertain, and either delivery of the message is withheld or the message is tentatively classified as ham or spam. As new information is gleaned, the classification of the message may be revised and, if so, it is delivered or moved to the appropriate file.
- Receiver engagement, in which the filter probes the recipient (or an administrator representing the recipient) to glean more information as a basis for classification. Active learning may occur in real-time (i.e., the information is gathered during classification) or in conjunction with deferred or tentative classification. An example of real-time active learning might be a user interface that solicits human adjudication from the user as part of the mail reading process. A more passive example is the use of an "unsure" folder into which messages are placed with the expectation that the user will adjudicate the messages and communicate the result to the filter.
- Sender engagement, in which the filter probes the sender or the sender's machine for more information. Examples are challenge–response systems and greylisting. These filters may have a profound effect on the ecosystem as they, through their probes, transmit information back to the sender. Furthermore, they introduce delays and risks of non-delivery that are difficult to assess [106]. It may be argued that these techniques which engage the sender do not fit our notion of "filter." Nevertheless, they are commonly deployed in place of, or in conjunction with, filters and so their effects must be considered.
- Collaborative filtering, in which the filter's result is used not only to classify messages on behalf of the user, but to provide information to other filters operating on behalf of other users. The motivation for collaborative filtering is that spam is sent in bulk, as is much hard-to-classify good email, so many other users are likely to receive the same or similar messages. Shared knowledge among the filters promises to

make such spam easier to detect. Potential pitfalls include risks to privacy and susceptibility to manipulation by malicious participants.

- Social network filtering, in which the sender and recipient's communication behavior are examined for evidence that particular messages might be spam.

1.6 Spam Filter Evaluation

Scientific evaluation, critical to any investigation of spam filters, addresses fundamental questions:

- Is spam filtering a viable tool for spam abatement?
- What are the risks, costs, and benefits of filter use?
- Which filtering techniques work best?
- How well do they work?
- Why do they work?
- How may they be improved?

The vast breadth of the spam ecosystem and possible abatement techniques render impossible the direct measurement of these quantities; there are simply too many parameters for any single evaluation or experiment to measure all their effects at once. Instead, we make various simplifying assumptions which hold many of the parameters constant, and conduct an experiment to measure a quantity of interest subject to those assumptions. Such experiments yield valuable insight, particularly if the assumptions are reasonable and the quantities measured truly illuminate the question under investigation. The validity of an experiment may be considered to have two aspects: *internal validity* and *external validity* or *generalizability*. Internal validity concerns the veracity of the experimental results under the test conditions and stated assumptions; external validity concerns the generalizability of these results to other situations where the stated assumptions, or hidden assumptions, may or may not hold. Establishing internal validity is largely a matter of good experimental design; establishing external validity involves analysis and repeated experiments using different assumptions and designs.

It is all too easy to fix on one experimental design and set of test conditions and to lose sight of the overall question being posed. It is similarly all too easy to dismiss the results a particular experiment due to the limitations inherent in its assumptions. For example, filters are commonly evaluated using tenfold cross validation [95], which assumes that the characteristics of spam are invariant over time. It would be wrong to conclude, without evidence, that the results of tenfold cross validation would be the same under a more realistic assumption. It would be equally wrong to dismiss out of hand the results of experiments using this method, to do so would entail dismissal of all scientific evidence, as there is no experiment without limiting assumptions. We would be left with only testimonials, or our own uncontrolled and unrepeatable observations, to judge the efficacy of various techniques. Instead, it is appropriate to identify assumptions that limit the generalizability of current results, and to conduct experiments to measure their effect.

The key to evaluation is to conduct experiments that glean the most informative results possible with reasonable effort, at reasonable cost, in a reasonable time frame. Simple assumptions — such as the assumption that the characteristics of spam are time-invariant — yield simple experiments whose internal validity is easy to establish. Many such experiments may reasonably be conducted to explore the breadth of solutions and deployment scenarios. Further experiments, with different simple assumptions, help to establish the external validity of the results. These experiments serve to identify the parameters and solutions of interest, but are inappropriate for evaluating fine differences. Experimental designs that more aptly model real filter deployment tend to be more complex and costly due to challenges in logistics, controlling confounding factors, and precisely measuring results. Such experiments are best reserved for methods and parameters established to be of interest by simpler ones.

Among the common assumptions in spam filter evaluation are:

- Batch or on-line filtering.
- Existence of training examples.
- Accurate "true" classification for training messages.

- Accurate "true" classification for test messages.
- Recipient behavior, e.g., reporting errors.
- Sender behavior, e.g., resending dropped messages.
- Availability of information, e.g., whitelists, blacklists, rule bases, community adjudication, etc.
- Language of messages to be filtered, e.g., English only.
- Format of messages to be filtered, e.g., text, html, ASCII, Unicode, etc.
- Quantifiable consequences for misclassification or delay [96].
- Time invariance of message characteristics [57].
- Effect (or non-effect) of spam filter on sender.
- Effect (or non-effect) of spam filter on recipient.

Laboratory and field experiments play complementary roles in scientific investigation. Laboratory experiments investigate the fundamental properties of filters under controlled conditions that facilitate reproducibility, precise measurement, and ongoing evaluation. Such conditions necessitate the adoption of simplifying assumptions such as those listed above. Field experiments, on the other hand, rely on different assumptions, are very difficult to control and their results very difficult to compare. Methods from scientific fields such as epidemiology [139] may be used to measure the effects of spam filters, however, such methods are considerably more expensive and less precise than laboratory experiments.

1.7 Evaluation Measures

An ideal spam filter would autonomously, immediately, and perfectly identify spam as spam and non-spam as non-spam. To evaluate a spam filter, we must somehow measure how closely it approximates this ideal. Furthermore, whatever measurement we use should reflect the suitability of the filter for its intended purpose.

Our ideal suggests four dimensions along which filters should be judged: autonomy, immediacy, spam identification, and non-spam identification. It is not obvious how to measure any of these dimensions separately, nor how to combine these measurements into a single one

for the purpose of comparing filters. Nevertheless, reasonable standard measures are useful to facilitate comparison, provided that the goal of optimizing them does not replace that of finding the most suitable filter for the purpose of *spam filtering*.

A fully autonomous spam filter would require no configuration, no updates, no training, and no feedback. Is a filter that receives nightly signature files from a central source more or less autonomous than one that requires user feedback on errors? Is the burden collecting a sample of labeled messages for training more or less onerous than delivering updates or user feedback? We cannot imagine a quantitative measure that could capture the differences between filters in this regard. They must be controlled when evaluating the other dimensions, but the relative amounts that filters employing these techniques diverge from the ideal will remain a matter of qualitative, not quantitative, evaluation.

An immediate filter would introduce no CPU, disk or network overhead, and would not defer its decision pending the arrival of new information. We may measure or analyze the efficiency of the filter; modeling external delay is more difficult. Reasonable delays may not matter much, but it is difficult to quantify reasonable. A two second delay per message may be reasonable for an end user, if the filter runs continuously. If, however, the filter is launched only when the inbox is opened, a user with 100 new messages may find him or herself waiting for several minutes. A mail server supporting 100 clients may also find a 2 second delay per message acceptable; a server supporting 100,000 clients may not.

Failures to identify non-spam and spam messages have materially different consequences. Misclassified non-spam messages are likely to be rejected, discarded or placed in quarantine. Any of these actions substantially increases the risk that the information contained in the message will be lost, or at least delayed substantially. Exactly how much risk and delay are incurred is difficult to quantify, as are the consequences, which depend on the nature of the message. Some messages are simply more important than others, while others are more likely to be missed, or delivered by separate channels, if they go astray. For example, advertising from a frequent flier program is less important than an electronic ticket receipt, but the latter is certain to be missed

and retrieved, either from quarantine or from a different medium. On the other hand, failure to deliver immediately a message from one's spouse to "come home right away" could have serious consequences. For these reasons, one must be cautious about characterizing failures to deliver non-spam in terms of a simple proportion, as such failures are rare events with causes and consequences that defeat statistical inference. With this caveat, *false positive rate* (*fpr*) — the proportion of non-spam messages identified as spam (cf. Table 4.1) — is a reasonable first-order measure of failures to identify non-spam.

Failures to identify spam also vary in importance, but are generally less important than failures to identify non-spam. Viruses, worms, and phishing messages may be an exception, as they pose significant risks to the user. Other spam messages have impact in proportion to their volume; so *false negative rate* (*fnr*) — the proportion of spam identified as non-spam — is an apt measure.

The overall efficacy of a hard classifier may be characterized by the pair (*fpr*, *fnr*). A classifier with lower *fpr* and *fnr* than another is superior.[4] Whether a classifier with a lower *fpr* and higher *fnr* is superior or inferior depends on the user's sensitivity to each kind of error.

The efficacy of a soft classifier with an adjustable threshold t may be characterized by the set of all distinguishable (*fpr*, *fnr*) pairs for different values of t. This set of points defines a receiver operating characteristic (ROC) curve (cf. [58, 82, 166]). A filter whose ROC curve is strictly above that of another is superior in all deployment situations, while a filter whose ROC curve crosses that of another is superior for some threshold settings and inferior for others.

The area under the ROC curve (AUC) provides an estimate of the effectiveness of a soft classifier over all threshold settings. AUC also has a probabilistic interpretation: it is the probability that the classifier will award a random spam message a higher score than a random ham message. In the spam filtering domain, typical AUC values are of the order of 0.999 or greater, for clarity, we often report $(1 - AUC)\%$, the

[4] Under the assumption that all messages have equal misclassification cost. See Kolcz et al. [96]

area above the ROC curve, as a percentage. So $AUC = 0.999$ would be reported instead as $(1 - AUC)\% = 0.1$.

False positive rate, false negative rate, and receiver operating characteristic curves are the standard measures of (e.g., medical) diagnostic test effectiveness [66]. This review uses primarily these measures; a spam filter is a diagnostic test applied to email for which a positive result indicates spam, and a negative result indicates non-spam. In Section 4.6, we review the diverse set of measures that have been applied to spam filters, and argue that diagnostic test methods are most suitable for comparative analysis.

1.8 Systematic Review

Spam filters have evolved quickly — and somewhat separately — in several milieux with different histories, objectives, evaluation methods, and measures of success. Practitioners have been concerned primarily with keeping their heads above water, delivering spam filters as quickly as possible to combat an ever-increasing tide of spam. Academics have, in large part, studied the problem as an application of the techniques and methods of information retrieval, machine learning and computer systems. Commercial product development and product testing involve yet another set of interests, methods, and measures of success. These groups have had limited interaction; as a consequence, it is exceedingly difficult to deduce from the literature or other sources the relative performance and promise of current and proposed spam filter methods.

The literature, including the so-called gray literature (dissertations, technical reports, popular press articles, commercial reports, web publications, software documentation and cited unpublished works) was searched for articles describing a spam filter or spam filtering method and an evaluation of its effectiveness. Articles were characterized by their methods and assumptions according to the taxonomy presented here. Where sufficient information was given in the article, quantitative results were recast as (fpr, fnr) or summarized using $1 - AUC$ expressed as a percentage, otherwise the results were omitted from this review. Results derived using incorrect methodology, or results that

are insufficiently noteworthy because their results are represented better elsewhere, were similarly omitted. Several hundred articles were considered for this review; perhaps one-third of them met our selection criteria.

Certain aspects of spam filtering are well represented in the literature, while others are hardly represented or not represented at all. This review reflects this uneven coverage, reporting some aspects in detail while leaving others as largely uncharted territory.

2

Spam Classifiers — Hand-Crafted

At the core of every spam filter is a classifier that estimates, based on available information, whether a given message is spam or not. The exact nature of the available information depends on the deployment scenario and the requirements of the classifier, the estimate is either categorical (hard classification), or ordinal or continuous (soft classification).

For the purpose of this review, we separate available information into two components: the message m itself, and extrinsic information e gathered from elsewhere. m is defined broadly to include not only the text of the message, but all information specific to it, such as header and envelope information, recipient, time of arrival, and so on. e encompasses all other information, including the filter's memory and external sources (see Figure 1.5) as well as rules encoded in the filter itself.

Nearly all filters in use today employ hand-crafted classifiers, or machine learning methods (cf. [153]), or both. The overall objective is the same — to classify messages as spam or non-spam — but the approaches, and the perspectives of those investigating the approaches, differ considerably. To reflect this diversity, this discussion is divided into two sections. The present section defines formally the classification

problem, and reviews hand-crafted spam classifiers within the context of this definition. The following section reviews machine learning methods to construct spam classifiers.

2.1 Definitions

We assume that every message m is either spam or not; the universe of messages M is partitioned into two sets, *spam* and *non-spam* such that every $m \in M$, where $M = spam \cup non\text{-}spam$ and $spam \cap non\text{-}spam = \varnothing$. Membership in *spam* is specified by our definition in Section 1, or by a similar subjective definition, which can be neither formalized nor determined with certainty for all messages. The key challenge for spam filtering is to build a classifier — a concrete function that answers accurately, for any m, the question "given m, is $m \in spam$?"

An ideal spam classifier would be a total function

$$isspam : M \rightarrow \{true,\ false\}$$

such that $isspam(m) = true$ if and only if $m \in spam$. Due to the definition of spam, such a classifier cannot be totally realized, instead we build an approximate classifier c such that $c \approx isspam$. Hard and soft classifiers employ different notions of *approximate*. A hard classifier

$$c : M \rightarrow \{true,\ false\}$$

approximates $isspam$ to the extent that $c(m) = isspam(m)$ for all $m \in M' \subseteq M$. A soft classifier

$$c : M \rightarrow \mathbb{R}$$

approximates $isspam$ to the extent that $c(m) > c(m')$ for all $(m, m') \in P \subseteq spam \times non\text{-}spam$. A hard classifier c_h may be defined in terms of a soft classifier c_s and a fixed threshold t:

$$c_h(m) = \begin{cases} true & (c_s(m) > t) \\ false & (c_s(m) \leq t) \end{cases} .$$

A utility measure quantifies the extent to which c approximates the ideal classifier. A naive utility measure for a hard classifier is $accuracy = \frac{|M'|}{|M|}$. Alternatively, a cost measure such as

$error = 1 - accuracy$ quantifies the extent to which c departs from the ideal. *Accuracy* and *error* are commonly reported and commonly optimized in filter construction, notwithstanding their shortcomings as measures of spam filter effectiveness. As a pair, the cost functions

$$fpr = \frac{|non\text{-}spam \cap \{m|c(m) = true\}|}{|non\text{-}spam|}$$

and

$$fnr = \frac{|spam \cap \{m|c(m) = false\}|}{|spam|}$$

distinguish between spam and non-spam error rates. For a soft classifier, we use the cost function

$$1 - AUC = \frac{|spam \times non\text{-}spam \cap \{(m, m')|c(m) < c(m')\}|}{|spam| * |non\text{-}spam|}.$$

It is useful to formalize the process of building the classifier from available evidence separately from the classifier itself. A classifier constructor C takes as input extrinsic information e from domain E and yields either a hard classifier $c : M \to \{true,\ false\}$ or a soft classifier $c : M \to \mathbb{R}$; that is $C : E \to (M \to \{true,\ false\})$ or $C : E \to (M \to \mathbb{R})$. Although represented formally as a function that returns a function, C may be effected by hand, for example, when rules or patterns are embedded into a program that implements c, the programmer implements C by hand. On the other hand, C may be an explicit automatic step in filtering, for example, when a supervised learning algorithm induces c from a set of labeled training examples. Or C may be implicit, in effect constructing a new classifier based on new evidence e for every message m, such is the case with on-line filters that consider all information available at the time of classification. Whether the constructor is manual, explicit or implicit, it is useful to consider a spam filter $f : (E \times M) \to \{true,\ false\}$ or $f : (E \times M) \to \mathbb{R}$ to be a function on two inputs defined in terms of applying a classifier constructor to e, and then the resulting classifier to m; that is, $f(e, m) = (C(e))(m)$.

2.2 The Human Classifier

If no automated filter is installed, the role of filtering or spam avoidance is assumed by the human recipient, that is, c is constructed and applied in the user's head. A human can usually identify email spam from a summary line containing the sender's name, subject, and time of delivery, that is, based on a subset of m. Occasionally the user will mistakenly open a spam message[1] thinking it is non-spam, or delete or overlook a non-spam message thinking it is spam. These occurrences constitute false negatives and false positives, respectively. The human classifier c therefore has nonzero fpr and fnr which could be measured by a suitable experiment.

Studies which assess the costs and benefits of manual filtering are rare. Commercial studies [2, 60, 59] have estimated categorizing spam to cost of the order of 10 minutes of productivity per day, based on self-reporting through surveys or assumptions regarding the number of spams received and the time required to identify each spam message. Total spam volume has increased considerably since these studies were conducted, while filter use has become common. The studies' assumptions imply that, absent filter use, lost productivity would be proportional to the volume of spam and hence considerably higher than 10 minutes per day.

The risk of email loss due to deployment of a spam filter must be compared with the risk due to human filtering. Yerazunis [194] carefully adjudicated the same 1900 email messages on two separate occasions, observing a disagreement on 3 messages. Since Yerazunis examined the full text of each message and spent considerably more time than assumed by the aforementioned studies, his disagreement rate (0.16%) would appear to be a lower bound on that achievable by humans in the course of reading a mixture of *spam* and *non-spam*.[2] Yih et al. [199] report a 3% error rate by Hotmail users in manually classifying a random selection of their incoming messages. Using a

[1] A spam message may be able to exploit the user interface to deliver its payload without any explicit action on the part of the user.

[2] Assuming that Yerazunis' sample and efforts are representative of email and user's capabilities in general.

Web interface, Graham-Cumming [73] collected 357,380 judgments for 92,186 *spam* and *non-spam* messages from self-selected participants. His results show an average disagreement among participants of about 10%, and an average disagreement with the official classification of about 6.75% [149].

Spam senders employ social engineering to increase the error rate of human filtering by composing the messages so as to convince the user to read and respond to the message. Benign subject lines or urgent subject lines that may appear to be from an acquaintance or authority are common (cf. Figure 2.1). Spam may be designed to exploit system insecurities to deliver its payload regardless of whether or not the message is selected to be read. Examples include automatically rendered preview images, automatically transmitted delivery receipts, or short subject lines that contain the payload itself.

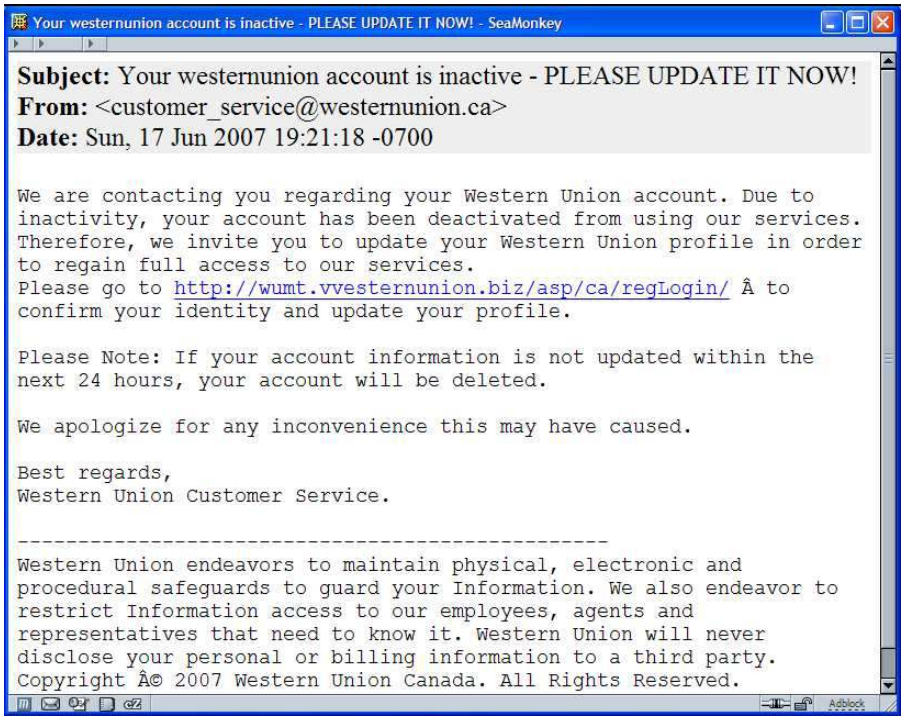

Fig. 2.1 Social engineering.

2.3 *Ad Hoc* Classifiers

Ad hoc filtering techniques are devised in response to a particular form of spam or in response to the particular needs of a single user or restricted set of users. A user or system administrator may choose to block or quarantine, for example, messages that

- Contain a particular word or phrase (e.g., `teen` or `MAKE MONEY`) in the subject.
- Contain attached files of a particular type (e.g., `scr`).
- Are sent in a particular format (e.g., Unicode).
- Appear to be sent from one of a particular set (a *black list*) of users, hosts or domains (e.g., `.ru`).
- Appear *not* to be sent from one of a trusted set (a *white list*) of users, hosts or domains (e.g., `gvcormac@uwaterloo.ca`).

Most email servers and clients support the definition of simple patterns to implement these policies. Because the patterns are written and modified in response to specific threats and needs, it is very difficult to study their efficacy and generalizability in a systematic way.

Ad hoc rules may have unforeseen consequences. Features that "obviously" indicate spam may occur, albeit rarely, in important non-spam messages. In an effort to block a flood of non-delivery messages resulting from a spam campaign that impersonated the author, an *ad hoc* filter was created to block messages quoting a "`From:`" line containing his email address, but not his name as would be contained in any message actually sent by him. This approach was very effective, except for two unfortunate circumstances that resulted in false positives:

(1) The filter blocked a response from a foreign embassy to an inquiry regarding the status of a visa application. The embassy response quoted the inquiry verbatim, except that for some reason (possibly security) it removed the sender's name from the `From:` field.
(2) A conference management system, configured to send email on behalf of the author, did not include his name.

The benefits and risks of *ad hoc* filtering are exceedingly difficult to measure, such reports appear to be absent from the literature.

2.4 Rule-Based Filtering

While it is difficult to draw a firm distinction between *ad hoc* and rule-based systems, we generally distinguish them by the following criteria:

- Rules are specified using a specific formal notation, as opposed to being embedded in the various tools that implement them.
- Multiple rules are brought to bear in classifying a message.
- The classifier is itself easily identified as a "black box" with specific inputs and outputs.
- Rule bases are designed to be more general to facilitate widespread deployment.

The ubiquitous SpamAssassin [164] open-source spam filter is the leading example of a rule-based filter.[3] While SpamAssassin consists of several thousand lines of Perl code, the rules themselves (dubbed "tests" in SpamAssassin parlance) are encapsulated in a few configuration files. Each test may be specified as a pattern which is applied to the email message, or as a query to a built-in routine that performs computation or data access that cannot be effected by a simple pattern. An example of the former is the test `MILLION_USD`, implemented as the pattern

```
/Million\b.{0,40}\b(?:United States? Dollars?|USD)/i,
```

which indicates a Nigerian scam. Examples of the latter are `USER_IN_BLACKLIST` and `USER_IN_WHITELIST`. With each test is associated a score to indicate whether the test indicates spam or non-spam, and how strongly so. Overall, the message is classified as spam if the

[3] SpamAssassin has now evolved into a hybrid filter, incorporating within its rules the results of many non-rule-based components, such as whitelists, blacklists, and machine-learning. Furthermore, combining weights for rules are derived using machine learning. See, for example, Figure 2.2 which displays the result of applying SpamAssassin to the message in Figure 2.1.

sum of the scores of the tests that "fire" exceeds some threshold, typically 5. SpamAssassin is shipped with a default set of about 800 rules, along with default scores and a default threshold. The user or administrator may alter the rules and scores, many custom rule-sets are maintained by third parties and available for download.

Spam countermeasures against rule-based filters employ the same general techniques as against *ad hoc* filters: spam messages are engineered to avoid the rules. To the extent that rules are better formalized and shared, spammers are better able systematically to test whether or not their spam avoids triggering the rules. Spammers routinely install and test against common filters like SpamAssassin.

2.5 Whitelists

A whitelist is a list of senders whose email should be delivered, notwithstanding the outcome of the spam filter. The address book of the recipient is commonly used as a simple whitelist, under the assumption that spam would be unlikely to arrive from these email addresses. The whitelist need not be an address book, and it may contain domain names or IP addresses instead of email addresses. Then the user may, using tools provided by his or her email client, maintain his or her whitelist explicitly, or automated tools may deduce the whitelist from email traffic (e.g., the auto-whitelist feature of SpamAssassin). Global whitelists may be maintained by email service providers or other organizations.

Such whitelists may be compromised if the spammer can determine or guess an address on the list and "spoof" the spam message so as to appear to have been sent from that address. One way to guess addresses likely to be whitelisted is simply to use real email addresses harvested from the web or elsewhere. Today, almost all spam is sent from such an address, to maximize the probability, however small, that address is in the recipient's whitelist. More sophisticated data mining techniques cluster email addresses by domain or by their co-occurrence on web pages. The author, for example, receives a disproportionate number of spam messages spoofed to appear to be sent by his co-authors for various published works. Viruses and other malware can easily harvest

the address books and other files of compromised machines for email addresses likely to be found in correspondents' whitelists.

Spoofing email is all too easy. The internet message format enforces no authentication at all, the sender's email address means nothing more than the return address written on the envelope of conventional mail. Email messages contain so-called `Received:` headers, akin to postmarks, which provide some evidence of the veracity of the sender, but they are not easy to parse and it is not easy to verify that they themselves are not forged [111].

2.6 Blacklists

A blacklist is a list of senders, domains or IP addresses messages from which are deemed to be spam. While whitelists and blacklists are, in a sense, complementary, they are not symmetric. Unlike a whitelist, which must be used in conjunction with another filter, a blacklist acts as a filter in its own right. However, a single user cannot compile a large enough list of possible spam senders for a personal blacklist to be of much use as a general spam filter. Instead, organizations compile large databases of IP addresses or domain names known to or likely to send spam, and spam filters query these databases in real-time.

The first major example of such a list was the Real-time Blackhole List (RBL) for the Mail Abuse Prevention System (MAPS) [116]. RBL and similar databases, collectively known as Domain Name Server Blacklists (DNSBL) use the internet Domain Name Server (DNS) as an efficient mechanism to query the blacklists [15].

Populating and maintaining comprehensive, accurate blacklists presents substantial challenge. Spam is sent from a vast number of sources, and the blacklist's effectiveness depends on being able to identify the source of most messages. One method is to examine a large number of spam messages and to add the source of spam messages to the blacklist. This approach dictates that a very large number of messages be examined, few organizations have the resources to do so themselves. One approach is to rely on the community at large to report spam senders, another is to use *spam traps* or *honeypots* — email addresses which have no legitimate purpose — and assume that

all email sent to such addresses is spam [130]. Technical aspects of the IP address, such as the fact that it is allocated by the Dynamic Host Control Protocol (DHCP) may be used to populate a blacklist. Such a *dial-up blacklist* is effective because a large fraction of spam is sent by compromised machines or by unscrupulous clients of retail internet service providers, these machines typically have dynamic IP addresses.

The effectiveness of a blacklist depends on its completeness, its accuracy, and the inability of spam to spoof non-blacklisted addresses.

2.7 Collaborative Spam Filtering

The nature of spam is such that each message is typically sent to a vast number of recipients. Chances are that a particular recipient is not the first to receive any particular message; it is likely to have not only been received but also recognized as spam by somebody else (or somebody else's spam filter or honeypot). Collaborative spam filtering is the process of capturing, recording, and querying these early judgments [46, 50, 157]. The false positive and false negative rates of this approach depend on both human and technical factors which limit the timeliness, completeness, and accuracy of these three steps. Ideally, collaborative filtering could approach, in the aggregate, a false negative rate of $\frac{1}{n}$, where n is the number of copies of each message sent by the spammer, and a false positive rate of 0.

The completeness of the approach is limited by the number of users who participate in the system, the effective value of n, therefore, is not the number of messages sent by the spammer, but the number of messages sent by the spammer to a participant in the collaborative filtering system. Participants may fail to recognize a message as spam, or, they may not bother to record their judgments. Spammers may send out thousands of spam messages in a few minutes in an effort to capitalize on the time it takes participants to recognize and report spam. Participants may inadvertently mark non-spam messages as spam, malicious users — perhaps the spammers themselves — may do so deliberately to increase the false positive rate and therefore compromise the system.

An essential component of a collaborative spam filtering system is a real-time database of known-spam messages that can be updated and

queried by diverse users. Practical considerations render it impossible to store entire messages in this database: it would be too large; query and update times would be excessive; privacy of messages might be compromised. A cryptographic hash function preserves privacy and allows for efficient query and update, but works only if the spam messages sent by a spammer are identical. In fact, they will differ somewhat due to their headers and other artifacts of delivery. And in an effort to defeat collaborative spam filters, spammers commonly generate messages that are trivially different. The crux of collaborative spam filtering is to record messages in such a way that nearly identical messages will be matched, while ones that differ more substantially will not [99, 97].

2.8 Challenge–Response

A challenge–response system [177] demands that a sender take special action in order to have his or her message delivered to and added to the whitelist of a particular recipient. The special action may be as simple as clicking a link or resending, or may involve supplying credentials, solving a puzzle [26], making a payment [171, 135], or performing a time-consuming computation [55]. The task is designed to as to be easy enough for a legitimate sender to accomplish, but too difficult, time consuming or expensive for a spammer sending thousands or millions of messages.

Challenge–response systems differ in the mechanisms used to communicate the challenge–response. A "classical" challenge–response, implemented by the recipient's email server, holds the message and issues a challenge message to the (purported) sender with instructions on how to respond. Since the vast majority of spam sender addresses are forged, these challenges themselves may be considered spam. It is apparent that if every user employed such a spam filter, no email would ever be delivered. Furthermore, some legitimate mail messages — for example, mailing lists and responses to web transactions — are sent by automated servers that would be unlikely to respond to challenges. For these reasons, their use is widely disfavored [76].

Pre-challenge systems [137] embed the challenge–response process into the sender's interface, evidence of a successful response is either

embedded in the message or transmitted by a separate interface. One of the commonest pre-challenge tasks is the *human interactive proof* [23, 25, 26, 159].

2.9 Greylisting

Greylisting is a form of challenge–response that engages the sender's mail delivery software rather than the sender [77, 106]. Messages are summarily rejected with a "soft" error code that invites retransmission. Under the assumption that non-spam is more likely to be re-sent than spam, this method acts as a spam filter.

Greylisting, like challenge–response, imposes delay and may result in lost legitimate messages, as it depends on the sender to retransmit and on the filter's ability to recognize the retransmission as such. Furthermore, lost messages are nearly impossible to detect or recover. Greylisting is trivial to defeat, all the spammer must do is to implement the invited retransmission. Presently, many spammers find it more cost effective not to, estimates range as high as 80%. The rate of lost legitimate email is difficult to measure, it is certainly nonzero [106].

2.10 Technical Measures

As noted above, many spam filtering techniques rely on being able to verify the identity of the sender or the sender's domain. Several proposals have been made to enhance the internet mail protocol to provide secure authentication of messages. These include Sender Policy Framework (SPF) [193], Sender-ID [115], and Domain Keys Identified Mail (DKIM) [110] which have been deployed by several major email service providers. To date, they are far from ubiquitous, and their impact is yet to be determined.

2.11 Combined Methods

Ad hoc and rule-based spam filters employ a "cocktail" of methods [11], composing separate classifiers to form one overall classifier. The composition itself is typically done in an *ad hoc* manner; in a pipeline, for example, filters sequentially examine the message until it is summarily

```
Content analysis details:   (10.2 points, 5.0 required)

 pts rule name              description
---- ---------------------- --------------------------------------------------
 1.0 NO_REAL_NAME           From: does not include a real name
 2.0 BIZ_TLD                URI: Contains an URL in the BIZ top-level domain
 0.0 BAYES_50               BODY: Bayesian spam probability is 40 to 60%
                            [score: 0.5249]
 4.1 URIBL_JP_SURBL         Contains an URL listed in the JP SURBL blocklist
                            [URIs: vvesternunion.biz]
 3.0 URIBL_OB_SURBL         Contains an URL listed in the OB SURBL blocklist
                            [URIs: vvesternunion.biz]
 0.1 TO_CC_NONE             No To: or Cc: header
```

Fig. 2.2 Virus spa.

blocked (or delivered) by one of them. SpamAssassin is a prototypical example in which many filters and tests are instead applied in parallel, and the results combined using a formula; the result of applying SpamAssassin to the message in Figure 2.1 is shown in Figure 2.2. More systematic methods of *stacking* autonomous filters using machine learning techniques [114, 154] are discussed in the next section.

3

Spam Classifiers — Machine-Learning

A classifier constructor C takes extrinsic information e as input and generates a classifier c as output. The methods reviewed in the previous section rely on hand-crafted implementations of C. The methods reviewed in the present section seek to replace much of the hand crafting by automatic machine learning methods. It is impossible to eliminate the hand-crafted component altogether, as in general e is a member of a huge amorphous universe E. e may be characterized as *"any data drawn from all possible sources."* E is therefore unsuitable as the domain for an automated constructor.

Instead we define a learning-based constructor to have two components: an abstractor $A : E \to D$ and a learner $L : D \to (M \to \{true, false\})$ or $L : D \to (M \to \mathbb{R})$ such that $C = L \cdot A$. We refer to D, the range of A and domain of L, as the *learner domain* D, and a concrete representation for each $d \in D$, are defined formally to facilitate the implementation of L by some machine learning algorithm. The hand-crafted definition of D and implementation of A are essential aspects of any machine-learning classifier; only L is automated.

The problem of designing an amenable learner domain is open-ended. Most designs can be characterized as having one of several

established *learning modes* and some particular *feature representation.* We consider the commonest examples in the following subsections, followed by specific techniques for abstraction and learning.

3.1 Learning Modes

3.1.1 Supervised Learning

Supervised learning [102] is a common mode for machine-learning classifiers. The learner's input (T, $label$) consists of a set $T \subseteq M$ of *training examples* and a function $label : T \to \{true,\ false\}$ that approximates $isspam$ over the sub-domain T. $label$ is typically hand-crafted using a human classifier. Under the assumptions that T is an independent and identically distributed (i.i.d.) sample of M and that $label(m) = isspam(m)$ for all $m \in T$, the learner induces the parameters of a model for $m \in spam$ and uses this model to construct c that optimizes some utility function, typically *accuracy*.

Supervised learning — along with its associated assumptions — is so common that it is often assumed without question. But it can be exceedingly difficult to obtain a sample — especially an i.i.d. sample — of messages to be classified. Indeed, many members of M (the very ones we are interested in classifying) exist only in the future, and are therefore simply impossible to sample. Constructing $label$ is sufficiently onerous and error-prone that the assumption of its existence is questionable. One should not assume that optimizing accuracy yields the classifier most suitable for its intended purpose: detecting spam.

3.1.2 Semi-Supervised Learning

Semi-supervised learning [204] assumes input (T, S, $label$), where $T \subseteq M$, $S \subset T$ and $label : S \to \{true,\ false\}$, that is, $label$ is defined for only a subset of the training examples. Semi-supervised learning accommodates the fact that obtaining sample messages may be considerably easier than labeling them, as is often the case with email spam. Semi-supervised learning assumes, like supervised learning, that T is an i.i.d. sample of M. This assumption may allow the learner to learn more about the distribution of M from the unlabeled examples

in $T \backslash S$. A vacuous semi-supervised learner is simply the supervised learner with input (S, *label*), this learning provides a convenient baseline against which semi-supervised learners may be compared.

3.1.3 Transductive Learning

Transductive learning [91] is a special case of semi-supervised learning with input (T, S, *label*) as before. The difference is that the output is a partial classifier $c : T \to \{true, \ false\}$ or $c : T \to \mathbb{R}$ defined only on the sub-domain T. Transductive learning methods may be used when classifier construction is implicit: the messages to be classified are known at the time of classifier construction, and the classifier is reconstructed for future messages.

3.1.4 Unsupervised Learning

Unsupervised learning [65] assumes no *label* function at all, that is, the input is simply a set $T \subseteq M$, and is rarely used directly to construct a classifier. Unsupervised learning methods may nevertheless be used in conjunction with others; for example, *clustering* methods may be used to find groups of similar messages under the assumption that each member of a group belongs to the same class.

3.1.5 Active Learning

Active learning [147] allows the classifier to request labels for some subset of the unlabeled training data. A spam filter may, for example, ask the user to mark several messages each as ham or spam, and classify the rest based on these examples. The prototypical method for active learning is *uncertainty sampling* [107], in which a soft classifier is applied to each unlabeled example, and labels are requested for those whose classifier result is closest to the threshold t.

3.1.6 On-line Learning

The modes discussed so far assume that M is a static set, that T is a sample of that set, and that (except for transductive learning) a classifier is to be constructed whose domain is all of M. On-line learning [36]

assumes that the messages in M are ordered chronologically, or, more generally, by some relation $<$. The semi-supervised on-line learner's input is $(T, <, m', S, label)$, where $T \subseteq \{m | m \leq m'\}$ for some $m' \in M$, $S \subseteq T$, and $label : S \to \{true, false\}$. The supervised learner assumes $S = T$. The domain of the constructed classifier $c : U \to \{true, false\}$ or $c : U \to \mathbb{R}$ is the set of later messages $U = \{m | m > m'\}$.

An on-line learner may be constructed trivially from the corresponding supervised or semi-supervised batch learner by ignoring the ordering relation, however,

- The training examples are certainly not an i.i.d. sample of the examples to be classified, as they are separated by the ordering. Therefore the assumptions of the batch learner are not met.
- Typical deployment of an on-line learner involves classifying a sequence of messages $m_1 < m_2 < \cdots < m_n$, each in turn. The maximum information is available to the learner if a new classifier c_i is defined for every $m_i (1 \leq i \leq n)$ with $T = \{m_{j<i}\}$. If the construction of c_i examines every training example in T_i, the time required will be at least proportional to i, that is, the time to classify the sequence of messages will be quadratic in n, effectively precluding on-line deployment.

Incremental learning may be used to reduce the overall cost of classifying a sequence. An incremental learner efficiently constructs c_{i+1} from the data structures representing c_i and m_i without necessarily examining all of the examples in T_i. The amenability of the learner to efficient incremental construction is an important criterion in the choice of a method for on-line filtering.

Incremental learning may be approximated using a non-incremental learner, batching, and a sliding window. Batching is predicated on the assumption that $c_{i+\Delta}$ may be approximated by c_i provided Δ is not large. Thus a new c_i is constructed whenever $i \equiv 0 \, (mod \, \Delta)$, otherwise $c_i = c_{\Delta \lfloor \frac{i}{\Delta} \rfloor}$, for some fixed not-too-large Δ. The overall effect is to improve learner efficiency by a factor of Δ. A sliding window considers only the most recent ω messages as the training set, that is,

$T_i = \{m_j | i - \omega \leq j < i\}$. The use of a sliding window renders overall learning time proportional to n.

Training example selection may yield better results than a sliding window. In general, we may select any $T_i \subseteq \{m_{j<i}\}$ such that $|T_i| \leq \omega$, and provided the selection is done without consulting any $m_k \notin T_i$. The sliding window is a special case of training example selection.

3.2 Feature Engineering

To construct a classifier $c : M \rightarrow \{true, false\}$ or $c : M \rightarrow \mathbb{R}$ it is necessary to define a concrete representation for every $m \in M$. This representation may be, but is typically not, simply the textual representation of the message. In general, a message is represented as a collection of *features* derived from the message, or from extrinsic information related to the message. The process of defining and extracting features likely to be useful to the classifier — feature engineering — has a profound impact on overall spam filter effectiveness. The reader should be skeptical regarding claims (positive or negative) for any particular learning method that fails to note the method of feature representation or the mode of learning.

The classical methods of feature engineering for spam filtering are largely derived from established methods for information retrieval and supervised machine learning. We present them first due to their historical weight, notwithstanding recent results suggesting the superiority of simpler methods for spam filtering.

3.2.1 Feature Vectors

A message m is typically represented a vector of n features $x = (x_1 x_2 \cdots x_n)$, where each x_i is a number or categorical value quantifying some evidence pertaining to the message that might be useful to the classifier. A feature extractor $Z : M \rightarrow X$ computes for a given message $m \in M$ the corresponding feature vector $x \in X$, where X is an n-dimensional space.

A simple and common example is the *bag of words* model. The distinct words contained in the training set are enumerated, assigning to each a unique dimension in X. For each message m, we might define

$Z(m)_i$ to be the number of times that the ith word occurs in m (the term-frequency model, cf. [53]). Or, we might divide that number by some function that increases with the total number of documents that contain the word (the tf-idf model, cf. [53]).

Some learners perform poorly, either in terms of efficiency or classification performance, for large n. *Feature selection* is commonly used to reduce n and hence the dimensionality of the feature space X. More generally, *dimensionality reduction* techniques may be used to transform and or project X to a space of smaller dimension.

Stop word elimination is a simple example of feature selection in the bag-of-words model, common words are simply not enumerated. Many statistical techniques have been proposed and used to identify the most *important* features, the rest are eliminated. *Stemming* is a simple example of dimensionality reduction: all morphological variants of the same word are mapped to the same dimension. More sophisticated methods, such as latent semantic indexing [63, 64], transform the entire space to one of smaller dimension.

Some feature engineering choices may interfere with incremental classifier construction, and hence efficient on-line deployment. The set of features or the set of values for a particular feature may grow as new messages are learned, amenable algorithms and data structures must be chosen. Global statistics like tf-idf must be recomputed when used, as the addition of a single document changes every tf-idf term.

Most importantly, feature selection or dimensionality reduction is problematic in an on-line environment. If the method uses statistics over the training set, these statistics will change with every new message, possibly occasioning the complete reconstruction of the classifier.

3.2.2 Tokenization

The bag of words model arose from early information retrieval research indicating that it worked as well for full text retrieval as more complex models that take, for example, word position, into account. Much of the spam filtering literature simply assumes this model.

Many variants of this model exist, depending on what one considers to constitute a *word*. In English text one may reasonably identify

words as strings of consecutive letters separated by punctuation or spaces. Some minor details, such as whether words may contain apostrophes or digits, have been found to be of little consequence. Much attention has been paid to the use of stemming, which seeks to treat all morphological variants of a common root word as the same. Stemming methods are necessarily approximate and heavily language dependent, and have shown indeterminate results in information retrieval and filtering. Similarly, the consideration of case — whether or upper case and lower case representations of the same letter are considered distinct — has shown indeterminate results. The overall consensus appears to be that these variants — and also those that identify synonyms — do not matter much [53].

The common assumption that email consists of English text, or even Roman text, is difficult to justify. Email is an international medium, containing messages in a vast number of languages using non-Roman alphabets, many of which have, like Chinese, no natural concept of "words." The bag-of-words model demands that we identify the language and devise language-specific tools to recognize words and word variants. Even an email message composed in English is not strictly English text. Email includes structured information which is not "English text" such as the header fields and MIME meta data. The character set may be encoded in many different ways. The text itself may be embedded in documents with a variety of formats, such as plain text, html, word-processor format, pdf, images, and so on. A substantial part of the message may involve non-textual media such as photographs, audio, and video. It is not obvious how to map these diverse data formats to the bag of words model, the alternatives range from complex format-specific interpretation to simply ignoring them.

Spammers have employed many obfuscation methods in an attempt to defeat spam filter tokenization [75, 81, 131] which in turn have given rise to deobfuscation techniques that aim to recover the original tokens [104, 105].

Recently, spammers have used distorted and noisy images to convey text messages, much attention has been paid to the specific problem of extracting features from these images [8, 14, 22, 52, 62, 186],

however, the overall impact of image-specific techniques has yet to be established [74].

3.2.3 Synthetic Words

The bag-of-words model may be extended, by the introduction of synthetic words, to capture additional information beyond which words appear somewhere in the message. Synthetic words are commonly denoted using a representation that would not actually be recognized as a word in the text, for example, by including a punctuation symbol. An effective use of synthetic words is to indicate the particular message field in which a word occurs [69]. `subject:money` might indicate the occurrence of the word "money" in the subject field of the message, whereas `body:money` would indicate its presence in the text of the message, and `money` would indicate its appearance anywhere in the message.

Virtually any feature may be inserted into the bag of words model by the use of synthetic words. The results of applying SpamAssassin's entire rule-set might be represented by 800 synthetic words. At some point, however, the use of synthetic words departs from the assumptions implied by the "bag of words" label, and loses the essence of the bag-of-words model. We have, for example, used Bogofilter — a bag-of-words based filter — to classify arbitrary feature vectors by rendering the vectors as a sequence of nonsense words, with each distinct word representing a particular feature [36].

3.2.4 Word Bigrams and Trigrams

Names and idioms often consist of several juxtaposed words. For example, the term "information retrieval" conveys a much more specific meaning than the words "information" and "retrieval" taken separately. Bigrams — pairs of consecutive words taken together as individual features — reflect the difference in meaning when words are taken together. A text is split into bigrams by taking every pair of adjacent words to form a synthetic word. For example, the text `to be or not to be` contains five bigrams: `to+be, be+or, or+not, not+to, to+be`. While the bigram model

captures information that is not captured by the bag-of-words model, it involves quadratically more features: each word may potentially appear in a bigram with each other word. This large number of features may be undesirable for two reasons: first, a learning methods may simply be unable to cope with such a large number; second, particular features may occur so rarely that they fail to provide meaningful input to the filter. For this reason, bigrams may be used in in addition to simple words (unigrams), and feature selection may be used to eliminate rare bigrams.

Trigrams (three adjacent words) and N-grams (N adjacent words) for $N > 3$ generate an exponential number of features, and appear to yield no substantive improvement over bigrams, even in identifying phrases consisting of more than two words. It turns out that messages containing "Hubble space" and "space telescope" but not "Hubble space telescope" are exceedingly rare, so the feature `hubble+space+telescope` contributes little.

3.2.5 Sparse Bigrams

Interactions among nearby (but not adjacent) words may be captured as sparse bigrams. A sparse bigram is simply a pair of words separated by no more than k words. $k = 0$ yields ordinary bigrams, while $k = 3$ is typical. A text with n words will generate $n(k + 1)$ sparse bigrams, and the interaction among words decreases rapidly with separation. For $k = 2$, the text `to be or not to be` generates the sparse bigrams `to+be`, `to+or`, `to+not`, `be+or`, `be+not`, `be+to`, `or+not`, `or+to`, `or+be`, `not+to`, `not+be`, `to+be`.

Orthogonal sparse bigrams (OSB) [158] further classify sparse bigrams by the number of intervening words, each of which we denote by "?": `to be or not to be` generates the sparse bigrams `to+be`, `to+?+or`, `to+?+?+not`, `be+or`, `be+?+not`, `be+?+?+to`, `or+not`, `or+?+to`, `or+?+?+be`, `not+to`, `not+?+be`, `to+be`. OSB derives from and simplifies earlier work on sparse binary polynomial hashing (SBPH), (cf. [158]) which treats as features all sub-sequences of length 2 or more contained in an interval of $k + 2$ words.

3.2.6 Character *N*-Grams

N-gram techniques may be applied to characters instead of words. Our running example could, for example, be represented as the character trigrams `to_`, `o_b`, `_be`, `be_`, `e_o`, and so on, where `_` represents a space in the input. Because the number of distinct characters is much smaller than the number of distinct words, larger values of N are useful and practical; $N = 3$ and $N = 4$ are typical. Although character n-grams make fewer linguistic assumptions than their word counterparts, evidence suggests that they capture inter-word as well as intra-word interactions, and are reasonably robust to spelling errors and morphological variants.

3.2.7 Meta Features

Meta features represent the results of other filters — perhaps based on other machine learning algorithms — applied to the message. Simple meta features might include message length, the proportion of upper case letters, the number of addressees, the results of *ad hoc* rules or other filtering techniques, and so on. The choice of information to represent is limited only by our imagination and the learning method's ability to harness it.

3.2.8 Feature Selection

Feature engineering techniques yield a potentially large number of features, a number which may be unwieldy or compromise the efficiency or efficacy of evidence combination. The bag-of-words approach yields one feature per word that occurs in any message, overall an unbounded number that commonly exceeds 10,000 and may approach 1 million over a large set of messages. Bigram and n-gram models potentially yield quadratically or exponentially more. The process of feature selection identifies those features likely to yield the most evidence to the classifier, discarding the rest.

It should be noted that the vectors representing these features are sparse, with the net effect that the overall number of non-vacuous elements $|\{E(m_i)_j \neq 0\}|$ representing any set of messages is proportional

to the overall size of the messages in the set. So long as the algorithms and data structures consume time and space nearly proportional to this size, the full feature set may be used. These time and space constraints are likely occasioned in any event by the application-specific requirements outlined in the introduction to this section. This observation has three consequences:

(1) Feature selection may not be justified by efficiency considerations.
(2) Application constraints may preclude selection methods with super-linear time and space requirements.
(3) Methods that require statistics over an entire set of messages, such as frequency counts, are not adaptive and may preclude on-line learning.

At the time of writing it is not obvious that any feature selection method is desirable within the context of spam filtering, or that it is meaningful to study feature selection separately from classifier construction (cf. [54]). This view appears to be unique to the spam filtering domain, stop-word and rare word elimination are standard approaches in IR, while statistical feature selection methods are the subject of much attention in general text classification research (cf. [153]).

In abstract terms, the feature selection problem may be characterized as that of finding the best subset of n features, for some definition of *best*. Primary considerations in defining *best* include the number $n' < n$ of features, the effectiveness of the classifier $c(\mathrm{x}')$, where x' is the feature vector with dimensions corresponding only to the selected features, and the tractability of an algorithm to effect the selection. If *best* is characterized by a formula or test outcome, a straightforward but prohibitively inefficient approach is to examine all 2^n subsets and pick the best. More commonly, greedy heuristics are applied that identify features in decreasing order of some characterization of their value, stopping at some suitable value of n'. Sebastiani [153] details several heuristics relying on statistics like term frequency and information gain. Regardless of the particular statistic, selecting the best over

a set is a challenge when the set is dynamic, as in adaptive and on-line filtering. Trivial approaches to feature selection, such as stop-word elimination, rely on fixed rules and are hence easy to apply in an on-line situation.

3.3 Probabilistic Classifiers

A probabilistic classifier computes an estimate $\text{Prob}(m \in spam|Z(m) = x)$ of the probability that a given message m, represented by x, is spam. This estimate may be used to define a soft classifier $c(m) = \text{Prob}(m \in spam|Z(m) = x)$ or a hard classifier $c(m) = \text{Prob}(m \in spam|Z(m) = x) > t$ for some fixed threshold $0 < t < 1$.

$Z(m)$ is typically a vector of features $x = x_1 x_2 \cdots x_n$. For each x_i, a separate estimate p_i may be computed for the probability that m is spam, considering only x_i as evidence. These estimates are then combined into an overall estimate that best reflects the evidence afforded by x as a whole.

When estimating and combining probabilities, it is often convenient to recast them as odds or log-odds:

$$\begin{aligned} Odds(z) &= \frac{\text{Prob}(z)}{1 - \text{Prob}(z)} \\ \text{Prob}(z) &= \frac{1}{1 + Odds(z)} \\ LogOdds(z) &= logit(\text{Prob}(z)) \\ logit(p) &= log\left(\frac{p}{1-p}\right) \\ logit^{-1}(q) &= \frac{1}{1 + e^{-q}}. \end{aligned}$$

In the following discussion, we abbreviate $Z(m)$ as $x^{[m]}$ (or simply x when m is understood) and $Z(m_i)$ as $x^{[i]}$.

3.3.1 Probability Estimates from Categorical Features

While the values of a binary feature $x_i : \{0,1\}$, have no intrinsic meaning, $x_i = 1$ typically represents the presence of some lexical feature (e.g.,

a word) in m while $x_i = 0$ indicates its absence. The probability that m is spam given $x_i = k$ is easily estimated as the fraction of messages with $x_i = k$ in the training set T:

$$\text{Prob}(m \in spam | x_i = k) = \frac{|\{m \in T | x_i^{[m]} = k\} \cap spam|}{|\{m \in T | x_i^{[m]} = k\}|},$$

it is convenient to recapitulate this estimate in terms of odds:

$$Odds(m \in spam | x_i = k) = \frac{|\{m \in T | x_i^{[m]} = k\} \cap spam|}{|\{m \in T | x_i^{[m]} = k\} \cap non\text{-}spam|}.$$

For example, if the word "money" occurs in 100 spam messages and 5 non-spam messages, the odds that a particular message m containing "money" is spam may be estimated to be $Odds(m \in spam | x_i = 1) = \frac{100}{5} = \frac{20}{1}$. The same estimate, expressed as a probability is $\text{Prob}(m \in spam | x_i = 1) = \frac{20}{1+20} = 0.952$. The value $k = 1$ is not particularly special, assuming T consists of, say, 1000 spam and 1000 non-spam messages, we may deduce that 900 spam messages and 995 non-spam messages have the $x_i = 0$, so the odds of message not containing "money" being spam are $Odds(m \in spam | x_i = 0) = \frac{900}{995} = \frac{0.9}{1}$, i.e., nearly even odds. Intuitively, the non-occurrence of "money" contributes little to solving the filtering problem, for this reason spam filters often ignore this information.

The ratio $\frac{a}{b}$ of the number of spam to non-spam messages a good odds estimate if these numbers, a and b, are sufficiently large. If they are small, the estimates will be unreliable due to chance, and if either or both is zero, the resulting odds are $\frac{0}{1}$, $\frac{1}{0}$ or $\frac{0}{0}$, none of which is a sensible estimate. A simple approach to mitigate this problem is to add small positive constants α and β to the numerator and denominator, respectively, that is, to use $\frac{a+\alpha}{b+\beta}$ as the odds estimate. In the case of $a = b = 0$ this yields an estimate of $\frac{\alpha}{\beta}$ while for large a and b being indistinguishable from $\frac{a}{b}$. Typically, $\alpha = \beta = 1$, this choice will be revisited when we consider the combination of estimates from various features.

3.3.2 Probability Estimates from Continuous Features

A direct way to estimate spam probability from a real-valued feature $x_i : \mathbb{R}$ is to transform its value to a binary value $b_i : \{0, 1\}$ by comparing to a threshold

$$b_i = \begin{cases} 1 & x_i > t \\ 0 & x_i \leq t \end{cases}$$

and to estimate $Odds(m \in spam | b_i = k)$ as described in the previous section. While a real value, like a discrete value, has no intrinsic meaning, features may be engineered so that a larger value of x_i indicates higher odds that $m \in spam$. In other words, that x_i is itself a soft classifier. And b_i is the corresponding hard classifier. An n-ary categorical value $d_i : \{0, 1, \ldots, n-1\}$ may be computed using n *bins* delimited by $n - 1$ threshold values

$$d_i = \begin{cases} n-1 & t_{n-2} < x_i \\ \cdots & \cdots \\ 1 & t_0 < x_i \leq t_1 \\ 0 & x_i \leq t_0 \end{cases},$$

thus effecting a piecewise approximation of the odds implied by the continuous value x_i.

The principal drawback to this approach is that as n increases, the number of messages with $d_i = k$ for any particular k decreases, and the odds estimates become less reliable. An alternative approach is to define a transformation $f : \mathbb{R} \to [0, 1]$ such that $\text{Prob}(m \in spam | x_i = k) \approx f(x_i)$. If x_i is, in effect, the result of a probabilistic classifier, f is trivial.

Parametric models may be used instead of simple counting to model the distributions of $x_i^{[m \in spam]}$ and $x_i^{[m \in non\text{-}spam]}$. For example, if a Gaussian distribution is assumed, the four parameters $\mu_{i,s}$, $\sigma_{i,s}$, $\mu_{i,n}$, $\sigma_{i,n}$, the mean and standard deviation for $x_i^{[m \in spam]}$ for $m \in spam$ and $m \notin spam$, respectively, fully characterize the distributions. Given these parameters,

$$\text{Odds}(m \in spam | x_i = k) \approx \frac{N_s \cdot g(\mu_{i,s}, \sigma_{i,s}, k)}{N_n \cdot g(\mu_{i,n}, \sigma_{i,n}, k)},$$

where N_s and N_n are the number of spam and non-spam in T, and g is the instantaneous Gaussian (normal) distribution at k [92].

An alternative approach [114] assumes that x_i is the result of a soft classifier and uses the tail of the cumulative empirical distribution to estimate

$$Odds(m \in spam | x_i = k) = \frac{|\{m | x^{[m]} \leq k\}|}{|\{m | x_i^{[m]} \geq k\}|}.$$

3.3.3 An Example

We illustrate the processes using two features derived from the TREC 2005 Public Spam Corpus [40]. Our features are chosen to harness the knowledge that all the messages in the corpus were delivered to individuals at one particular organization and that the organization's name — Enron in this instance — might have different prevalence in spam and non-spam messages. Our two features simply count the number of occurrences of the character sequence `enron` in the header and in the body, respectively, after converting all letters to lower case. Table 3.1

Table 3.1 Example from TREC 2005 Corpus.

Message tag	true class	`head:enron`	`body:enron`
016/201	spam	12	0
033/101	spam	11	0
050/001	spam	10	0
066/186	non-spam	7	24
083/101	non-spam	21	0
083/101	non-spam	21	0
100/001	non-spam	27	4
133/101	spam	12	17
148/013	non-spam	22	5
166/201	non-spam	13	23
183/101	spam	11	0
200/001	spam	14	4
216/201	non-spam	25	2
233/101	spam	13	20
250/001	non-spam	5	0
266/201	spam	12	0
283/101	spam	13	0
300/001	spam	11	22

presents these attributes for 18 messages selected from the corpus, 10 of which are spam and 8 of which are non-spam.

As discrete values indicating the presence of `enron` in the respective message components, these features are of limited use. `enron` occurs in *every* header, and therefore its presence yields no information beyond the ratio of spam to non-spam in the sample. `enron` occurs in the bodies of 4 spam and 5 non-spam messages yielding a 4:5 estimate of the odds that a message whose body contains `enron` is spam. Similarly it does not occur in the bodies of 6 spam and 3 non-spam message, yielding a 2:1 odds estimate for such messages. These estimates are compared with our best estimate of the *true* values — the gold standard computed over the entire corpus — in Table 3.2.

Table 3.3 shows the result of splitting the values of `head:enron` into three discrete ranges: [0–9], [10–19], [20–30], and the effect of two choices of α and β. Values in the center range clearly predict spam, while extreme values predict non-spam. Table 3.4 shows the predictions made for each possible value of `head:enron` assuming a Gaussian distribution with parameters computed from the sample: $\mu_s = 11.9$, $\sigma_s = 1.2$, $\mu_n = 17.6$, $\sigma_n = 8.3$. The model aptly estimates $\text{Prob}(m \in spam | head : enron = k)$ for small values of k, but dramatically underestimates it for larger values. These larger values are fairly rare, mitigating the effect of the underestimate, and even the underestimates will yield a correct

Table 3.2 Sample vs. gold standard spam estimates.

	Training		Gold standard	
Feature	Freq.	Prob	Freq.	Prob
`head:enron` > 0,	1	0.56	0.9999	0.57
`head:enron` $= 0$,	0	0.5	0.0001	0
`body:enron` > 0,	0.5	0.44	0.62	0.45
`body:enron` $= 0$,	0.5	0.67	0.38	0.77
No feature (all messages)	1	0.56	1	0.57

Table 3.3 Discrete range feature estimates.

	Training			Gold standard	
Feature	Freq.	$\alpha = \beta \to 0$	$\alpha = \beta = 1$	Freq.	Prob
$0 \leq$ `head:enron` < 10	0.11	0	0.25	0.18	0.05
$10 \leq$ `head:enron` < 20	0.61	0.91	0.85	0.74	0.75
$20 \leq$ `head:enron` < 30	0.28	0	0.14	0.05	0.19

Table 3.4 Sample vs. gold standard spam estimates, Gaussian model.

	Training		Gold standard	
`head:enron`	Freq.	Prob	Freq.	Prob
5	0.06	0.0000	0.00	0.0000
6	0	0.0000	0.01	0.0705
7	0.06	0.0017	0.08	0.0000
8	0	0.0311	0.05	0.0409
9	0	0.2315	0.03	0.1767
10	0.06	0.5880	0.07	0.6191
11	0.17	0.7735	0.28	0.8366
12	0.17	0.8049	0.19	0.7343
13	0.17	0.7158	0.09	0.7838
14	0.06	0.4371	0.04	0.7269
15	0	0.1079	0.02	0.6321
16	0	0.0094	0.01	0.4687
17	0	0.0004	0.01	0.4162
18	0	0.0000	0.01	0.4838
19	0	0.0000	0.01	0.3539
20	0	0.0000	0.01	0.5745
21	0.11	0.0000	0.01	0.4236
22	0.06	0.0000	0.01	0.4008
23	0	0.0000	0.00	0.5281
24	0	0.0000	0.00	0.1026
25	0.06	0.0000	0.02	0.0114
26	0	0.0000	0.00	0.0629
27	0.06	0.0000	0.00	0.0026

classification more often than not. Nevertheless, there is plenty of room to improve the model.

3.3.4 Combining Probability Estimates

We consider the problem of constructing the estimate $\text{Prob}(m \in spam|x = k)$ given $p_i = \text{Prob}(m \in spam|x_i = k_i)$ for all $1 \leq i \leq n$, where $k = (k_0 k_1 \cdots k_n)$. We consider the special cases $n = 0$, $n = 1$, and $n = 2$, and their generalization to $n > 2$.

$n = 0$ denotes the empty vector, so the estimate, which we denote $p_{()}$, reduces to $\text{Prob}(m \in spam|x = ())$; that is, $\text{Prob}(m \in spam)$ since $x = ()$ is a tautology. x may be considered a discrete feature with only one possible value, the empty vector. Using the method described in Section 3.3.1, we compute $Odds(m \in spam|x = ()) = \frac{|T \cap spam|}{|T \cap non\text{-}spam|}$, the ratio of spam to non-spam in the training set.

$n = 1$ is also trivial; we have $\text{Prob}(m \in spam | x = (k_1)) = \text{Prob}(m \in spam | x_0 = k_0) = p_1$.

$n = 2$ is more problematic; there is no general method of combining p_1 and p_2 into a common estimate without considering the evidence x_1 and x_2 and T from which they are derived. Under the assumption that x_0 and x_1 occur independently in both spam and non-spam, we may combine $p_{()}$, p_1 and p_2 using Bayes rule to compute a combined estimate p, thus forming a *naive Bayes* classifier, which is conveniently expressed using the logistic transform:

$$\begin{aligned} logit(p) &= logit(p_1) + logit(p_2) - logit(p_0), \\ i.e.\ logOdds(m \in spam | x) &= logOdds(m \in spam | x_0) \\ &\quad + logOdds(m \in spam | x_1) \\ &\quad - logOdds(m \in spam). \end{aligned}$$

The naive Bayes assumption seldom holds in practice, the word "sildenafil," for example, is far more likely to be found in email messages — spam and non-spam alike — that also contain the word "Viagra." Invalid assumptions aside, naive Bayes classifiers are commonly used because they are simple and perform adequately as hard classifiers with a threshold $t = 0.5$, even if their probability estimates are far from accurate [51].

A contrasting assumption is that x_1 and x_2 are dependent, for example, that the presence of "sildenafil" or "Viagra" are both indicators of spam, but whether one, or the other, or both occur in a particular message is of no consequence. In short, a message containing "sildenafil" *and* "Viagra" is no more or less likely to be spam than a message containing either term alone. Under this assumption p_1 and p_2 may be averaged, as they are both estimates of the same quantity and differ only in estimation error. Arguably the most apt form of average is a weighted average under the logistic transformation: $logit(p) = \beta_1 logit(p_1) + \beta_2 logit(p_2)$ for some $\beta_1 + \beta_2 = 1$, chosen to reflect the relative estimation error inherent in p_1 and p_2. Absent such an estimate, the choice $\beta_1 = \beta_2 = \frac{1}{2}$ yields an unweighted average, reflecting the (reasonably robust) assumption that the estimation errors are equal.

Table 3.5 Combining estimates.

	Prob		
Feature combination	Logit avg.	Naive Bayes	Gold std.
$0 \leq$ `head:enron` < 10 `body:enron` $= 0$	0.45	0.36	0.14
$0 \leq$ `head:enron` < 10 `body:enron` > 0	0.33	0.16	0.03
$10 \leq$ `head:enron` < 20 `body:enron` $= 0$	0.77	0.90	0.86
$10 \leq$ `head:enron` < 20 `body:enron` > 0	0.66	0.76	0.65
$20 \leq$ `head:enron` < 30 `body:enron` $= 0$	0.36	0.21	0.40
$20 \leq$ `head:enron` < 30 `body:enron` > 0	0.25	0.08	0.12

Table 3.5 compares the two methods using all combinations of values for the two discrete features in our running example. We see that averaging tends to yield conservative results, closer to $p_{()} = 0.55$, while those due to naive Bayes are more extreme. For some examples, averaging appears to yield the better estimate, for others, naive Bayes.

Reality[1] lies somewhere between these two assumptions. A common model — the logistic model — subsumes both:

$$logit(p) = \beta_0 logit(p_{()}) + \beta_1 logit(p_1) + \beta_2 logit(p_2).$$

The naive Bayes assumption is modeled by $\beta_0 = -1, \beta_1 = \beta_2 = 1$, the dependent evidence assumption by $\beta_0 = 0, \beta_1 + \beta_2 = 1$. More generally, one may choose $\beta_0, \beta_1, \beta_2$ so as to optimize some utility or cost measure for the resulting classifier. The most common utility measure is the *likelihood* of the training data, which may be optimized using logistic regression. This method of deriving a classifier is known as *maximum entropy* or *logistic regression*.

Logistic regression generalizes to any $n \geq 0$:

$$logOdds(m \in spam|x) = \beta_0 logit(p_{()}) + \sum_{i=1}^{n} \beta_i logit(p_i).$$

[1] Or, at least, a closer approximation of reality.

The naive Bayes assumption is modeled by $\beta_0 = 1 - n$, $\beta_i = 1 (1 \leq i \leq n)$ while the dependent evidence assumption is modeled by $\beta_0 = 0$, $\sum_{i=1}^{n} \beta_i = 1$.

It is unnecessary to transform discrete-valued features to probability estimates for the purpose of logistic regression. Instead, a feature $x_i : \{k_1, k_2, \ldots, k_l\}$ may be interpreted as l distinct binary features

$$x_{i,1}, x_{i,2}, \ldots, x_{i,l}, \quad \text{where } x_{i,j} = \begin{cases} 1 & (x_i = k_j) \\ 0 & (x_i \neq k_j) \end{cases}.$$

In this case, each coefficient $\beta_{i,j} = \beta_i \text{Prob}(m \in spam | x_i = k_j)$ in the model. Commonly, x_i is binary-valued and $x_{i,0}$ is discarded for the reasons stated in Section 3.3.1, so x_i is effectively replaced by $x_{i,1}$.

3.3.5 Practical Considerations

The choice of feature representation and combining method may have a dramatic effect on the simplicity and efficiency of the resulting spam filter, particularly in on-line deployment. We have previously mentioned that feature transformations such as tf-idf and statistical feature selection are difficult to reconcile with adaptive classifiers, probability-based interpretations that model global distributions entail similar difficulties. For this reason, discrete features that may be derived from individual messages, independent of others in the training set, are more amenable to on-line settings. In general, the literature shows that these simple feature representations work for spam filtering as well as, or better than, more complex and less adaptive ones based on global statistics.

Naive Bayes filters for discrete-valued features are easily implemented. Each message is converted to its feature representation, and a dictionary is used to count the number of spam and non-spam messages containing each feature value. A message may be classified by retrieving the counts for the feature values it contains and applying the formula to yield a hard or soft classification. New messages may be added to the training set by updating the relevant counts. Previously unseen features or feature values pose no problem to an on-line naive Bayes filter; they may simply be inserted into the dictionary, with associated counts of 0.

Self-training — a simple but effective method of semi-supervised learning — may be effected by classifying a message, and then adding the message and inferred classification to the training data. Assuming the filter is accurate, it will be much more likely to self-train correctly than not, generally increasing the accuracy of the filter. But there is also the risk that incorrect self-training, albeit rare, might cause the filter to go astray, thus "blessing" a particular kind of spam as non-spam, or vice versa. Should the user report a misclassification error to the filter, the effect of incorrect self-training is easily reversed by subtracting from the relevant counts.

Provided the coefficients are fixed, a filter using a logistic model is as easy to implement and update as a naive Bayes filter. Many filters labeled "Bayesian" are in fact not Bayesian at all, but derived from a technique dubbed χ^2 by Robinson [136] which combines estimates using geometric means of separate (but not necessarily independent) estimates for Prob($m \in$ *spam*) and Prob($m \in$ *non-spam*) — equivalent to the logit average of the presented here.

Logistic regression is traditionally viewed as a batch algorithm. Indeed, computing the β_i so as to maximize the likelihood of the training examples requires that the entire training set be examined. Logistic regression is a standard tool for statistical analysis [87], and sophisticated implementations are found in all major statistical software packages like SPSS, SAS, Stata, S and R, as well as the data mining toolkit Weka [192]. Stand-alone versions are available as well, notably LR-TRIRLS [100] which was designed specifically for building classifiers for large feature spaces. It is possible to use these tools in conjunction with batching and a sliding window, but effectiveness is compromised [36].

Goodman and Yih [68] describe a very simple iterative descent method for performing logistic regression, paraphrased in Figure 3.1. While this method converges more slowly than the sophisticated methods mentioned above, its simplicity makes it particularly attractive for both batch and on-line filtering. In the case of batch filtering, the algorithm may be applied to huge datasets (the author has found it effective on a dataset with 100 million 5-valued features). In the case of on-line filtering, it is attractive because the optimal solution for training set T is very nearly optimal when T is augmented by one message:

Input:
Set $T : M$ of training examples
$m \in T$ represented by $x^{[m]} = (1\, x_1^{[m]} x_2^{[m]} \ldots x_n^{[m]})$
Labeling $label : T \to \{0, 1\}$
Rate parameter δ
Output:
$\beta \cdot x^{[m]}$ is the maximum likelihood estimate of
$Prob(label(m) = 1)$ over $m \in T$
Method:
$\beta \leftarrow (0 \cdots 0)$
repeat until convergence:
for $m \in T$
let $p = \frac{1}{1+e^{-\beta \cdot x^{[m]}}}$
$\beta \leftarrow \beta + (label(m) - p) \cdot \delta \cdot x^{[m]}$

Fig. 3.1 Gradient descent logistic regression.

$T \leftarrow T \cup \{m\}$. For practical purposes, note Goodman and Yih, it suffices to apply the gradient descent step only to the new message m. Under this assumption it is unnecessary to retain the messages in T, the only persistent state required by the classifier is the coefficient vector $(\beta_0 \beta_1 \cdots \beta_n)$.

Self-training is easily effected using gradient descent, however, the effects of incorrect self-training are more difficult to undo. Training the classifier again with the same message and the corrected label is the only efficient correction method of which we are aware.

3.4 Linear Classifiers

A linear classifier views the feature vector $x^{[m]}$ for a message m as a point in n-dimensional space, where n is the number of features. The classifier consists of a vector of coefficients $\beta = (\beta_1 \beta_2 \cdots \beta_n)$ and a threshold t. The equation $\beta \cdot x = t$ defines a hyperplane that divides the space into half-spaces. All points on one side of the hyperplane $(\beta \cdot x^{[m]} > t)$ are classified as spam while the ones on the other side $(\beta \cdot x^{[m]} \leq t)$ are classified as non-spam. $\beta \cdot x = t$ is a *separating*

hyperplane if $\forall_{m \in spam} \beta \cdot x^{[m]} > t$ and $\forall_{m \in non\text{-}spam} \beta \cdot x^{[m]} \leq t$. A set of messages is said to be *linearly separable* if there exists a separating hyperplane for the set. The logit transform renders the probabilistic classifier developed in the previous section an example of a linear classifier. In this section, we describe a geometric interpretation and several construction methods.

For convenience, we limit our illustrations to the case of $n = 2$; it should be kept in mind that typical spam filtering applications involve many more features, and hence dimensions. Figure 3.2 shows the vector space representation for the 18 messages in our running example. The x-axis corresponds to the `head:enron` feature transformed using the Gaussian model (cf. Table 3.4). The y-axis corresponds to the `body:enron` feature represented as a simple count. The diagonal line is a separating hyperplane because all spam messages fall to one side and all non-spam to the other. As such, it is a perfect classifier — at least for the sample data! Figure 3.3 shows that the same line is

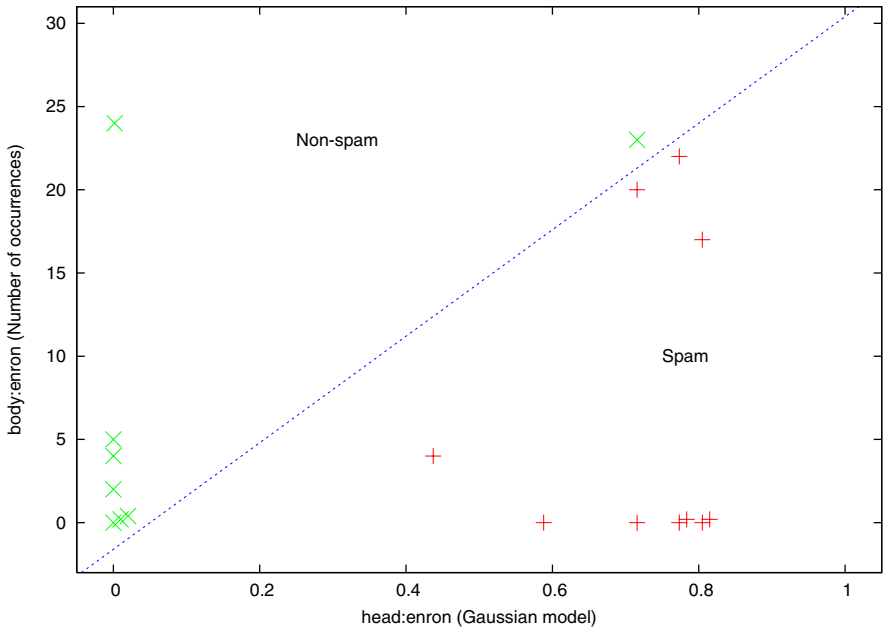

Fig. 3.2 Separating hyperplane.

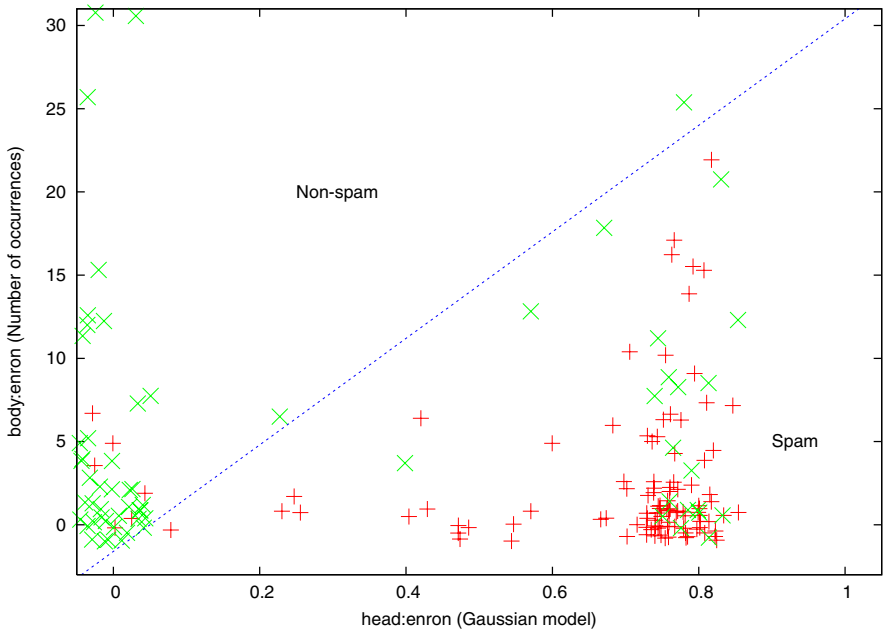

Fig. 3.3 Linearly inseparable sample.

not a separating hyperplane for a larger sample from the same source, indeed, none exists. Still, most spam lies on the *spam* side of the line while most non-spam lies on the other side. So the line is a reasonable classifier. But is it the *best* linear classifier within this vector space? And how may it be chosen using only the training data? The answer depends on the definition of *best*.[2]

If the points are linearly separable, there are, in general, an infinite number of separating hyperplanes. Any linear combination of the extreme curves shown in Figure 3.4, provided it has positive coefficients, will itself separate spam from non-spam. It is not apparent that the best classifier is a separating hyperplane, even if one exists. If one were to assume that the non-spam $(0.72, 23)$ were an outlier — perhaps a mistake in the training data — one might reasonably choose the vertical separator in Figure 3.5, which reflects the assumption that

[2] At this point, we adopt the definitions of *best* inherent in the classification algorithms of interest, their appropriateness within the context of spam filtering is addressed in Section 4.

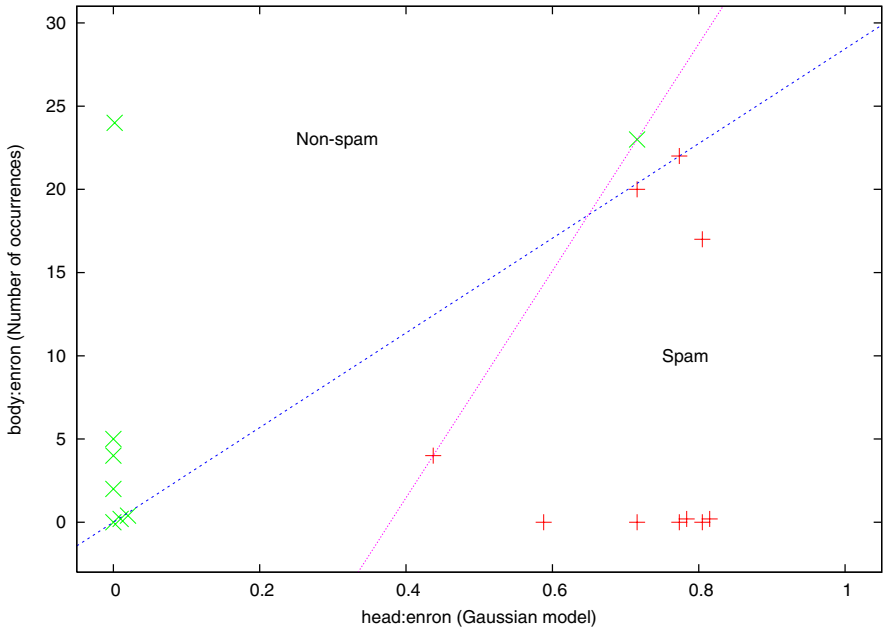

Fig. 3.4 Best separating hyperplane?

the second feature has no real effect. Hindsight (i.e., Figure 3.2) tells us intuitively that our original separator was more appropriate, however, we are concerned here with justifying the choice based on the training data alone.

Figures 3.2 and 3.5 represent two competing views of what constitutes the best classifier:

- One which correctly classifies all examples while maximizing the distance from the nearest example to the hyperplane.
- One which allows one or more examples to be misclassified while increasing the distance to the rest.

3.4.1 Perceptron Algorithm

The perceptron algorithm iteratively finds a separating hyperplane — any separating hyperplane, if one exists — by incrementing or decrementing the weights for every example on the wrong side (see

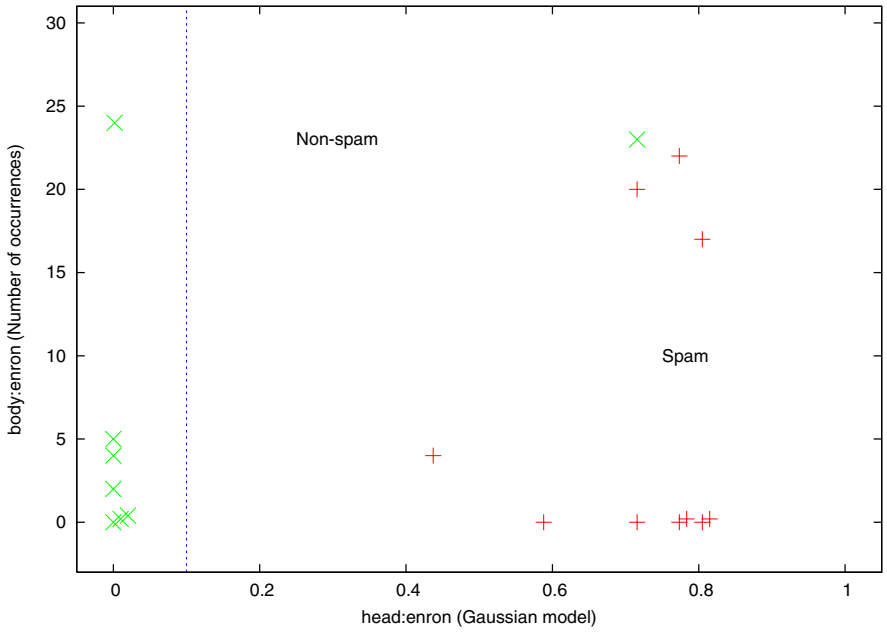

Fig. 3.5 Ignoring one point.

Figure 3.6). The algorithm ignores correctly classified examples. If the examples are linearly separable, the perceptron converges in a finite number of steps, otherwise it fails to terminate. For practical purposes it is sufficient to stop training after some time, under the assumption that a good, if not optimal in any sense, classifier has been found. The perceptron is attractive for spam filtering because it is simple, incremental, and adaptive.

The margin perceptron algorithm (Figure 3.7, cf. [152]) adds margin and rate parameters τ and δ which effect training on near-misses as well as errors, so as to bias the method to prefer higher margin separators, where the margin is defined to be the distance from the hyperplane to the nearest example.

3.4.2 Winnow Algorithm

The Winnow algorithm (cf. [158]) applies only to binary attribute vectors and considers only positive evidence, that is, it computes a weight

Input:
Set $T : M$ of training examples
$m \in T$ represented by $x^{[m]} = (1\, x_1^{[m]} x_2^{[m]} \cdots x_n^{[m]})$
Labeling $label : T \rightarrow \{-1, 1\}$
Output:
If linearly separable, β such that
$\beta \cdot x^{[m]} > 0$ iff $label(m) = 1$
else fails to terminate
Method:
$\beta \leftarrow (0 \cdots 0)$
while $\exists_{m \in T} \beta \cdot x^{[m]} \cdot label(m) < 0$
$\beta \leftarrow \beta + x \cdot label(m)$

Fig. 3.6 Perceptron learning algorithm.

Input:
Set $T : M$ of training examples
$m \in T$ represented by $x^{[m]} = (1\, x_1^{[m]} x_2^{[m]} \cdots x_n^{[m]})$
Labeling $label : T \rightarrow \{-1, 1\}$
Margin and rate parameters τ and δ
Output:
If linearly separable, β such that
$\beta \cdot x^{[m]} > 0$ iff $label(m) = 1$
else fails to terminate
Method:
$\beta \leftarrow (0 \cdots 0)$
while $\exists_{m \in T} \beta \cdot x^{[m]} \cdot label(m) < \tau$
$\beta \leftarrow \beta + x \cdot \rho \cdot label(m)$

Fig. 3.7 Perceptron with margins.

vector β with strictly positive elements. If the data are linearly separable, it will find β such that $\beta \cdot x^{[m]} > 1$ iff $label(m) = 1$. Like perceptron, Winnow trains on errors, and can also be adapted to train on near misses. Winnow uses multiplicative, in contrast to perceptron's additive, reinforcement for β. More specifically, elements of β corresponding to features present in the training sample are multiplied

by a promotion factor $\alpha > 1$ when $\beta \cdot x^{[m]} > 1$ and $label(m) = 0$, and by a demotion factor $0 < \beta < 1$ when $\beta \cdot x^m \leq 1$ and $label(m) = 1$. Since Winnow is asymmetric, it is common to combine the results of two Winnow methods — each trained independently, one predicting $label(m) = 1$ and the other predicting $label(m) = 0$ — into an overall prediction.

Siefkes et al. [158] report that it is useful to normalize the result by substituting $\frac{\beta \cdot x^m}{n}$ for $\beta \cdot x^m$, where n is the number of nonzero features.

3.4.3 Support Vector Machines

A support vector machine (SVM) directly computes the separating hyperplane that maximizes the margin or distance to the nearest example points. Several points will be at the same distance, these points are known as the *support* vectors as the classifier is a linear combination of them — all other points may be ignored. Thus the SVM would prefer the solution in Figure 3.2 over the ones in Figure 3.4, with support vectors of $(0, 0)$, $(0.72, 23)$ on the non-spam side and $(0.72, 20)$ on the spam side.

In the case of non-separable data, or separable data in which a few points dramatically affect the solution (e.g., Figure 3.5 or point $(0.72, 23)$ in our training data), it may be desirable to relax the requirement that all training data be correctly classified. Contemporary SVM formulations implement a tradeoff between maximizing the margin and minimizing the magnitude of training errors. The tradeoff parameter C determines the relative weight of the second objective to the first. $C = 0$ specifies the pure SVM detailed in the previous paragraph, $C = 1$ gives the objectives balanced weight, and is a typical default value, $C = 100$ gives the second objective substantial weight, and has been found to be appropriate for spam filtering [54, 152].

Sculley [150] describes a gradient method for efficient incremental on-line spam filtering using SVMs. Efficient implementations for batch SVM computation are available (see Figure 3.8).

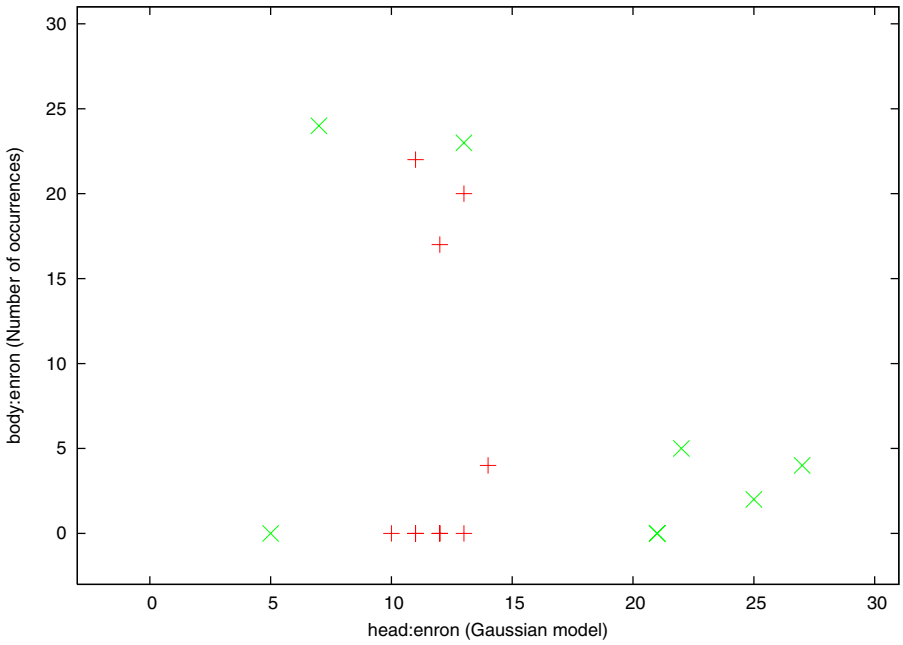

Fig. 3.8 Untransformed features.

3.5 Rocchio Method

The Rocchio classifier derives from Rocchio's method to harness relevance feedback in information retrieval, for which the objective is to rank documents by the estimated probability that they are relevant to a particular topic. Like Winnow, Rocchio's method takes into account only positive evidence; a classifier is formed by comparing the weighted results of two instances of the Rocchio feedback method: one considering spam to be the positive class, and one considering non-spam to be the positive class.

Rocchio's method typically uses the tf-idf-weighted bag-of-words model, and cosine as a similarity measure, i.e., $similarity(x,y) = \frac{x \cdot y}{|x| \cdot |y|}$ or, if x and y are always normalized, simply $similarity(x,y) = x \cdot y$. Let C_s be the centroid of all $x^{[m]}$ such that $label(m) = 1$, and C_n be the centroid for all $x^{[m]}$ such that $label(m) = 0$. $Similarity(x, C_s)$ represents the Similarity of x to spam while $similarity(x, C_n)$ represents the

similarity of x. A hard classifier for m is effected by comparing the weighted difference between these measures to a threshold t: $similarity(x, C_s) - \alpha \cdot similarity(x, C_n) > t$. That is, $\beta \cdot x > t$, where $\beta = C_s - \alpha \cdot C_n$. Joachims [90] argues that Rocchio classifiers are uncompetitive for general text classification, and we are unaware of any results that contradict this conclusion within the context of spam. Sebastiani [153] provides further details.

3.6 Nearest Neighbor Methods

A nearest neighbor or memory-based classifier [156] computes the distance between the feature vector $x^{[m]}$ and several examples $x^{[m' \in T]}$ for which $label(m')$ is known. In the simplest case, the class of m is estimated to be that of the nearest example, i.e., $label(m) = label(m')$, where $m' = \arg\max_{m'} x^m \cdot x^{m'}$. A simple variant is *k-nearest neighbor* (kNN) in which, for some fixed k, the k most similar examples are identified, and the majority class among these examples is assigned to x. Or some threshold $0 \leq t < k$ may be chosen, and x deemed to be spam if $n > t$, where n is the number of among the k that represent spam. *Weighted nearest neighbor* weights each of the k nearest examples by its similarity to x and compares the sum of the weights to a threshold. Several authors consider the use of clustering and kNN methods for spam filtering, but none report strong performance [6, 47, 143, 144, 197].

Duplicate or near duplicate detection [97, 98] is a special case of nearest neighbor in which $similarity(x, y)$ yields a binary result rather than a distance. Fingerprinting and signature files are names given to methods in which a database D of known spam messages is maintained and queried to determine, for a message to be classified, whether $similarity(x^{[m]}, x^{[m']}) = true$ for some $m' \in D$. The motivation for this approach is that spammers often send vast numbers of similar messages which may, after the first few, be recognized as spam. The challenges include efficient implementations of D and defining *similarity* so that it cannot easily be defeated by spam messages that are continuously perturbed so as to defeat duplicate detection.

3.7 Logic-Based Methods

Decision rules use logic to express the interactions among features. Decision rule inference systems like RIPPER [30] compete with decision trees in terms of efficiency and effectiveness.

Decision trees like C4.5 [132] successively split the training data (and, hopefully, the data to be classified) according to the value of one feature at a time, eventually arriving at a subspace populated exclusively by members of one class: spam or non-spam. In the worst case, such a tree may be of exponential size. The trick is to choose the order in which to examine the features and to prune the tree so as to partition the data with many fewer nodes. Decision stumps, consisting of just one node, represent the extreme case. Heuristics to do this typically make similar assumptions to those used in feature selection, ordering the dimensions by some measure of the degree of evidence afforded. Typically, these pruned decision tree classifiers are rather weak classifiers, they may be improved substantially by ensemble methods like bagging and boosting which combine the results of a large number of randomly generated classifiers [24, 54, 179].

As for feature selection, incremental and adaptive methods for rule inference and tree construction remain elusive, particularly in combination with bagging or boosting.

3.8 Data Compression Models

A data compression model D estimates the *information content* of a sequence $s = s_1 s_2 \cdots s_n$ of symbols from a finite alphabet Σ. $I(s^{[m]}) = -log_2(\text{Prob}(s = s^{[m]}))$ is the information content of the sequence $s^{[m]}$ representing a particular message m, a lower bound on the number of bits required to represent m. $\text{Prob}_D(s = s^{[m]})$ is the *likelihood* of $s^{[m]}$ under D, and $I_D(s^{[m]}) = -log_2(\text{Prob}_D(s = s^{[m]}))$ estimates $I(s^{[m]})$. D_1 models $s^{[m]}$ *better than* D_2 if $I_{D_1}(s^{[m]}) < I_{D_2}(s^{[m]})$. For data compression, the objective is to find the best possible model so as to minimize the encoded length over all s of interest.

For classification, we construct two models: D_{spam} and $D_{non\text{-}spam}$ which model spam and non-spam, respectively. The classifier is

predicated on the assumption that D_{spam} will model $s^{[m \in spam]}$ better than $D_{non\text{-}spam}$ and that $D_{non\text{-}spam}$ will model $s^{[m \in non\text{-}spam]}$ better than D_{spam}. The hard classifier

$$c(m) = \begin{cases} true & (I_{D_{spam}}(s^{[m]}) < i_{D_{non\text{-}spam}}(s^{[m]})) \\ false & otherwise \end{cases}$$

follows from this definition, as does a soft classifier

$$c(m) = I_{D_{non\text{-}spam}}(s^{[m]}) - I_{D_{spam}}(s^{[m]}).$$

The soft classifier is amenable to probabilistic interpretation. Define the *correct model* for m to be

$$\begin{aligned} D^{[m]} &= \begin{cases} D_{spam} & (m \in spam) \\ D_{non\text{-}spam} & (m \in non\text{-}spam) \end{cases}. \\ I_{D_{spam}}(s^{[m]}) &= -\log(\text{Prob}(s = s^{[m]} | D^{[m]} = D_{spam})) \\ I_{D_{non\text{-}spam}}(s^{[m]}) &= -\log(\text{Prob}(s = s^{[m]} | D^{[m]} = D_{non\text{-}spam})) \\ c(m) &= \log\left(\frac{\text{Prob}(s = s^{[m]} | D^{[m]} = D_{spam})}{\text{Prob}(s = s^{[m]} | D^{[m]} = D_{non\text{-}spam})}\right) \\ &= \log\left(\frac{\text{Prob}(s = s^{[m]} | m \in spam)}{\text{Prob}(s = s^{[m]} | m \in non\text{-}spam)}\right) \\ &= \log\left(\frac{Odds(m \in spam | s = s^{[m]})}{Odds(m \in spam)}\right), \end{aligned}$$

$$logOdds(m \in spam | s = s^{[m]}) = logOdds(m \in spam) + c(m).$$

That is, given an estimate of the prior probability p that $m \in spam$, data compression models estimate the probability of m being spam to be $logit^{-1}(logit(p) + c(m))$.

3.8.1 Sequential Models

Sequential compression models like prediction by partial matching (PPM) [29], dynamic Markov compression (DMC) [38], and context-tree weighting (CTW) [190], process a message sequentially, in effect

constructing a separate model D_k for each prefix $s_1 s_2 \cdots s_k$ of its input to compute $I_{D_k}(s_{k+1})$. The information content of the message is

$$I_D(s) = \sum_{i=1}^{n} I_{D_{i-1}}(s_i).$$

For spam filtering [16], we form two sequences S and N by concatenating all the spam messages in T and all the non-spam messages in T, respectively. Using the same method, two models D_S and D_N are constructed yielding $I_D(S)$ and $I_D(N)$, respectively. A message m to be classified is concatenated to S and to N and new models D_{Sm} and D_{Nm} are constructed yielding $I_D(Sm)$ and $I_D(Nm)$. Using Bayes' rule, we deduce

$$\begin{aligned} I_{D_{spam}}(m) &= I_D(Sm) - I_D(S) \\ I_{D_{non\text{-}spam}}(m) &= I_D(Nm) - I_D(N) \\ c(m) &= I_{D_{non\text{-}spam}} - I_{D_{spam}}. \end{aligned}$$

In practice it is unnecessary to compute D_{Sm} and D_{Nm} from scratch. Sequential compression models efficiently construct D_{Sm} and D_{Nm} from stored representations of D_S and D_N. Incremental training is easily effected by replacing D_S by D_{Sm} when $m \in \textit{spam}$ and D_N by D_{Nm} when $m \in \textit{non-spam}$. Incorrect self-training is not easily reversed.

Figure 3.9 illustrates DMC applied to the `To:` fields[3] of the 18 messages in our running example, in their natural order (cf. Table 3.1). The first column of Figure 3.9 shows the true class of the message, the second $c(m)$, the third shows the `To:` field of the message. Each character x_i is colored to indicate its "spamminess" within the string: bold indicates spam ($c_{i-1}(x_i) \gg 0$), while italic indicates non-spam ($c_{i-1}(x_i) \ll 0$) and normal indicates neither ($c_{i-1}(x_i) \approx 0$). DMC colors the first message Grey, as there are no previous messages so the spam and non-spam models are identical. The next two messages are classified (correctly) as spam, which is perhaps not surprising as there are no non-spam examples with which to compare. Indeed the fourth message — the first

[3] For brevity of illustration, the method has been applied only to the *To:* field extracted from the message. Superior results are achieved if the method is applied to the whole message, or to a fixed-length prefix of the message.

```
S  0.0 To: emclaug@enron.com
S  3.9 To: Skean@enron.com
S  0.1 To: <joydish@bareed.alburaq.net>
H  1.1 To: "Shapiro, Richard" <Richard.Shapiro@ENRON.com>
H -0.8 To: "Adams, Jacqueline P." <Jacqueline.P.Adams@ENRON.com>,
H -4.4 To: "Adams, Jacqueline P." <Jacqueline.P.Adams@ENRON.com>,
H  1.6 To: pete.davis@enron.com
S  1.0 To: KAM.KEISER@enron.com
H -2.1 To: "Abel, Chris" <Chris.Abel@ENRON.com>,
H -1.7 To: "Moran, Tom" <Tom.Moran@ENRON.com>,
S -0.1 To: nqeb5e@msn.com
S  1.6 To: skean@enron.com
H -1.3 To: pete.davis@enron.com
S  0.2 To: kholst <kholst@enron.com
H -1.1 To: "Scott, Susan M." <Susan.M.Scott@ENRON.com>
S  0.9 To: mmotley@enron.com
S  3.3 To: kholst@enron.com
S  3.1 To: keith.holst@enron.com
```

Fig. 3.9 DMC results on training set.

non-spam — is incorrectly classified. As more examples are learned, the models are better able to distinguish spam from non-spam. The last seven messages are correctly classified. The colors of individual characters reveal that some key indicators of non-spam are:

- quotation marks,
- `ENRON` in upper case,
- specific names like `Adams` and `pete.davis`.

Indicators of spam are:

- `enron` in lower case,
- upper case sequences other than `ENRON`,
- variants of the name `kholst`.

The data compression models used for spam filtering harness the correlation between adjacent symbols in the message. DMC (Dynamic Markov Compression) uses a bit-wise dynamic Markov model that incrementally adapts to model longer sequences which occur frequently. Prediction by Partial Matching (PPM), in contrast, tacitly uses an n-gram character model ($4 \leq n \leq 8$ is typical) for which a suffix tree representation is more amenable than a feature vector, because it consumes linear space. Context-tree weighting is asymptotically optimal

under certain theoretical assumptions, but is more complex and does not appear to yield better results than DMC or PPM.

Methods that employ suffix trees [127], n-gram [28], or language models [120] rely on similar evidence to that of compression models.

3.9 Meta Classifiers

The methods outlined in this section all estimate the same quantity: the truth value of the proposition that a message is spam. For the reasons stated in Section 3.3.4 an average of the estimates is likely to yield better precision than any single one. The synthesis of classifier results — which goes by a plethora of names like voting, stacking, fusion, ensemble, committee, cocktail, and pipeline — finds common use in spam filtering.

SpamAssassin, for example, combines the results of its 800 tests, many of which are classifiers in their own right, using a linear classifier originally constructed by hand, and later using a genetic algorithm, a perceptron and logistic regression. SpamAssassin's meta classifier is constructed off-line using training messages collected by volunteers, and the resulting weight vector β is distributed periodically. In contrast, on-line meta classifiers [114, 154] employ incremental and adaptive methods to synthesize the results of other on-line methods.

Bagging [19] and boosting [145] combine several weak classifiers generated by applying the same construction method to randomly perturbed data.

4

Evaluation Methods and Measures

Although email is ubiquitous, privacy issues limit our ability to use it for comparative study. Email from public sources like mailing lists may not adequately represent personal or corporate email, either in its spam or non-spam content. Removing or obfuscating sensitive content is a formidable challenge, and may dramatically compromise spam filter effectiveness. Real-time or meta information may be difficult to capture; its absence may also compromise filter effectiveness. The true labels for messages — essential to precise evaluation — are difficult to determine, especially in real-time. In spite of these challenges, more realistic test collections, along with more realistic laboratory and field test methods are being developed on an ongoing basis. This section outlines the issues that might arise and the approaches that might be brought to bear on the problem of spam filter evaluation. Section 5 describes the major efforts and results to date.

4.1 Test Corpora

A test corpus is a collection of email messages with an associated *gold standard* closely approximating the true classification for some or all of

the messages. In addition, the corpus may be chronologically sequenced. For the purpose of comparative evaluation, splits may be defined that separate the email messages into one or more pairs of test and training sets, or the messages may be used in sequence for on-line evaluation.

Technically, it is a simple matter to capture all the email delivered to a recipient or a set of recipients. Publishing this email as a corpus, or using it for field testing, is not so simple. Few individuals are willing to publish their email, because doing so would compromise their privacy and the privacy of their correspondents. A choice must be made between using a somewhat artificial published collection of messages and using a more realistic collection that must be kept private. Published collections facilitate comparative evaluation because diverse methods may be tested under exactly the same circumstances, private collections facilitate comparison only through experiments run by the proprietors of the data.

The gold standard represents, as accurately as is practicable, the result of adjudicating each message in the collection according to the definition of spam. The gold standard plays two distinct roles in the evaluation framework. The gold standard is assumed to be *truth* in measuring the filter's error rates. The gold standard is also a source of training labels. It may be appropriate to bifurcate the gold standard for these two purposes, while errors are never desirable in evaluating the filter's error rates, they may aptly reflect the training data available to the filter in real-world deployment.

Human adjudication is a necessary component of gold standard creation. One approach is to ask the email recipient to sort his or her email, a tedious and error-prone process. Another approach is to have the recipient use a spam filter, and to report only the errors made by the filter, assuming that unreported messages were correctly classified by the filter. This method also has a substantial error rate [42]. A third party adjudicator — other than the recipient — may also perform the task.

A bootstrap method can improve both efficiency and accuracy [42]. An initial gold standard G_0 is created using the method above. One or more filters is run, using G_0 for training and evaluation. Each message for which the filter and G_0 disagree is re-adjudicated and, where G_0 is

found to be wrong, it is corrected. The result of all corrections is a new standard G_1. This process is repeated, using different filters, to form G_2, and so on, to G_n. The final gold standard, G_n, may be expected to have a much lower error rate than that of the adjudicator or any of the filters alone. Segal [155] investigates the use of uncertainty sampling [107] as a method for efficiently generating gold standard labels. Graham-Cumming [73] investigates the use of exhaustive volunteer-based adjudication.

Corpus testing is valuable in that it allows a vast number of filters and methods to be tested under identical circumstances. It is limited to the extent that the messages in the corpus are a realistic, timely sample of real email. It is further limited in that the interaction with the user and with external resources such as blacklists are difficult to emulate.

4.2 Real-Time Aspects

The most direct way to capture the real-time features that a filter might use is to operate the filter in real-time, perhaps using a standard interface so that a number of filters may be plugged in and tested — either sequentially or in parallel for the same user (a crossover study design) or for different users (a randomized controlled trial design). Another approach is to find existing installations of particular techniques, and to measure their effect (a case-control design). Practitioners generally rely on *ad hoc* methods of evaluation that resemble crossover or case-control studies, but lack the controls normally associated with scientific evaluation. To our knowledge no such studies have yet been reported in the literature; however, CEAS 2007 has launched a live spam challenge [170], which is a real-time parallel crossover study. In addition, the email stream and gold standard are captured as a corpus so that the real-time results may be compared with post-hoc laboratory experiments.

An alternate approach is to collect pertinent real-time information as part of a test collection, and to reproduce it in a later simulation. Some of this information, like the time of delivery and message envelope information is easy to capture. Its inclusion in corpora marks the best practice in simulating real-time aspects after the fact. Other

information, like the state of external servers (DNS, blacklists, and the like) is more problematic. If the exact information that might be queried is known, it might also be collected and stored with the corpus. However, this information that is queried is filter-specific, with the upshot that a historical trace of the entire server content would be needed. As vast amounts of disk storage are cheaply available, it may be feasible to capture all the updates to a server so as to be able to reproduce its state at any particular moment.

A hybrid approach is to collect the information as for a corpus, but to do so and conduct the experiments within a short time interval, so that time-dependent factors will not have changed much between collection and experiment (e.g., [160]).

4.3 User Interaction

The most direct way to capture user interaction effects is through the user's actual interaction with the filter, using one of the designs for live real-time evaluation.[1] The main difference is that the user's behavior is affected by the filter's behavior, and the filter's is in turn affected by the user's. Blind — even double blind — experiments may be conducted provided all interaction is through a standard interface, and the filter itself plugs in through a standard protocol so that neither the user nor the tester is aware of the filter being tested.

User studies might also be conducted using simulated real-time message delivery based on a corpus, and a randomized controlled design. However, the messages would not be specifically addressed to the subject users, so the validity of their interactions would be questionable. The complementary approach is to use either actual or simulated real-time data, and to simulate the user's behavior. TREC 2005 [40] simulates an idealized user, who notices and immediately corrects any filter errors. Any number of other user interactions may be modeled as simulated within the TREC design. TREC 2006 [33] simulated delayed feedback so as to model the user's reading mail only occasionally. Further experiments may be designed to model incomplete and inaccurate

[1] However, the most powerful design — the parallel crossover study — would not be amenable because the user could interact with only one filter at a time.

feedback, with model parameters estimated so as to approximate a range of human behaviors. Separate experiments may be designed to validate these models.

The CEAS design, being real-time, requires real-time construction of a gold standard for the purpose of simulating user feedback. It is only necessary for the real-time gold standard to be accurate enough and timely enough to reasonably simulate user behavior. This affords ample time for a panel of adjudicators to employ several iterations of the batch or uncertainty sampling techniques. Further post-hoc adjudication may be effected for the purpose of evaluation.

4.4 Sender Interaction

Evaluation of filters involving sender interaction encounter an additional challenge over and above those arising from user interaction. The potential that interaction may alter the sender's behavior is more acute, as the sender is likely to be uncooperative or even adversarial. Furthermore, it is possible to observe the sender's behavior only from the perspective of the recipient.

The tradeoffs in experimental design between live testing and simulation are affected in the following ways. First, a user typically interacts with many different senders, whose behavior must be modeled for evaluation. Even a reasonably large study would lack sufficient data from which to estimate the parameters to model each of the senders.

Estimates of false negative rates are reasonably easy to achieve, as messages classified as non-spam are delivered to the user and may be adjudicated using any of the techniques we have outlined previously. Estimates of false positive rates are much more difficult to achieve, as messages classified as spam are often never delivered. Without adjudication of these non-delivered messages, false positive rates cannot be measured at all.[2] Arguments that legitimate senders would necessarily behave so as to ensure delivery ($fpr = 0$), without empirical evidence, are unavailing.

[2] Furthermore, the value of these techniques cannot be directly compared with others — even those that do not rely on sender interaction — unless the entire set of sent messages is available.

One approach to adjudication is to have the filter capture all messages for later adjudication. Some filters do so in the course of their interaction, while others do not. Challenge–response systems, for example, typically accept and hold the transmitted message pending response to the challenge. Greylisting systems may also capture the entire message before returning a soft error to the sender, however, many do not and modifying them to do so materially changes their behavior as observed by the sender. Tarpitting, in contrast, is predicated on a strategy of inducing the sender to abort transmission, a strategy whose effect cannot possibly be measured (at least from the receiver's perspective) while capturing the entire message.

It may be possible to adjudicate undelivered messages based on partial information, such as the IP or email address of the sender. Such adjudication would likely be more difficult and error prone than were it based on full messages.

Even the ability to capture all attempted transmissions does not completely capture the effect of sender interaction. Mailing list software, for example, commonly deletes or suspends the subscriptions for users whose messages it fails to deliver. Many senders — this author included — refuse to respond to challenges and, when challenged, never again attempt communication. Hardware or network failure may occur during the interaction process,[3] thwarting or materially delaying delivery. Future messages may therefore be left unsent as a direct result of the filter's action. While one may argue whether future unsent messages fit the narrow definition of false positives, their non-transmission is caused by the filter, and our overarching principle of measuring the suitability of the filter for its intended purpose comes to bear: preempting message transmission has similar consequences to losing transmitted messages, so, in the absence of a explicit measurement of this effect, including it in *fpr* seems more apt than ignoring it. If it can be measured, that is.

[3] Indeed, failure or delay may be wholly or partially caused by the interaction process because of general load increases, or due to the potential of runaway interactions. Imagine, for example, the situation in which both sender and receiver both use naive challenge–response systems. Arguments that more sophisticated challenge–response protocols entirely prevent such situations are, without supporting evidence, unavailing.

4.5 Community Interactions

To the extent that a filter is a passive consumer of information collected from a global community of friendly and adversarial users, its efficacy may be evaluated using designs already elaborated for capturing real-time effects. If the information changes slowly it might be appropriate to capture an evaluation corpus and to evaluate filters using global information available at a later time — the time of experiment.

An alternative approach is to capture a snapshot, or partial snapshot, of available community information, and to reproduce this information at the time of evaluation. This approach is limited by the feasibility of capturing the information, and the representativeness of any particular subset of the global information space.

The fundamental property that distinguishes community-based filters from others is that they are active, not passive, participants, influencing as well as consuming global information. For a small-scale deployment, this influence may be inconsequential, however, if a filter is widely deployed the overall influence may be substantial, not only for users of the filter, but for other community members, friend and foe alike. Economic or biological models [167, 200] attempt to predict such influences, using techniques such as game theory [5, 161, 49]. Validation of such models presents a considerable challenge, as common study designs are generally infeasible.

4.6 Summary Measures

The purpose of a summary measure is to estimate the effectiveness of a spam filter, given the filter's result and the gold standard for a collection of messages. As discussed previously, the effectiveness of a hard classifier may be characterized as the two-dimensional quantity (fnr, fpr), and the effectiveness of a soft classifier may be characterized a set of such pairs — an ROC curve. These measures evaluate the filter itself, and are insensitive to deployment-specific parameters such as the proportions of ham and spam to be classified, and the consequences of errors. These parameters may be combined with fnr and fpr to yield an overall cost and benefit estimate for any particular deployment.

While these measures are standard in the diagnostic testing literature [66], they are far from ubiquitous in the spam filtering literature (see Table 4.1). A large number of studies report *precision* and *recall* or F_1 from information retrieval [180], others report *accuracy, weighted accuracy* or *total cost ratio* [4] (see Tables 4.2 and 4.3).

4.6.1 Diagnostic Testing Measures

The filter's result and the gold standard for a particular collection of messages may be represented by a contingency table as shown in Table 4.4. In Table 4.4, tn, tp, fn, fp represent the number of messages correctly classified as ham, correctly classified as spam, incorrectly classified as ham, and incorrectly classified as spam. $n = tn + tp + fn + fp$ is the total number of messages; $tn + fp$ is the number of ham messages, $tp + fn$ is the number of spam messages, $tn + fn$ is the number of messages classified as ham, $tp + fp$ is the number of messages classified as spam. Table 4.5 shows the contingency table for the results in Figure 3.9.

Table 4.1 Signal detection and diagnostic test measures.

Label	Description
fpr	*false positive rate.* The fraction of non-spam that is misclassified. Identical to *fallout* in IR evaluation.
fnr	*false negative rate.* The fraction of spam that is misclassified.
prev	*prior probability, a.k.a. prevalence.* The fraction of all email that is spam.
lr_+	*positive likelihood ratio.* The relative likelihood of a positive filter result, under the pair of assumptions that the message is spam and non-spam.
lr_-	*negative likelihood ratio.* The relative likelihood of a negative filter result, under the pair of assumptions that the message is spam and non-spam.
ppv	*positive predictive value.* The probability that a message is spam, given a positive filter result. Identical to *precision* in IR.
npv	*negative predictive value.* The probability that a message is non-spam, given a negative filter result.
ROC	*receiver operating characteristic curve.* The curve representing *tpr* (1-*fnr*) as a function of *fpr* (1-*tnr*) for all possible threshold values.
AUC	*area under the ROC curve.* The area under the ROC curve; alternatively, $\text{Prob}(c(m_1) > c(m_2) \mid m_1 \in \textit{spam}, m_2 \in \textit{non-spam})$
(1−AUC)%	*percent area above the ROC curve.* $100 \cdot (1 - AUC)$.
DOR	*diagnostic odds ratio.* $\frac{lr_+}{lr_-}$
lam	*logistic average misclassification.* $lam = logit^{-1}(\frac{logit(fpr)+logit(fnr)}{2})$.
d′	*discrimination.* The number of standard deviations separating the spam and non-spam score distributions, assuming both are Gaussian.

Table 4.2 IR measures.

$Precision = ppv$. From an information retrieval perspective, the fraction of results returned from a search that are relevant to the request. Unavailing as a measure of spam filter effectiveness, especially if spam is considered, as it commonly is, the positive class: the fraction of all quarantined messages that are spam. Proportional to prevalence if fpr and fnr are invariant.
$Recall = tpr = 1 - fnr$. The fraction of all relevant documents that are returned from a search. The fraction of spam correctly identified.
$Fallout = fpr$. The fraction of all irrelevant documents that are retrieved. The fraction of non-spam that is misclassified.
Recall-fallout curve. The curve representing recall as a function of fallout depending on the number of documents retrieved. Equivalent to ROC curve.
Recall-precision curve. The curve representing precision and a function of recall depending on the number of documents retrieved. Akin to ROC curve, with ppv substituted for $(1 - fpr)$.
Average precision. The area under the recall-precision curve.
$F_\beta = \frac{(1+\beta)\cdot precision \cdot recall}{\beta \cdot precision + recall}$ The weighted harmonic mean of recall and precision. Commonly, $\beta = 1$.

Table 4.3 Accuracy based measures.

$Accuracy = prevalence \cdot tpr + (1 - prevalence) \cdot tnr$. The fraction of all documents that are correctly identified. Poorly regarded for evaluating diagnostic tests, as, like precision, it depends on prevalence [3].
$Error = 1 - accuracy$.
$Weighted accuracy = \lambda \cdot prevalence \cdot tpr + (1 - \lambda) \cdot (1 - prevalence) \cdot tnr$, $(0 < \lambda < 1)$. The fraction of all documents correctly identified, assuming each spam to appear λ times [6].
$Total cost ratio$. The ratio between the weighted accuracy of a filter and the weighted accuracy of the *trivial rejecter* with $tnr = 0, tpr = 1$ [6].
Diluted false positive rate. $dfpr = \frac{fpr}{prevalence}$. The fraction of all messages that are both non-spam and misclassified as spam. Inflates fpr by a factor of $\frac{1}{prevalence}$ [160].

Table 4.4 Contingency table.

	Gold standard	
Filter	ham	spam
ham	tn	fn
spam	fp	tp

fnr and fpr are defined as $fnr = \frac{fn}{tp+fn}$ and $fpr = \frac{fp}{tn+fp}$. Note that $fpr = \text{Prob}(c(m) = spam | m \notin spam)$ is the likelihood that the filter will misclassify a ham message. Similarly, $fnr = \text{Prob}(c(m) \neq spam | m \in spam)$ is the likelihood that the filter will misclassify a spam message. Table 4.6 shows the values of fpr, fnr and the measures defined below for the contingency table in Table 4.5.

Table 4.5 Contingency table for results in Table 3.9.

	Gold standard		
Filter	ham	spam	total
ham	6	2	8
spam	2	8	10
total	8	10	18

Table 4.6 Measures derived from Table 4.5.

fpr	fnr	$prev$	lr_+	lr_-	ppv	npv	dor	lam	$1 - AUC(\%)$
0.25	0.2	0.56	3.2	0.27	0.8	0.75	12	0.22	15.0

The *predictive value* of a test is the probability that the outcome of the test is correct, more specifically, the *positive* predictive value $ppv = \text{Prob}(m \in spam | c(m) = spam)$ and the *negative* predictive value $npv = \text{Prob}(m \notin spam | c(m) \neq spam)$. For the test deployment denoted in the table, these values may be estimated directly: $ppv = \frac{tp}{tp+fp}$, $npv = \frac{tn}{tn+fn}$. Using Bayes' rule, it is possible to compute predictive value for other deployment scenarios in which the proportion or *prevalence* of spam is different from the test deployment, using only fnr and fpr from the test results. We have $Odds(m \in spam) = \frac{prev}{1-prev}$. Define the *positive likelihood ratio* $lr_+ = \frac{1-fnr}{fpr}$ and *negative likelihood ratio* $lr_- = \frac{fnr}{1-fpr}$. Then $Odds(m \in spam | c(m) = spam) = Odds(m \in spam) \cdot lr_+$ and $Odds(m \in spam | c(m) \neq spam) = Odds(m \in spam) \cdot lr_-$ (cf. Table 4.6).

Figure 4.1 shows the ROC curve for the same test results. The ROC curve traditionally plots true positive rate (otherwise referred to as *sensitivity* or $1 - fnr$) as a function of fpr (also referred to as 1-*specificity*). The ROC curve for a good filter will approach the point $(fpr = 0, fnr = 0)$, that is, the top-left corner of the plot. AUC, the area under the curve, summarizes the proximity of the curve to this ideal point, for this example, $AUC = 0.85$, that is, $(1 - AUC)\% = 15.0$.

4.6.2 Threshold Independent Measures

While AUC gives a reasonable summary measure for soft classifier performance independent of threshold setting, it does not directly indicate the (fpr, fnr) result that might be achieved for a particular threshold

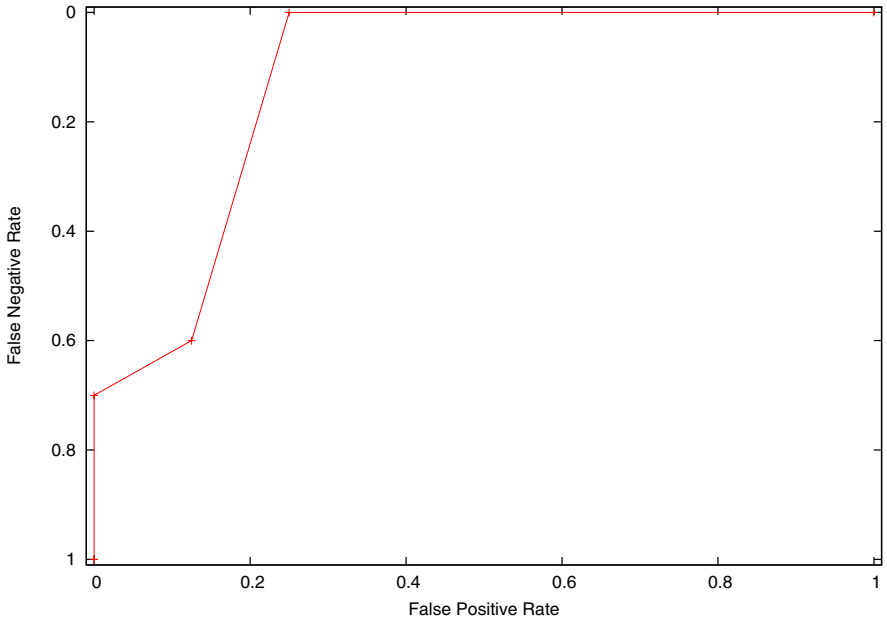

Fig. 4.1 Receiver operating characteristic curve for Table 4.5.

setting. Similarly, the (fpr, fnr) pair for a hard classifier gives no direct indication how the classifier would behave at different thresholds settings and hence what AUC result it might achieve. In both cases we might wish, for example, to estimate the value of fnr that would result if the threshold were adjusted to achieve $fpr = 0.1\%$.

It has been observed [66] that the diagnostic odds ratio, $dor = \frac{lr_+}{lr_-} = \frac{tp \cdot tn}{fp \cdot fn}$ is, for many diagnostic tests, effectively invariant over a large number of threshold settings. Intuitively, a change in threshold setting that increases lr_+ by some multiplicative factor tends to decrease lr_- by the same factor. Therefore dor is a useful summary measure largely uninfluenced by threshold setting. The same effect has been observed at TREC [40], giving rise to the measure logistic average misclassification rate, $lam = logit^{-1}(\frac{logit(fpr)+logit(fnr)}{2}) = logit^{-1}(log(dor^{-0.5}))$. In Table 4.6, $dor = 12$, $lam = 0.22$. Note that the value lam is necessarily between fpr and fnr; for small values it is indistinguishable from their geometric mean.

Under the assumption that *dor* is invariant, it is possible to estimate (fpr', fnr') from (fpr, fnr) by solving the equation

$$\frac{(1 - fpr) \cdot (1 - fnr)}{fpr \cdot fnr} \approx \frac{(1 - fpr') \cdot (1 - fnr')}{fpr' \cdot fnr'}.$$

It is further possible to estimate $AUC \approx \int_0^1 (fnr)\, d(fpr)$. Using $fpr = 0.25$ and $fnr = 0.2$ from Table 4.6, the *dor* invariance assumption yields a remarkably accurate estimate of $(1 - AUC)\% \approx 15.6$.

4.6.3 Other Signal Detection Measures

An alternative summary measure is the discrimination factor d' which is the standardized difference between the mean classifier results for spam and ham messages, that is, d' is the number of standard deviations that separate the two, under the assumption that the two sets of results is normally distributed with equal variance.

Two alternative representations for ROC curves better illustrate the differences among classifiers with very high AUC values. The TREC spam track plots the curves on the *logit* scale which, under the same assumptions as those that underlie the diagnostic odds ratio, tends to show each filter's performance as a horizontal line with slope 1. As for conventional ROC plots, curves closer to the top-left corner indicate superior effectiveness. DET curves [118] use the normal deviate (*probit*) scale which, under the same assumptions as those that underlie the discrimination factor, tends to show each filter's performance as a horizontal line with slope 1. The *logit* and *probit* functions yield similar results except for extreme values.

Table 4.2 summarizes the measures based on signal detection and diagnostic test theory.

4.6.4 IR Measures

Although signal detection measures have been proposed and used in classical information retrieval, they have been supplanted in this domain by *recall* and *precision* and summary measures based on them [166, 180, 185]. The task of classical information retrieval is to identify,

from a large collection of documents, as many *relevant* documents as possible while identifying as few irrelevant documents as possible. To apply the metaphor to spam filtering, one must either deem the spam messages to be relevant and the ham messages to be irrelevant, or the ham messages to be relevant and the spam messages to be irrelevant. The commonest, although not particularly apt, interpretation is the former. In this presentation, the unqualified terms *precision* and *recall* assume this "spam is relevant" interpretation, the qualified terms *ham-precision and ham-recall* assume the alternate "ham is relevant" interpretation.

Recall (or *ham-recall*) is identical to true positive rate: $recall = tpr = 1 - fnr = \frac{tp}{tp+fn}$. Precision is identical to positive predictive value: $precision = ppv = \frac{tp}{tp+fp}$. In terms of the metaphor, recall is the fraction of spam correctly identified by the spam filter, while precision is the fraction of quarantined messages that are spam. This measure is not particularly relevant to the intended purpose of a spam filter, and it says nothing whatsoever about fpr — the fraction of all ham messages that are quarantined. Furthermore, it is impossible to compute fpr from recall and precision alone. This fact may be demonstrated by the fact that neither definition incorporates the value tn on which fpr relies. The net result is that *precision* and *recall* may be used to compare the performance of filters on a particular test, but they yield little insight as to the effectiveness of the filter for its intended purpose, and none as to the relative performance on a different test with different spam prevalence. On the other hand, if the prevalence of spam in the test is known, or if ham recall is known, it is possible to deduce tn and hence fpr and the other diagnostic test measures.

The harmonic mean of recall and precision, $F_1 = \frac{2 \cdot recall \cdot precision}{recall + precision}$, is a one-dimensional measure commonly used to rank spam filters [180], more generally, $F_\beta = \frac{(1+\beta) \cdot recall \cdot precision}{recall + \beta \cdot precision}$. *Recall-precision curves* and *average precision* [185] are direct analogues of ROC and AUC, substituting precision and recall for fpr and fnr. The utility of these precision-based measures (Table 4.2) for spam filter comparison is questionable [42].

4.6.5 Cost-Sensitive Measures

Perhaps the simplest evaluation measures are $accuracy = \frac{fn+fp}{fn+fp+tn+tp}$ and $error = 1 - accuracy$. Although accuracy and error are poor measures [153, 42] they are commonly optimized and reported. *Weighted accuracy*, $WAcc = \frac{fn+\lambda \cdot fp}{fn+tp+\lambda \cdot fp+\lambda \cdot tn}$ for some $\lambda \geq 1$ models the assumption that false positives are more important than false negatives, weighting them accordingly [7]. It is not obvious how to choose an appropriate value of λ [83], values of $\lambda \in \{1, 9, 99, 999\}$ are typically reported, though the larger values have been found to be less useful [7]. One issue with accuracy, and weighted accuracy in particular, is that a vacuous filter — one that always yields the same classification — can receive a high score. For example, a filter that labels *all* messages as spam would yield $accuracy = 0.95$ if the prevalence of spam processed by the filter was 0.95. For this reason, *total cost ratio* — the ratio of the filter's accuracy to that of the vacuous filter — is often reported instead.

Weighted accuracy, like accuracy, depends on threshold setting as well as the prevalence of spam. While these parameters may be learned [108], there is substantial risk that evaluations based on accuracy or weighted accuracy will be confounded by arbitrary parameter settings that happen to work well (or poorly) for the data.

Accuracy and weighted accuracy are linear utility measures, defined more generally as $utility = \frac{w_{tn} \cdot tn + w_{tp} \cdot tp + w_{fn} \cdot fn + w_{fp} \cdot fp}{tn+tp+fn+fp}$, where w_{tn}, w_{tp}, w_{fn}, w_{fp} are real-valued coefficients that reflect the utility and costs of correctly and incorrectly classifying messages of each class. General utility measures have been reported for text classification (cf. [153]) but not, to date, for spam filtering.

It is possible that, even within a particular class, messages have different importance and hence different utility and cost associated with their correct and incorrect classification. Estimating the cost for every message presents a considerable challenge, and the literature reports such an effort only for limited experiments on a private collection [96]. It is possible, but labor intensive, to classify the messages into several genres, and to compute an evaluation measure on each genre separately [40, 42, 168].

4.6.6 Precision of Measurement

The purpose of measuring the effectiveness of a filter in a particular experiment is to predict its effectiveness in different situations. The accuracy of this prediction depends on two factors: the degree to which the circumstances of the experiment model those of the situations to be predicted, and the degree of chance error in the measurements. The first issue has been addressed in the preceding sections. The role of chance in measurement is addressed by the techniques of statistical inference.

Associated with any measurement is the notion — either tacit or explicit — of precision: the extent to which the measurement is free from chance variation. If we were told that a person's height is 173 cm, we would not be surprised to learn that person's true height was 172.6 cm, or 173.3 cm, because the notation used implies that the measurement was taken to the nearest cm. If we are told that another person's height is $5'8''$, any true height between about $5'7\frac{1}{2}''$ and $5'8\frac{1}{2}''$ would be unsurprising. Suppose we were asked, "who is taller?" An appropriate answer might be that it is impossible to tell because the given measurements lack the precision necessary to answer the question. Or that, for practical purposes, they are the same height because the difference is too small to matter. Yet the specious answer — that the first person is taller because $173\,\text{cm} > 5'8'' = 172.72\,\text{cm}$ — is common.

The measures used to evaluate spam filters, while more abstract, are subject to the same considerations regarding precision of measurement.[4] Whenever a measure is stated, consideration must be given to the precision of the estimate. Whenever measured results are compared, consideration must be given not simply to which is bigger, but to the magnitude of the difference, and whether or not the difference matters. For any statistical measure, best practice is to state the measured value along with a *confidence interval* consisting of a *lower confidence limit* and an *upper confidence limit* as well as a *degree of confidence* (cf. [41]). In comparing two measurements x and y, the most useful information is conveyed by reporting the value of the difference $d = x - y$ along

[4] Note that "precision" of measurement is unrelated to the IR measure of the same name.

with a confidence interval for d. Significance tests convey strictly less information than confidence intervals; "significant" means that the confidence interval for d does not include 0, "not significant" means that it does. Significance tests are often mistakenly interpreted as being properties of the difference rather than of the measurement, "not significant" is misconstrued as "no difference" or "unimportant difference" while "significant" is misconstrued as "important." In our height example, the difference in measurements tells us nothing about which individual is taller, yet we know that no two individuals have *exactly* the same height. A more precise measurement (perhaps a back-to-back differential measurement) might answer the question — at one particular moment in time — but that does not mean that the difference in height would be of any consequence. While significance tests are common, many methodologists advocate that they be supplanted by the measurement of differences [139].

Until recently, the majority of experiments in spam filtering and related fields have reported neither confidence intervals nor significance results, substantially compromising the value of those results. Dietterich [48] discusses the use of significance tests for cross-validation results. Cormack and Lynam discuss confidence intervals and significance tests for the measures presented here, and estimate confidence intervals for published studies that originally lacked them [42]. The TREC spam filter evaluation toolkit contains software to compute the major test-based measures with 95% confidence intervals.

5

Results and Benchmarks

A systematic comparison of evaluation efforts must reconcile their diverse objectives, methods, datasets, and measures. The following sections first review the major laboratory evaluations to date, followed by field evaluations. We report results only when they may be recast as (fpr, fnr) or AUC. Where AUC is not reported directly, it is estimated under the assumption that lam is constant (Section 4.6.2). Where the results of several methods are reported using the same data, methods and measures, they give some indication of the relative performance of the filters. Where the results of the same methods are reported using different data, they give some indication of the "difficulty" of the data which can facilitate indirect comparisons between methods tested on only one or the other.

To facilitate intuitive comparison, Table 5.1 illustrates summary measures estimated from typical values of lam:

- $fnr(\%)$ at $fpr = 0.1\%$ is the false negative rate when the threshold is adjusted so as to yield a false positive rate of 0.1%, that is, one good email per thousand is misclassified,

Table 5.1 Equivalent measures for fixed *lam*.

lam	$fnr(\%)$ at $fpr = 0.1\%$	$fnr(\%)$ at $fpr = 1.0\%$	$(1 - AUC)(\%)$
0.3	1.0	0.1	0.01
0.6	3.0	0.3	0.03
1.0	10.0	1.0	0.08
2.0	30.0	4.0	0.3
6.0	80.0	30.0	2.0
10.0	92.0	55.0	4.0

- $fnr(\%)$ at $fpr = 1.0\%$ is the false negative rate when the threshold is adjusted so as to yield a false positive rate of 1%, one good email per hundred is misclassified,
- $(1 - AUC)(\%)$ is the area above the ROC curve.

5.1 Prototypical Studies

Sahami et al. [141] report early experiments on the use of Naive Bayes spam filters, using email collected from a user's inbox. Sahami et al. use a total of 1789 messages — 1578 spam and 211 non-spam — split chronologically into training and test sets with 1538 and 251 messages, respectively. Three feature engineering methods are evaluated: words alone, words plus 35 hand-crafted phrases, words plus phrases plus 20 hand-crafted non-textual patterns thought to indicate spam. Feature selection is effected using mutual information, resulting in 500 features. Recall and precision are reported, considering both spam and non-spam to be the positive class, from these statistics we may deduce *fpr*, *fnr*, and *lam*, summarized in Table 5.2. To test generalizability of the result, the same method was trained on all the email received by an individual over an interval of time, and tested on email received by the same interval in the subsequent week. These results, shown in Table 5.2, are substantially worse than those achieved on the collected messages.

Table 5.2 Results by Sahami et al.

Data	*fpr* (%)	*fnr* (%)	$(1-AUC)(\%)$ [est.]
collected	0	1.7	0.1
live	1.7	20	2.0

The general approach — selecting a small set of available messages for evaluation — is common to many studies. It assumes that the messages are representative of those that might be filtered, an assumption that was tested, and found not to hold, by comparison with a more realistic sample. The small test set size yields large confidence intervals, and results might not be indicative of those that would be achieved with a larger one. The results suggest that a spam filter employing this method would have too high an error rate to be useful for its intended purpose.

Drucker et al. [53] consider the use of support vector machine, boosted C4.5 decision tree, RIPPER and Rocchio classifiers for spam filtering. A dataset of 850 spam and 2150 non-spam messages collected from a single user, and a second dataset of 314 spam and 303 nonspam collected from colleagues were used in tenfold cross validation. The effects of many parameters were investigated, with results reported as the *fpr* achieved with the threshold adjusted so as to achieve a specific value of *fnr* (1% or 5%).

This study is one of the first and most comprehensive studies of spam filtering techniques, considering a vast number of feature engineering methods, learning methods, and tuning parameters. While the precision of measurement is inadequate to yield final definitive answers, the Drucker and Vapnik offer the following observations which have yet to be contradicted:

- SVM and boosted trees have comparable accuracy, but SVM is better able to achieve low false positive rates,
- RIPPER and Rocchio are not competitive,
- boosted trees require inordinately long training time,
- large values of C work better for SVM, reaching a plateau at $C \geq 50$,
- binary features work better for SVM than term frequencies; term frequencies work better for boosted trees,
- feature selection causes undesirable overhead and is best considered part of the learning algorithm,
- stop words should not be excluded,
- SVM does not benefit from term selection.

Table 5.3 Drucker et al. Results $(1-AUC)(\%)$ [est.].

Method	Stopwords	No stopwords
SVM (TF)	0.81	0.56
SVM (binary)	0.93	0.50
Boosting (TF)	0.60	0.42
Ripper	2.15	2.00
Rocchio (TF-IDF)	2.32	1.29

Table 5.3 summarizes the primary results.

5.2 The Spambase Public Dataset

Spambase [165] was the first public spam test collection. As part of the UCI Repository of Machine Learning Databases, it has been used as an example in studies of a vast number of general machine learning algorithms and classifiers (e.g., [126, 128, 169]). Feature-vector representations for each of 1813 spam and 2788 nonspam messages, but not the messages themselves, are included in the dataset. Each vector contains 57 real-valued attributes indicating the frequencies of certain words and characters, as well as lexical characteristics such as the average length of same-character runs and the number of capital letters in the message. Due to privacy concerns, no further information concerning the messages is available. This terse vector representation removes in large part the information that contemporary spam filters use to classify messages, calling into question the validity of Spambase-derived results [85, 202] as a measure of spam filter effectiveness.

5.3 The Ling Spam Corpus

The Ling Spam corpus [4, 112] consists of several abstracted versions of 841 spam email messages combined with 2412 non-spam messages sent to a mailing list. Except for the subject line, header information is removed. Markup was eliminated and all words were converted to lower case. In variants of the corpus stop words were removed or words were replaced by their roots. Duplicate messages were removed. Standard splits are identified so as to facilitate comparable evaluations using tenfold cross validation.

Although spam and nonspam messages are drawn from disparate sources, and information useful to some filters is removed, Ling Spam represented a much more realistic evaluation suite than any available at the time of its release, and a large number of studies have used it for evaluation. The results of these studies are typically expressed in terms of precision and recall without statistical confidence intervals. It is possible, using these statistics and the number of spam and nonspam messages in the corpus, to deduce the contingency table so as to compute (*fpr*, *fnr*) and estimate AUC. Some studies report only accuracy, weighted accuracy, or total cost ratio alone (e.g., [201]), from which it is impossible to recover *fpr* and *fnr*, the results of studies are excluded from presentation. Other studies report ROC curves which represent a range of possible (*fpr*, *fnr*) pairs.

Figure 5.1 plots the results for studies from which it is possible to derive one or more (*fpr*, *fnr*) pairs or an ROC curve. Results reported as ROC curves appear as curves. Table 5.4 relates the labels on the points and curves to particular studies and methods.

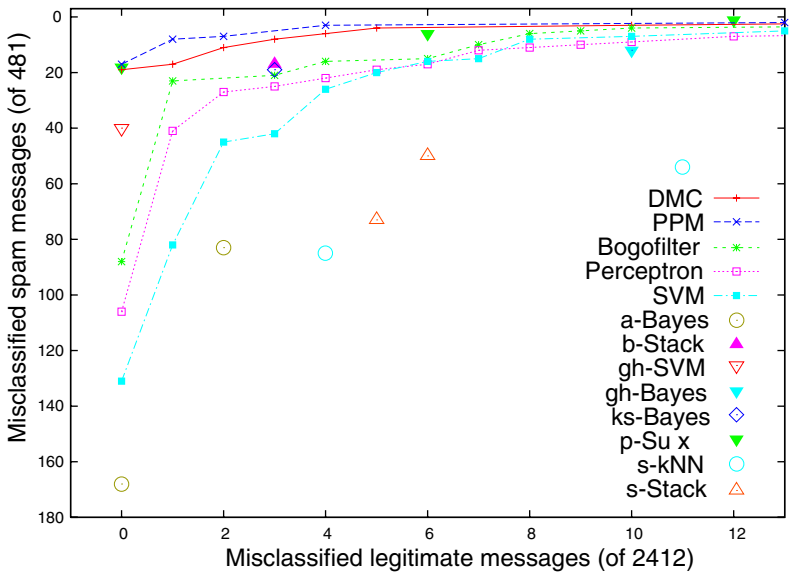

Fig. 5.1 Ling Spam results.

Table 5.4 Caption key for Ling Spam and PU results.

Label	Description
a-Bayes	Naive Bayes, multi-variate Bernoulli model on binary features [4].
a-FlexBayes	Flexible naive Bayes — uses kernel density estimation for estimating classconditional probabilities of continuous valued attributes [7].
a-LogitBoost	LogitBoost (variant of boosting) with decision stumps as base classifiers [7].
a-SVM	Linear kernel support vector machines [7].
b-Stack	Stacking of linear support vector machine classifiers built from different message fields [18].
Bogofilter	Version 0.94.0, default parameters [134].
c-AdaBoost	Boosted decision trees with real-valued predictions [24].
DMC	Dynamic Markov Compression [16].
gh-Bayes	Naive Bayes (exact model unknown) with weighting of training instances according to misclassification cost ratio [83].
gh-SVM	Linear support vector machine with weighting of training instances according to misclassification cost ratio [83].
h-Bayes	Multinomial naive Bayes [88].
ks-Bayes	Multinomial naive Bayes [146].
p-Suffix	Pattern matching of character sequences based on suffix tree data structure and heuristic scoring functions [127].
m-Filtron	Support vector machines with linear kernels [122].
s-kNN	k-nearest neighbors with attribute and distance weighting [143].
s-Stack	Stacking of naive Bayes and k-nearest neighbors [142].
SVM	Support vector machine, linear kernel with $C = 1$ [175].
Perceptron	Perceptron algorithm [175].
PPM	Prediction by Partial Matching [16].

The format of the messages in Ling Spam renders them unsuitable as input for many spam filters. Cormack and Lynam applied three popular filters directly to the Ling Spam messages, observing pathologically poor performance (Table 5.5). However, with suitably reformatted messages, one of these filters (Bogofilter [134]) shows competitive performance [36].

Table 5.5 Real filter results on Ling Spam corpus.

Filter	*fpr(%)*	*fnr(%)*	*(1−AUC)(%)*
SpamAssassin [164]	0	95.4	n/a
Bogofilter [134]	0	59.5	n/a
SpamProbe [21]	0	31.3	n/a
CRM114 [195]	54.9	11.2	22.0

5.4 PU1, PU2, PU3, and PUA Corpora

The PU1, PU2, PU3, and PUA corpora [122] differ from Ling Spam in two substantial ways, but are otherwise similar:

- they are derived from real email messages sent to individuals,
- to obfuscate the content of these private messages, they are abstracted further by replacing each distinct word by an arbitrarily chosen integer.

Headers (except for the subject) and duplicate messages were removed as per Ling Spam, variants employing stop words and stemming are available for the PU1 corpus only.

The corpora themselves, as well as tools to do the abstraction, are available on the internet [112]. PU1 was created first, and is used in a number of studies. PU3 is substantially larger than PU1; PU2 and PUA are small and seldom cited. Figures 5.2 and 5.3 give the results of studies using PU1 and PU3 for which spam and nonspam misclassification error counts may be deduced.

It is perhaps interesting to note that filters based on sequential compression models show superior performance on these corpora, although one might expect them to be disadvantaged by obfuscation. Bogofilter also shows reasonably good performance [36] when the numbers representing words are transliterated to an alphabetic representation.

5.5 The Enron-Spam Corpus

The *Enron-spam* corpus [112] (not to be confused with the Enron email dataset [94] or the trec05p-1 corpus) is a successor to the Ling Spam and PU corpora that contains chronological splits of email messages received by six Enron employees combined with spam from various sources. The corpus consists of a suite of six pairs of training sets and test sets, in which the messages in each training set predate those in the corresponding test set. Two versions of the corpus are provided, one preprocessed in the same manner as Ling Spam, the other containing raw messages. The corpus has been used by its creators to study feature representation for Bayesian spam filters [121].

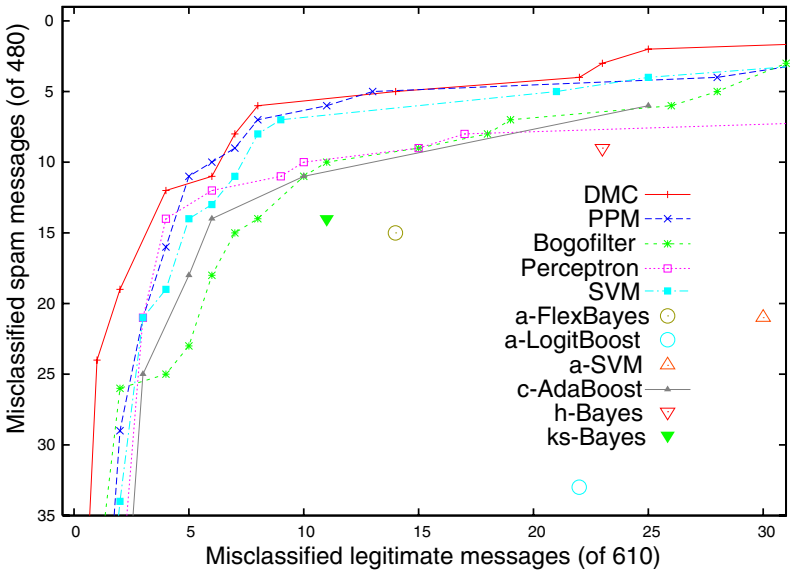

Fig. 5.2 PU1 results.

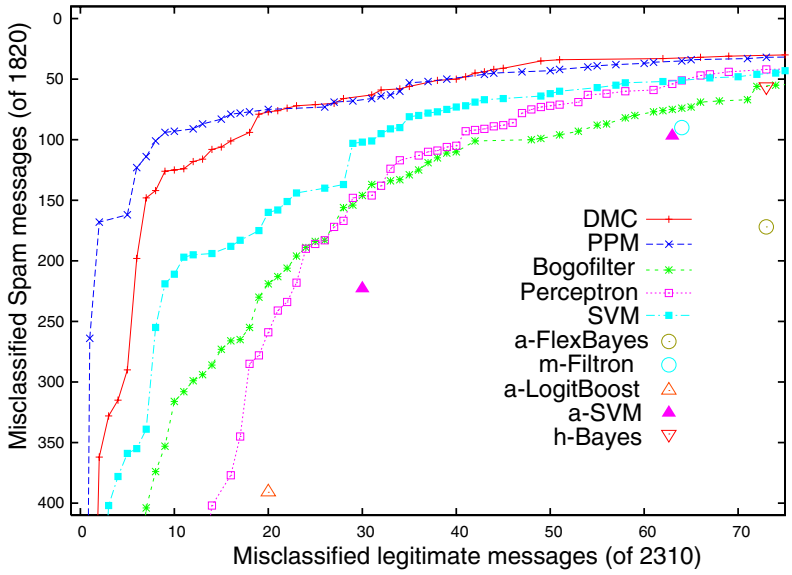

Fig. 5.3 PU3 results.

5.6 SpamAssassin Corpus

The SpamAssassin corpus [163] consists of 4147 real email messages, 31% spam, with complete headers. The messages come from several sources, including mailing lists and messages sent to and by SpamAssassin developers. Some host names are altered for privacy, in most cases the messages appear exactly as received. The corpus has been used, for the most part, for the development of SpamAssassin and other practical spam filters. A standard evaluation method using the corpus has not evolved; studies have reported different measures derived from tenfold cross validation [86, 201] (see Cormack and Lynam [42] for comparison). On-line evaluation has been investigated using random [196] and chronological [114] sequences of email. A chronologically ordered version of the SpamAssassin corpus suitable for on-line evaluation using the TREC method is a toolkit.

5.7 TREC Spam Track

The TREC Spam Track is the largest and most realistic laboratory evaluation to date. In the three years it ran, ten test corpora with a total of 721,461 messages were used to test filters submitted by 35 participants. Each participant submitted up to four different filters, and these filters were the subject of various experiments using the corpora. The results of more than one thousand experimental runs may be found in the summaries and appendices of the TREC proceedings [40, 33, 34]. Four of the ten corpora are available on the web for free download, subject to a click-through usage agreement [172, 173, 174].

The core task at TREC — the *immediate feedback* task — simulates the on-line deployment of a spam filter with idealized user feedback. Variants of the task, including *delayed feedback*, *partial feedback*, and *active learning* explore the impact of imperfect or limited user feedback.

The core task, along with methods and evaluation measures embodied in an open-source toolkit [113], was developed by Cormack and Lynam [42] prior to TREC 2005, and run on a corpus consisting of 49,086 email messages captured for the purpose. This corpus, which came to be known as the *Mr X* corpus, was also used for TREC 2005,

along with three new corpora: *tm, sb*, and *trec05p-1. tm* and *sb* were, like *Mr X,* private corpora formed by capturing and labeling all the email sent to an individual during an interval of many months.

5.7.1 TREC Corpora

trec05p-1, the TREC 2005 Public 5.6 Corpus, contains the stored email of 150 Enron executives, collected and released to the public domain as a result of the US Federal investigation of Enron's collapse. These messages were labeled and augmented by additional spam messages from a public source [78], altered to appear to have been delivered to Enron during the same period [40].

TREC 2005 established that private and public corpora could yield reasonably consistent results, although some of the corpora were more "difficult" than others, the results and especially the relative results were not dramatically different — well within differences that could be explained by chance.

TREC 2006 saw the creation of two new private corpora — *Mr X2* and *sb2*, captured later from the same sources as *Mr X* and *sb*, as well as English and Chinese public corpora — *trec06p* and *trec06c*. trec06p was captured from the Web and augmented with public spam, while trec06c was captured from a mailing list and augmented with spam from a honeypot at the same site.

TREC 2007 used one private corpus — *Mr X3* — and one public corpus — *trec07p*. The *Mr X3* corpus contains 161,975 messages, as opposed to *Mr X's* 49,086. The difference is due entirely to an increase in the proportion of spam received from 82% to 95% over the interval 2004–2007. trec07p contains exactly the messages delivered to a particular server over three months.

5.7.2 TREC Variants

Four variants — *ham25, ham50, spam25*, and *spam50* — are included with the trec05p-1 distribution. Each specifies a random sub-sample of either spam or non-spam, so as to measure the sensitivity of filter performance to the proportion of spam.

TREC 2006 introduced the *delayed feedback task* which entailed extensions to the toolkit [39] to specify exactly the interleaving of classify and train operations. A standard delayed interleaving for trec05p-1 is available with the revised toolkit, the trec06p and trec06c distributions specify the interleaving used in the delayed feedback task at TREC 2006.

TREC 2006 also introduced a *(batch) active learning task* in which filters were presented with an unlabeled training set and were permitted to use the results of k queries of the form $label(m)$, for some quota k. Tools to implement (batch) active learning are included in the toolkit.

Cormack and Bratko [36] use the delayed feedback extensions to specify standard splits for tenfold cross validation using trec05p-1, and also a single 9:1 chronological split suitable for testing non-adaptive filtering methods.

Sculley [147] proposed the TREC 2007 *on-line active learning task* and extended the toolkit to implement it. TREC 2007 also incorporated two variants of delayed feedback: (*extreme*) *delayed feedback* in which feedback is given immediately for the first few thousand messages and never given for the rest, *partial feedback* in which feedback is given only for messages addressed to a subset of the recipients. Standard interleavings for (extreme) delayed feedback and for partial feedback are distributed with the trec07p corpus.

The 2007 CEAS Live Spam Challenge [170] altered the TREC methodology in several ways:

(1) A stream of email was captured and distributed to participants in real-time.
(2) The stream was communicated to filters using SMTP instead of a command-line interface.
(3) Filters were required to return only a hard classification.
(4) Filters were permitted unrestricted access to external resources such as name servers and blacklists.

Except for these differences, the Live Spam Challenge paralleled the TREC delayed feedback task. The messages came from the same source as trec07p, and were preserved so as to support direct comparison with

laboratory filters using the TREC toolkit. Due to technical difficulties with the SMTP interface, the test was abbreviated and some filters were unable to operate properly. At the time of writing, neither the corpus nor the results have officially been released, but plans are underway to resolve the technical difficulties and reprise the Live Spam Challenge in 2008.

5.7.3 TREC and Associated Results

Results related to the TREC corpora and methodologies are discussed below. AUC measures derived from the three TREC public corpora, conducted at TREC and elsewhere, are summarized in Table 5.6; results derived from the three *Mr X* corpora are summarized in Table 5.7.

Cormack and Lynam [42] describe and justify the on-line evaluation methods used at TREC, the creation of the Mr X corpus and the results of testing several popular spam filters, including Bogofilter, SpamProbe, SpamBayes, and SpamAssassin, all of which include a "Bayesian" classifier implementing Robinson's χ^2 learning method. SpamAssassin also incorporates a hand-crafted component, whose effectiveness was measured separately and in conjunction with the learning component. All the learning filters demonstrated similar performance.

Table 5.6 $1-AUC(\%)$ — TREC public corpora.

Method/Features	Refs.	trec05p-1	trec06p	trec07p
Fusion	[33, 114]	0.007	0.020	
SVM/4-gram	[150]	0.008	0.023	
ROSVM/4-gram	[150, 151]	0.009	0.024	0.009
DMC+LR/4-gram	[35]			0.006
LR/4-gram	[35]	0.012		0.006
DMC	[16, 35]	0.013	0.033	0.008
Perceptron/string	[152]	0.017	0.041	
PPM	[16]	0.019	0.061	0.011
Clustering/string	[125]			0.011
OSBF-Lua	[10]	0.019	0.054	0.028
LR/words	[68]	0.022		
Bogofilter	[40]	0.048		
SpamAssassin "Bayes"	[40]	0.059		
LR/words	[36]	0.068		
SVM/words	[36]	0.075		
SpamAssasin Rules+Bayes	[40]	0.345		

Table 5.7 $1-AUC(\%)$ — Mr X corpora.

Method/Features	Refs.	Mr X	Mr X2	Mr X3
ROSVM/4-gram	[150, 151]			0.004
DMC+LR/4-gram	[35]			0.008
LR/4-gram	[35]	0.03		0.010
Clustering/string	[125]			0.015
DMC	[16, 35]	0.04	0.05	0.022
Perceptron/string	[152]	0.04	0.06	
LR/words	[68]	0.05		
Bogofilter	[40]	0.05	0.09	
OSBF-Lua	[10]	0.07	0.05	0.028
SpamAssassin Rules + Bayes	[42]	0.06		
PPM	[16]	0.08	0.08	0.023
SpamProbe	[42]	0.09		
SpamBayes	[42]	0.16		
SpamAssassin "Bayes"	[42]	0.15		
SpamAssassin Rules	[42]	0.80		
Human + filter	[42]	0.88		
DSPAM	[42]	1.03		
CRM114	[42]	1.10		

SpamAssassin's hand-crafted component showed inferior performance to the learning component, while the combination of learning and hand-crafted was marginally superior. Two other learning filters — CRM114 and DSPAM — were included in the study, showing inferior performance. Cormack and Lynam show ROC curves for all results and report $1 - AUC$ as well as *fpr* and *fnr* with 95% confidence limits. In addition, misclassified spam and non-spam are categorized into genres indicating the general nature of the messages.

The full TREC results occupy nearly one hundred pages of appendices in each year's proceedings [182, 183, 184]. We summarize noteworthy results, in particular those that are particularly strong and those that provide baselines for comparison. Perhaps the most notable and surprising result from TREC 2005 was the commanding performance of the PPM-based filter due to Bratko and Filipic [17]. Bogofilter and the learning component of SpamAssassin performed well, but somewhat more poorly than PPM. SpamAssassin's combined learning and hand-crafted components gave insubstantially different results from the learning component alone, superior on two corpora and inferior on two. Based on these results, we chose PPM and Bogofilter as standards for comparison.

Following TREC 2005, the PPM results were independently reproduced and bettered using the Dynamic Markov Compression method [16, 38]. The authors observe that using a 2500 byte prefix of each message, as opposed to the entire message, yields both increased efficiency and increased effectiveness.

It is notable that all the well-performing methods at TREC 2005 were based on what are considered fairly weak methods in the machine learning community. In response, Goodman and Yih [68] devised a very simple on-line logistic regression method that showed strong performance on the trec05p-1 and Mr X corpora, bettering the TREC results for all but PPM.

At the same time, Cormack and Bratko [36] investigated the thesis that sophisticated learning methods were not represented at TREC due to the challenges of feature engineering and on-line deployment. They found:

(1) Substantially different results were obtained using on-line evaluation, tenfold cross validation and a chronological split of the same data (the trec05p-1 corpus).
(2) Under tenfold cross-validation, Bogofilter's performance about equaled logistic regression and a support vector machine (linear kernel, $C = 1$) using binary bag-of-words features. Compression methods were superior.
(3) Using a chronological split with the first 90% of the corpus as training examples and the last 10% as test examples, the compression methods strongly outperformed Bogofilter, logistic regression and the support vector machine.
(4) Compression methods showed dramatically superior performance to that of on-line versions of logistic regression and SVM constructed using batching and a sliding window. Bogofilter's performance was in between.
(5) Compression methods compared favorably with any reported in the literature, when compared using tenfold cross validation on the Ling Spam, PU1, PU2, and PU3 corpora (cf. Sections 5.3, 5.4). Bogofilter, logistic regression and support

> vector machine showed competitive performance, generally inferior to that of the compression methods.

Sculley and Brodley [148] point out that data compression classification methods may be viewed as learning methods with a feature space consisting of all substrings of the message.

Lynam and Cormack [114] used several on-line meta-classifiers to combine the results of all 51 filters evaluated at TREC 2005. Even the simplest method — voting — produced a substantial improvement over the best filter. The most sophisticated method — logistic regression on the log-odds-transformed soft results of the individual classifiers — produced the best results reported to date for trec05p-1.

TREC 2006 saw a general improvement in filter effectiveness, with several approaches equaling or bettering that of PPM. OSBF-Lua [9] due to Assis [10] is perhaps the best overall performer in the core task. OSBF-Lua uses orthogonal sparse binomial word features [158], iterative threshold training, and a weighted average scheme for combining frequency estimates. Sculley et al. [152] show strong performance combining inexact string matching for tokenization with the margin perceptron learning algorithm.

Filter performance was generally worse for the delayed learning task, no participant investigated a technique to mitigate this effect. Notable results in the active learning task were achieved by Bruckner et al. [20] using orthogonal sparse bigrams, Winnow, pre-training, and threshold training to achieve strong initial performance in the (batch) active learning task. Bratko et al. [16] addresses the active learning problem by using only the most recent training data. The rationale for this choice is the observation that on-line classifiers work much better on chronologically ordered data, hence more recent training examples should be more apt.

Sculley et al. [150] investigate the use of a relaxed on-line support-vector machine (ROSVM) for spam filtering using TREC and other spam corpora. The best performance is yielded by a configuration using binary features derived from character 4-grams and an adaptive on-line SVM implementation with a linear kernel and $C = 100$. Effectiveness on the trec05p-1 and trec06p corpora was nearly as good as that of

the fusion methods due to Lynam and Cormack — much better than any single spam filter to date. A linear-time variant, using a sliding window and relaxed optimization criteria, reduces execution time on the 92,000-message trec05p-1 corpus by an order of magnitude — to a fraction of a second per message.

At TREC 2007, ROSVM [151] showed the best results on *Mr X3* and second best results on *trec07p* in the core (immediate feedback) task. Insubstantially different results were shown by Cormack's method fusing DMC with on-line logistic regression using 4-gram features [35], which was second-best on *Mr X3* and best on trec07p. In isolation, the logistic regression component fared better than DMC and nearly as well as the combination. Strong results were obtained by Niu et al. [125] using clustering and and on-line character-based linear classifier. Yerazunis [198] achieved similar effectiveness using OSB features and iterative threshold training.

The delayed feedback and partial feedback tasks showed decreased filter performance — the partial feedback task remarkably so (see Sculley et al. [151] for discussion). The on-line active learning task was dominated by versions of ROSVM and on-line logistic regression adapted to use uncertainty sampling: filters requested the label for any soft classification result falling within some fixed distance of the hard classification threshold t.

5.8 ECML/PKDD Discovery Challenge

The TREC partial feedback task is an on-line adaptation of the 2006 ECML/PKDD Discovery Challenge which had previously addressed similar issues in a batch context. The purpose of the Challenge was to model spam filter deployment in which neither feedback nor user-specific training data is available. Task A and Task B both involved training on one or more labeled mixtures of spam and non-spam from distinct sources, and then classifying several mixtures of spam and non-spam from different distinct sources. Task A used a single training set with 4000 messages, and three test sets with 2500 messages each. Task B used three training sets with 100 messages each, and 15 test sets with 400 messages each. Messages were represented as term frequency

Table 5.8 1−AUC(%) — ECML/PKDD Discovery Challenge Task A.

Method	Refs.	Task A
SVM (unofficial)	[205]	3.28
Bayesian	[93]	4.93
Graph	[129]	5.09
Bayesian	[80]	5.13
Positive only	[176]	6.35

Table 5.9 $1-AUC(\%)$ — ECML/PKDD Discovery Challenge Task B.

Method	Refs.	Task B
DMC	[32]	5.35
SVM (unofficial)	[205]	6.16
Positive only	[176]	8.18
Bayesian	[80]	9.26

vectors, absent terms occurring in fewer than four messages. Neither the words themselves nor the text of the messages were available to filters. The test corpora are available for free download [56].

Participants were evaluated in terms AUC (see Tables 5.8 and 5.9 for results whose methods are described in the literature). A number of semi-supervised and transductive methods have been applied to the problem, both within the context of the Challenge and after. The joint winners (because their scores were not separated with 95% confidence) of Task A were:

- A two-pass supervised and semi-supervised Bayesian classifier by Junejo et al. [93].
- An efficient adaptation due to Pharinger [129] of the graph-based method of Zhou et al. [203].
- A Bayesian method by Gupta et al. [79].

The sole winner of Task B iteratively applied DMC with self-training to a string representation of the feature vectors [32]. Runner-up in both tasks was a method of training only on positive examples by Trogkanis and Paliouras [176]. Zien, and unofficial participant, achieved strong results with a semi-supervised support vector machine [205].

Bickel and Scheffer [13] describe the foundations of the biased sample problem underlying the Challenge and a hierarchical Bayesian model to address it. Bickel et al. [12] present a discriminative approach

to the problem. It remains to be seen how well these methods may be applied to the problem of on-line spam filtering.

5.9 Gen Spam

Gen Spam is a newer corpus [120] consisting of two sets of real email, with header information reformatted in XML and identifying information like email addresses and hostnames obfuscated. Otherwise, the content of the messages is preserved. One set of email is from a different source than the other. The first, consisting of 38,246 messages, is used as a training set. The second is split into an adaptation set with 600 messages and a test set with 1551 messages. The intended use is to model spam filter deployment in which a large amount of generic data (the training set) is available for training, and a smaller amount of recipient-specific data (the adaptation set). Table 5.10 compares the results of several filters on the Gen Spam corpus.

5.10 Product Evaluation Reports

Product evaluation reports are typically the subject of analyst reports, magazine articles, and advertising. Although they are not subject to scientific peer review, they may have considerable influence, this section reviews the most visible product evaluation efforts.

5.10.1 Network World Test

Snyder [160] conducted a parallel test of 41 commercial spam filters. Over a two-week period the message traffic to a server was captured, screened for viruses and backscatter, and passed on to the spam filters under test. 11,000 passed the screening process and were then passed to each of the filters for classification.

Table 5.10 Gen Spam corpus results.

Method	$(1 - AUC)(\%)$
Bogofilter	0.44
DMC	0.03
Medlock [120]	0.48
PPM	0.05

During the test, filters were connected to the internet and allowed to interact with external resources, for example, to receive updates. No user feedback was provided.

Later, the messages were adjudicated and 8027 spam and 2386 non-spam messages were selected as an evaluation set, 600 (5.5%) were discarded for the following reasons:

- Messages with viruses, not caught in screening.
- Backscatter not caught in screening.
- Messages with duplicate message identifiers.
- Messages for which the adjudicators were unable to determine the correct classification.

Snyder incorrectly reports "diluted" $fpr = \frac{|non\text{-}spam \cap \{m|c(m)=true\}|}{|non\text{-}spam|+|spam|}$, which we correct to $fpr = \frac{|non\text{-}spam \cap \{m|c(m)=true\}|}{|non\text{-}spam|}$ in Table 5.11.

Strengths. This study approximates many aspects of the actual deployment of spam filters more closely than laboratory investigations. Among field investigations it stands out in measuring appropriate statistics in a controlled environment. The methodology has been adopted by others [162].

Limitations. Commercial filters are tested as black boxes, so it is difficult to generalize beyond the particular versions tested. Filters requiring user feedback were excluded. The environment is not reproducible and therefore not amenable to further testing. The screening process introduces delay that may affect filter performance. The screening and

Table 5.11 Network world results.

Vendor	*fpr* (%)	*fnr* (%)	$1-AUC$(%) [est.]
BorderWare	0.12	1	0.01
Postini	0.24	3	0.06
CipherTrust	0.36	3	0.09
Symantec	0.48	3	0.12
Advascan	0.57	3	0.14
NetCleanse	0.46	5	0.17
Proofpoint	0.87	3	0.18
Barracuda	0.9	5	0.32
Cloudmark	1.05	5	0.36
Spamfighter	1.02	14	0.91

selection processes introduce substantial selection bias, as the proportion of discarded messages substantially exceeds measured error rates and these messages might be more likely to cause difficulties to filters. Internet connectivity during the course of the test may be a source of interaction among filters, or between filters and vendors whose actions may confound the test. Sample size yields insufficient precision to differentiate the better filters. Mischaracterization of *fpr* in reported results may encourage invalid comparison with other work.

5.10.2 Veritest Anti-Spam Benchmark Service

Veritest [181] is a commercial test service that evaluates filters by installing them in a test environment and sending them a stream of spam mixed with simulated non-spam messages, reporting *tpr* and diluted *fpr* (labeled *false positive percentage* or *false positive rate*, see Table 5.12 for corrected results). Filters are tested four times per year, concurrently, over a 13 day period. Spam messages are captured from long-standing spam traps: accounts and domains that were created for the purpose or have been abandoned and no longer receive legitimate email. Non-spam messages come from two sources: legitimate subscription-based sources like mailing lists and news services, and messages sampled from real business and personal correspondence.

Messages are forwarded to particular filters under test by repeatedly altering the DNS (dynamic name server) entries that map hostnames (e.g., xxx.yyy.com) to IP addresses (e.g., 111.222.000.123). Thus, mail addressed to a particular host is sent to one IP address — the IP address corresponding to a particular filter — for some parts of the test and to others at other times. Each message is sent to exactly one filter, but

Table 5.12 Veritest Results — Autumn 2005.

Vendor	*fpr* (%)	*fnr* (%)	$1-AUC$(%) [est.]
Message Labs	0.3	0.4	0.01
Solution A	0.1	1.9	0.02
Solution B	1.3	4.5	0.40
Sophos	1.0	1.4	0.11
Tumbleweed	0.3	0.5	0.02

the assignment of hostnames to filters is rotated so each filter receives overall a set of messages with comparable characteristics.

Filters typically receive about 10,000 messages each — 6,000 spam and 4,000 non-spam — during the course of the test. Each filter sorts messages into its own spam and non-spam folders, which are adjudicated. Messages which cannot be adjudicated as spam or non-spam are excluded from the results.

Opus One use the same methodology for ongoing commercial spam filter evaluation.

Strengths. Filters are tested live in a real-time environment. Spam is delivered directly from spam sources without delay or forwarding. Filters may employ sender interaction methods like greylisting and tarpitting [109].

Limitations. Spam sources may not adequately cover the range typical sources. Generated and subscription-sourced messages may not adequately represent the non-spam typically delivered to a user or set of users. User feedback is difficult to incorporate or simulate, as the message stream (and hence user) is artificial, and no gold standard is available for simulation. DNS updates, and delay in DNS propagation may introduce bias or artefacts in the streams of spam and non-spam delivered to filters. While sender interaction with spam senders is aptly captured, interaction with non-spam senders may not be, because the sender is simulated. In particular, false positives are likely to be underestimated. Sample sizes yield insufficient precision for fine comparison, especially since different filters are tested on different samples. Number of samples needed rises in proportion to number of filters tested. Exclusion of messages which cannot be adjudicated results in selection bias, non-blind evaluation of messages (i.e., previously sorted by a particular filter) may yield adjudication bias.

5.10.3 West Coast Labs Anti-Spam Test Methodology

West Coast Labs [189] performs comparative tests between pairs of spam filters inserted into a stream of email (see Table 5.13). The spam messages are those delivered to an established spam-trap domain, while the non-spam messages are manually created and sent from various

Table 5.13 West Coast Lab Results — January 2007.

Vendor	*fpr* (%)	*fnr* (%)	$1-AUC$(%) [est.]
BitDefender	0	3	n/a
Email Systems	0	1	n/a
Kaspersky Labs	0	3	n/a
MXSweep	0	1	n/a
SoftScan	0	3	n/a
Sophos	0	3	n/a
SurfControl	0	1	n/a

sources so as to resemble real email. The incoming mail stream is routed periodically to one filter or the other using NAT (Network Address Translation). Thus each filter receives messages in the stream about half the time, and no messages other times. A particular test runs for 10 days, with about 500 messages delivered to each filter per day. *tpr* and *tnr*, but not the prevalence of spam, were reported.

Strengths. The method intercepts a single stream of email, and NAT rerouting is immediate.

Limitations. The sources of spam may not be representative. A sample of 500 messages is too small to yield a nonzero *fpr* estimate. Hand-created non-spam may not be representative and is but one genre of legitimate message that is likely to occur in a real email stream. User feedback would be difficult to simulate. Filters see temporal gaps in message transmission. The method would be difficult to scale to test many filters, as the incoming stream must be multiplexed among them.

6

Discussion

By information retrieval standards, spam filters yield outstanding results. Laboratory testing shows that a content-based learning filter can correctly classify all but a few spam messages per hundred and all but a few thousand non-spam messages per thousand ($fpr \approx 0.2\%$, $fnr \approx 2\%$, $(1 - AUC) \approx 0.03\%$). There is some evidence that similar results may be achieved in practice either by machine learning methods or by other methods like blacklisting, greylisting, and collaborative filtering. The controlled studies necessary to measure the effectiveness of all types of filters — and combinations of filters — have yet to be conducted. We argue that understanding and improvement of the effectiveness of spam filters is best achieved by a combination of laboratory and field studies, using common measures and statistical methods. To this end, we have reviewed existing results with respect to common criteria and have outlined designs for future studies.

The TREC Spam Track offers standard methodologies, tools and datasets for large-scale spam filter evaluation. Each of the three years of the Spam Track saw substantial improvement in overall filter effectiveness. Bayesian filters using words as features were eclipsed by sequential compression models using no explicit features, and by on-line versions of

logistic regression and support vector machines using character 4-grams as features. Methods based on orthogonal sparse bigrams showed excellent results. The best known method for the TREC datasets is achieved by combining all other methods using logistic regression.

The Spam Track is a laboratory evaluation whose baseline task simulates on-line filter deployment with user feedback. Tests of common filters on public and private email collections indicate that results achieved on the public datasets predict well those on private ones representing real sequences of email. This observation suggests that the use of public corpora is an effective approach to identifying those approaches worthy of more costly testing on "real" data. Variants of the baseline task have shown that delayed, incomplete or incorrect feedback degrade filter performance somewhat. The TREC methods apply equally well to batch or real-time deployment. The extent to which these variants better model actual spam filter use remains a subject of future research.

The methods evaluated at TREC may be used as references for comparison in different evaluation contexts. The best methods compare favorably to all previously reported methods when evaluated using the Ling Spam, PU1 and PU3 corpora. We suggest that any future evaluation, whether laboratory or field experiment, should include one or more filters tested at TREC as a control.

The role of user feedback in spam filtering is the subject of some controversy. One point of view is that spam filters must operate without any feedback from the user, the opposing point of view is that user feedback is the most useful information from which to construct a classifier. Each approach involves tradeoffs that merit further evaluation. Can filters, without user feedback, achieve acceptable error rates? What are the hidden infrastructure costs of constructing and training filters that use no feedback? Can filters be designed to achieve good error rates with minimal feedback?

Legislative or technical measures may some day substantially abate email spam, but are unlikely to eliminate it. The creativity and efforts of those choose to violate laws and social norms to deliver their message will provide a continuing challenge to filter designers.

References

[1] "You might be an anti-spam kook if...," http://www.rhyolite.com/anti-spam/you-might-be.html.

[2] 2004 National Technology Readiness Survey: Summary report, http://www.smith.umd.edu/ntrs/NTRS_2004.pdf, 2005.

[3] A. J. Alberg, J. W. Park, B. W. Hager, M. V. Brock, and M. Diener-West, "The use of overall accuracy to evaluate the validity of screening or diagnostic tests," *Journal of General Internal Medicine*, vol. 19, no. 1, 2004.

[4] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, and C. D. Spyropoulos, "An evaluation of Naive Bayesian anti-spam filtering," *CoRR*, vol. cs.CL/0006013, Informal Publication, 2000.

[5] I. Androutsopoulos, E. F. Magirou, and D. K. Vassilakis, "A game theoretic model of spam e-mailing," in *CEAS 2005 — The Second Conference on Email and Anti-Spam*, 2005.

[6] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, and P. Stamatopoulos, "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 200)*, pp. 1–13, 2000.

[7] I. Androutsopoulos, G. Paliouras, and E. Michelakis, "Learning to filter unsolicited commercial E-Mail," Tech. Rep. 2004/2, *NCSR "Demokritos"*, October 2004.

[8] H. B. Aradhye, G. K. Myers, and J. A. Herson, "Image analysis for efficient categorization of image-based spam e-mail," in *Proceedings of the Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, 2005.

[9] F. Assis, "OSBF-Lua," http://osbf-lua.luaforge.net/.
[10] F. Assis, "OSBF-Lua — A text classification module for Lua the importance of the training method," in *Fifteenth Text REtrieval Conference (TREC-2006)*, Gaithersburg, MD: NIST, 2006.
[11] A. Berg, "Creating an antispam cocktail: Best spam detection and filtering techniques," http://searchsecurity.techtarget.com/tip/1,289483,sid14_gci1116643,00.html, 2005.
[12] S. Bickel, M. Bruckner, and T. Scheffer, "Discriminative learning for differing training and test distributions," *International Conference on Machine Learning (ICML)*, 2007.
[13] S. Bickel and T. Scheffer, "Dirichlet-Enhanced spam filtering based on biased samples," *Neural Information Processing Systems (NIPS)*, 2007.
[14] B. Biggio, G. Fumera, I. Pillai, and F. Roli, "Image spam filtering by content obscuring detection," in *CEAS 2007 — The Third Conference on Email and Anti-Spam*, 2007.
[15] Blacklists compared, http://www.sdsc.edu/ jeff/spam/Blacklists_Compared.html.
[16] A. Bratko, G. V. Cormack, B. Filipič, T. R. Lynam, and B. Zupan, "Spam filtering using statistical data compression models," *Journal of Machine Learning Research*, vol. 7, pp. 2673–2698, December 2006.
[17] A. Bratko and B. Filipič, "Spam filtering using character-level markov models: Experiments for the TREC 2005 Spam Track," in *Proceedings of 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005.
[18] A. Bratko and B. Filipič, "Exploiting structural information for semi-structured document categorization," *Information Processing and Management*, vol. 42, no. 3, pp. 679–694, 2006.
[19] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
[20] M. Bruckner, P. Haider, and T. Scheffer, "Highly scalable discriminative spam filtering," in *Proceedings of 15th Text REtrieval Conference (TREC 2006)*, Gaithersburg, MD, November 2006.
[21] B. Burton, *SpamProbe — A Fast Bayesian Spam Filter*. 2002. http:// spamprobe.sourceforge.net.
[22] B. Byun, C.-H. Lee, S. Webb, and C. Pu, "A discriminative classifier learning approach to image modeling and spam image identification," in *CEAS 2007 — The Third Conference on Email and Anti-Spam*, 2007.
[23] CAPTCHA: Telling humans and computers apart automatically, http://www.captcha.net/.
[24] X. Carreras and L. Márquez, "Boosting trees for anti-spam email filtering," in *Proceedings of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*, 2001.
[25] K. Chellapilla, K. Larson, K. Simard, and M. Czerwinski, "Designing human friendly human interactive proofs (HIPS)," in *CHI '05: SIGCHI Conference on Human Factors in Computing Systems*, pp. 711–720, 2005.
[26] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski, "Computers beat humans at single character recognition in reading based human interaction

proofs," in *CEAS 2005 — The Second Conference on Email and Anti-Spam*, 2005.

[27] S. Chhabra, *Fighting Spam, Phishing and Email Fraud*. University of California, Riverside, 2005.

[28] A. Ciltik and T. Gungor, "Time-efficient spam e-mail filtering using n-gram models," *Pattern Recognition Letters*, vol. 29, pp. 19–33, 2008.

[29] J. G. Cleary and I. H. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, vol. 32, pp. 396–402, April 1984.

[30] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the 12th International Conference on Machine Learning*, (A. Prieditis and S. Russell, eds.), pp. 115–123, Tahoe City, CA: Morgan Kaufmann, July 9–12 1995.

[31] G. Cormack, J. M. G. Hidalgo, and E. P. Sánz, "Feature engineering for mobile (SMS) spam filtering," in *30th ACM SIGIR Conference on Research and Development on Information Retrieval*, Amsterdam, 2007.

[32] G. V. Cormack, "Harnessing unlabeled examples through iterative application of Dynamic Markov Modeling," in *Proceedings of the ECML/PKDD Discovery Challenge Workshop*, Berlin, 2006.

[33] G. V. Cormack, "TREC 2006 Spam Track Overview," in *Fifteenth Text REtrieval Conference (TREC-2006)*, Gaithersburg, MD: NIST, 2006.

[34] G. V. Cormack, "TREC 2007 Spam Track Overview," in *Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD: NIST, 2007.

[35] G. V. Cormack, "University of waterloo participation in the TREC 2007 spam track," in *Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD: NIST, 2007.

[36] G. V. Cormack and A. Bratko, "Batch and on-line spam filter evaluation," in *CEAS 2006: The Third Conference on Email and Anti-Spam*, Mountain View, CA, 2006.

[37] G. V. Cormack, J. M. G. Hidalgo, and E. P. Sánz, "Spam filtering for short messages," in *CIKM '07: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, pp. 313–320, USA, New York, NY: ACM Press, 2007.

[38] G. V. Cormack and R. N. S. Horspool, "Data compression using dynamic Markov modelling," *The Computer Journal*, vol. 30, no. 6, pp. 541–550, 1987.

[39] G. V. Cormack and T. R. Lynam, *TREC Spam Filter Evaluation Toolkit*. http://plg.uwaterloo.ca/~gvcormac/jig/.

[40] G. V. Cormack and T. R. Lynam, "TREC 2005 Spam Track Overview," http://plg.uwaterloo.ca/~gvcormac/trecspamtrack05, 2005.

[41] G. V. Cormack and T. R. Lynam, "Statistical precision of information retrieval evaluation," in *SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 533–540, USA, New York, NY: ACM Press, 2006.

[42] G. V. Cormack and T. R. Lynam, "On-line supervised spam filter evaluation," *ACM Transactions on Information Systems*, vol. 25, no. 3, 2007.

[43] D. Crocker, "Challenges in Anti-spam Efforts," *The Internet Protocol Journal*, vol. 8, no. 4, http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_8-4/anti-spam_efforts.html, 2006.

[44] N. N. Dalvi, P. M. Domingos, S. K. Sanghai, and D. Verma, "Adversarial classification," in *KDD*, (W. Klm, R. Kohavi, J. Gehrke, and W. DuMouchel, eds.), pp. 99–108, 2004.

[45] R. Dantu and P. Kolan, "Detecting spam in VoIP networks," in *SRUTI'05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, pp. 5–5, USA, Berkeley, CA: USENIX Association, 2005.

[46] J. Deguerre, "The mechanics of Vipul's Razor," *Network Security*, pp. 15–17, September 2007.

[47] S. J. Delany, P. Cunningham, and L. Coyle, "Case-based reasoning for spam filtering," *Artificial Intelligence Review*, vol. 24, no. 3–4, pp. 359–378, 2005.

[48] T. Dietterich, *Statistical Tests for Comparing Supervised Classification Learning Algorithms*. Oregon State University, 1996.

[49] V. Dimitrios and E. M. Ion Androutsopoulos, "A game-theoretic investigation of the effect of human interactive proofs on spam e-mail," in *CEAS 2007 — The Fourth Conference on Email and Anti-Spam*, 2007.

[50] N. Dimmock and I. Maddison, "Peer-to-peer Collaborative Spam Detection," *ACM Crossroads*, vol. 11, no. 2, 2004.

[51] P. Domingos and M. J. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, no. 2–3, pp. 103–130, 1997.

[52] M. Dredze, R. Gevaryahu, and A. Elias-Bachrach, "Learning fast classifiers for image spam," in *CEAS 2007 — The Third Conference on Email and Anti-Spam*, 2007.

[53] H. Drucker, D. Wu, and V. Vapnik, "Support vector machines for spam categorization," *IEEE-NN*, vol. 10, no. 5, pp. 1048–1054, 1999.

[54] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, 1999.

[55] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *CRYPTO '92*, 1992.

[56] ECML/PKDD Discovery Challenge, http://www.ecmlpkdd2006.org/challenge.htm, 2006.

[57] T. Fawcett, "'In vivo' spam filtering: A challenge problem for data mining," *KDD Explorations*, vol. 5, no. 2, December 2003.

[58] T. Fawcett, *ROC Graphs: Notes and Practical Considerations for Researchers*. HP Laboratories, 2004. http://home.comcast.net/~tom.fawcett/public_html/papers/ROC101.pdf.

[59] D. Ferris and R. Jennings, "*Calculating the Cost of Spam for Your Organization*," http://http://www.ferris.com/?p=310061, 2005.

[60] D. Ferris, R. Jennings, and C. Williams, *The Global Economic Impact of Spam*. Ferris Research, http://www.ferris.com/?p=309942, 2005.

[61] Final ultimate solution to the spam problem, http://craphound.com/spamsolutions.txt.

[62] G. Fumera, I. Pillai, and F. Roli, "Spam filtering based on the analysis of text information embedded into images," *Journal of Machine Learning Research*

(special issue on Machine Learning in Computer Security), vol. 7, pp. 2699–2720, 12/2006 2006.

[63] W. Gansterer, A. Janecek, and R. Neumayer, "Spam filtering based on latent semantic indexing," in *SIAM Conference on Data Mining*, 2007.

[64] K. R. Gee, "Using latent semantic indexing to filter spam," in *SAC '03: Proceedings of the 2003 ACM symposium on Applied Computing*, pp. 460–464, USA, New York, NY: ACM Press, 2003.

[65] Z. Ghahramani, "Unsupervised learning," in *Advanced Lectures in Machine Learning*, pp. 72–112, Lecture Notes in Computer Science, vol. 3176, 2004.

[66] A. S. Glas, J. G. Lijmer, M. H. Prins, G. J. Bonsel, and P. M. M. Bossuyt, "The diagnostic odds ratio: A single indicator of test performance," *Journal of Clinical Epidemiology*, vol. 56, no. 11, pp. 1129–1135, 2003.

[67] J. Goodman, D. Heckerman, and R. Rounthwaite, "Stopping spam," *Scientific American*, vol. 292, pp. 42–88, April 2005.

[68] J. Goodman and W.-T. Yih, "Online discriminative spam filter training," in *The Third Conference on Email and Anti-Spam*, Mountain View, CA, 2006.

[69] P. Graham, *Better Bayesian Filtering*. http://www.paulgraham.com/better.html, 2004.

[70] J. Graham-Cumming, "How to beat an adaptive spam filter," in *The Spam Conference*, 2004.

[71] J. Graham-Cumming, "People and spam," in *The Spam Conference*, 2005.

[72] J. Graham-Cumming, "Does Bayesian poisining exist?," *Virus Bulletin*, February 2006.

[73] J. Graham-Cumming, "SpamOrHam," *Virus Bulletin*, 2006-06-01.

[74] J. Graham-Cumming, "The rise and fall of image-based spam," *Virus Bulletin*, 2006-11-01.

[75] J. Graham-Cumming, "The spammer's compendium: Five yars on," *Virus Bulletin*, 2007-09-20.

[76] J. Graham-Cumming, "Why I hate challenge-response," *JGC's Anti-Spam Newsletter*, February 28, 2005.

[77] Greylisting: The next step in the spam control war, http://projects.puremagic.com/greylisting/, 2003.

[78] B. Guenter, *Spam Archive*. http://www.untroubled.org/spam/.

[79] K. Gupta, V. Chaudhary, N. Marwah, and C. Taneja, *ECML-PKDD Discovery Challenge Entry*. Inductis India Pvt Ltd, 2006.

[80] K. Gupta, V. Chaudhary, N. Marwah, and C. Taneja, "Using positive-only learning to deal with the heterogeneity of labeled and unlabeled data," in *Proceedings of ECML/PKDD Discovery Challenge Workshop*, Berlin, 2006.

[81] B. Hayes, "How many ways can you spell Viagra?," *American Scientist*, vol. 95, 2007.

[82] J. M. G. Hidalgo, "Evaluating cost-sensitive unsolicited bulk email categorization," in *Proceedings of SAC-02, 17th ACM Symposium on Applied Computing*, pp. 615–620, Madrid, ES, 2002.

[83] J. M. G. Hidalgo, "Evaluating cost-sensitive unsolicited bulk email categorization," in *SAC '02: Proceedings of the 2002 ACM Symposium on Applied Computing*, pp. 615–620, Madrid: ACM Press, March 2002.

[84] J. M. G. Hidalgo, G. C. Bringas, E. P. Sanz, and F. C. Garcia, "Content based SMS spam filtering," in *DocEng '06: Proceedings of the 2006 ACM Symposium on Document Engineering*, pp. 107–114, USA, New York, NY: ACM Press, 2006.

[85] J. M. G. Hidalgo, M. M. López, and E. P. Sanz, "Combining text and heuristics for cost-sensitive spam filtering," in *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning*, pp. 99–102, USA, Morristown, NJ: Association for Computational Linguistics, 2000.

[86] S. Holden, "Spam Filtering II," *Hakin9, 02/2004*, pp. 68–77, 2004.

[87] D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression*. New York: Wiley, 2000.

[88] J. Hovold, "Naive bayes spam filtering using word-position-based attributes," in *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS 2005)*, Palo Alto, CA, July 2005.

[89] M. Ilger, J. Strauss, W. Gansterer, and C. Proschinger, *The Economy of Spam*. Vol. FA384018-6, Instituted of Distributed and Multimedia Systems, Univeristy of Vienna, 2006.

[90] T. Joachims, "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization," in *Proceedings of ICML-97, 14th International Conference on Machine Learning*, (D. H. Fisher, ed.), pp. 143–151, US, Nashville, San Francisco: Morgan Kaufmann Publishers, 1997.

[91] T. Joachims, *Transductive Inference for Text Classification Using Support Vector Machines*. 1999.

[92] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Eleventh Conference on Uncertainty in Artificial Integelligence*, pp. 338–345, 1995.

[93] Y. Junejo and A. Karim, "A two-pass statistical approach for automatic personalized spam filtering," in *Proceedings of ECML/PKDD Discovery Challenge Workshop*, Berlin, 2006.

[94] B. Klimt and Y. Yang, "Introducing the Enron corpus," in *CEAS 2004 — The Conference on Email and Anti-Spam*, 2004.

[95] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *IJCAI*, pp. 1137–1145, 1995.

[96] A. Kolcz and J. Alspector, "SVM-based filtering of E-mail spam with content-specific misclassification costs," *TextDM 2001 (IEEE ICDM-2001 Workshop on Text Mining)*, 2001.

[97] A. Kolcz and A. Chowdhury, "Hardening fingerprints by context," in *CEAS 2007 — The Third Conference on Email and Anti-Spam*, 2007.

[98] A. Kolcz and A. Chowdhury, "Lexicon randomization for near-duplicate detection with I-match," *Journal of Supercomputing*, vol. DOI 10.1007/s11227-007-0171-z, 2007.

[99] A. Kolcz, A. Chowdhury, and J. Alspector, "The impact of feature selection on signature-driven spam detection," in *CEAS 2004 — The Conference on Email and Anti-Spam*, 2004.

[100] P. Komarek and A. Moore, "Fast robust logistic regression for large sparse datasets with binary outputs," in *Artificial Intelligence and Statistics*, 2003.
[101] A. Kornblum, "Searching for John Doe: Finding spammers and phishers," in *CEAS 2004 — The Conference on Email and Anti-Spam*, 2004.
[102] S. Kotsiantis, "Supervised learning: A review of classification techniques," *Informatica*, vol. 31, pp. 249–268, 2007.
[103] B. Krebs, "In the fight agains spam E-mail, Goliath wins again," *Washington Post*, May 17 2006.
[104] H. Lee and A. Y. Ng, "Spam deobfuscation using a hidden Markov model," in *CEAS 2005 – The Second Conference on Email and Anti-Spam*, 2005.
[105] S. Lee, I. Jeong, and S. Choi, "Dyamically weighted hidden Markov model for spam deobfuscation," in *IJCAI 07*, pp. 2523–2529, 2007.
[106] J. R. Levine, "Experiences with greylisting," in *CEAS 2005: Second Conference on Email and Anti-Spam*, 2005.
[107] D. D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," in *Proceedings of ICML-94, 11th International Conference on Machine Learning*, (W. W. Cohen and H. Hirsh, eds.), pp. 148–156, US, New Brunswick, San Francisco: Morgan Kaufmann Publishers, 1994.
[108] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka, "Training algorithms for linear text classifiers," 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.), (Zürich, CH), pp. 298–306, New York, US: ACM Press, 1996.
[109] K. Li, C. Pu, and M. Ahamad, "Resisting SPAM delivery by TCP damping," in *CEAS 2004 — The Conference on Email and Anti-Spam*, 2004.
[110] B. Lieba and J. Fenton, "DomainKeys identified email (DKIM): Using digital signatures for domain verification," in *CEAS 2007: The Third Conference on Email and Anti-Spam*, 2007.
[111] B. Lieba, J. Ossher, V. T. Rajan, R. Segal, and M. Wegman, "SMTP path analysis," in *2nd Conference on Email and Anti-spam*, 2005.
[112] Ling-Spam, PU and Enron Corpora, http://www.iit.demokritos.gr/skel/i-config/downloads/.
[113] T. Lynam and G. Cormack, *TREC Spam Filter Evaluation Took Kit*. http://plg.uwaterloo.ca/~trlynam/spamjig.
[114] T. R. Lynam and G. V. Cormack, "On-line spam filter fusion," in *29th ACM SIGIR Conference on Research and Development on Information Retrieval*, Seattle, 2006.
[115] J. Lyon and M. Wong, *Sender-ID: Authenticating E-mail RFC 4406*. Internet Engineering Task Force, 2006.
[116] Mail abuse prevention system, http://www.mail-abuse.com/, 2005.
[117] M. Mangalindan, "For bulk E-mailer, pestering millions offers path to profit," *Wall Street Journal*, November 13, 2002.
[118] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET curve in assessment of detection task performance," in *Proceedings of Eurospeech '97*, pp. 1895–1898, Rhodes, Greece, 1997.

[119] R. McMillan, "US Court threatens Spamhaus with shut down," *InfoWorld*, October 09 2006.

[120] B. Medlock, "An adaptive, semi-structured language model approach to spam filtering on a new corpus," in *Proceeding of CEAS 2006 — Third Conference on Email and Anti-Spam*, Mountain View, CA, 2006.

[121] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Naive Bayes — Which Naive Bayes?," in *Proceedings of CEAS 2006 — Third Conference on Email and Anti-Spam*, Mountain View, CA, 2006.

[122] E. Michelakis, I. Androutsopoulos, G. Paliouras, G. Sakkis, and P. Stamatopoulos, "Filtron: A learning-based anti-spam filter," in *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS 2004)*, Mountain View, CA, July 2004.

[123] G. Mishne and D. Carmel, *Blocking Blog Spam with Language Model Disagreement*. 2005.

[124] E. Moustakas, C. Ranganathan, and P. Duquenoy, "Chunk-Kwei: A pattern-discovery-based System for the automatic identificaton of unsolicited email messages (spam)," in *CEAS 2004 — The Conference on Email and Anti-Spam*, 2004.

[125] J. Niu, J. Xu, J. Yao, J. Zheng, and Q. Sun, "WIM at TREC 2007," in *Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD: NIST, 2007.

[126] C. S. Oliveira, F. G. Cozman, and I. Cohen, "Splitting the unsupervised and supervised components of semi-supervised learning," in *ICML 2005 LPCTD Workshop*, 2005.

[127] R. M. Pampapathi, B. Mirkin, and M. Levene, "A suffix tree approach to email filtering," Tech. Rep., Birkbeck University of London, 2005.

[128] C. Perlich, F. Provost, and J. S. Simonoff, "Tree induction vs. logistic regression: A learning-curve analysis," *Journal of Machanic Learning and Research*, vol. 4, pp. 211–255, 2003.

[129] B. Pfahringer, "A semi-supervised spam mail detector," in *Proceedings of ECML/PKDD Discovery Challenge Workshop*, Berlin, 2006.

[130] Project Honeypot, http://www.projecthoneypot.org/.

[131] C. Pu and S. Webb, "Observed trends in spam construction techniques," in *Proceedings of CEAS 2006 — Third Conference on Email and Anti-Spam*, Mountain View, CA, 2006.

[132] J. R. Quinlan, *C4.5: Programs for Machine Learning*. USA, San Francisco, CA: Morgan Kaufmann Publishers Inc., 1993.

[133] A. Ramachandran, D. Dagon, and N. Feamster, "Can DNS-based blacklists keep up with bots?," in *CEAS 2006 — The Second Conference on Email and Anti-Spam*, 2006.

[134] E. S. Raymond, D. Relson, M. Andree, and G. Louis, "BogoFilter," http://bogofilter.sourceforge.net/, 2004.

[135] F. R. Rideau, *Stamps vs. Spam: Postage as a Method to Eliminate Unsolicited Email*. http://fare.tunes.org/articles/stamps_vs_spam.html, 2002.

[136] G. Robinson, "A statistical approach to the spam problem," *Linux Journal*, vol. 107, no. 3, March 2003.

[137] R. Roman, J. Zhou, and J. Lopez, "An anti-spam scheme using pre-challenges," *Computer Communications*, vol. 29, no. 15, pp. 2739–2749, 2006.
[138] C. Rossow, "Anti-Spam measures of European ISPs/ESPs: A survey based analysis of state-of-the-art technologies, current spam trends and recommendations for future-oriented anti-spam concepts," *Institute for Internet Security*, August 2007.
[139] K. J. Rothman and S. Greenland, *Modern epidemiology*. Lippinscott Williams and Wilkins, 1998.
[140] R. Rowland, *Spam, Spam, Spam: The Cyberspace Wars*. CBC, http://www.cbc.ca/news/background/spam/, 2004.
[141] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail," in *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, AAAI Technical Report WS-98-05, 1998.
[142] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos, "Stacking classifiers for anti-spam filtering of e-mail," in *Empirical Methods in Natural Language Processing (EMNLP 2001)*, pp. 44–50, 2001.
[143] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos, "A memory-based approach to anti-spam filtering for mailing lists," *Information Retrieval*, vol. 6, no. 1, pp. 49–73, 2003.
[144] M. Sasaki and H. Shinnou, "Spam detection using text clustering," in *CW '05: Proceedings of the 2005 International Conference on Cyberworlds*, pp. 316–319, USA, Washington, DC: IEEE Computer Society, 2005.
[145] R. E. Schapire and Y. Singer, "Improved boosting using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.
[146] K. M. Schneider, "A comparison of event models for naive bayes anti-spam e-mail filtering," in *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, 2003.
[147] D. Sculley, "Online active learning methods for fast label-efficient spam filtering," in *Proceeding of the CEAS 2007 — Fourth Conference on Email and Anti-Spam*, Mountain View, CA, 2007.
[148] D. Sculley and C. E. Brodley, "Compression and machine learning: A new perspective on feature space vectors," in *Data Compression Conference (DCC 06)*, pp. 332–341, Snowbird, 2006.
[149] D. Sculley and G. V. Cormack, *Filtering Spam in the Presence of Noisy User Feedback*. Tufts University, 2008.
[150] D. Sculley and G. M. Wachman, "Relaxed online support vector machines for spam filtering," in *30th ACM SIGIR Conference on Research and Development on Information Retrieval*, Amsterdam, 2007.
[151] D. Sculley and G. M. Wachman, "Relaxed online SVMs in the TREC Spam filtering track," in *Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD: NIST, 2007.
[152] D. Sculley, G. M. Wachman, and C. E. Brodley, "Spam classification with on-line linear classifiers and inexact string matching features," in *Proceedings*

of the 15th Text REtrieval Conference (TREC 2006), Gaithersburg, MD, November 2006.

[153] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[154] R. Segal, J. Crawford, J. Kephart, and B. Leiba, "SpamGuru: An enterprise anti-spam filtering system," in *First Conference on Email and Anti-Spam (CEAS)*, 2004.

[155] R. Segal, T. Markowitz, and W. Arnold, "Fast uncertainty sampling for labeling large e-mail corpora," in *Proceedings of the CEAS 2006 — Third Conference on Email and Anti-Spam*, Mountain View, CA, 2006.

[156] Shakhnarovish, Darrell, and Indyk, *Nearest-Neighbor Methods in Learning and Vision,* (Shakhnarovish, ed.), MIT Press, 2005.

[157] V. Sharma and A. O'Donnell, "Fighting spam with reputation systems," *ACM Queue*, November 2005.

[158] C. Siefkes, F. Assis, S. Chhabra, and W. S. Yerazunis, "Combining winnow and orthogonal sparse bigrams for incremental spam filtering," in *PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 410–421, USA, New York, NY: Springer-Verlag, New York, Inc., 2004.

[159] P. Y. Simard, R. Szeliski, J. Benaloh, J. Couvreur, and I. Calinov, "Using character recognition and segmentation to tell computer from humans," in *ICDAR '03: Seventh International Conference on Document Analysis and Recognition*, 2003.

[160] J. Snyder, "Spam in the wild, the sequel," *Network World 12/20/04*, 2004.

[161] E. Solan and E. Reshef, "The effects of anti-spam methods on spam mail," in *CEAS 2006 — The Third Conference on Email and Anti-Spam*, 2006.

[162] Spam testing methodology, http://www.opus1.com/www/whitepapers/spamtestmethodology.pdf, 2007.

[163] spamassassin.org, *The Spamassassin Public Mail Corpus*. http:// spamassassin.apache.org/publiccorpus, 2003.

[164] spamassassin.org, *Welcome to SpamAssassin*. http://spamassassin.apache.org, 2005.

[165] Spambase, http://mlearn.ics.uci.edu/databases/spambase/.

[166] J. A. Swets, "Effectiveness of information retrieval systems," *American Documentation*, vol. 20, pp. 72–89, 1969.

[167] T. Takemura and H. Ebara, "Spam mail reduces economic effects," in *Second International Conference on the Digital Society*, pp. 20–24, 2008.

[168] W. Tau Yih, R. McCann, and A. Kolcz, "Improving spam filtering by detecting gray mail," in *Proceedings of CEAS 2007 — Fourth Conference on Email and Anti-Spam*, Mountain View, CA, 2007.

[169] D. M. J. Tax and C. Veenman, "Tuning the hyperparameter of an AUC-optimized classifier," in *Seventeenth Belgium-Netherlands Conference on Artificial Intelligence*, pp. 224–231, 2005.

[170] The CEAS 2007 Live Spam Challenge, http://www.ceas.cc/2007/challenge/challenge.html, 2007.

[171] The penny black project, http://research.microsoft.com/research/sv/PennyBlack/.
[172] TREC 2005 Spam Corpus, http://plg.uwaterloo.ca/~gvcormac/treccorpus, 2005.
[173] TREC 2006 Spam Corpora, http://plg.uwaterloo.ca/~gvcormac/treccorpus, 2006.
[174] TREC 2007 Spam Corpus, http://plg.uwaterloo.ca/~gvcormac/treccorpus, 2007.
[175] K. Tretyakov, "Machine learning techniques in spam filtering," Tech. Rep., Institute of Computer Science, University of Tartu, 2004.
[176] N. Trogkanis and G. Paliouras, "Using positive-only learning to deal with the heterogeneity of labeled and unlabeled data," in *Proceedings of ECML/PKDD Discovery Challenge Workshop*, Berlin, 2006.
[177] H. Tschabitscher, *What you Need to Know about Challenge-Response Spam Filters*. http://email.about.com/cs/spamgeneral/a/challenge_resp.htm.
[178] D. Turner, M. Fossi, E. Johnson, T. Mack, J. Blackbird, S. Entwisle, M. K. Low, D. McKinney, and C. Wueest, *Symantec Global Internet Security Threat Report: Trends for July-December 07*. Symantec, http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiii_04-2008.en-us.pdf, 2007.
[179] A. Tuttle, E. Milios, and N. Kalyaniwalla, "An evaluation of machine learning techniques for enterprise spam filters," Technical Report CS-2004-03, Halifax, NS: Dalhousie University, 2004.
[180] C. J. Van Rijsbergen, *Information Retrieval*. Department of Computer Science, University of Glasgow, Second ed., 1979.
[181] Veritest Anti-Spam Benchmark Service Autumn 2005 Report http://www.tumbleweed.com/pdfs/VeriTest_Anti-Spam_Report_Vol4_all_c.pdf, 2005.
[182] E. Voorhees, *Fourteenth Text REtrieval Conference (TREC-2005)*. Gaithersburg, MD: NIST, 2005.
[183] E. Voorhees, *Fifteenth Text REtrieval Conference (TREC-2005)*. Gaithersburg, MD: NIST, 2006.
[184] E. Voorhees, *Sixteenth Text REtrieval Conference (TREC-2005)*. Gaithersburg, MD: NIST, 2007.
[185] E. M. Voorhees and D. K. Harman, eds., *TREC — Experiment and Evaluation in Information Retrieval*. Boston: MIT Press, 2005.
[186] Z. Wang, W. Josephson, Q. LV, M. Charikar, and K. Li, "Filtering image spam with near-duplicate detection," in *CEAS 2007 — The Third Conference on Email and Anti-Spam*, 2007.
[187] Web Spam Challenge, 2008.
[188] S. Webb, J. Caverloo, and C. Pu, "Introducing the webb spam corpus: Using email spam to identify web spam automatically," in *Proceedings of CEAS 2006 — Third Conference on Email and Anti-Spam*, Mountain View, CA, 2006.
[189] West Coast Labs, http://www.westcoastlabs.com.
[190] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The context-tree weighting method: Basic properties," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 653–664, 1995.

[191] G. L. Wittel and S. F. Wu, "On attacking statistical spam filters," in *CEAS 2004 — The Conference on Email and Anti-Spam*, 2004.

[192] I. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. Cunningham, *Weka: Practical Machine Learning Tools and Techniques with Java Implementations*. 1999.

[193] M. Wong and W. Schlitt, *Sender Policy Framework (SPF) for Authorizing Use of Domains in E-mail*. Vol. RFC 4408, 2006.

[194] W. Yerazunis, *Correspondence with Paul Graham*. http://www. paulgraham.com/wsy.html, 16 October 2002.

[195] W. S. Yerazunis, *CRM114 — the Controllable Regex Mutilator*. http://crm114.sourceforge.net/, 2004.

[196] W. S. Yerazunis, "The spam-filtering accuracy plateau at 99.9% accuracy and how to get past it," in *2004 MIT Spam Conference*, January 2004.

[197] W. S. Yerazunis, "Seven hypothesis about spam filtering," in *Proceedings 15th Text REtrieval Conference (TREC 2006)*, NIST, Gaithersburg, MD, November 2006.

[198] W. S. Yerazunis, "Seven Hypothesis about Spam," in *Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD: NIST, 2007.

[199] W. Yih, J. Goodman, and G. Hulten, "Learning at low false positive rates," in *Proceedings of the 3rd Conference on Email and Anti-Spam*, 2006.

[200] X. Yue, A. Abraham, Z.-X. Chi, Y.-Y. Hao, and H. Mo, "Artificial immune system inspired behavior-based anti-spam filter," *Soft Computing*, vol. 11, pp. 729–740, 2007.

[201] L. Zhang, J. Zhu, and T. Yao, "An evaluation of statistical spam filtering techniques," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 3, no. 4, pp. 243–269, 2004.

[202] W. Zhao and Z. Zhang, "An email classification model based on rough set theory," in *Active Media Technology, 2005. (AMT 2005)*, 2005.

[203] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *18th Annual Conference on Neural Information Processing Systems*, 2003.

[204] X. Zhu, *Semi-supervised Learning Literature Survey*. Vol. TR 1530, University of Wisconsin, 2007.

[205] A. Zien, "Semi-supervised support vector machines and application to spam filtering," in *Oral Presentation, ECML/PKDD Discovery Challenge Workshop*, Berlin, 2006.

[206] A. Zinman and J. Donath, "Is Britney Spears spam?," in *Proceedings of CEAS 2007 — Fourth Conference on Email and Anti-Spam*, Mountain View, CA, 2007.