UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

EVALUATION OF THE HIGHEST PROBABILITY SVM NEAREST NEIGHBOR CLASSIFIER WITH VARIABLE RELATIVE ERROR COST

Enrico Blanzieri and Anton Bryl

May 2007

Technical Report # DIT-07-025

Evaluation of the Highest Probability SVM Nearest Neighbor Classifier with Variable Relative Error Cost

Enrico Blanzieri,
University of Trento, Italy,
and
Anton Bryl
University of Trento, Italy,
Create-Net, Italy
anton.bryl@dit.unitn.it

May 24, 2007

Abstract

In this paper we evaluate the performance of the highest probability SVM nearest neighbor (HP-SVM-NN) classifier, which combines the ideas of the SVM and k-NN classifiers, on the task of spam filtering, using the pure SVM classifier as a quality baseline. To classify a sample the HP-SVM-NN classifier does the following: for each k in a predefined set $\{k_1, ..., k_N\}$ it trains an SVM model on k nearest labeled samples, uses this model to classify the given sample, and transforms the output of SVM into posterior probabilities of the two classes using sigmoid approximation; than it selects that of the $2 \times N$ resulting answers which has the highest probability. The experimental evaluation shows, that in terms of ROC curves the algorithm is able to achieve higher accuracy than the pure SVM classifier.

1 Introduction

The problem of unsolicited bulk email, or *spam*, is today well-known to every user of the Internet. Spam not only causes misuse of time and computational resources, thus leading to financial losses, but it is also often used to advertise illegal goods and services or to promote online frauds [16]. According to a study by Siponen and Stucke [19] the most popular way of anti-spam protection is spam filtering. Many classification algorithms are proposed in order to have accurate and reliable spam filtering, the Support Vector Machine (SVM) classifier being among the best [9, 12, 21].

In this paper we evaluate the performance of the highest probability SVM nearest neighbor classifier, which is an improvement over the SVM nearest neighbor classifier, on the task of spam filtering. SVM nearest neighbor (SVM-NN) [5] is a combination of the SVM and k-NN classifiers. In order to classify a sample x, it first selects k training samples nearest to the sample x, and then uses this k samples for training an SVM model which is further used to make the decision. This method is able to achieve a smaller generalization error bound in comparison to pure SVM because of bigger margin and a smaller ball containing the points. The motivation for using this classifier for spam filtering is the following. Spam is not uniform, but rather consists of messages on different topics [10] and in different genres [8]. The same can be applied also to legitimate mail. This suggests that a classifier which works on a local level can achieve good results on this data. As such, this algorithm proposes no way of estimation of the pa-

rameter k. Blanzieri and Bryl [4] made an attempt to estimate k by internal training and testing on the training data, but this approach brought uncertain results. Instead, with the Highest Probability SVM-NN (HP-SVM-NN), we propose to select the parameter k that minimizes the posterior probability of error. The probability of error is estimated from the output of SVM using the sigmoid approximation [17]. The description and some preliminary experimental evaluation of the method is given in our technical report [3]. In particular, we showed that with equal error cost the proposed method is able to outperform the pure SVM classifier.

In this paper we present further experimental evaluation of the HP-SVM-NN classifier, paying attention to the possibility to adjust the balance between the two types of errors. The experiments show that the proposed algorithm is able to outperform the pure SVM classifier. We also discuss two ways of building a practical spam filter based on this classifier. In fact, the low speed of the classifier together with its high accuracy suggest that it is reasonable to use it not as a separate filter, but as the last step in a cascade of classifiers, so that "easy" spam is previously filtered out by a faster algorithm, for example by pure SVM or by Naïve Bayes.

The rest of the paper is organized as follows. In Section 2 we give an overview of the related work, in Section 3 we present the description of the algorithm, Section 4 contains description and discussion of the experimental evaluation, Section 5 addresses the issue of building a practically useful filter based on this classifier, and Section 6 is a conclusion.

2 Related Work

A great number of learning-based classifiers were proposed for spam filtering. Some of them exploit the knowledge about the structure of the message header, retrieving particular kinds of technical information and classifying messages according to it. For example, Leiba et al. [13] propose to analyze IP addresses in the SMTP path contained in the header. Another group of approaches to spam filtering uses human language technologies to analyze the content of the message, for example the approach proposed by Medlock [14] is based on smooth n-gram language modeling.

However, there is a large group of learning-based spam filters that observe an email message just as a set of tokens. Such filters can use data from the message content, or from the message header, or from both. The Naïve Bayes filter [18] is the most well-known in this group. Also, the Support Vector Machine (SVM) classifier was proposed for spam filtering by Drucker et al. [9], and proved to show good results in comparison to other methods [9, 12, 21], which makes it a reasonable quality baseline for new algorithms. We must also mention here the filtering approaches based on maximum entropy model [20] and boosting [9], both comparable in accuracy to SVM [21]. A spam filter based on the k-nearest neighbor (k-NN) algorithm was introduced by Androutsopoulos et al. [1], and showed comparatively low results on the spam filtering task [12, 21]. A more detailed overview of the existing approaches to anti-spam protection can be found in the survey by Blanzieri and Bryl [2].

3 The Algorithm

In order to present the highest probability SVM nearest neighbor classifier we need first to give descriptions of the SVM classifier and the SVM nearest neighbor classifier.

3.1 Support Vector Machines

Support Vector Machine (SVM) is a state of the art classifier [6]. A detailed description of it can be found, for example, in the book by Cristianini and Shawe-Taylor [7]. Below we describe it briefly. Let there be n labeled training samples that belong to two classes. Each sample x_i is a vector of dimensionality d, and each label y_i is either 1 or -1 depending on the class of the sample. Thus, the training data set can be described as follows:

$$T = ((x_1, y_1), (x_2, y_2), ..., (x_n, y_n)),$$

$$x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}.$$

Algorithm: The SVM Nearest Neighbor Classifier

Require: sample x to classify;
training set $T = \{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$;
number of nearest neighbors k.

Ensure: decision $y_p \in \{-1, 1\}$

1: Find k samples (x_i, y_i) with minimal values of $K(x_i, x_i) - 2 * K(x_i, x)$
2: Train an SVM model on the k selected samples
3: Classify x using this model, get the result y_p
4: **return** y_p

Figure 1: The SVM Nearest Neighbor Classifier: pseudocode.

Given this training samples and a predefined transformation $\Phi : \mathbb{R}^d \to F$, which maps the features to a transformed feature space, the classifier builds a decision rule of the following form:

$$y_p(x) = sign\left(\sum_{i=1}^{L} \alpha_i y_i K(x_i, x) + b\right),$$

where $K(a, b) = \Phi(a) \cdot \Phi(b)$ is the kernel function and α_i and b are selected so as to maximize the margin of the separating hyperplane. It is also possible to get the result not in the binary form, but in the form of a real number, by dropping the sign function:

$$y'_p(x) = \sum_{i=1}^{L} \alpha_i y_i K(x_i, x) + b,$$

As it will be discussed further, such real-number output can be useful for estimating the posterior probability of the classification error. Also the SVM classifier allows to consider unequal error cost by introducing an additional parameter in the training process [15].

Some applications require not a binary classification decision, but rather posterior probabilities of the classes $P(class|input)$. SVM provides no direct way to obtain such probabilities. Nevertheless, several ways of approximation of the posterior probabilities for SVM are proposed in the literature. In particular, Platt [17] proposed to approximate the posterior probability of the $\{y = 1\}$ class with a sigmoid:

$$P(y = 1|y'_p) = \frac{1}{1 + e^{Ay'_p + B}} \quad (1)$$

where A and B are the parameters obtained by fitting on an additional training set. Platt observes, that for the SVM classifier with the linear kernel it is acceptable to use the same training set both for training the SVM model and for fitting the sigmoid. The description of the procedure of fitting the parameters A and B can be found in the Platt's original publication [17].

3.2 The SVM Nearest Neighbor Classifier

The SVM Nearest Neighbor (SVM-NN) classifier [5] combines the ideas of the SVM and k-NN classifiers. In order to classify a sample x, the algorithm first selects k samples nearest to the sample x, and then uses this k samples to train an SVM model and perform the classification. The pseudocode of the basic version of the algorithm is given in Figure 1. Samples with minimal values of $K(x_i, x_i) - 2K(x_i, x)$ are the closest samples to the sample x in the transformed feature space, as can be seen from the following equality:

$$||\Phi(x_i) - \Phi(x)||^2 = \Phi^2(x_i) + \Phi^2(x) - 2\Phi(x_i) \cdot \Phi(x) =$$

$$= K(x_i, x_i) + K(x, x) - 2K(x_i, x)$$

A problem is that this algorithm, as such, provides no procedure of finding an appropriate value for the parameter k. In previous works the parameter k was determined by means of a validation set [4, 5]. The approach proved to be useful for remote sensing data, where the procedure outperformed SVM, but not for

Algorithm: The Highest Probability SVM Nearest Neighbor Classifier

Require: sample x to classify;
training set $T = \{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$;
set of possible values for the number of nearest neighbors $\{k_1, k_2, ..., k_N\}$;
parameter C which allows to adjust the balance between the two types of errors.

Ensure: decision $y_p \in \{-1, 1\}$

Order the training samples by the value of $K(x_i, x_i) - 2 * K(x_i, x)$ in ascending order.
MinErr = 1000
Value = 0
if the first k_1 values are all from the same class c **then**
  return c
end if
for all k **do**
  Train SVM model on the first k training samples in the ordered list.
  Classify x using the SVM model with equal error costs, get the result y'_p.
  Classify the same training samples using this model.
  Fit the parameters A and B for the estimation of $P(y = 1|y'_p)$.
  ErrorPositive = $1 - P(y = 1|y'_p)$
  ErrorNegative = $P(y = 1|y'_p)$
  if ErrorPositive < MinErr **then**
    MinErr = ErrorPositive
    Value = 1
  end if
  if ErrorNegative * C < MinErr **then**
    MinErr = ErrorNegative * C
    Value = -1
  end if
end for
return Value

Figure 2: The Highest Probability SVM Nearest Neighbor Classifier: pseudocode.

spam classification, where the results were not conclusive. In both cases the SVM-NN methods outperformed the plain k-NN classifier based on the majority vote.

3.3 The Highest Probability SVM Nearest Neighbor Classifier

The highest probability SVM Nearest Neighbor (HP-SVM-NN) classifier is based on the idea of selecting the parameter k from a predefined set $\{k_1, ..., k_N\}$ separately for each sample x which must be classified. To do this, the classifier first performs the following actions for each considered k: k training samples nearest to the sample x are selected; an SVM model is trained on this k samples; then, the same k samples are classified using this model, and the output is used to fit the parameters A and B in the equation (1); then, the sample x is classified using this model, and the real-number output of the model is used to calculate the estimation of the probabilities $P(legitimate|x)$ and $P(spam|x) = 1-P(legitimate|x)$; in this way, $2\times N$ answers are obtained. Then, the answer with the highest posterior probability estimate is chosen. An additional param-

eter C can be used to adjust the balance between the two types of errors. In this case, the probability of error for the negative answer must be not just lower then the probability of error for the positive answer, but at least C times lower to be selected. If false positives are less desirable than false negatives, $C < 1$ should be used

We must mention, that from the point of view of speed such algorithm is not the same as N runs of basic SVM-NN classifier, because the costly operation of distance calculation is performed only once for each classified sample. Nevertheless, in this form the algorithm is very slow and needs some optimization to allow practical usage or fast experimental evaluation. Such optimization is possible because for some samples with the smallest considered k all the nearest neighbors selected are from the same class. In this degenerated case the estimate of posterior probability of the class from which the neighboring samples come is equal to 1.0, an so cannot be exceeded. If such case occurs, the decision is taken immediately and no further search is performed. This version of the algorithm is faster then the initial one. The pseudocode of this optimized version of the algorithm is presented on Figure 2.

4 Experimental Evaluation

4.1 Experimental Procedures

In order to evaluate the performance of the proposed algorithm and to compare it with the pure SVM classifier, we established an experiment using ten-fold cross-validation on the SpamAssassin corpus[1]. The corpus contains 4150 legitimate messages and 1897 spam messages (spam rate is 31.37%). The partitioning of data is the same for all the runs. The linear kernel was used in both classifiers. The following set of 12 possible values of the parameter k was used: {50, 150, 250, 350, 500, 700, 1000, 1400, 2000, 2800, 4000, 5400}. The following values of the number of features d were used: 100, 200, 500, 1000, 2000, 3000.

Feature extraction is performed in the following way. Each part of the message, namely the message body and each field of the message header, is considered as an unordered set of strings (tokens) separated by spaces. Presence of a certain token in a certain part of the message is considered a binary feature of this message. Then, the d features with the highest information gain are selected. The information gain of a feature is defined as follows:

$$IG(f_k) = \sum_{c \in \{c_1, c_2\}} \sum_{f \in \{f_k, \overline{f}_k\}} P(f, c) \times log \frac{P(f, c)}{P(f) \times P(c)},$$

where f_k is a binary feature, and c_1 and c_2 are the two classes of samples. Thus, each message is represented with a vector of d binary features.

In order to build ROC curves, for the HP-SVM-NN classifier the parameter C (see Figure 2) is changed, and for the SVM classifier the balance between the two types of errors is changed by modifying the relative error cost parameter. For all the other parameters default values are used. Inside the HP-SVM-NN the SVM classifier is always used with default parameters. The implementation of SVM used for this experiment is a popular set of open-source utilities called SVM*light*[2] [11]. Feature extraction is performed by means of a Perl script. The HP-SVM-NN classifier is implemented as a Perl script which uses SVM*light* utilities as external modules for SVM training and classification.

4.2 Results

In Figure 3 the comparison of ROC curves for the SVM classifier and the HP-SVM-NN classifier is presented. We can see that HP-SVM-NN performs better with all the numbers of features considered, except for $d = 200$, for which the results are unsure. Since both methods have high accuracy, the difference between the curves may seem quite small. However, having true positive rate, for example, of 99% instead of 98% means in fact twice less spam in one's mailbox.

The obvious disadvantage of our classifier is its low speed. Our implementation, with the number of features $d = 500$, number of possible values of k equal to 12 and the training dataset of about 5400

[1] http://spamassassin.apache.org/publiccorpus/

[2] http://svmlight.joachims.org/

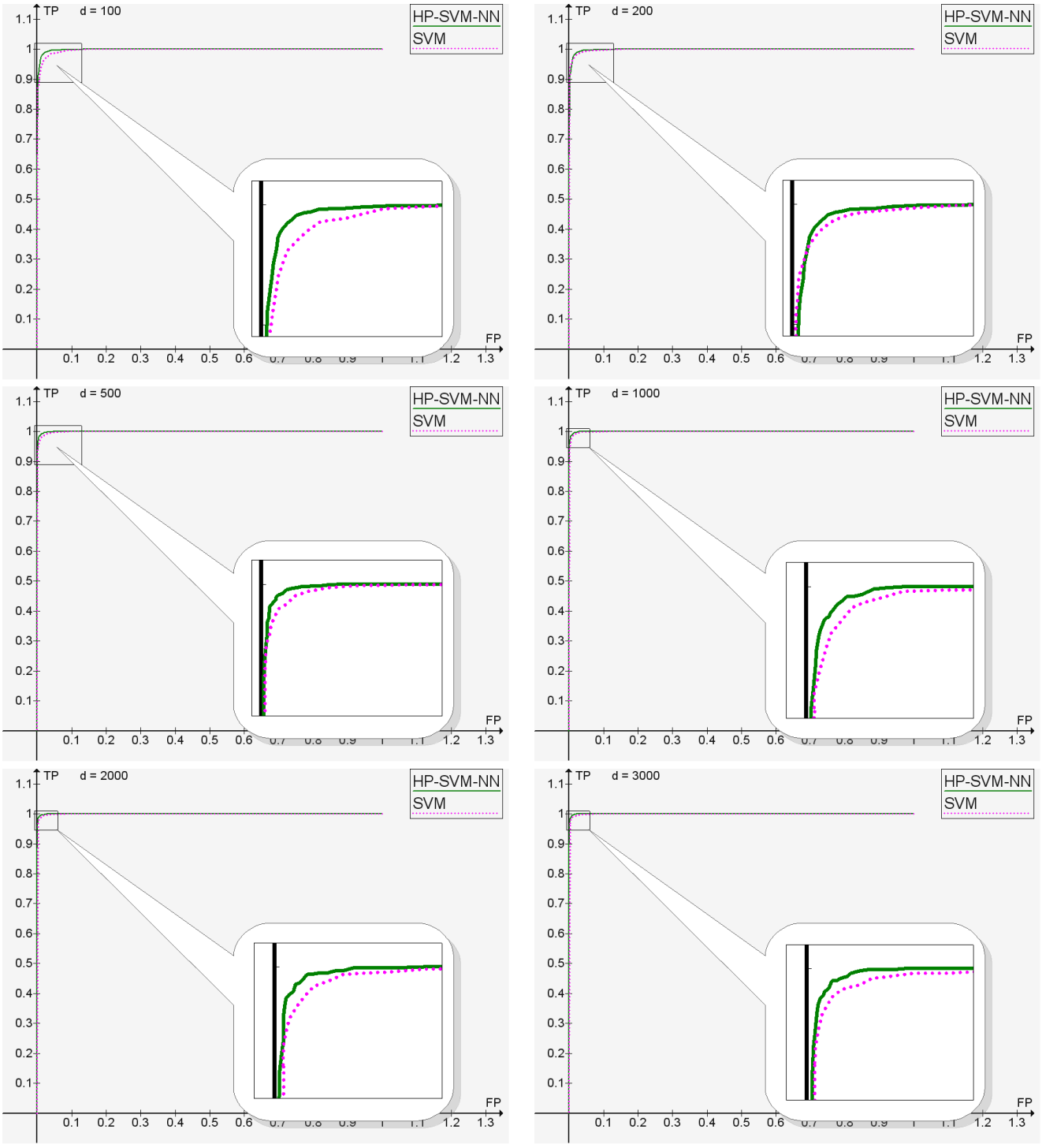
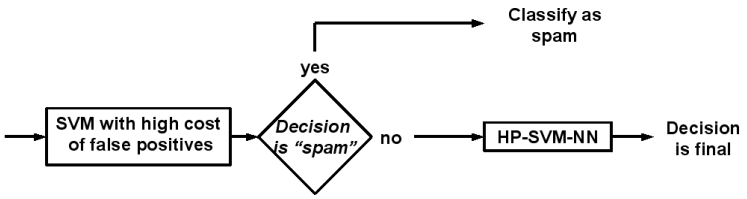
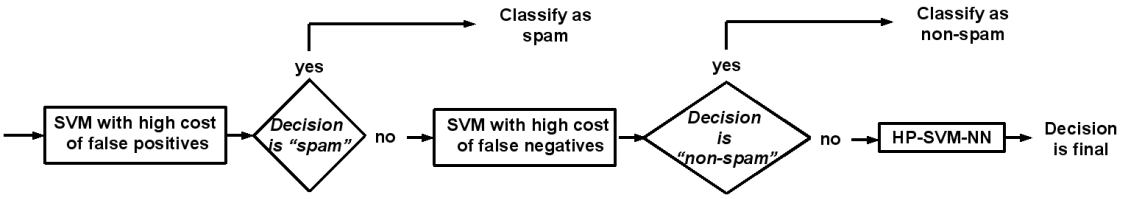


Figure 3: Comparison of SVM and HP-SVM-NN in terms of ROC curves. *SVM* is the pure SVM classifier, as implemented in SVM*light*. *HP-SVM-NN* is the highest probability SVM-NN classifier, as described in section 3.3. d is the number of features. Positive class is spam, negative class is legitimate mail.



(a) First variant.



(b) Second variant.

Figure 4: Building Two Hybrid Filters. The first variant is for the systems which receive much spam and small amount of legitimate mail. The second variant is for the systems which receive huge number of both spam and legitimate mail.

samples, classifies an "easy" sample (with degenerate neighborhood) in about a second and a usual sample in about ten seconds on a PC with CPU speed of 2.50GHz. However, there is much space for optimization at the software level, by which improvement on speed can be achieved.

5 Building a Hybrid Filter

The low speed of the HP-SVM-NN classifier makes it unreasonable to build a filter based on this classifier alone. In order to make the solution practically useful, it makes sense to combine it with other algorithms, so that only "hard" messages are classified by HP-SVM-NN. If the system receives much spam and comparatively small amount of legitimate mail, the following procedure can be used to classify a message. First, the message is classified using a faster classification algorithm, for example SVM, with the cost of false positives set to a very high value. If it is classified as spam, this decision is final; if it is classified as legitimate mail, the final decision is taken by re-classifying this message with HP-SVM-NN. The scheme of this procedure is shown on Figure 4(a). If the system receives large amount of both spam and legitimate mail, the following, a bit more complicated, procedure can be used to classify a message. First, as in the previous case, a fast classifier with the cost of false positives set to a very high value is applied to the message, and if it is classified as spam, no further check is performed. If it classified as legitimate mail, this decision is re-checked by the same classifier, but this time with high cost of false negatives. If the answer is again "legitimate mail", this decision is final, else the HP-SVM-NN classifier is used to make the final decision. The scheme of this procedure is shown on Figure 4(b). In this way, only a small subset of the data will be classified with the HP-SVM-NN algorithm, which will allow to profit from its high accuracy without loosing too much in classification speed.

6 Conclusion

In this paper we evaluated the highest probability SVM nearest neighbor (HP-SVM-NN) classifier applied to the task of spam filtering with variable relative error cost. HP-SVM-NN is a local SVM classifier, which uses k samples in the neighborhood of the sam-

ple which must be classified, with the parameter k selected among a pool of values dynamically, depending on the posterior probabilities of the classes estimated as proposed by Platt [17]. Experimental comparison with the pure SVM classifier is performed, showing that our classifier is able to achieve higher accuracy than SVM. Thus locality proved to be a viable way of increasing the accuracy of the classification at the price of extra computation. Two ways of building a practical filter based on this classifier are discussed. The proposed filter architectures need empirical evaluation on other datasets. Such evaluation is subject to the nearest future work.

References

[1] Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Georgios Sakkis, Constantine Spyropoulos, and Panagiotis 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, PKDD 2000*, pages 1–13, 2000.

[2] Enrico Blanzieri and Anton Bryl. A survey of anti-spam techniques. Technical report #DIT-06-056. 2006.

[3] Enrico Blanzieri and Anton Bryl. Highest probability SVM nearest neighbor classifier for spam filtering. Technical report #DIT-07-007. 2007.

[4] Enrico Blanzieri and Anton Bryl. Instance-based spam filtering using SVM nearest neighbor classifier. In *Proceedings of FLAIRS-20*, pages 441–442, 2007.

[5] Enrico Blanzieri and Farid Melgani. An adaptive SVM nearest neighbor classifier for remotely sensed imagery. In *Proceedings of 2006 IEEE International Geoscience And Remote Sensing Symposium*, 2006.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[7] Nello Cristianini and John Shawe-Taylor. *Support Vector Machines*. Cambridge University Press, 2000.

[8] Wendy Cukier, Susan Cody, and Eva Nesselroth. Genres of spam: Expectations and deceptions. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences, HICSS '06*, volume 3, 2006.

[9] Harris Drucker, Donghui Wu, and Vladimir Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural networks*, 10(5):1048–1054, 1999.

[10] Geoff Hulten, Anthony Penta, Gopalakrishnan Seshadrinathan, and Manav Mishra. Trends in spam products and methods. In *Proceedings of the First Conference on Email and Anti-Spam, CEAS'2004*, 2004.

[11] Thorsten Joachims. *Making large-Scale SVM Learning Practical*. MIT-Press, 1999.

[12] Chih-Chin Lai and Ming-Chi Tsai. An empirical performance comparison of machine learning methods for spam e-mail categorization. *Hybrid Intelligent Systems*, pages 44–48, 2004.

[13] Barry Leiba, Joel Ossher, V. T. Rajan, Richard Segal, and Mark Wegman. SMTP path analysis. In *Proceedings of Second Conference on Email and Anti-Spam, CEAS'2005*, 2005.

[14] Ben Medlock. An adaptive approach to spam filtering on a new corpus. In *Proceedings of the Third Conference on Email and Anti-Spam, CEAS'2006*, 2006.

[15] Katharina Morik, Peter Brockhausen, and Thorsten Joachims. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *Proceedings 16th International Conference on Machine Learning*, pages 268–277. Morgan Kaufmann, San Francisco, CA, 1999.

[16] Evangelos Moustakas, C. Ranganathan, and Penny Duquenoy. Combating spam through legislation: A comparative analysis of us and european approaches. In *Proceedings of Second Conference on Email and Anti-Spam, CEAS'2005*, 2005.

[17] John Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A.J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, *Advances in Large Margin Classiers*, pages 61–74, 2000.

[18] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*. AAAI Technical Report WS-98-05, 1998.

[19] Mikko Siponen and Carl Stucke. Effective anti-spam strategies in companies: An international study. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences, HICSS '06*, volume 6, 2006.

[20] Le Zhang and Tianshun Yao. Filtering junk mail with a maximum entropy model. In *Proceeding of 20th International Conference on Computer Processing of Oriental Languages, ICCPOL03*, pages 446–453, 2003.

[21] Le Zhang, Jingbo Zhu, and Tianshun Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):243–269, 2004.