Information Retrieval, 6, 49–73, 2003

A Memory-Based Approach to Anti-Spam Filtering for Mailing Lists

GEORGIOS SAKKIS gsakis@iit.demokritos.gr
Institute of Informatics and Telecommunications, National Centre for Scientific Research (NCSR) "Demokritos", GR-153 10 Ag. Paraskevi, Athens, Greece

ION ANDROUTSOPOULOS ion@aueb.gr
Department of Informatics, Athens University of Economics and Business, Patission 76, GR-104 34, Athens, Greece

GEORGIOS PALIOURAS paliourg@iit.demokritos.gr
VANGELIS KARKALETSIS vangelis@iit.demokritos.gr
CONSTANTINE D. SPYROPOULOS costass@iit.demokritos.gr
Institute of Informatics and Telecommunications, National Centre for Scientific Research (NCSR) "Demokritos", GR-153 10 Ag. Paraskevi, Athens, Greece

PANAGIOTIS STAMATOPOULOS T.Stamatopoulos@di.uoa.gr
Department of Informatics, University of Athens, TYPA Buildings, Panepistimiopolis, GR-157 71, Athens, Greece

Received September 17, 2001; Revised August 12, 2002; Accepted October 16, 2002

Abstract. This paper presents an extensive empirical evaluation of memory-based learning in the context of anti-spam filtering, a novel cost-sensitive application of text categorization that attempts to identify automatically unsolicited commercial messages that flood mailboxes. Focusing on anti-spam filtering for mailing lists, a thorough investigation of the effectiveness of a memory-based anti-spam filter is performed using a publicly available corpus. The investigation includes different attribute and distance-weighting schemes, and studies on the effect of the neighborhood size, the size of the attribute set, and the size of the training corpus. Three different cost scenarios are identified, and suitable cost-sensitive evaluation functions are employed. We conclude that memory-based anti-spam filtering for mailing lists is practically feasible, especially when combined with additional safety nets. Compared to a previously tested Naive Bayes filter, the memory-based filter performs on average better, particularly when the misclassification cost for non-spam messages is high.

Keywords: text categorization, machine learning, unsolicited commercial e-mail, spam

1. Introduction

This paper presents a thorough empirical evaluation of memory-based learning in the context of a novel cost-sensitive application, that of filtering unsolicited commercial e-mail messages.

The increasing popularity and low cost of electronic mail have intrigued direct marketers to flood the mailboxes of thousands of users with unsolicited messages. These messages are usually referred to as *spam* or, more formally, *Unsolicited Commercial E-mail* (UCE), and may advertise anything, from vacations to get-rich schemes. Spam messages are extremely annoying to most users, as they waste their time and prolong dial-up connections. They also waste bandwidth, and often expose minors to unsuitable content by advertising pornographic sites. A 1998 study found that spam messages constituted approximately 10% of the incoming messages to a corporate network (Cranor and LaMacchia 1998). The situation seems to be worsening, and without appropriate counter-measures, spam messages could eventually undermine the usability of e-mail.

The proposed counter-measures have been either regulatory or technical, with regulatory measures having limited effect so far.[1] Technical measures are based on *anti-spam filters*, which attempt to discriminate between spam and non-spam, hereafter *legitimate*, messages. Typical anti-spam filters currently in the market employ blacklists of known spammers, and handcrafted rules that block messages containing specific words or phrases. Blacklists, however, are of little use, as spammers often use forged addresses. Handcrafted rules are also problematic: to be most effective, they need to be tuned to the incoming messages of particular users or groups of users, a task requiring time and expertise that has to be repeated periodically to account for gradual changes in the characteristics of spam messages (Cranor and Lamacchia 1998).

The success of machine learning techniques in text categorization (Sebastiani 2001) has recently led researchers to explore the applicability of learning algorithms in anti-spam filtering.[2] A supervised learning algorithm is fed with a corpus of messages that have been classified manually as spam or legitimate, and builds a classifier, which is then used to detect incoming spam messages. Apart from collecting separately spam and legitimate training messages, the learning process is fully automatic, and can be repeated to tailor the filter to the incoming messages of particular users or groups, or to capture changes in the characteristics of spam messages. Anti-spam filtering differs from other electronic mail and news categorization tasks (Lang 1995, Cohen 1996, Payne and Edwards 1997), in that spam messages cover a very wide spectrum of topics, and hence are much less homogeneous than other categories that have been considered in the past. Another difference is that anti-spam filtering is a case of *cost-sensitive* classification, an area that has not been explored as intensively in text categorization as in other classification tasks.[3] The cost of accidentally blocking a legitimate message can be much higher than letting a spam message pass the filter, and this cost difference must be taken into account during both training and evaluation.

Sahami et al. (1998) experimented with an anti-spam filter based on Naive Bayes (Mitchell 1997). In similar anti-spam experiments, Pantel and Lin (1998) found Naive Bayes to outperform Ripper (Cohen and Singer 1999). Drucker et al. (1999) experimented with Ripper, Rocchio's classifier (Rocchio 1971, Joachims 1997), Support Vector Machines (Cristianini and Shawe-Taylor 2000), and boosted decision trees (Quinlan 1993, Schapire and Singer 2000), with results showing that Support Vector Machines and boosted decision trees achieve very similar error rates, both outperforming Rocchio's classifier. A direct comparison of these previous results, however, is impossible, as they are based on different and not publicly available data sets. Furthermore, the reported figures can be misleading, since they are not formulated within a cost-sensitive framework.

Research on text categorization has benefited significantly from the existence of publicly available, manually categorized document collections, like the Reuters corpora, which have been used as standard benchmarks.[4] Producing similar corpora for anti-spam filtering is complicated by privacy issues. Publicizing spam messages does not pose a problem, because spam messages are distributed blindly to very large numbers of recipients, and, hence, they are effectively already publicly available; but legitimate e-mail messages cannot usually be released without violating the privacy of their recipients and senders. There is, however, a type of anti-spam filtering where researchers can share benchmark corpora without violating privacy constraints: constructing filters that will guard against spam messages sent to mailing lists with public archives.[5] In this case, rather than examining the messages that arrive at a user's individual mailbox, the anti-spam filter examines the messages that arrive at the server of the list, before sending the messages to the subscribers of the list.

Mailing lists are often targeted by spammers, who either cannot distinguish between personal addresses and addresses of mailing lists, or deliberately send their messages to lists to reach their subscribers. In either case, the result can be a major waste of bandwidth and storage, especially in mailing lists with large numbers of subscribers. To avoid this waste and other abuses, many lists are moderated, i.e., a person is assigned the task of reading each incoming message before allowing it to be circulated to the subscribers. Lists with intense traffic, however, can overwhelm their moderators, and in many cases there may be nobody willing to act as a moderator. Hence, a filter that would automatically detect spam postings to mailing lists, or that would report suspicious postings to a moderator for further inspection would be particularly useful.

Most mailing lists focus on particular topics. Hence, an anti-spam filter trained for a particular mailing list can use features indicating that a message is off-topic (e.g., absence of particular terminology) as hints that the message is spam. Assuming that the messages that most users receive are less topic-specific than the messages of a mailing list, it is reasonable to expect that the performance of an anti-spam filter for a list will be better than the performance of an anti-spam filter for a personal mailbox. One cannot, therefore, safely generalize conclusions drawn from experimenting with mailing lists to anti-spam filters for personal mailboxes, although conclusions of the first kind can be seen as preliminary indications of the viability of anti-spam filtering in other settings, and, as already mentioned, anti-spam filtering for mailing lists is valuable in its own right. We also note that in anti-spam filtering approaches that examine only the *content* of the messages, as in our case, it makes no difference whether the messages are circulated via a list server or a Usenet-like newsgroup. Hence the work described here applies equally well to newsgroups.

Along these lines, we have recently introduced *Ling-Spam*, a publicly available collection of spam messages and legitimate messages from a mailing list on linguistics, as well as suitable cost-sensitive evaluation measures, which were used to conduct a detailed evaluation of a Naive Bayes anti-spam filter (Androutsopoulos et al. 2000a). Continuing that strand of work, this paper presents a thorough empirical evaluation of a *memory-based* anti-spam filter, using Ling-Spam and the same evaluation framework as in our previous experiments, thus contributing towards standard benchmarks. Memory-based classifiers are particularly promising for anti-spam filtering, on the grounds that spam messages form a rather incoherent class in terms of topics. Hence, a classifier that predicts the class of a new message by recalling similar already classified messages is likely to perform at least as well as classifiers that build a unique model for each message class. A disadvantage of memory-based classifiers is that they can be computationally expensive in their classification phase, due to their "lazy" character. Very efficient implementations of memory-based classifiers, however, are available that address this issue (Daelemans et al. 1997, 2000). Furthermore, our experiments (Section 6.4) indicate that in many cases memory-based anti-spam filtering is viable with small training sets, which can be handled efficiently even by less sophisticated implementations of memory-based classifiers.

A preliminary investigation of memory-based anti-spam filtering was presented in a previous article (Androutsopoulos et al. 2000c), where we experimented with a simplistic version of the *k*-Nearest Neighbor algorithm (Mitchell 1997) with promising results. Gomez Hidalgo et al. (2000) have reported similar experiments. The work that will be presented here is much more detailed, in that it considers the effect of several extensions to the basic *k*-Nearest Neighbor algorithm that have not been explored in previous anti-spam experiments, including different schemes for attribute and distance weighting, as well as the effect of the neighborhood size, the size of the training corpus, the size of the attribute set, and different cost scenarios. In all cases, we attempt to justify our observations, and thus increase our confidence that similar behavior is likely to appear in other similar cost-sensitive applications. Overall, our results indicate that memory-based anti-spam filtering for mailing lists is practically feasible, especially when combined with additional safety nets. Compared to the Naive Bayes filter, the memory-based filter performs on average better, particularly when the misclassification cost for legitimate messages is high.

The rest of this paper is organized as follows: Section 2 presents our benchmark corpus; Section 3 describes the preprocessing that is applied to the messages to convert them to training or testing instances; Section 4 discusses the basic memory-based learner; Section 5 introduces the cost-sensitive evaluation measures; Section 6 presents our experimental results, investigating separately the effect of each one of the extensions to the basic algorithm that we have considered; Section 7 concludes and suggests directions for further research.

2. Benchmark corpus

This section describes Ling-Spam, the corpus that was used in our experiments. Ling-Spam is a mixture of spam messages, and legitimate messages sent via the Linguist list, a moderated mailing list about the science and profession of linguistics.[6] The corpus consists of 2893 messages:

- 2412 legitimate messages, obtained by randomly downloading digests from the list's archives, breaking the digests into their messages, and removing text added by the list's server.
- 481 spam messages, received by one of the authors. Attachments, HTML tags, and duplicate spam messages received on the same day have not been included.

Spam messages constitute approximately 16% of Ling-Spam, a rate close to those reported by Cranor and LaMacchia (1998), and Sahami et al. (1998). The Linguist messages are less topic-specific than one might expect. For example, they contain job postings, software availability announcements, and even flame-like responses.

The number of messages in Ling-Spam is small when compared to established benchmarks for text categorization, such as the Reuters corpora (Section 1). As a partial remedy, we used *10-fold stratified cross-validation* (Kohavi 1995) in all of our experiments, a technique that increases the confidence of experimental findings when using small datasets. That is, Ling-Spam was partitioned in 10 parts, with each part maintaining the same ratio of legitimate and spam messages as in the entire corpus. Each experiment was repeated 10 times, each time reserving a different part as the testing corpus and using the remaining 9 parts as the training corpus. Performance scores were then averaged over the 10 iterations.

To the best of our knowledge, the only other publicly available collection of spam and legitimate messages is Spambase (Gomez Hidalgo et al. 2000).[7] This is a collection of 4601 vectors (Section 3), each representing a spam or legitimate message, with spam messages constituting approximately 39% of the total. Each vector contains the values of 58 pre-selected attributes, including the category. Spambase is much more restrictive than Ling-Spam, since the original texts are not available. For example, unlike Ling-Spam, with Spambase one cannot experiment with more attributes, different attribute selection algorithms, or attributes corresponding to phrases, rather than individual words.

3. Message representation and preprocessing

For the purposes of our experiments, each message is converted into a vector $\vec{x} = \langle x_1, x_2, x_3, \ldots, x_n \rangle$, where $x_1, \ldots, x_n$ are the values of attributes $X_1, \ldots, X_n$, as in the vector space model (Salton and McGill 1983). All attributes are binary: $X_i = 1$ if some characteristic represented by X_i is present in the message; otherwise $X_i = 0$. In our experiments, attributes represent words, i.e., each attribute shows if a particular word (e.g., "adult") occurs in the message. To avoid treating forms of the same word as different attributes, a lemmatizer was applied to the corpora to convert each word to its base form (e.g., "was" becomes "be").[8]

It is also possible to use attributes corresponding to phrases (e.g., "be over 21") or non-textual characteristics (e.g., whether or not the message contains attachments, or whether or not it was sent on a Sunday). Previous work (Sahami et al. 1998) indicates that using both word and phrasal attributes can lead to marginally better results than using only word attributes. However, a separate, not yet published, strand of our work found no evidence that using phrasal attributes in anti-spam filtering can lead to consistent improvements. The results of Sahami et al. (1998) also show that the inclusion of non-textual attributes can be more beneficial. Using non-textual attributes, however, requires an additional manual preprocessing stage to devise candidate attributes of this kind (e.g., one may observe that, unlike spam messages, the legitimate messages of a particular mailing list are rarely sent over weekends, and rarely contain attachments, and hence consider these characteristics as candidate attributes). As we are interested in anti-spam filters that can be trained fully automatically, we did not investigate the use of non-textual attributes, though operational filters may indeed benefit from attributes of this type.

To reduce the high dimensionality of the instance space, attribute selection was performed. First, words occurring in less than 4 messages were discarded, i.e., they were not considered

Table 1. Attributes with the highest *IG* scores in the entire Ling-Spam corpus, ordered by decreasing score.

1. language
2. !
3. university
4. remove
5. free
6. linguistic
7. your
8. you
9. click
10. money
11. sell
12. english
13. @
14. million
15. our
16. income
17. today
18. market
19. advertise
20. get
21. business
22. product
23. just
24. edu
25. guarantee
26. linguistics
27. internet
28. bulk
29. company
30. %
31. save
32. papers
33. conference
34. day
35. $
36. mailing
37. win
38. thousand
39. now
40. purchase
41. earn
42. best
43. de
44. buy
45. easy
46. dollar
47. com
48. every
49. hundred
50. customer

as candidate attributes. Then, the Information Gain (*IG*) of each candidate attribute X with respect to variable C, denoting the category, was computed as in (1) below, and the attributes with the m highest *IG*-scores were selected, with m varying in our experiments from 50 to 700 by 50. The probabilities were estimated from the training corpora using m-estimates (Mitchell 1997).

$$IG(X, C) = \sum_{x \in \{0,1\}, c \in \{spam, legit\}} P(X = x, C = c) \cdot \log_2 \frac{P(X = x, C = c)}{P(X = x) \cdot P(C = c)} \qquad (1)$$

Yang and Pedersen (1997) report that it is feasible to remove up to 98% of the candidate attributes using *IG* or other similar functions, and preserve, or even improve, generalization accuracy. Table 1 shows the attributes with the highest *IG* scores in the entire Ling-Spam corpus. They are mostly words that are common in spam messages (e.g., "remove", "free", "your") and rare in messages about linguistics, or words that are frequent in Linguist messages (e.g., "language", "university", "linguistic") and uncommon in spam messages. Unlike typical text categorization tasks, punctuation and other special symbols (e.g., "!", "@", "%", "$") were not discarded during preprocessing, but they were treated as "words". It can be seen from Table 1 that many of these symbols are among the best attributes in Ling-Spam, which is not surprising given that these symbols are used much more frequently in spam messages than in normal text.

4. Memory-based learning

Memory-based, or “instance-based”, methods do not construct a unique model for each category, but simply store the training examples (Aha et al. 1991, Wilson 1997). Test instances are then classified by estimating their similarity to the stored examples. In its simplest form, memory-based learning treats instances as points in a multi-dimensional space defined by the attributes that have been selected. Classification is usually performed through a variant of the basic k-Nearest-Neighbor (k-NN) algorithm (Cover and Hart 1967), which assigns to each test instance the majority class of its k closest training instances (its *k-neighborhood*).

Various metrics can be used to compute the distance between two instances (Giraud-Carrier and Martinez 1995, Wilson and Martinez 1997). With symbolic (nominal) attributes, as in our case, the *overlap* metric is a common choice. This metric counts the attributes where the two instances have different values. Given two instances $\vec{x}_i = \langle x_{i1}, x_{i2}, \ldots, x_{in} \rangle$ and $\vec{x}_j = \langle x_{j1}, x_{j2}, \ldots, x_{jn} \rangle$ their overlap distance is:

$$d(\vec{x}_i, \vec{x}_j) \equiv \sum_{r=1}^{n} \delta(x_{ir}, x_{jr}) \tag{2}$$

where

$$\delta(x, y) \equiv \begin{cases} 0, & \text{if } x = y \\ 1, & \text{otherwise} \end{cases}$$

In our experiments, we used the TiMBL memory-based learning software (Daelemans et al. 2000). TiMBL implements the basic k-NN classifier as above, except that the k-neighborhood is taken to contain *all* the training instances at the k closest *distances*, rather than the k closest instances. As a result, if there is more than one neighbor at some of the k closest distances, the neighborhood will contain more than k neighbors.

We have also experimented with different *attribute-weighting* and *distance-weighting* schemes. Unlike the basic k-NN classifier, where all the attributes are treated as equally important, attribute-weighting extensions assign different importance scores to the attributes, depending on how well they discriminate between the categories, and adjust the distance metric accordingly (Aha 1992, Wettschereck et al. 1995). Distance weighting takes memory-based learning one step further, by considering neighbors closer to the input instance as more important, assigning greater voting weight to them (Dudani 1976, Bailey and Jain 1978, Wettschereck 1994). This can reduce the sensitivity of the classifier to the k parameter, the neighborhood size. Attribute and distance weighting is considered in more detail in Section 6.

5. Cost-sensitive classification and evaluation

In anti-spam filtering there are two types of possible error: blocking a legitimate message (classifying a legitimate message as spam), and accepting a spam message (classifying a spam message as legitimate). Let $L \rightarrow S$ and $S \rightarrow L$ denote the two error types, respectively. Previous work on anti-spam filtering (Sahami et al. 1998; Androutsopoulos et al. 2000a, *Table 2*. Cost matrix used in this paper.

	Classified as legitimate	Classified as spam
Legitimate message	$c(L \rightarrow L) = 0$	$c(L \rightarrow S) = \lambda$
Spam message	$c(S \rightarrow L) = 1$	$c(S \rightarrow S) = 0$

2000c) has assumed that $L \rightarrow S$ is generally more severe an error than $S \rightarrow L$. This is based on the assumption that most users can tolerate a small percentage of mistakenly admitted spam messages, while they consider losing legitimate messages much more damaging. Invoking a decision-theoretic notion of cost (Lewis 1995), we assume that $L \rightarrow S$ is λ times more costly than $S \rightarrow L$. More precisely, we use the cost matrix of Table 2, where $L \rightarrow L$ and $S \rightarrow S$ denote the cases where the filter classifies correctly a legitimate or spam message, respectively. Here cost is intended to reflect the effort that a subscriber of the list, or the moderator if there is one, wastes to recover from the failures of the filter.[9] Correctly classifying a message ($L \rightarrow L$ or $S \rightarrow S$) is assigned zero cost, since in this case there is no effort to be wasted. Misclassifying a spam message ($S \rightarrow L$) is assigned unary cost, and misclassifying a legitimate message ($L \rightarrow S$) is taken to be λ times more costly. The value of the λ parameter depends on the usage scenario of the filter, as will be discussed below; for example, whether the filter deletes messages classified as spam, or simply flags them as low-priority.

Let $W_L(\vec{x})$ and $W_S(\vec{x})$ be the degrees of confidence of the classifier that instance $\vec{x}$ is legitimate or spam, respectively. For the basic k-NN algorithm of the previous section, a suitable measure of the confidence that a test instance belongs in a category (legitimate or spam) is the percentage of training instances in the k-neighborhood that belongs to that category. We classify a test instance as spam iff the expected cost of classifying it as legitimate is greater than the expected cost of classifying it as spam, i.e.,

$$W_S(\vec{x}) \cdot c(S \rightarrow L) + W_L(\vec{x}) \cdot c(L \rightarrow L) > W_L(\vec{x}) \cdot c(L \rightarrow S) + W_S(\vec{x}) \cdot c(S \rightarrow S)$$

According to the cost matrix, this is equivalent to $W_S(\vec{x}) > W_L(\vec{x}) \cdot \lambda$, i.e., we use the following criterion:

$$\vec{x} \mapsto S \quad \text{iff} \quad \frac{W_S(\vec{x})}{W_L(\vec{x})} > \lambda \tag{3}$$

$W_L(\vec{x})$ and $W_S(\vec{x})$ can be scaled to the [0,1] interval, so that their sum equals to 1. In this case, criterion (3) is equivalent to (4), where t is the classification threshold. A message exceeding this threshold is classified as spam; otherwise it is classified as legitimate.

$$\vec{x} \mapsto S \quad \text{iff} \quad W_S(\vec{x}) > t, \quad \text{with } t = \frac{\lambda}{1+\lambda}, \lambda = \frac{t}{1-t} \tag{4}$$

If $W_L(\vec{x})$ and $W_S(\vec{x})$ are accurate estimates of the conditional probabilities $P(C = legit \mid \vec{X} = \vec{x})$ and $P(C = spam \mid \vec{X} = \vec{x})$, respectively, criteria (3) and (4) achieve optimal results (Duda and Hart 1973).

In the work by Sahami et al. (1998), which considered anti-spam filters for personal mailboxes, the threshold t was set to 0.999, which corresponds to $\lambda = 999$, i.e., blocking a legitimate message was taken to be as bad as letting 999 spam messages pass the filter. This cost scenario introduces a very high bias for classifying messages as legitimate, which may be reasonable when blocked messages are deleted automatically without further processing, because most users would consider losing legitimate messages from their mailboxes unacceptable. For the sake of compatibility with previously published work, we include this cost scenario ($\lambda = 999$) in our experiments, though alternative scenarios are possible, especially in the case of anti-spam filtering for lists, where lower λ values are reasonable.

For example, an anti-spam filter may be used as a preprocessor, to reduce the number of spam messages that the moderator of a list has to consider. In this case, the moderator examines only messages that pass the filter. Rather than being deleted, a message blocked by the filter can be returned to its sender, explaining the reason of the return and asking the sender to repost the message to the moderator for manual approval (see also Hall 1998). The reply of the filter would not contain verbatim the address of the moderator (e.g., "Send the message to moderator*mylist+com, replacing the star with an at-sign and the plus with a dot."), to prevent the address of the moderator from being harvested by spam robots that process the replies they receive. In this scenario, $\lambda = 9$ ($t = 0.9$) seems more reasonable: blocking a legitimate message is penalized mildly more than letting a spam message pass, to account for the fact that recovering from a blocked legitimate message is more costly (counting the senders' extra work to repost it and their possible frustration) than recovering from a spam message that passed the filter (which requires the moderator to delete the message manually).

In a third scenario, the filter could simply flag messages it suspects to be spam (e.g., by adding a prefix like "[spam?]" to their subjects), without removing them, to help the subscribers of the list or newsgroup prioritize the processing of their incoming messages. In this case, $\lambda = 1$ ($t = 0.5$) seems reasonable, since none of the two error types is more significant than the other.

Apart from the classification threshold, cost must also be taken into account when defining evaluation measures. Cost-sensitive measures have been employed in recent TREC text filtering tasks (Hull and Robertson 2000), and were recently introduced in anti-spam filtering (Androutsopoulos et al. 2000a, 2000c). In cost-insensitive classification tasks, *accuracy* (*Acc*) and its complementary *error rate* (*Err* = 1 − *Acc*) are often used. In our context:

$$Acc = \frac{N_{L\to L} + N_{S\to S}}{N_L + N_S} \qquad Err = \frac{N_{L\to S} + N_{S\to L}}{N_L + N_S},$$

where $N_{Y\to Z}$ is the number of messages in category Y that the filter classified as Z, $N_L = N_{L\to L} + N_{L\to S}$ is the total number of legitimate messages to be classified, and $N_S = N_{S\to S} + N_{S\to L}$ the total number of spam messages.

Accuracy and error rate assign equal weights to the two error types ($L \to S$ and $S \to L$). To make these measures sensitive to cost, each legitimate message is treated, for evaluation purposes, as if it were λ messages. That is, when a legitimate message is blocked, this counts as λ errors; and when it passes the filter, it counts as λ successes. This leads to the following definitions of *weighted accuracy* (*WAcc*) and *weighted error rate* ($WErr = 1 - WAcc$):

$$WAcc = \frac{\lambda \cdot N_{L \to L} + N_{S \to S}}{\lambda \cdot N_L + N_S} \qquad WErr = \frac{\lambda \cdot N_{L \to S} + N_{S \to L}}{\lambda \cdot N_L + N_S}$$

In terms of cost, the numerator of *WErr* above is equal to the total cost incurred by using the filter on the $N_L + N_S$ messages, while the denominator is a normalizing factor equal to the incurred cost of the worst possible filter, which misclassifies all the messages. *WAcc* is simply the complement of the normalized incurred cost.

When one of the categories is more frequent than the other, as in our case and especially when $\lambda = 9$ or 999, the values of accuracy, error rate, and their weighted versions are often misleadingly high. To get a more realistic picture of a classifier's performance, it is common to compare its accuracy or error rate to those of a simplistic baseline approach. We consider the case where no filter is present as our baseline: legitimate messages are (correctly) never blocked, and spam messages (mistakenly) always pass. The weighted accuracy and weighted error rate of the baseline are:

$$WAcc^b = \frac{\lambda \cdot N_L}{\lambda \cdot N_L + N_S} \qquad WErr^b = \frac{N_S}{\lambda \cdot N_L + N_S}$$

The *total cost ratio* (*TCR*) allows the performance of a filter to be compared easily to that of the baseline:

$$TCR = \frac{WErr^b}{WErr} = \frac{N_S}{\lambda \cdot N_{L \to S} + N_{S \to L}}$$

Greater *TCR* values indicate better performance. For $TCR < 1$, not using the filter is better. If cost is proportional to wasted effort, *TCR* expresses how much effort is wasted to delete manually all spam messages when no filter is used (N_S), compared to the effort wasted to delete manually any spam messages that passed the filter ($N_{S \to L}$) plus the effort needed to recover from mistakenly blocked legitimate messages ($\lambda \cdot N_{L \to S}$).

We also present our results in terms of *recall* (R) and *precision* (P), which in our case are defined as below:

$$R = \frac{N_{S \to S}}{N_{S \to S} + N_{S \to L}} \qquad P = \frac{N_{S \to S}}{N_{S \to S} + N_{L \to S}}$$

Recall measures the percentage of spam messages that the filter manages to block (intuitively, its effectiveness), while precision measures the degree to which the blocked messages are indeed spam (intuitively, the filter's safety). Despite their intuitiveness, comparing different filter configurations using recall and precision is difficult: each filter configuration yields a pair of recall and precision results, and without a single combining measure, like *TCR*, that incorporates the notion of cost, it is difficult to decide which pair is better.[10] In our experiments, the aim was to optimize the filter's performance in terms of *TCR*.

6. Experimental results

We now proceed with the presentation of our experimental results, providing at the same time more information on the attribute-weighting and distance-weighting extensions to the basic k-NN classifier of Section 4 that we have considered.

We first investigated the impact of *attribute weighting*, using a weighting scheme based on Information Gain (*IG*, Section 3). We performed two sets of experiments on Ling-Spam, with and without attribute weighting. In each set of experiments, three different cost scenarios were tested, corresponding to three different values of λ, as discussed in Section 5. In each scenario, we varied the number of selected attributes from 50 to 700 by 50, each time retaining the attributes with the highest *IG* scores. Furthermore, each experiment was repeated with three different neighborhood sizes ($k = 1, 2$, and 10); the results suggested that there was no need to try more values of k to conclude that attribute weighting has a positive effect.

In the second set of experiments, we investigated the effect of *distance weighting*. Using the *IG*-based attribute weighting scheme, various distance-sensitive voting schemes were examined, for each cost scenario (λ value) and number of selected attributes, with the size of the neighborhood (k) now set to the maximum size of those considered (10). In a third set of experiments, we examined the effect of *dimensionality* (number of selected attributes) and *neighborhood size* (k), using the best-performing configuration in terms of attribute weighting and distance-weighting. Finally, in a fourth set of experiments, we examined the effect of the *training corpus size*, for each cost scenario, using the best configuration of the previous experiments.

In all of the experiments, *10-fold stratified cross-validation* was employed (Section 2). *WAcc* was averaged over the 10 iterations, and *TCR* was computed as $WErr^b$ over the average *WErr* (Section 5).

6.1. Attribute weighting

The basic k-NN classifier assigns equal importance to all the attributes. In real-world applications, however, irrelevant or redundant attributes are often included in the representation of the instances. This causes the classification accuracy of k-NN to degrade, unless appropriate weights are assigned to the attributes, corresponding to their relevance to the classification task. The distance metric, that computes the distance between two instances, has to be adjusted accordingly, to incorporate the attribute weights. Equation (2) becomes:

$$d(\vec{x}_i, \vec{x}_j) \equiv \sum_{r=1}^{n} w_r \cdot \delta(x_{ir}, x_{jr}), \tag{5}$$

where w_r is the weight assigned to r-th attribute.

6.1.1. Attribute-weighting scheme. Information Gain (*IG*) was presented in Section 3 as our attribute selection function. The same function can be used for attribute weighting. An equivalent expression of (1) in information theoretic terms is the following:

$$IG(X, C) = H(C) - \sum_{x \in \{0,1\}} P(X = x) \cdot H(C \mid X = x), \tag{6}$$

where $H(C)$ is the *entropy* of variable C, denoting the category. $H(C)$ measures the uncertainty on the category of a randomly selected instance, and is defined as:

$$H(C) = -\sum_{c \in \{spam, legit\}} P(C = c) \cdot \log_2 P(C = c). \tag{7}$$

$H(C \mid X = x)$ is defined similarly, replacing $P(C = c)$ in (7) by $P(C = c \mid X = x)$. $H(C \mid X = x)$ measures the uncertainty on the category given the value of attribute X. Equation (6) subtracts from the entropy of the category ($H(C)$) the expected value of the entropy when the value of X is known, averaged over all the possible values of X. *IG* is therefore a measure of how much knowing the value of X reduces the entropy of C. The larger the reduction, the more useful X is in predicting C.

The TiMBL software that we used (Section 4) supports both *IG* and the standard no-weighting scheme (equal weights for all attributes), hereafter *EW*. We also experimented with *Gain Ratio* (Quinlan 1986), an attribute weighting scheme intended to correct the bias of *IG* towards attributes with many uniformly distributed values. In our case, however, where all attributes are binary, there is nothing to be gained from using Gain Ratio instead of *IG*, and the results that we obtained confirm this. Recent versions of TiMBL support two additional attribute-weighting measures, namely chi-squared and shared variance. Although we did not explore these measures thoroughly, the experiments we conducted on some randomly selected settings showed no significant difference from *IG* in agreement with Yang and Pedersen (1997).

6.1.2. Results of the attribute-weighting experiments. Figures 1 and 2 show the *TCR* of 10-NN ($k = 10$) for $\lambda = 1$ and $\lambda = 999$, respectively, for *IG* and *EW* and varying numbers

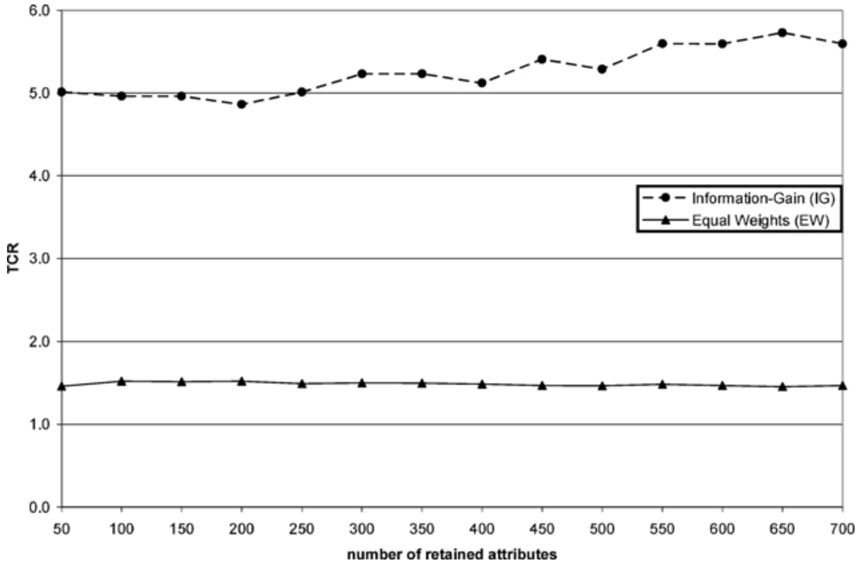

Figure 1. *TCR* of 10-NN for $\lambda = 1$ and two attribute-weighting schemes.

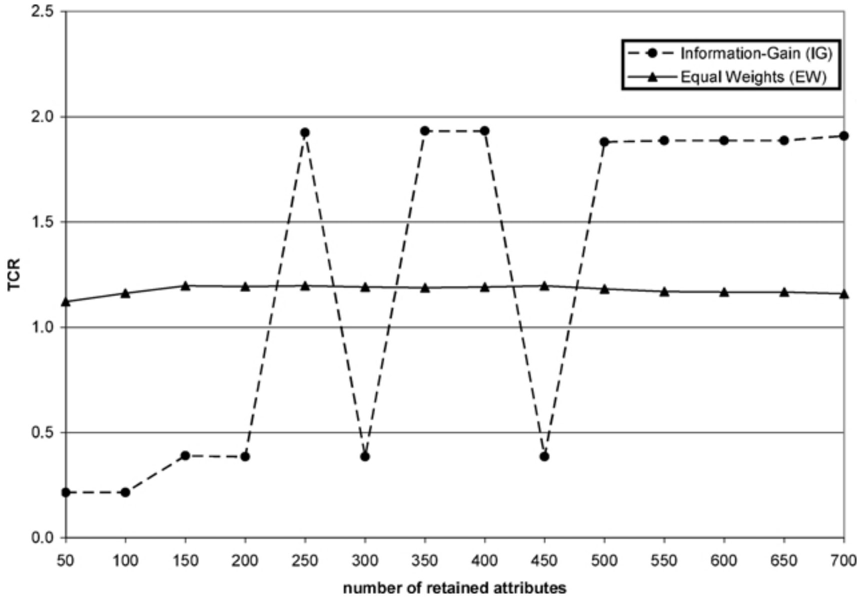

Figure 2. *TCR* of 10-NN for $\lambda = 999$ and two attribute-weighting schemes.

of attributes. The corresponding curves for $\lambda = 9$ are similar to those for $\lambda = 1$, and are omitted to save space.

For $\lambda = 1$ (as well as $\lambda = 9$), the conclusion is that 10-NN with *IG* outperforms 10-NN with *EW* (no attribute weighting). The same pattern was observed in the experiments that we performed with $k = 1$ and $k = 2$. Another interesting pattern is the dependence on the dimensionality: when using *IG*, the performance improved continuously as we retained more attributes; in contrast, with *EW* the number of attributes does not seem to affect the performance.

For $\lambda = 999$ (figure 2), the picture is quite different. The distinguishing characteristic of this scenario is the unstable behavior of the filter, which was already noticed in our previous work with the Naive Bayes classifier (Androutsopoulos et al. 2000a, 2000c). The reason is that $L \rightarrow S$ errors are penalized so heavily, that a *single* blocked legitimate message causes the baseline to outperform the memory-based filter in terms of *WAcc*, and *TCR* to drop below 1. Given this instability, the objective in this scenario is to select a *reliable* configuration, which maintains *TCR* constantly above 1, even if that configuration does not achieve the highest *TCR* score.

Bearing the above goal in mind, the most reliable option for 10-NN when $\lambda = 999$ is to use no attribute weighting (*EW*), since it attains consistently $TCR > 1$. However, its recall does not exceed 17%, which is a rather low performance. On the other hand, *IG* seems to be less reliable, while its recall reaches 47%, blocking almost half of the spam messages in this demanding scenario. Unlike 10-NN, 1-NN and 2-NN do not reach the baseline ($TCR < 1$) in this scenario for any of the attribute-weighting schemes. The general impression formed by these results is that high dimensionality and a large k-neighborhood provide more reliably

TCR scores above the baseline, when $\lambda = 999$. This impression is strengthened by the results presented in the following sections.

6.1.3. Interpretation of the results of the attribute-weighting experiments. The first striking observation in figure 1 is the poor performance of 10-NN when no attribute weighting is used (*EW*). This may seem odd, as k-NN has been used successfully in many domains without attribute weighting. This phenomenon can be explained by the fact that *EW* led to large numbers of instances at equal distances in the k-neighborhood. (The reader is reminded that in our experiments the k-neighborhood comprises all the neighbors at the k closest *distances*; see Section 4.) For example, 10-NN with 700 attributes and *EW* gave rise to k-neighborhoods of typically 100–200 instances, while the respective number of instances for 50 attributes often exceeded 2000! Since the vast majority of messages are legitimate, it is reasonable that within neighborhoods of hundreds or thousands of instances, most of them will also be legitimate. This forces 10-NN to classify almost all of the messages as legitimate, which is why its performance is very close to that of the baseline. A similar explanation can be given to the fact that for $\lambda = 999$ (figure 2), the behavior of 10-NN with *EW* is closer to the behavior of the baseline, compared to the cases where *IG* is used.

The question that arises is why there are so many instances in the k-neighborhood when *EW* is used. The answer lies in the representation of the instances and the use of the overlap distance metric (Eq. (2)). As mentioned in Section 4, the metric counts the number of attributes where the instances have different values. At the same time, the representation results in sparse instance vectors, i.e., vectors with many zeros, indicating the absence of attribute-words in the document. Thus, it is frequently the case that many messages differ in the same *number* of features, but not necessarily the same features. With *EW*, all these messages are considered equally distant from an incoming new message. On the other hand, *IG* and most attribute-weighting functions avoid this problem: since every attribute weighs differently (Eq. (5)), two instances are equally distant from an incoming message practically only when they are identical; and finding two different messages at the same distance from a new message becomes further unlikely as the dimensionality increases.

A second issue to look into is the effect of dimensionality on the two weighting schemes. In the case of *EW*, the large neighborhoods that are formed are also responsible for the stability of the curves with different numbers of retained attributes (figures 1 and 2): the majority class (legitimate) prevails most of the times, and the filter's behavior is very similar to that of the baseline, regardless of the selected attributes. With *IG*, there is an exponential decrease in the weight of the last retained attribute, as more attributes are added; this can be seen in figure 3. The overall marginal increase in performance (figures 1 and 2) as the number of attributes increases, suggests that the weights assigned to the attributes are appropriate. That is, using the *IG* weights, the classifier seems to be taking advantage of even inferior attributes, by assigning them appropriately lower importance.

The main conclusion of the discussion above is that *IG* attribute-weighting has a positive effect. Hence, we use this scheme in the experiments of the following sections.

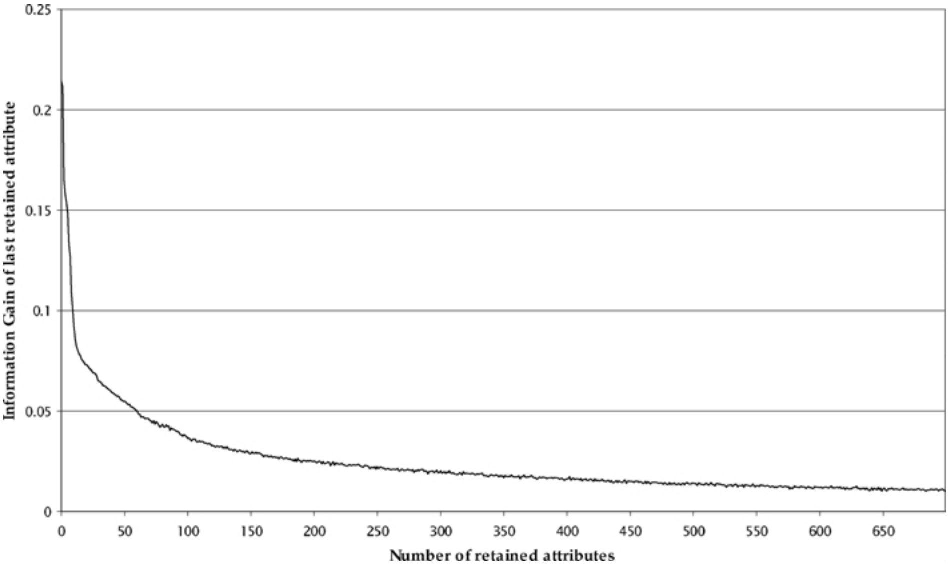

Figure 3. Information Gain scores of best word attributes in Ling-Spam.

6.2. Distance weighting

Having chosen to use the *IG* attribute-weighting scheme, we now examine the effect of distance-weighting. Distance-weighting does not treat all the instances in a k-neighborhood as equally important, but weighs them according to their distance from the incoming instance. The advantages of distance weighting, or else weighted voting, in memory-based methods have been discussed extensively in the literature (Dudani 1976, MacLeod et al. 1987). The main benefit is that distance weighting reduces the sensitivity of k-NN to the choice of k. A k value that may be suitable for sparsely populated regions may be unsuitable for dense regions, generating in the latter case neighborhoods that contain many irrelevant instances. Weighted voting undervalues distant neighbors, without ignoring them completely, in order to adapt the effective size of the neighborhood to the local distribution of instances.

6.2.1. Distance-sensitive voting schemes. Various voting schemes have been proposed for distance weighting. We experimented with simple voting schemes that use a distance-sensitive function to weigh the vote of each instance in the k-neighborhood. The first scheme uses a linear function to weigh the votes:

$$f_0(d) = d_{\max} - d,$$

where $f_0(d)$ is the weight assigned to a neighbor at distance d from the incoming instance, and $d_{\max}$ is the maximum obtainable distance. The maximum distance occurs when two instances differ in every attribute; hence, it is equal to the sum of all attribute weights. The other voting schemes use hyperbolic functions, as below:[11]

$$f_n(d) = \frac{1}{d^n}, n = 1, 2, 3, \ldots .$$

In all of the voting schemes mentioned above, when one or more neighbors are identical to the incoming instance (i.e., $d = 0$), we classify the incoming instance to the majority class of the identical instances, ignoring all other neighbors.

With distance-weighting, the confidence level $W_c(\vec{x})$ that the incoming instance $\vec{x}$ belongs to class c, is computed by the following formula:

$$W_c(\vec{x}) = \sum_i f_n(d(\vec{x}, \vec{x}_i)) \cdot \bar{\delta}(c, C(\vec{x}_i)),$$

where $\bar{\delta}(x, y) = 1 - \delta(x, y)$, $\vec{x}_i$ ranges over the instances in the k-neighborhood, and $C(\vec{x}_i)$ is the class of neighbor $\vec{x}_i$. This formula simply weighs the contribution of each neighbor by its distance from the incoming instance. As in the basic k-NN classifier, the confidence levels for the two categories can be scaled to the [0,1] interval, so that their sum equals to 1 (Section 5).

6.2.2. Results of the distance-weighting experiments. Figure 4 shows the *TCR* of 10-NN for $\lambda = 1$. Each curve corresponds to one voting scheme, and there is an additional curve for

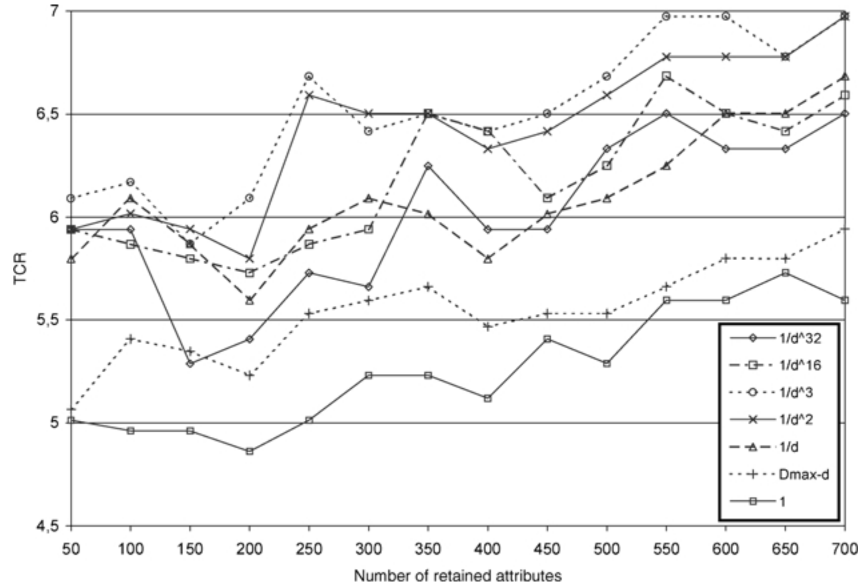

Figure 4. *TCR* of 10-NN for $\lambda = 1$ and different distance-sensitive voting schemes (using *IG* for attribute weighting).

majority voting (i.e., no distance weighting). The selected value for k ($k = 10$) is the highest of those considered in our experiments, because distance weighting is usually combined with a large value for k: one of the principal reasons for employing distance weighting is the smoothing of the classifier's performance for varying k, so that the optimal selection of its value is less crucial. The corresponding figure for $\lambda = 9$ is similar to that for $\lambda = 1$, with respect to the conclusions drawn, and is therefore omitted. Finally, for $\lambda = 999$ the curves are almost uninfluenced by the use of distance weighting, and the corresponding figure is much like figure 2.

Figure 4 shows clearly the improvement brought by distance weighting. The improvement is greater when distant neighbors are mildly undervalued. TCR clearly improves when moving from $f_1(d)$ to $f_2(d)$ to $f_3(d)$, where the best results are obtained. However, the two highest curves, $f_2(d)$ and $f_3(d)$, are quite close to each other, suggesting that $f_n(d)$ functions for $n > 3$ will not lead to further significant improvements in performance. In fact, the results obtained for $f_{16}(d)$ and $f_{32}(d)$, as well as higher values of n that are not shown in figure 4 for the sake of readability, show a reduction in performance as distant neighbors get heavily undervalued.

6.2.3. Interpretation of the results of the distance-weighting experiments. The fact that performance improves as close neighbors are slightly overvalued suggests that local classification is preferable in this task. It is interesting to note that an important contributor to the success of the distance-sensitive voting schemes that we explored is that when one or more neighbors are identical to the incoming instance, the incoming instance is classified to the majority class of the identical instances, ignoring all the other neighbors; this produces almost always the correct results. Even when there are no identical neighbors, functions that prioritize the nearest neighbors reward the local minority class, if its members are closer to the incoming instance. (The good performance of the *IG* attribute-weighting scheme helps further, by bringing the "right" instances closer to the incoming instance.)

It should be stressed that mild distance-weighted voting differs from a reduction of the neighborhood size, in that it does not ignore distant neighbors altogether. In other words, reducing the value of k does not improve performance. This is also the reason for the reduced performance obtained by voting schemes that greatly undervalue distant neighbors. As the value of n in the $f_n(d)$ function becomes very large, the behavior of the method approaches the effect of using 1-NN, which is a suboptimal choice.

6.3. Neighborhood size and dimensionality

The results of the distance-weighting experiments have indicated a clear superiority of voting schemes that favor mildly close neighbors. The question that arises is whether the value of the k parameter, which determines the size of the neighborhood, can still affect the performance of the classifier when such voting schemes are employed. We continued our investigation towards that direction, by examining the influence of k, in combination with the dimensionality of the instance space. In all the experiments below, we used *IG* for attribute weighting and $f_3(d) = 1/d^3$ for distance weighting, following the conclusions of the previous experiments.

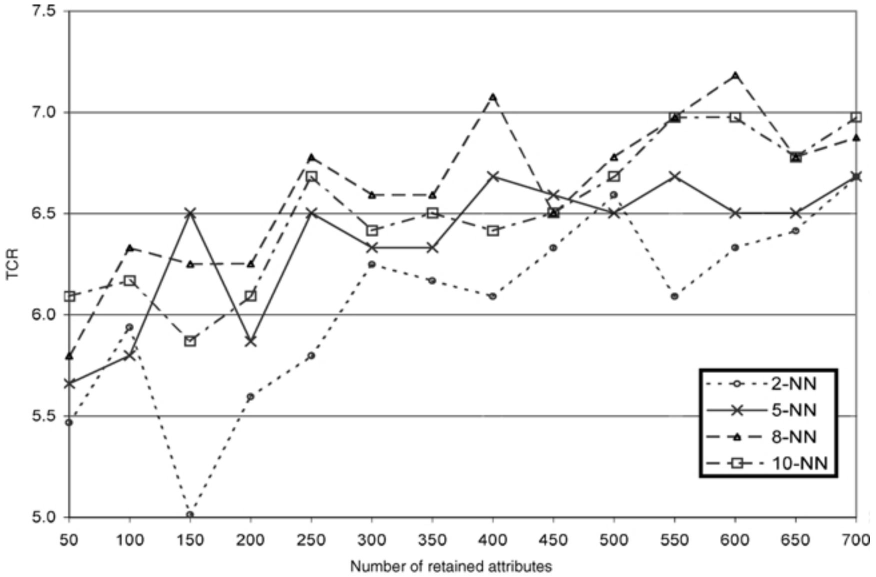

Figure 5. *TCR* of k-NN for $\lambda = 1$ and different k values (using *IG* for attribute weighting and $1/d^3$ for distance weighting).

6.3.1. Results of the neighborhood size and dimensionality experiments. Figures 5–7 show some representative curves for various k-neighborhood sizes in each one of the cost scenarios ($\lambda = 1$, 9 and 999), respectively. Although we carried out experiments for every k from 1 to 10, we present only some representative curves, for the sake of brevity. The discussion below, however, refers to the whole set of results.

In figure 5, a general trend involving the dimensionality and the performance of the classifier is observed: as the number of retained attributes increases, so does *TCR*. The relationship between the size of the neighborhood and the performance of the classifier is less clear: *TCR* seems to be improving as k increases, up to $k = 8$, when it starts to deteriorate slightly. However, the extensive overlap of the curves does not allow any safe conclusions to be drawn.

Figure 6 presents the two best and two worst curves for $\lambda = 9$. The behavior of the classifier is different here, in that the best performance occurs for very small neighborhoods ($k = 2$, 3). As the neighborhood grows, *TCR* declines gradually, but steadily. Interestingly enough, 1-NN performs the worst, which can be attributed to the fact that no distance weighting can be used in this case.

Finally, in figure 7, we observe the same steep fluctuations of *TCR* as in figure 2, indicating transitions from below baseline performance to above, and vice versa. One exception is k-NN for $k < 4$, which is almost always below the baseline. Another interesting phenomenon is the fact that 4-NN achieves the highest *TCR* globally for 250 attributes, and yet it is not useful, as its performance is not steadily above the baseline; in practice, it is not possible to predict accurately the exact dimensionality of such a narrow peak. The general conclusion is that satisfactory performance is obtained reliably using a high dimensionality and a large k-neighborhood.

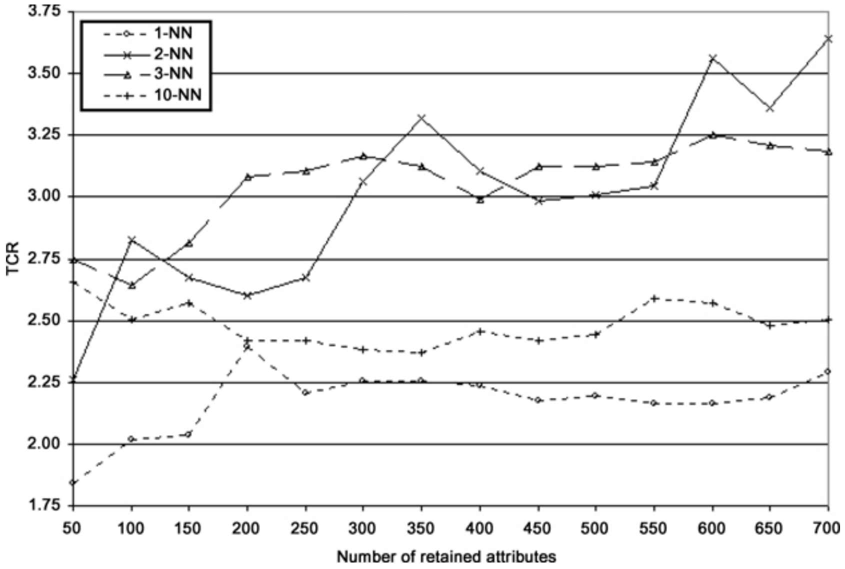

Figure 6. *TCR* of k-NN for $\lambda = 9$ and different k values (using *IG* for attribute weighting and $1/d^3$ for distance weighting).

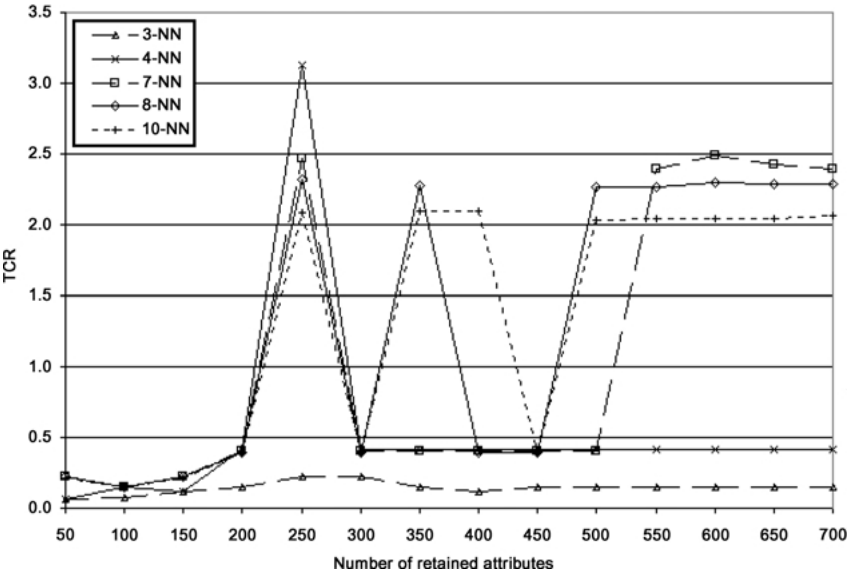

Figure 7. *TCR* of k-NN for $\lambda = 999$ and different k values (using *IG* for attribute weighting and $1/d^3$ for distance weighting).

6.3.2. Interpretation of the results of the neighborhood size and dimensionality experiments. The conclusions to be drawn here are less clear than those in the previous experiments. The optimal value of k depends heavily on the selected scenario (λ parameter), and also correlates with the dimensionality in an unintelligible way. The experiments showed that for a given scenario, there is no clear ranking of k-NN classifiers for various values of k. Furthermore, the effect of the number of retained attributes does not follow a coherent pattern. For example, k_1-NN may be better than k_2-NN for 50–250 retained attributes, k_2-NN may outperform k_1-NN for 300–400 attributes, and in a third interval (e.g., 450–600 features) k_1-NN may again be better than k_2-NN. To some extent this confusing picture is justified by the use of the distance-weighting function, which reduces significantly the dependence of the classifier on the choice of k. It should also be stressed that the difference between the best and worst performing configuration is much smaller in this set of experiments than in the previous two sets. The performance of the worst classifier in figure 5 does not fall below $TCR = 5$, which is comparable only to the best classifier of figure 1, where no distance weighting was used.

6.3.3. Best results overall and comparison to a Naive Bayes filter. It is interesting to translate these results into recall and precision, which are perhaps more intuitive than the combined *TCR* measure. Table 3 summarizes the best configurations witnessed for each scenario. In every case, *IG* was used for attribute weighting and $f_3(d) = 1/d^3$ for distance weighting. A recall around 89% and a precision of over 97% attained for $\lambda = 1$ make a satisfactory, if not sufficient, performance. The gain from moving to the second scenario with $\lambda = 9$ is, on the other hand, questionable. A small increase of precision by 1.4% is accompanied by a nearly five times greater decrease in recall. However, this might be acceptable, as almost no legitimate messages are misclassified. In the third scenario ($\lambda = 999$), the filter's safety (not blocking legitimate messages) becomes the crucial issue, and therefore precision must be kept to 100% at any cost. Ignoring abrupt peaks (e.g., the peaks at 250 attributes in figure 7), which do not correspond to stable configurations with respect to minor changes to the number of retained attributes, the highest recall is approximately around 60% (with $k = 7$ and 600 attributes); this is notable though far from perfect.

The results presented in Table 3 are directly comparable to the results of our earlier work with the Naive Bayes classifier on the same corpus. Table 4 reproduces our earlier best results, as presented in (Androutsopoulos et al. 2000b). The comparison of the two tables shows that the memory-based approach compares favourably to the Naive Bayes classifier. In terms of *TCR*, the memory-based classifier is clearly better for $\lambda = 1$, and slightly worse for $\lambda = 9$ and slightly better again for $\lambda = 999$. In the strict scenario ($\lambda = 999$), it should be noted that the performance of the Naive Bayes classifier is very unstable, i.e., the result

Table 3. Best configurations per usage scenario and the corresponding performance.

λ	k	Dimensionality	Recall (%)	Precision (%)	*TCR*
1	8	600	88.60	97.39	7.18
9	2	700	81.93	98.79	3.64
999	7	600	59.91	100	2.49

Table 4. Best results of the Naive Bayes filter.

λ	Dimensionality	Recall (%)	Precision (%)	*TCR*
1	100	82.35	99.02	5.41
9	100	77.57	99.45	3.82
999	300	63.67	100	2.86

shown in Table 4 corresponds to a very narrow peak with respect to the number of retained attributes. Apart from that peak, the Naive Bayes classifier never exceeded $TCR = 1$ for $\lambda = 999$. In contrast, as already discussed, there are intervals where the memory-based classifier achieves *TCR* steadily above 1. Examining recall and precision, it is clear that the memory-based classifier improves recall for $\lambda = 1$ and $\lambda = 9$, at a small cost of precision.

6.4. *Corpus size*

Having examined the basic parameters of the classifier, we now turn to the size of the training corpus, which was kept fixed to 2603 messages (90% of the whole corpus) in the experiments presented above. As before, in every ten-fold experiment the corpus was divided into ten parts, and a different part was reserved for testing at each iteration. From each of the remaining nine parts, only $x\%$ was used for training, with x ranging from 10 to 100 by 10. For every cost scenario, the best configuration was employed (as in Table 3), with *IG* attribute weighting and $f_3(d) = 1/d^3$ used for distance weighting.

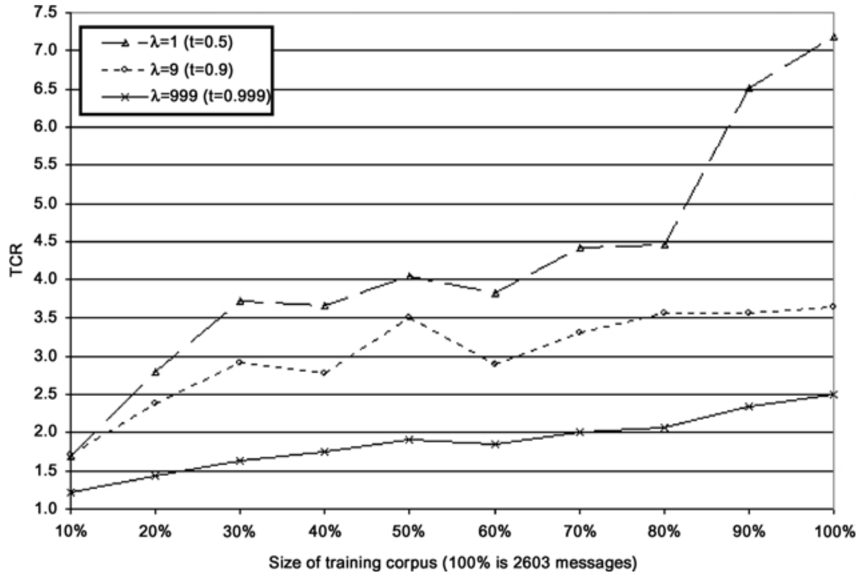

Figure 8. *TCR* for variable sizes of training corpus (using *IG* for attribute weighting and $1/d^3$ for distance weighting).

Figure 8 presents the resulting learning curves in terms of *TCR*. In all three scenarios, the diagram shows a clear trend of improvement as the size of the training corpus increases. By studying the corresponding recall and precision measures, we observed that the increase in performance is primarily due to an increase in recall. For example, the transition from 80 to 100% of the training corpus for $\lambda = 1$ raises recall by nearly 10%, decreasing at the same time precision by only 1.6%.

The increase of *TCR* in figure 8 is generally mild, with the exception of a leap from 4.5 to 6.5 for $\lambda = 1$. There is no indication that the learning curves have approached an asymptote for any size of the training set, which suggests that a larger corpus might give even better results. This is particularly encouraging for the strict scenario ($\lambda = 999$), where there is still a large scope for improvement. It is also notable that, when $\lambda = 999$, *TCR* remains above the baseline for all the sizes of the training corpus. In contrast, the Naive Bayes filter that we examined in previous work (Androutsopoulos et al. 2000a) had reached $TCR > 1$ only for 100% of the training corpus. These findings indicate that a memory-based anti-spam filter for mailing lists may be viable in practice, even when absolute precision is required.

7. Conclusions

In this paper, we presented a thorough empirical evaluation of a memory-based approach to anti-spam filtering for mailing lists. In contrast to anti-spam filters for personal mailboxes, anti-spam filters for mailing lists, which apply equally well to Usenet-like newsgroups, are more tractable in terms of evaluation, since publicly available archives can be used as standard benchmarks without privacy violations. We introduced cost-sensitive evaluation measures, along with indicative cost scenarios, discussing how these scenarios relate to additional safety nets that may be available in mailing lists, such as using the filter as an aid to a moderator. Our experimental results show that the use of a memory-based filter can be justified, even in the strictest scenario, where the blocking of a legitimate message is practically unacceptable. Furthermore, our memory-based filter compares favorably to the probabilistic classifier that we used in our earlier work on this problem. Overall, then, our work indicates that anti-spam filtering using text categorization techniques is feasible, at least for mailing lists and newsgroups, despite the fact that spam messages cover a very wide spectrum of topics, and hence are much less homogeneous than other categories that have been considered in the past. Further improvements may be possible by incorporating non-textual attributes, though this requires an additional manual preprocessing stage to devise candidate attributes of this kind.

The most important contribution of this article is the exploration of various parameters of the memory-based method, such as attribute weighting, distance weighting, and neighborhood size. Our experiments show that an attribute-weighting scheme based on Information Gain has a positive effect, and that voting schemes that mildly devalue distant neighbors are beneficial. We have also shown that by using the right attribute-weighting and distance-weighting schemes, the size of the neighborhood of the incoming instances becomes less important. In addition to the parameters of the classification method, we have explored two important parameters of the particular classification task: the dimensionality and the size of the training corpus. Regarding the dimensionality of the task, which is determined by the number of retained attributes after the initial selection, our results show that its effect on classification performance is positive when using attribute-weighting, i.e., the performance improves as the number of retained attributes increases. Similarly, the performance improves as the size of the training corpus increases, which is an indication that a larger training corpus might lead to even better results.

The experiments presented here have opened a number of interesting research issues, which we are currently examining. In the context of the memory-based classifier, we are examining non-binary representations of the messages, by taking into account the frequency of a word within a message. The standard TFIDF weighting, with cosine normalization to cope with variable document length, may be tried as well. Additionally, we would like to examine other attribute-weighting functions and their relationship to the chosen representation of instances. Weighted voting can also benefit from functions that do not depend on the absolute distance from the input instance, but take into account the local properties of the neighborhood, as shown by Zavrel (1997). Finally, our main interest is in combining memory-based, probabilistic, and other induced classifiers, within a classifier ensemble framework, such as stacking (Wolpert 1992). Initial results (Sakkis et al. 2001) indicate that this can improve anti-spam filtering performance further.

Acknowledgments

The authors wish to thank the anonymous reviewers for their constructive comments.

Notes

1. Consult http://www.cauce.org/, http://spam.abuse.net/, and http://www.junkemail.org/ for further information on UCE and related legal issues.
2. See http://www.esi.uem.es/~jmgomez/spam/index.html for a collection of resources related to machine learning and anti-spam filtering.
3. An on-line bibliography on cost-sensitive learning can be found at http://home.ptd.net/~olcay/cost-sensitive.html.
4. See http://about.reuters.com/researchandstandards/corpus/.
5. An alternative path is to share suitably encrypted mailboxes, which will allow different representation and learning techniques to be compared, while still maintaining privacy. We have recently started to explore this path as well (Androutsopoulos et al. 2000b).
6. The Linguist list is archived at http://listserv.linguistlist.org/archives/linguist.html.
7. Spambase was created by M. Hopkins, E. Reeber G. Forman and J. Suermondt. It is available from http://ww.ics.uci.edu/~mlearn/MLRepository.html.
8. We used *morph*, a lemmatizer included in the GATE system. See http://www.dcs.shef.ac.uk/research/groups/nlp/gate/.
9. An alternative analysis, e.g., from the view-point of an ISP provider, could also take into account the cost of the bandwidth that is wasted by mistakenly admitting spam messages.
10. The F-measure, which is often used in text classification to combine recall and precision (e.g., Riloff and Lehnert 1994), cannot be used here, because it is unclear how its weighting factor (β parameter) relates to the cost ratio of the two error types (λ) in our experiments.
11. $f_1(d)$ was proposed by Dudani (1976). $f_0(d)$ is a simplified version of a similar function also proposed by Dudani (1976). $f_n(d)$, where $n > 1$, are our own stricter versions, that follow naturally from $f_1(d)$.

References

Aha WD (1992) Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. International Journal of Man-Machine Studies, 36:267–287.

Aha WD, Kibler D and Albert MK (1991) Instance-based learning algorithms. Machine Learning, 6:37–66.

Androutsopoulos I, Koutsias J, Chandrinos KV, Paliouras G and Spyropoulos CD (2000a) An evaluation of Naive Bayesian anti-spam filtering. In: Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000), Barcelona, Spain, pp. 9–17.

Androutsopoulos I, Koutsias J, Chandrinos KV and Spyropoulos CD (2000b) An experimental comparison of Naive Bayesian and keyword-based anti-spam filtering with encrypted personal e-mail messages. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000), Athens, Greece, pp. 160–167.

Androutsopoulos I, Paliouras G, Karkaletsis V, Sakkis G, Spyropoulos CD and Stamatopoulos P (2000c) 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 2000), Lyon, France, pp. 1– 3.

Bailey T and Jain AK (1978) A note on distance-weighted *k*-nearest neighbor rules, IEEE Transactions on Systems, Man, and Cybernetics, 8(4):311–313.

Cohen WW (1996) Learning rules that classify e-mail. In: Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access, Palo Alto, US, pp. 18 –25.

Cohen WW and Singer Y (1999) Context-sensitive learning methods for text categorization. ACM Transactions on Information Systems, 17(2):141–173.

Cover T and Hart P (1967) Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13:21–27.

Cranor LF and LaMacchia BA (1998) Spam!, Communications of ACM, 41(8):74–83.

Cristianini N and Shawe-Taylor J (2000) An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press.

Daelemans W, Van den Bosch A and Weijters A (1997) IGTREE: Using trees for compression and classification in lazy learning algorithms. Artificial Intelligence Review, 11:407–423.

Daelemans W, Zavrel J, van der Sloot K and van den Bosch A (2000) TiMBL: Tilburg Memory Based Learner, version 3.0, Reference Guide. ILK, Computational Linguistics, Tilburg University. http:/ilk.kub.nl/∼ilk/papers.

Drucker HD, Wu D and Vapnik V (1999) Support vector machines for spam categorization. IEEE Transactions On Neural Networks, 10(5):1048–1054.

Duda RO and Hart PE (1973) Bayes decision theory. Chapter 2 in Pattern Classification and Scene Analysis, John Wiley, pp. 10–43.

Dudani AS (1976) The distance-weighted *k*-nearest neighbor rule. IEEE Transactions on Systems, Man and Cybernetics, 6(4):325–327.

Giraud-Carrier C and Martinez RT (1995) An efficient metric for heterogeneous inductive learning applications in the attribute-value language. Intelligent Systems, pp. 341–350.

Gómez Hidalgo JM, Maña López M and Puertas Sanz E (2000) Combining text and heuristics for cost-sensitive spam filtering. In: Fourth Computational Natural Language Learning Workshop, CoNLL-2000, Lisbon, Portugal, pp. 99–102.

Hall RJ (1998) How to avoid unwanted email. Communications of ACM, 41(3):88–95.

Hull D and Robertson S (2000) The TREC-8 filtering track final report. In: Proceedings of the Eighth Text Retrieval Conference (TREC-8), NIST Special Publication 500-246, pp. 35–56.

Joachims T (1997) A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In: Proceedings of ICML-97, 14th International Conference on Machine Learning, Nashville, US, pp. 143–151.

Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Morgan Kaufmann, pp. 1137–1143.

Lang K (1995) Newsweeder: Learning to filter netnews. In: Proceedings of the 12th International Conference on Machine Learning, Stanford, CA, pp. 331–339.

Lewis D (1995) Evaluating and optimizing autonomous text classification systems. In: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '95), New York, pp. 246–254.

MacLeod ESJ, Luk A and Titterington DM (1987) A re-examination of the distance-weighted k-nearest neighbor classification rule. IEEE Transactions on Systems, Man, and Cybernetics, 17(4):689–696.

Mitchell TM (1997) Machine Learning. McGraw-Hill.

Pantel P and Lin D (1998) SpamCop: A spam classification and organization program. Learning for Text Categorization—Papers from the AAAI Workshop, Madison Wisconsin, pp. 95–98. AAAI Technical Report WS-98-05.

Payne TR and Edwards P (1997) Interface agents that learn: An investigation of learning issues in a mail agent interface. Applied Artificial Intelligence, 11(1):1–32.

Quinlan JR (1986) Induction of Decision Trees. Machine learning, 1(1):81–106.

Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, California.

Rocchio, J. (1971). Relevance feedback information retrieval. In: Salton G, ed., The Smart Retrieval System—Experiments in Automatic Document Processing, Prentice-Hall, Englewood Cliffs, NJ, pp. 313–323.

Sahami M, Dumais S, Heckerman D and Horvitz E (1998) A Bayesian approach to filtering junk e-mail. Learning for Text Categorization—Papers from the AAAI Workshop, Madison Wisconsin, pp. 55–62. AAAI Technical Report WS-98-05.

Sakkis G, Androutsopoulos I, Paliouras G, Karkaletsis V, Spyropoulos CD and Stamatopoulos P (2001) Stacking classifiers for anti-spam filtering of e-mail. In: *Proceedings of the Sixth Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, Carnegie Mellon University, Pittsburgh, PA, USA, pp. 44–50.

Salton G and Buckley C (1988) Term-weighting approaches in automatic text retrieval. Information Processing and Management, 24(3):513–523.

Salton G and McGill MJ (1983) Introduction to Modern Information Retrieval. McGraw-Hill.

Schapire RE and Singer Y (2000) BoosTexter: A boosting-based system for text categorization. Machine Learning, 39(2/3):135–168.

Sebastiani F (2001) Machine Learning in Automated Text Categorization. Revised version of Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy.

Wettschereck D (1994) A Study of Distance-Based Machine Learning Algorithms. PhD thesis, Oregon State University.

Wettschereck D, Aha WD and Mohri T (1995) A Review and Comparative Evaluation of Feature Weighting Methods for Lazy Learning Algorithms, Technical Report AIC-95-012, Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.

Wilson DR (1997) Advances in Instance-Based Learning Algorithms. PhD thesis, Brigham Young University.

Wilson DR and Martinez RT (1997) Improved heterogeneous distance functions. Journal of Artificial Intelligence Research, 6(1):1–34.

Wolpert D (1992) Stacked generalization. Neural Networks, 5(2):241–260.

Yang Y and Pedersen JO (1997) A comparative study on feature selection in text categorization. In: Proceedings of ICML-97, 14th International Conference on Machine Learning, Nashville, US, pp. 412–420.

Zavrel J (1997) An empirical re-examination of weighted voting for k-NN. In: Proceedings of the 7th Belgian-Dutch Conference on Machine Learning (BENELEARN-97), Tilburg, The Netherlands.