Journal of Computer Security 00 (2008) 1–30 1
DOI 10.3233/JCS-2008-0319
IOS Press

Applying effective feature selection techniques with Hierarchical Mixtures of Experts for spam classification

Petros Belsis [a,b,*], Kostas Fragos [c], Stefanos Gritzalis [a] and Christos Skourlas [b]

[a] *Department of Information and Communication Systems Engineering, University of the Aegean, Samos, 83200 Greece*
[b] *Department of Informatics, Technological Education Institute of Athens, Egaleo, 12210 Greece*
[c] *Department of Electrical and Computer Engineering, National Technical University of Athens, Athens, 15771 Greece*

E-mail abuse has been steadily increasing during the last decade. E-mail users find themselves targeted by massive quantities of unsolicited bulk e-mail, which often contains offensive language or has fraudulent intentions. Internet Service Providers (ISPs) on the other hand, have to face a considerable system overloading as the incoming mail consumes network and storage resources. Among the plethora of solutions, the most prominent in terms of cost efficiency and complexity are the text filtering approaches. Most of the approaches model the problem using linear statistical models. Despite their popularity – due both to their simplicity and relative ease of interpretation – the non-linearity assumption of data samples is inappropriate in practice. This is mainly due to the inability of other approaches to capture the apparent non-linear relationships, which characterize these samples. In this paper, we propose a margin-based feature selection approach integrated with a Hierarchical Mixtures of Experts (HME) system, which attempts to overcome limitations common to other machine-learning based approaches. By reducing the data dimensionality using effective algorithms for feature selection we evaluated our system with publicly available corpora of e-mails, characterized by very high similarity between legitimate and bulk e-mail (and thus low discriminative potential). We experimented with two different architectures, a hierarchical HME and a perceptron HME. As a result, we confirm the domination of our Spam Filtering (SF) – HME method against other machine learning approaches, which present lesser degree of recall, as well as against traditional rule-based approaches, which lack considerably in the achieved degrees of precision.

Keywords: Spam mail, machine learning based processing, Hierarchical Mixtures of Experts

1. Introduction

E-mail has become lately the dominant way of remote communication. The low complexity of setting up an e-mail server, and the virtually zero cost of sending e-mails comparing to traditional massive marketing notification techniques [16,17],

*Corresponding author: Department of Information and Communication Systems Engineering, University of the Aegean, Karlovassi, Samos, 83200 Greece. Tel.: +30 22730 82234; Fax: +30 22730 82009; E-mail: pbelsis@aegean.gr.

make it an attractive way for unethical advertisers to communicate with potential customers. Unsolicited bulk e-mail or most commonly spam, is responsible for a variety of problems: On one side the Internet Service Providers (ISPs) have to face a slowdown in their system's performance, due to the consumption of network and storage resources; on the other hand, mail users spend a lot of time trying to filter their e-mails. Moreover, the content of these e-mails often contains offensive language or in worst cases some e-mails have fraudulent intentions (most commonly known as scam mails). To ensure that e-mail continues to be a valuable business and personal communications tool, there is an urgent necessity for efficient techniques able to ensure real time access to e-mail content, filtering all unwanted information. Several solutions have been proposed towards the alleviation of the problem, from technical to regulatory and economic [20].

In Europe, the EU launched the "E-Privacy Directive" (directive 2002/58/EC) concerning the processing of personal data and the protection of privacy in electronic communications. The EU directive says that all bulk e-mail should be opt-in, which means that people who receive the mail have stipulated that they want to receive information about the product being advertised (usually by clicking a box on a webpage). In US the Congress adopted the so called CAN-SPAM Act, which took effect on January 1, 2004, and requires that unsolicited commercial e-mail should include a valid return address, the option to opt-out, a valid subject line identifying the e-mail as an advertisement, and a valid sender postal address. Sending fraudulent e-mail, and unlabeled or falsely labeled sexually oriented e-mail, is a criminal act subject to fines and imprisonment. Automated address harvesting and dictionary attacks, based on randomly created addresses, are also prohibited as "aggravated violations". Unfortunately legislative measures against spamming have not been very effective, mainly because most of the spam comes from outside the EU and US; at the same time it is easy for spammers to change locations of their servers and continue to exercise their annoying and unethical practice.

The rapid acceptance of e-mail was based on the simplicity of the Simple Mail Transfer Protocol (SMTP). While the SMTP was designed to be open – a fact that contributed in making e-mail so popular – it does not offer any means of authorization. This allows spammers to easily mask their identities by hacking unprotected mail servers or forging return addresses in a message's "mail from" command. Some recent proposals suggest a change in the way the protocols work; this solution is not desirable, since it would not allow retaining the simplicity of e-mail as a communication means and it would contribute in restricting the communication between Internet users. We will refer in brief to some proposals as well as to their deficiencies in Section 2.

Filtering is among several popular technical solutions [18,19]. Several commercial or open source mail clients offer various filtering capabilities to the average user, while other (server side mail processing) products require manual configuration and constant updates by administrators. These approaches are distinguished by their high

cost and administrator's personal commitment as well as for their ineffectiveness and constant necessity to manually upgrade the knowledge base [27].

Text categorization techniques have become the dominant paradigm in building anti-spam filters, due to their effectiveness and relatively low development cost [19]. Most of these research approaches attempt to classify mail into interesting and uninteresting ones, on the basis of machine learning techniques [1–3,10,14,15,30]. Even though these techniques are characterized by high degrees of precision, they suffer from relatively lower accuracy ratings; in other words they allow categorization of unsolicited mail as legitimate. In order to measure the efficiency of a filtering method, the following parameters are used:

$$\text{Recall} = \frac{\textit{Categories\_Found\_Correct}}{\textit{Total\_Categories\_Correct}},$$

$$\text{Precision} = \frac{\textit{Categories\_Found\_Correct}}{\textit{Total\_Categories\_Found}}.$$

For instance, if we want to define the Spam Recall and $N_{S\to S}$ is the number of spam messages identified correctly as spam, $N_{S\to L}$ is the number of spam messages identified as legitimate, and $N_{L\to S}$ the number of legitimate messages identified as spam, then we would have

$$\text{Spam\_Precision} = \frac{N_{S\to S}}{N_{S\to S} + N_{L\to S}} \quad \text{and}$$

$$\text{Spam\_Recall} = \frac{N_{S\to S}}{N_{S\to S} + N_{S\to L}}.$$

Recall and precision for legitimate messages can be defined in a similar manner. Thus, high spam recall means better spam filtering. High legitimate recall means less false positives [37]. Our approach focuses not only on achieving high precision and recall, but also on requiring lower training times; retraining is also a simple and fast process, which is an essential advantage for an efficient filter.

The main contributions of the paper are the following: A novel machine-learning based approach is proposed for spam filtering. The proposed approach includes a three phase combination of appropriate techniques and algorithms, that: (i) preprocess and produce an appropriate feature set out of the training and test corpora; (ii) identify the most appropriate features using different algorithms in order to reduce the dimensionality of data (in the first experiment the most representative features are used; in the second the optimal feature set is being used); (iii) by using a Hierarchical Mixtures of Experts (HME) approach and an Expectation–Maximization algorithm we perform carefully designed different classification experiments and provide evidence about the correctness of the main selection criteria of our method, by comparing our method experimentally with other approaches.

The rest of the paper is organized as follows: In Section 2 we present a state-of-the-art review in the area of e-mail filtering, providing a brief comparison with our approach. In Section 3 we present the main challenges and describe the criticality of the feature selection process when building accurate and fast classifiers. We present the main principles of our approach and provide a detailed analysis of our system architecture. Section 4 discusses our data-set characteristics, which characterize it as one of the most difficult benchmarks, and discusses in detail our experiments and the results we obtained with the two different SF-HME architectures; we also provide a comparative evaluation to other approaches. Section 5 concludes the paper and provides directions for future work.

2. Related work

The area of e-mail filtering and classification has recently attracted much research focus. Among other solutions, text-based filtering rises to prominence. In this section, we present a review and also attempt to classify research work on the area of spam filtering, according to the techniques applied. Section 2.1 presents systems, which filter e-mails by applying rule-based techniques. Section 2.2 describes the statistical-based approaches, with major focus on the naïve-Bayesian classifier, which has proved so far to be among the most effective by both means of accuracy and training costs [14,19]. Section 2.3 presents other approaches which belong to the area of artificial intelligence (such as artificial neural networks or genetic programming), which could not be classified in any of the previous categories. Section 2.4 presents works based on combined application of different machine learning algorithms and their relative effectiveness comparison. Section 2.5 discusses proposals about modification of existing mail protocols.

2.1. Rule-based approaches

Cohen [1] uses a system, which learns a set of keyword-potting rules based on the RIPPER rule-learning algorithm to classify e-mails into predefined categories. He reports a performance comparable to traditional TF-IDF weighting method. In general, building a rule-based system often involves acquisition and maintenance of a huge set of rules with an extremely higher cost compared to the purely statistical approach. Furthermore, such a system is cumbersome from a scalability perspective.

Cunningham et al. [29] applied case-based reasoning, a method whose main advantage is the ability to adjust in order to track concept drift, still though the reported experiments were on a very low number of test data. The characteristics of the data-set used for test purposes were also not referenced. In addition, this method has the disadvantage of transferring the burden of labeling the data to the user. Even in cases where a collaborative approach would be used to update the database with the spam messages, the case-based approach will label an e-mail as spam if it looks like any

spam in the training data. There are often many cases where a legitimate e-mail contains spam-like features, and for these reasons the data-set that we used to evaluate our approach (provided by the SpamAssasin community [35]) is considered as one of the most challenging ones; this is due to the fact that many of its legitimate messages contain several spammish features, still a careful examination allows intelligent filters to identify their legitimacy.

Kolcz et al. [34] explored the impact of feature-based selection on signature-based classification. By applying the I-Match algorithm, they explored the possibility of creating a server-side filter, which identifies spam messages through techniques of near-duplicate document detection. Their hypothesis was that spam often consists of highly similar messages sent in high volume. Nevertheless this technique is vulnerable to dedicated spamming attacks, such as frequent content alteration.

2.2. Statistical-based approaches

Statistical filters automatically learn and maintain filtering rules and easily adapt to new circumstances when new data arrive. The most popular and effective statistical spam filter is the naïve-Bayes spam filter. Sahami et al. [2] analyzed a manually categorized mail corpus based on the use of words and phrases. In their research, they applied naïve-Bayesian learning based on: words only, words and phrases, words-phrases and concurrent incorporation of domain specific characteristics, such as the inspection of the sender's server domain (.edu, .gov, etc.). They achieved high percentages of recall especially for the latter case, which is based on characteristics added externally by the user; however they could not guarantee the accuracy of the results. For example, the use of too many quotation marks might indicate spam but it might be dependent upon the specific authoring style of the sender.

Androutsopoulos et al. [3,14] preprocessed manually categorized mail into four separate corpora using a lemmatizer and a stop list. Their investigation examines the effect of attribute-set size, training corpus size, lemmatization and a stop-list, that were not explored in Sahami et al.'s experiments [14]. Even though they achieved fairly high degrees of precision, their recall accuracy was rather low [30].

O'Brien et al. performed a comparative test of naïve-Bayes classifier versus Chi by degrees of freedom to classify spam mail [27]. This method has been applied in the past effectively for authorship identification methods. The advantage of this method (considering that each author has textual fingerprints) is that vast majority of spam could be blocked by identifying the specific style of a spammer, given that the system has been trained in advance. On the negative side, there is no author-specific spam data-set so far that could be used as a reliable benchmark. Moreover, in their experiments the authors achieve an unimpressively lower recall than that of other approaches.

Gee [30] applied latent semantic indexing analysis improving the low recall, though this method was reported to suffer from serious errors, namely categorizing legitimate e-mail as illegal, which consists to be an error with very high importance [3,10,14,19].

Drucker et al. [10] analyzed their corpus by applying Ripper, Rocchio boosting and Support Vector Machines (SVM) and they found that SVM is somewhat lower in accuracy than boosting; however it dominates in the necessary training time. In their work they reported that the smallest error rates are achieved by using the subject and body without a stop-list and that SVM should be used with binary features. Although high percentages of precision were achieved, some of the tested algorithms demand long training times.

Nicholas [22] applied a different boosting algorithm (AdaBoost [31]) with decision stumps, in attempt to overcome the extremely slow training times of C4.5 examined by Drucker et al. [10]; though the results did not reveal any superiority to the naïve-Bayes method.

2.3. Other approaches

Drewes [32] created an artificial neural-network based e-mail classifier; still the reported precision was significantly lower than that of other machine learning approaches. Furthermore, neural networks are not an appropriate choice for this type of problem, due to the extensive time they demand for training purposes [10].

Katirai et al. in [21] applied genetic programming algorithms and performed a comparison with the naïve-Bayesian classifier. Even though the results on their set of e-mails were comparatively equal to the Bayesian classifier, there was not any obvious proof to indicate substitution of Bayesian filters with genetic algorithms.

2.4. Algorithm effectiveness comparisons

Kiritchenko et al. [26] compared the performance of naïve-Bayes versus SVM, applying co-training on unlabeled data, and reported the superiority of SVM. Using this method the users do not have to label the data themselves; however the reported accuracy is significantly lower than the one recorded by other experiments [2,3,14].

Hidalgo [19] evaluated a number of algorithms, namely C4.5, naïve-Bayes, Rocchio and SVM and did not distinguish any significant domination between the tested algorithms.

Carreras et al. [28] applied the AdaBoost algorithm [31] on a publicly available corpus – the PU1 corpus produced for the needs of the experiments described in [14] – and reported that this algorithm outperforms significantly the performance of Decision Trees and slightly the performance of naïve Bayes. Still as reported in [28], the PU1 corpus is too small and too easy. Default parameters produced very good results and tuning parameters result only in slight improvements. For this reason we did not use in our experiments the PU1 but a much harder corpus, especially created for testing e-mail filters.

Zhao [40] combines three different classifiers (algorithms), k-nearest neighbour, Gaussian, and boosting with MultiLayer Perceptron (MLP), into a ME (Mixture of Experts) approach, which yields overall better performance than any of individual

contributors. Each of the three individual algorithms plays the role of an expert. The gate keeps a record of each expert's past behaviour and reduces dynamically the weight of experts which make higher number of wrong predictions. The system focuses on the experts which have higher prediction accuracy. Zhao's analysis is interesting but due to time and resource constrains, as mentioned in [40], the cross validation of the results was limited. Also, the experiments where done using the spam e-mail database (Spambase) [42] from UCI's machine learning data repository, which does not have the characteristics and difficulty of the data-set used in our experiments.

2.5. Protocol-oriented solutions

The open architecture of the Simple Mail Transfer Protocol (SMTP) resulted in the widespread use of e-mail; still, this openness was the reason for the wide abuse of this useful means of communication. E-mail filtering is an effective solution but has the disadvantage that it needs constant adjustment to the techniques used by spammers to bypass the filters; thus, a reasonable proposal would be a change in the way protocols work, which would eliminate e-mail abuse. One proposal is to change the domain name system so that domains could publish all of the IP addresses legitimately associated with them – a type of reverse mail exchange (RMX) record – in their own DNS databases [43]. If an e-mail sender transmits a message and spoofs his address to make it look like it came from a specific domain (such as computer.org), e-mail servers receiving the transmission could check the sender's real IP address listed in the message's header, against addresses listed in the spoofed domain's RMX record. The server could then verify if the message actually came from the domain. Still the openness and the current size of the Internet threaten the effectiveness of the proposed anti-spam standards. The presence of so many servers around the world would make impractical to verify the origin of every e-mail message. In addition it would be hard to decide whether to allow contact with servers that so far somebody is not familiar with [43].

Since SMTP lacks authentication, any vital change in the way it works would take many years to be implemented by all the SMTP servers around the world. Plus, many of the spam messages are sent by worms or zombies (as they are called); these programs acquire e-mail addresses from an infected PC and send directly e-mail messages (by implementing a simple SMTP engine). A common practice against this type of spam attack is greylisting [47] that forces the sending Mail Transfer Agent (MTA) to resend the message. Greylisting places significant latency on communication (or in worst cases rejects legitimate mail); greylisting is also easy to bypass even for simple worms by implementing a little more sophisticated SMTP engine. Another proposal is to implement HTTP tar pits or SMTP tar pits. The first intends to trap e-mail harvesters in a continuous loop and therefore does not allow the harvester to visit all the web pages in a server and collect addresses. The second solution attempts to slow down the MTA of the agent. The first solution is hard to implement

since it requires a network slowdown in order to be more effective; the latter does not cause serious delays to a spammer, unless implemented by many servers around the world [41].

Ioannidis [9] proposes the encapsulation of policy in the e-mail address. Users can use single purpose addresses (SPAs) which can be cut and pasted. Each single purpose address is associated with a legitimate user. Therefore for spammers the sender address would not match the encoded address in the e-mail header. In case a spammer acquires an address and learns also the sender, he can forge the "From" header. In these cases the user would have to revoke SPAs or to change the encryption key. E-mails not complying could be deleted or the sender could be challenged. Obviously the drawbacks of such an approach are: the difficulty to scale, or the complication of e-mail usage for the average user. Moreover, challenging approaches especially when an image is sent to the sender are not viewable by users using text-only e-mail clients.

Our approach is based on a combination of algorithms applied effectively and independently in the past for feature selection and classification purposes and which presented high precision and accuracy ratings [5,8]. For benchmarking purposes we applied our method to a spam sample with very low discrimination potential between spam and non-spam samples so as to prove the superiority of our method. We also provided a comparative evaluation of our approach with the naïve-Bayesian approach which is acknowledged to be one of the most dominant in recent research in the field.

3. The proposed SF-HME system

3.1. Setting the scene

Spammers usually attempt to trick the filters by inserting words that would not attract suspicions or by inserting several characters that would alter the appearance of a suspicious word but still would be readable by a human. An example is given in the following Fig. 1. As we can see, html-like code is inserted in several parts of

From: khysvqtxifx@botcnnct.com
Reply-To: khysvqtxifx@botcnnct.com
Subject: Announcement about Vlagr@
for our subscribers

V_i_a_g_r_a Pro (S@ALE 50%) -
Increase P_c_r_f_o_r_m_a_n_c_e by 200%
Low p*r*i*c*e*s
M/o/n/e/y back g%uarantc%e

Fig. 1. An example of an e-mail attempting to bypass spam filters. The first and last lines are humanly understandable but do not contain a spam indication for a simple filter.

the e-mail, while words in last line contain underscore symbols between individual characters.

It is obvious that spam filtering needs to adjust constantly to the evolving variations of class representations; hence, it could be seen as a multi-class classification problem. Our approach is based on the Hierarchical Mixtures of Experts (HME) algorithm, which previously has been successfully applied on classification tasks [7,8]. In order to improve the classification accuracy of the algorithm, we applied feature selection algorithms based on a margin-selection strategy on the training data. In the following paragraphs we describe the implementation choices, starting by explaining the criticality of the feature selection process in the calculations.

One of the most challenging tasks in the classification process is the selection of suitable features to represent the instances of a particular class [4]. Choosing the best candidate features can be a real disadvantage for the selection algorithm in relation to effort and time consumption [6,7].

We consider that all e-mails (e-mails of the training set, or incoming unclassified e-mails) could be represented as vectors of binary features: $e = (f_1, f_2, \ldots, f_N)$, where N is the number of features. Using a simple approach for a given e-mail, the feature f_j is assigned a value of 1 if the e-mail contains the feature f_j and 0 otherwise. In the case of e-mails from the training set, the vector has a Label as an extra component: $(f_1, f_2, \ldots, f_N, \mathit{Label})$. According to the characteristics of the training set, we are interested in assigning to each incoming e-mail a (classification) label. Hence, the "relevance" of the incoming new e-mail is determined by the (classification) labels that are extracted from the training data. Table 1 illustrates another straightforward way of representing a simplified collection of four e-mails (using frequencies of features). Instead of using values of 0 or 1 for each feature we use the frequency of the features. In the first column we have an indicative collection of 15 features extracted from the training set (all the possible features are usually much more). In the second column we have the collection feature frequency (cff) (how many times the feature occurs in the training set); the rest columns record the feature frequency (ff) (how many times the feature occurs in the specific e-mail). In the example of Table 1 we assume (see the label of the first column of the table) that the feature selection process was conducted before creating the table and only 15 features are finally used to represent all the e-mails. In an alternative approach (supposing again that all the features of the collection are 15 as in the previous case) we can use a feature selection algorithm, to eliminate features and select only the necessary ones for the representation. We could also use alternative interesting representations of the e-mails assigning weights on specific features. Therefore, we could simplify the necessary space for feature representation using two columns for each e-mail: one column depicting the existence or not of a feature (and assigning only the values of 1 or 0) and another column depicting simple weights (e.g. a number which declares how many times every feature occurs in a specific e-mail divided by the collection feature frequency). An alternative modified form of Table 1 is given in Table 2 to illustrate these ideas.

Table 1
Example of e-mail classification by measuring the frequency of a feature in a document

Features extracted	Frequency cff	#number of e-mails	E-mail1	E-mail2	E-mail3	E-mail4
Feature 1	$cff_1 = 3$	2	$ff_{1\,1} = 0$	$ff_{2\,1} = 1$	$ff_{3\,1} = 0$	$ff_{4\,1} = 2$
Feature 2	$cff_2 = 3$	2	$ff_{1\,2} = 0$	$ff_{2\,2} = 2$	$ff_{3\,2} = 1$	$ff_{4\,2} = 0$
Feature 3	$cff_3 = 3$	3	$ff_{1\,3} = 0$	$ff_{2\,3} = 1$	$ff_{3\,3} = 1$	$ff_{4\,3} = 1$
Feature 4	$cff_4 = 4$	2	$ff_{1\,4} = 0$	$ff_{2\,4} = 1$	$ff_{3\,4} = 0$	$ff_{4\,4} = 3$
Feature 5	$cff_5 = 4$	2	$ff_{1\,5} = 3$	$ff_{2\,5} = 0$	$ff_{3\,5} = 1$	$ff_{4\,5} = 0$
Feature 6	$cff_6 = 4$	3	$ff_{1\,6} = 2$	$ff_{2\,6} = 1$	$ff_{3\,6} = 0$	$ff_{4\,6} = 1$
Feature 7	$cff_7 = 5$	3	$ff_{1\,7} = 1$	$ff_{2\,7} = 3$	$ff_{3\,7} = 0$	$ff_{4\,7} = 1$
Feature 8	$cff_8 = 5$	3	$ff_{1\,8} = 3$	$ff_{2\,8} = 1$	$ff_{3\,8} = 0$	$ff_{4\,8} = 1$
Feature 9	$cff_9 = 5$	3	$ff_{1\,9} = 2$	$ff_{2\,9} = 0$	$ff_{3\,9} = 2$	$ff_{4\,9} = 1$
Feature 10	$cff_{10} = 5$	3	$ff_{1\,10} = 1$	$ff_{2\,10} = 2$	$ff_{3\,10} = 2$	$ff_{4\,10} = 0$
Feature 11	$cff_{11} = 6$	3	$ff_{1\,11} = 1$	$ff_{2\,11} = 0$	$ff_{3\,11} = 2$	$ff_{4\,11} = 3$
Feature 12	$cff_{12} = 7$	2	$ff_{1\,12} = 0$	$ff_{2\,12} = 6$	$ff_{3\,12} = 0$	$ff_{4\,12} = 1$
Feature 13	$cff_{13} = 8$	4	$ff_{1\,13} = 2$	$ff_{2\,13} = 2$	$ff_{3\,13} = 1$	$ff_{4\,13} = 3$
Feature 14	$cff_{14} = 8$	4	$ff_{1\,14} = 3$	$ff_{2\,14} = 2$	$ff_{3\,14} = 1$	$ff_{4\,14} = 2$
Feature 15	$cff_{15} = 9$	3	$ff_{1\,15} = 1$	$ff_{2\,15} = 4$	$ff_{3\,15} = 0$	$ff_{4\,15} = 4$
Label			1	1	1	0

Note: The last row indicates the label assigned to the document. Label ←0 means that the e-mail is legitimate. The last four columns consist of a possible representation of the e-mails as vectors.

Table 2
Reducing the number of necessary features for e-mail representation

All the features	Frequency cff	#number of e-mails	···	E-mail2-binary features	E-mail2 weights
Feature 1	$cff_1 = 3$	2		1 (occurs in the e-mail)	$ff_{2\,1} = 1/3$
Feature 2	$cff_2 = 3$	2		1	$ff_{2\,2} = 2/3$
Feature 3	$cff_3 = 3$	3		1	$ff_{2\,3} = 1/3$
Feature 4	$cff_4 = 4$	2		1	$ff_{2\,4} = 1/4$
Feature 5	$cff_5 = 4$	2		0 (does not occur)	$ff_{2\,5} = 0/4$
Feature 6	$cff_6 = 4$	3		1	$ff_{2\,6} = 1/4$
···				···	···
Label					1

Note: The last column records weights assigned to specific features.

In the rest of this paper, we focus on the description and discussion of our experiments, based on the latter case of simplified yet robust vector representation of e-mails using only binary features (1 if the feature exists in the e-mail and 0 in a different case). Weights were also calculated using the Simba feature selection algorithm, in order to calculate the hypothesis margin.

In our experiments, we have decided to select features from all the available fields of an incoming e-mail. All header information was kept, since this practice achieves better results as indicated in [10] and [27]. While preprocessing the e-mails all the features were used rather than a subset; words from the subject and body were used without a stop list so as to achieve smaller error rates [10]. It is apparent that for a given number of e-mails the number of features can grow significantly. Therefore, a good choice of features is essential in the process as most of the algorithms require excessive periods of time to choose the optimal feature set (which is an important task when building compact and accurate classifiers). The main disadvantage when searching for the best features is that it requires excessive time in training algorithms, which is often an unacceptable condition. The optimum set of features depends on the data and algorithm used. From this very large number of candidate features the most relevant ones should be considered for efficient classification. This is consistent with many researchers [23–25], who estimated that systems using 1–3% of the total words in a category demonstrated little or no loss in performance.

3.2. Feature selection – feature quality estimation

In order to select the most appropriate features in a classification problem several solutions and algorithms can be adopted. It is essential not only to be able to determine the most appropriate features but also to estimate the quality of a solution. Margins are a useful concept introduced in order to measure the quality of a feature set [38]. A margin is a geometric measure that evaluates the confidence of a classifier with respect to its decision [31]. Bachrach et al. [5] present two new feature selection algorithms: the Greedy Feature Flip (G-flip algorithm) and the Iterative Search Margin Based Algorithm (Simba algorithm). These algorithms use margins for 1-nearest neighbour [39]. The use of margins for 1-nearest neighbour guarantees good performance for any feature selection scheme which selects a small set of features while keeping the margin large.

There are two ways to define the margin of an instance with respect to a classification rule: the sample margin measure and the hypothesis margin measure. The sample margin measures the distance of a given instance from the decision boundary. This margin is unstable and difficult to calculate, since for a large number of instances and considering small variations in the instances location it would demand difficult calculations and would cause large variations in the result. The hypothesis margin calculates how much the boundaries can move without changing the assigned label to a given instance. It can be proved that the hypothesis margin for a given instance can be calculated as

$$\theta_P^W(x) = \frac{1}{2}(\|x - \mu\|_W - \|x - \lambda\|_W), \tag{1}$$

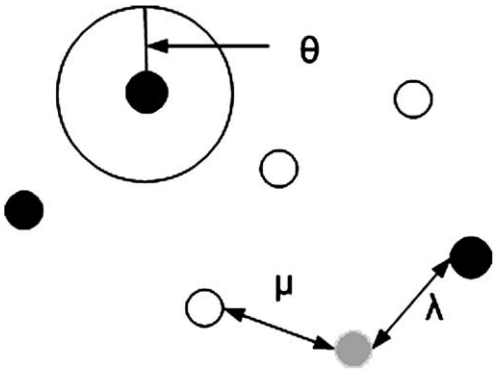

Fig. 2. The hypothesis margin θ is the distance the hypothesis can travel without assigning a different label to the instances. In the figure, for the instance in red color, the distance is given by Eq. (1) where μ and λ are the distances from the closest instances with different labels.

where μ and λ are the nearest points to x in P with the same and different label (Fig. 2) respectively and

$$\|z\|_W = \sqrt{\sum_i w_i^2 z_i^2}. \tag{2}$$

Note that a chosen set of features affects the margin through the distance measure.

3.2.1. Feature selection using the Iterative Search Margin Based Algorithm (Simba)

We applied the Iterative Search Margin Based Algorithm (Simba) in order to select the most relevant features [5]. The reason for selecting Simba is that it seems to outperform other classical statistical approaches such as the relief algorithm, mutual information criterion, etc. [5]. The algorithm provides a weighted vector: $w = (w_1, w_2, \ldots, w_N)$, where N is the number of candidate features and each w_j ranks the importance of feature f_j in the classification task.

For a training set of instances P (in our case e-mails) we calculate the hypothesis margin for an instance $x \in P$ using formula (1). The algorithm at the start point initializes the weighted vector $w = (1, 1, \ldots, 1)$ and in a number of iterations T using a stochastic gradient ascent over the sum of $\sum_i \theta_p(x_i)$ for all the instances x_i it updates the vector w: $w = w + \Delta$, where vector Δ is calculated from the following equation:

$$\Delta i = \sum_{x \in P} \frac{\partial \theta(x_i)}{\partial w_i} = \frac{1}{2} \sum_{x \in P} \left(\frac{(x_i - \mu)^2}{\|x - \mu\|_w} - \frac{(x_i - \lambda)^2}{\|x - \lambda\|_w} \right). \tag{3}$$

The algorithm finally provides after a typical number of iterations a weighted vector w containing the relevancy ranks for the features.

3.2.2. Feature selection using the Greedy Feature Flip Algorithm (G-flip)

We experimented with an alternative margin based feature selection strategy, the Greedy Feature Flip Algorithm (G-flip) [39]; this is a parameter-free approach, or in other words there is no need to tune the number of features and the value of threshold. The G-flip algorithm is a greedy search algorithm for maximizing $e(F)$, where F is a set of features. It repeatedly iterates over the feature set and updates the set of chosen features. Upon every iteration the algorithm decides to remove or add the current feature to the selected set by calculating the evaluation function

$$e(w) = \sum_{x \in S} \theta^{W}_{S \setminus x}(x), \tag{4}$$

with and without this feature. The pseudo-code below which is adapted from [39], describes the basic steps involved in the selection of the optimal feature set:

Begin

1. *(In the beginning) The set F of all the chosen features is the empty set*: $F = \emptyset$;
2. *For all the instances in the training set* $1, 2, \ldots$

   a. *Initialize a random permutation s of the N features*
   b. *for* $i = 1$ *to* N
      calculate $e_1 = e(F \cup s(i))$// *include the* i*th feature*
      calculate $e_2 = e(F \setminus \{s(i)\})$// *exclude the* i*th feature*
      if $e1 > e2$ *then include in* F *the* i*th feature* $F = \{F \cup s(i)\}$
      else exclude the i*th feature* $F = F \setminus \{s(i)\}$
   c. *if no change made in step* (b) *break*

End

The algorithm attempts to maximise the evaluation function as its step increases its value.

These two feature algorithms can be used effectively for feature selection in supervised multi-class classification systems. Simba returns a weight – vector that allows us to choose the features with highest weight and the G-flip returns an "optimal" set of features.

3.3. Hierarchical Mixtures of Experts algorithm

We can define an observation over the given data-set as a collection of numerical measurements, denoted by a vector $x = (x_1, x_2, \ldots, x^k)$, where $x \in \mathbb{R}^k$. In classification applications a mapping $f : \mathbb{R}^k \to \{0, 1\}$ is usually defined. This mapping is usually referred as the classifier. Hence, it is supposed that the new observation ("fresh" data) has an unknown nature which is referred to as the label $y = f(x) \in \{0, 1\}$.

A classifier can accordingly be defined in a generalized manner by considering features of each entity under consideration. Therefore, a classifier is any mapping $f : \mathbb{R}^k \rightarrow \text{CC}$ – from the feature space $\mathbb{R}^k$ to the set of class (target) labels $\text{CC} = \{\text{cc}_1, \text{cc}_2, \ldots, \text{cc}_n\}$. Feature selection is the task of choosing a small subset of features sufficient to predict the target labels well. A classifier accordingly makes a prediction using the provided feature set. There are many ways in which classifiers can be specified and used. One such case is the Mixture of Experts (ME) classifier [8,13] utilised in our research. According to Jordan et al. [7] the "Mixture of Experts type of training constitutes a special niche in the group of dynamic combiner methods". The Mixture of Experts (ME) classifier implements the principle of "divide and conquer": Instead of solving the classification problem over the entire feature space, we can divide it into several regions (subspaces) and try to solve the problem locally and then combine the solutions (outputs). The subspaces are defined through the gating functions g_m. In each region local classifiers ("experts") y_m are assigned and used. If a new sample is to be classified, each of the experts can produce outputs based on the given data and the gate decides which expert to be called upon for the set of inputs. Thus, the ME classifier combines the decisions of several local ("expert") classifiers. To summarize, the ME classifier-system could be seen as a collection of M local experts (expert systems, classifiers) y_m. In each region those experts are combined using the gating functions g_m.

MEs try to solve the problems using a divide-and-conquer strategy by decomposing the whole (usually complex) problem into simpler sub-problems. MEs belong to the class of probabilistic models and consist of a set of experts (which model conditional probabilistic processes) and a gate (which combines the probabilities of the experts). The gating network of ME's learns to classify the input space into patterns in a soft way. Figure 3 shows a mixture of expert's model of two experts and one gate. The standard choices for experts are generalized linear models [7] and multilayer perceptrons [11].

3.3.1. Generalized linear HMEs

We consider that our problem can be modeled using generalized linear models of the form: $y_i = w_i^{\text{T}} x$, where w_i parameters. The output of the expert network of Fig. 3 is the weighted (by the gating network outputs) mean of the expert outputs given by

$$y(x) = \sum_i g_i(x) y_i(x), \tag{5}$$

where $g_i(x)$ denotes the probability that input x is attributed in expert i. In a classification problem we are always interested to compute the *a-posteriori* probability of class label y given the evidence x. In other words, in terms of a ME model we measure the conditional probability $p(y|x)$ of the output y given the input x. This can be formulated by

$$p(y|x) = \sum_i g_i(x) \phi_i(y|x), \tag{6}$$

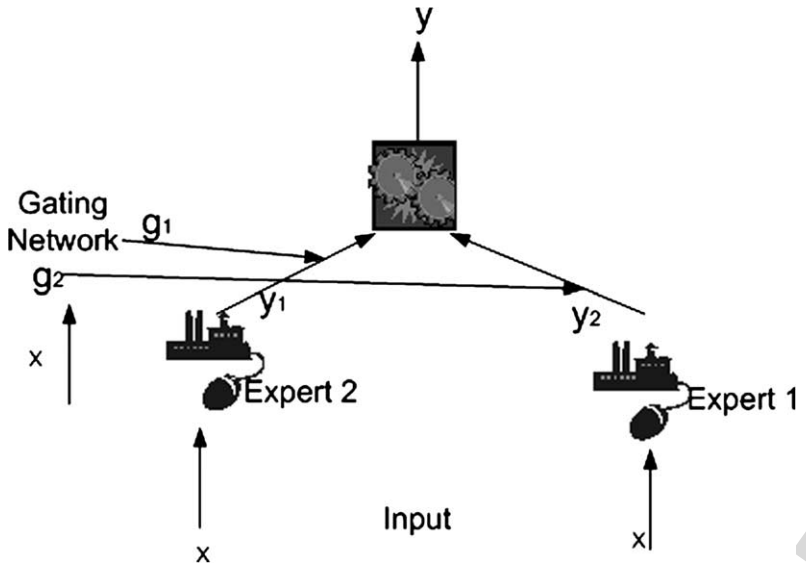

Fig. 3. A mixture of experts' model consisting of two experts E_1, E_2 and one gate G.

where ϕ_i (y_i in Fig. 3) represents the conditional densities of target Y given the expert i. In order to ensure a probabilistic interpretation to the model, the activation function g_i of the gate is chosen to be the soft-max function [12]:

$$g_i = \exp(z_i) \Big/ \sum_j \exp(z_j), \tag{7}$$

where z_i are the gating network outputs before thresholding. Using this function, we achieve a non-negative sum from the gating network; in addition, this sum equals to one.

It is highly desirable in statistical models to model non-linear functions. However, the non-linear functions that a ME model can represent are somewhat restricted since the gate can only form linear boundaries between adjacent expert regions in the input space. A complementary approach proposed by Jordan and Jacobs [7] is to use experts which are by themselves mixtures-of-experts' models. This approach is easily implemented as a generalization of the mixture of experts' model. The result is known as *hierarchical* mixtures-of-experts model (HME) and may be visualized as a tree structure. Such a model is depicted in Fig. 4. The architecture of these models consists of two levels of gates with binary branches at each non-terminal node. The output of the terminal experts' E_3, E_4, E_5, E_6 are y_3, y_4, y_5, y_6, respectively; the outputs of the gates G_1, G_2 rooted at the non-terminal nodes in the second level are g_3, g_4, g_5, g_6. For the outputs of the non-terminal nodes in the second level we

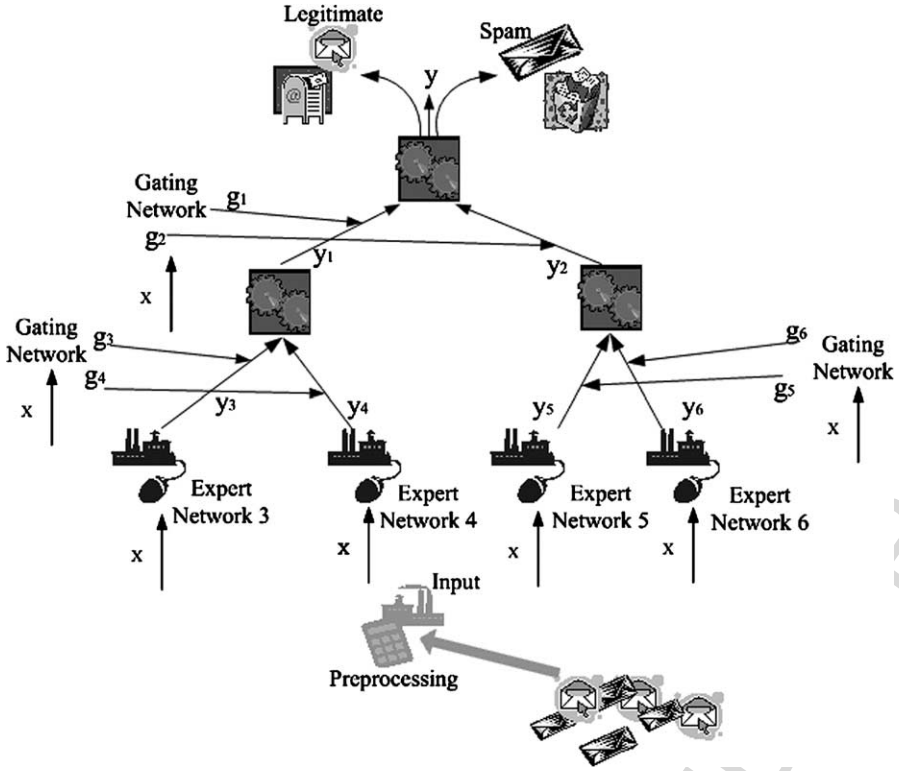

Fig. 4. Tree structure of a hierarchical mixture of experts with binary branches at each non-terminal node, and a depth of 2.

have $y_1 = g_3 y_3 + g_4 y_4$, $y_2 = g_5 y_5 + g_6 y_6$ and finally the output of the system is $y = g_1 y_1 + g_2 y_2$.

The training phase which aims to estimate the system parameters is considered of vital importance for a classification system. For the purposes of our classification task the model must be trained over a suitable number of training instances in order to estimate the parameters, i.e. the functions g_i, ϕ_i. For g_i we use the soft-max function (Eq. (7)) and for experts generalized linear models. The distribution of Eq. (6) forms the basis for the mixture of experts' error function, which can be further optimized using gradient descent or the Expectation–Maximization (EM) algorithm [7]; in our case we use the EM algorithm.

The EM algorithm functions in an iterative way in problems where data is missing or hidden. In the case of mixture of experts' models, missing data are considered the outputs of experts. Moreover, EM is an attractive method for training since it enables the optimization of a ME or HME model to break up into a set of optimizations, one for each expert and gate. It is commonly used to train Gaussian mixtures and other

mixture models. The principle of maximum likelihood is a standard way to motivate error functions. Given a set of independently distributed training data $\{x^n, t^n\}$, $n = 1, \ldots, N$, the likelihood L of the data is given by

$$L = \prod_n p(x, t) = \prod_n p(t|x)p(x). \tag{8}$$

By taking the negative logarithm of the likelihood and dropping the term $p(x)$ (because it does not depend on the model parameters), we can obtain a cost function

$$E = -\sum_n p(t|x). \tag{9}$$

Taking into account Eq. (6), the cost function for this classification task can be formulated as follows:

$$E = -\sum_n \ln \sum_i g_i(x)\phi_i(t|x). \tag{10}$$

This cost function must be minimized to find the optimal parameters using the EM algorithm, a complete description of which can be found in [7].

3.3.2. Perceptron HMEs

We attempted so far to model the problem by creating a hierarchical tree structure of experts where terminal nodes correspond to expert units, the output of which is usually $y_i = w_i^{\mathrm{T}} x$ (the generalized linear model) with x the input and w_i the parameters. The hierarchical structure was necessary because there is no apparent linear relationship between the data-sets. An alternative way to model the experts is to use an artificial neuron, which is a mapping $\mathbb{R}^{k+1} \to Y \subseteq [-1, 1]$, where Y can be discrete assuming only the elements $\{\pm 1\}$. The mapping is given by some function $f(\cdot)$, referred to as the activation function. A common choice for the activation function is the logistic sigmoid $1/(1 + \mathrm{e}^{-x})$ *or* the hyperbolic tangent – $\tanh(x)$. We have chosen to model the experts using a two-layer perceptron. The perceptron can compute an output y from various inputs by forming a linear combination of weights w_i (for each input x_i) and then using some non-linear activation function: $y = f(\sum_{i=1}^n w_i x_i + b)$, where w is the vector of weights, x is the vector of inputs and f is the activation function.

The output of the neuron typically depends on a weighted sum of inputs and is influenced by the sum exceeding certain thresholds.

The perceptrons can be used as building blocks of a larger structure. Multilayer perceptrons are powerful tools and can be used to solve complex problems with suitable training. A typical *multilayer* perceptron (MLP) (Fig. 5(b)) network consists of a set of source nodes forming the *input layer*, one or more *hidden layers* of computation nodes and an *output layer* of nodes. We have expanded our experiments by

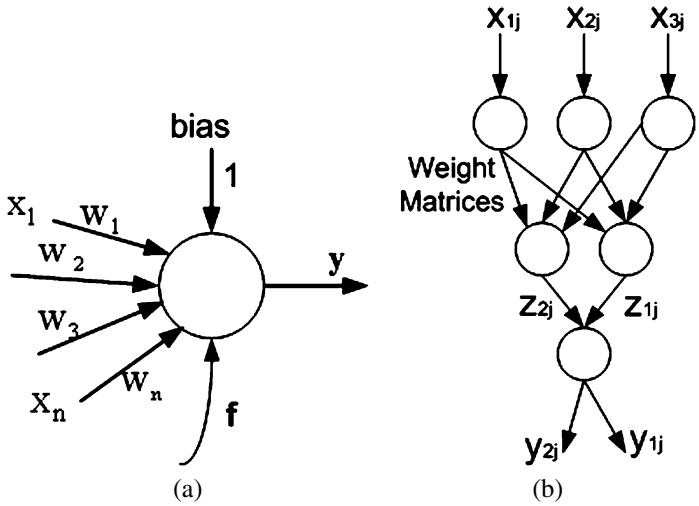

Fig. 5. (a) A simple perceptron. (b) A two layer perceptron with two units in the hidden layer.

modifying the system using two layer perceptrons to model the expert units. A problem that needs to be faced has to do with the presence of the hidden units whose values do not appear in the likelihood function. We will describe in brief how this problem is treated considering the hidden unit values as unobservable data; for a more detailed analysis of how HME MLP architectures may be trained using the EM algorithm the interested reader may refer to [48,51,52].

As we mentioned, the EM algorithm is applied broadly to the computation of Maximum Likelihood (ML) estimates, in cases of missing or incomplete data. It is based on the idea of solving a number of simpler problems by augmenting the original data (the incomplete data) with a number of variables that are unobservable or unavailable to the user. These unobserved data are referred as the missing z-data (in case of an MLP neural network architecture as missing data we consider the hidden layer's values) and together with the originally observed data they consist the complete data input of the classification system.

We consider as

$$(\mathbf{x}_1^{\mathrm{T}}, \mathbf{y}_1^{\mathrm{T}})^{\mathrm{T}}, \ldots, (\mathbf{x}_n^{\mathrm{T}}, \mathbf{y}_n^{\mathrm{T}})^{\mathrm{T}} \tag{11}$$

the n samples available to train the network, where superscript T denotes vector transpose. Let $X_j = (x_{1j}, \ldots, x_{pj})$ be an input feature vector and $y_j = (y_{1j}, \ldots, y_{gj})$ an output vector with $j = 1, \ldots, n$. We consider a vector Θ consisting of the unknown parameters which needs to be estimated by means of the ML statistical technique. Within the EM framework the Θ vector can be estimated by means

of the complete-data log likelihood (of both the observed and missing data), which is given by

$$\log L_c(\Theta; y, z, x) \approx \log P(Y|x, z; \Theta) + \log P(Z|x; \Theta). \tag{12}$$

From the last equation it is apparent that we need to specify the distribution of random variable Z (conditional on x) and the conditional distribution of Y given x and z.

There are two steps for every iteration of the EM algorithm, called the expectation (E) step and the maximization (M) step:

- The E-step, which involves the computation of the so-called Q-function, which is given by the conditional expectation of the complete-data log likelihood given the observed data and the current estimates.
- The M-step, which updates the estimates that maximizes the Q-function over the parameter space.

On the $(k+1)$th iteration of the algorithm, the E-step computes the Q-function:

$$Q\big(\Theta; \Theta^{(\kappa)}\big) = E_{\Theta^{(\kappa)}}\{\log L_c(\Theta; y, z, x)|y, x\} \tag{13}$$

with $E_{\Theta}^{(\kappa)}$ denoting the expectation operator using the current value $\Theta^{(\kappa)}$ for Θ. The M-step of the algorithm updates $\Theta^{(\kappa)}$ taking $\Theta^{(\kappa+1)}$ to be the value of Θ that maximizes Q over all admissible values of Θ; thus, in the applied version of the EM algorithm the complete log-likelihood function is approximated by replacing the random vector z by its conditional expectation.

For the MLP neural network with m hidden units we can work as follows: Let z_{hj} ($h = 1, \ldots, m$ the number of hidden units and $j = 1, \ldots, n$ the number of input instances) be the realization of the zero–one random variable Z_{hj} for which its conditional distribution given $\mathbf{x}_j = (1, x_{1j}, x_{2j})$ is specified by

$$P(Z_{hj} = 1|x_j) = \frac{\exp(w_h^{\mathrm{T}} x_j)}{1 + \exp(w_h^{\mathrm{T}} x_j)}, \tag{14}$$

where $\mathbf{w}_h$ is the synaptic is the synaptic weight vector of the hth hidden unit. The bias term is included in $\mathbf{w}_h$ by adding a constant input $x_{0j} = 1$ for all $j = 1, \ldots, n$; therefore the input becomes $x_j = (x_{0j}, x_{1j}, \ldots, x_{pj})^{\mathrm{T}}$, or

$$w_h^{\mathrm{T}} x_j = \sum_{l=0}^{p} w_{hl} x_{lj}. \tag{15}$$

It is apparent that the variable Z_{hj} has a Bernoulli distribution.

The output of the zero–one indicator variable Y_j is specified by

$$P(Y_{ij} = 1|x_j, z_j) = \frac{\exp(v_i^{\mathrm{T}} z_j)}{\sum_{r=1}^{g} \exp(v_r^{\mathrm{T}} z_j)} \tag{16}$$

for $i = 1, \ldots, g$ (g the number of outputs), where $\mathbf{v}_i$ is the synaptic weigh vector of output unit. The bias term is included in $\mathbf{v}_i$, by adding a constant hidden unit $z_{0j} = 1$ for all $j = 1, \ldots, n$. Thus, we have $\mathbf{u}_i^{\mathrm{T}} = \sum_{h=0}^{m} u_{ih} z_{hj}$. The term on the right side of Eq. (16) is referenced in literature as the normalized exponential or softmax function [50]. Taking Eqs (14) and (16) the vector of all the unknown parameters is given by $\Theta = (w_1^{\mathrm{T}}, \ldots, w_m^{\mathrm{T}}, v_1^{\mathrm{T}}, \ldots, v_{g-1}^{\mathrm{T}})^{\mathrm{T}}$. In other words, by applying the EM algorithm we estimate vector Θ. In more detail we have

$$P(Y|x, z; \Theta) = \prod_{j=1}^{n} \prod_{i=1}^{g} o_{ij}^{yij}, \tag{17}$$

$$P(Z|x; \Theta) = \prod_{j=1}^{n} \prod_{h=1}^{m} u_{hj}^{z_{hj}} (1 - u_{hj})^{(1-z_{hj})}, \tag{18}$$

where

$$u_{hj} = P_r(Z_{hj} = 1|x_j) = \frac{\exp(\sum_{l=0}^{p} w_{hl} x_{lj})}{1 + \exp(\sum_{h=0}^{p} w_{hl} x_{lj})} \tag{19}$$

and

$$o_{gj} = P_r(Y_{gj} = 1|x_j, z_j) = \frac{\exp(\sum_{h=0}^{m} u_{ih} z_{hj})}{1 + \exp(\sum_{r=1}^{g-1} u_{rh} z_{hj})} \tag{20}$$

for $i = 1, \ldots, g - 1$ and

$$o_{gj} = P_r(Y_{gj} = 1|x_j, z_j) = \frac{1}{1 + \sum_{r=1}^{g-1} \exp \sum_{h=0}^{m} u_{rh} z_{hj}}. \tag{21}$$

It follows on application of the EM algorithm in training MLP networks that on the $(k + 1)$th iteration of the E-step, we calculate the expectation of conditional $\log L_c(\Theta; y, z, x)$ on the current estimate of the parameter Θ^k and the observed input and output vectors.

Using (19)–(21) we can compute the expectation of conditional $\log L_c(\Theta; y, z, x)$ which finally can be decomposed into two parts Qw and $Q\upsilon$: The Qw which is a function of vector $\mathbf{w}$ and is linear in $\mathbf{z}$ and the $Q\upsilon$ which is a function of vector $\boldsymbol{\upsilon}$

and is non-linear in **z**. This decomposition of the conditional log $L_c(\Theta; y, z, x)$ enables us to update separately the estimates of w_h and v_i by maximizing Qw and Qv respectively. For a small number of hidden units the EM algorithm provides an efficient training algorithm. For larger number of hidden units a Monte Carlo approach may be used to implement the E-step (since it can be proved that the computational complexity grows significantly with then number of hidden units) [48].

We used routines from the open source Matlab package Bayes Net ToolBox (BNT) [49] in order to implement the HMEs classification system. Within this framework we have built an HME model; we have also used multilayer perceptron models as experts in the place of Softmax functions and used the EM algorithm to train the system.

4. SF-HME system evaluation

Our experiments were performed on a publicly available corpus, provided by the Open Project Spam Assassin for evaluation purposes and benchmarking of unsolicited bulk e-mail filters [35]. In recent bibliography very few databases have been publicly available for evaluation purposes. For some of them the reader may refer to [19,36]. One of the most extensively exploited corpora is the PU1 e-mail corpus [28], collected for the experiments described in [3,14]. In our experiments the basic feature set consisted of words extracted from our corpus; we have also included several characteristics that have all been removed from the PU1 corpus (such as the presence of HTML code); their presence in our test-data makes the classification process more challenging. Furthermore, in order to handle the privacy issues rising when it comes to mail corpora, the PU1 corpus has been encrypted prior to publicizing and therefore has reduced processing capabilities; for example it is not appropriate for co-processing with lexical thesauri or ontological processing, etc. In order to overcome these limitations, the samples we used are not encrypted, and can be freely downloaded from [35].

4.1. Evaluation on a publicly available corpus

4.1.1. Sample data characteristics

We performed our experiments by applying our SF-HME method to the large public spam corpus provided by the Spam Assassin project, as we described in the previous paragraph. This is a selection of mail messages, created especially for benchmarking of spam-filtering systems. The *20030228_spam_2* collection was selected for our experiments. The legitimate corpus consists of two collections: the *20030228_hard_ham_*2 and *20030228_easy_ham* containing 250 and 2500 non-spam messages respectively.

The *hard_ham_2* corpus contains non-spam messages that are difficult to discriminate from spam messages because of their high similarity to typical spam; thus, they

contain several spam-like features: use of HTML, unusual HTML markup, colored text, "spammish-sounding" phrases, etc. The *easy_ham* corpus contains non-spam messages that are relatively easy to be discriminated from spam messages since they do not contain any "spammish" signatures.

4.1.2. Experimental details – the case of generalized linear HME

In order to test the performance of our SF-HME system (the Simba feature selection strategy coupled with the HME classification algorithm) we scanned html code from these corpora and extracted everything that can be used as a candidate feature for classification; mainly words, plus a number of other textual and non-textual features (fields like *received_from*, X-*keywords*, *Content Type*, *subject*, *body*, *size* and other types of information like html tags for *fonts* and *colors*, URLs for multimedia resources, etc.). In order to avoid simplifying the classification process, we did not mix the two non-spam corpora to make a single non-spam corpus; instead we performed two separated experiments one for each corpus.

For the *easy_ham* corpus our algorithm performed as it was expected extremely excellent results achieving 100% discrimination accuracy. Next we describe the experiments in detail while using the *hard_ham* corpus: We divided the 1397 spam messages of the *20030228_spam_2* collection into 5 groups, each group containing 240 messages (150 for training and 90 for testing). From the 250 messages of the *hard_ham_2* corpus the 150 were used for training and the 90 for testing (we used 90 because our program separated only 243 discrete e-mails from the *hard_ham_2* corpus). We performed 5 evaluation experiments using as evaluation measures the average *precision* and *recall*. All the results appearing in Table 3 are the mean values from the five experiments. The total number of features was: 515.219. The number of discrete features reached the number 31.628.

We selected the 300 most representative features by the Simba feature selection algorithm after stemming – a technique that has been proved to enhance e-mail-

Table 3

The 20 most representative features for the classification task as selected by Simba feature selecting algorithm

Feature	Simba score	Feature	Simba score
netnoteinc	1	Deliveri	0.4086
2002	0.61775	http	0.34849
yyyy@netnoteinc	0.60678	Uid	0.34538
taint	0.561	Copyright	0.29373
postfix	0.55541	0000	0.2922
2001	0.5481	Keyword	0.28825
Tm	0.5096	v1	0.26284
text/plain	0.44924	Subscript	0.25903
newslett	0.43718	Juli	0.25345
//www	0.4086	Qmail	0.24557

filtering efforts [14] – and conversion to lower case and removal of punctuation marks.

The log-likelihood before learning was measured to be −182.028879. After only 3 iterations of the EM algorithm the log-likelihood was found to be −0.000957. Table 4 summarizes the results from the first experiment.

4.1.3. Evaluation of perceptron SF-HME

We evaluated the second SF-HME architecture consisting of the two layer perceptron expert units using both the Simba and the G-flip feature selection algorithms and compared its performance on the Assassin corpus with the naïve-Bayes classifier. In the first stage, we ran the G-flip algorithm on the same training data-set as the one applied to the Simba algorithm. The algorithm returned an optimal set of 5717 features which we used to encode the test set of e-mails. The value of the evaluation function (Eq. (4)) at the end of the algorithm running was 2369,1.

Table 5 shows the first 20 features from the optimal set as selected by the G-flip algorithm.

Accordingly, from the *20030228_spam_2* collection we used 50% of the e-mails for the training phase and the remaining for the testing phase of the 1-NN classifier. Similarly, from the *hard_ham_2* corpus we selected 50% of the e-mails used for the training phase and the remaining for the testing phase. The training corpus appeared to have a total of 2105 features.

Table 4

Recall and precision ratings achieved in our experiments for legitimate and spam mail

	Recall	Precision
Spam	92.22%	80.58%
Legitimate	77.78%	90.91%

Table 5

The 20 most representative features for the classification task as selected by G-flip feature selection algorithm

	Feature		Feature
1	to	11	ilug@jmason
2	com	12	x
3	receiv	13	authent
4	127	14	host
5	postfix	15	165
6	with	16	version
7	fetchmail	17	content
8	root@lugh	18	type
9	slashnull	19	text/plain
10	G72LqWv13294	20	ascii

Table 6

Evaluation results on *20030228_spam_2* collection for HMEs with two layer perceptron experts in comparison with naïve-Bayesian classifier

Feature selection strategy Simba	Classifier			
	HME with perceptron experts		Naïve-Bayes classifier	
	Precision	Recall	Precision	Recall
Spam	86.79%	92%	77.5%	62%
Legitimate	91.49%	86%	68.33%	82%

Note: In both cases we selected the most representative features using the Simba feature selection algorithm.

Table 7

Evaluation results on *20030228_spam_2* collection for HME with two layer perceptron experts' nodes and naïve-Bayes classifier using the G-flip feature selection algorithm

Feature selection strategy G-flip	Classifier			
	HME with perceptron experts		Naïve-Bayes classifier	
	Precision	Recall	Precision	Recall
Spam	84.90%	90%	76.19%	64%
Legitimate	89.36%	84%	68.97%	80%

In order to test the performance of our perceptron SF-HME, we performed two different experiments using the *20030228_spam_2* collection: in the first we used the 300 most representative features extracted from the total feature set using the Simba algorithm; in the second experiment we used the optimal feature set extracted using the G-flip algorithm.

We measured the performance of our SF-HME system, in both cases. We compared the system's performance with the results from the naïve-Bayes classifier which proves to be one of the most widely used in related literature and one of the most well-performing classifiers [2,14,19]. The comparative results are shown in Tables 6 and 7; Figs 6, 7 show that our approach outperforms the Bayesian classifier.

4.1.4. Results and discussion

Through our experiments the recorded results provide indications about the robustness of our method (legitimate e-mails are very hard to discriminate from spam in the corpus we used). Other research attempts used different corpora for their experiments and reported high precision [3,14,19]; however, they were performed on test data with low similarity between legitimate and spam mail a factor that makes the classification process an easy task (and with little or no effect when applying tuning parameters) [28]. In addition, with more recent versions of the same corpora used in experiments and by applying SVM, lower degrees of precision and recall have been reported [33]. Even on updated versions of these corpora, HTML com-

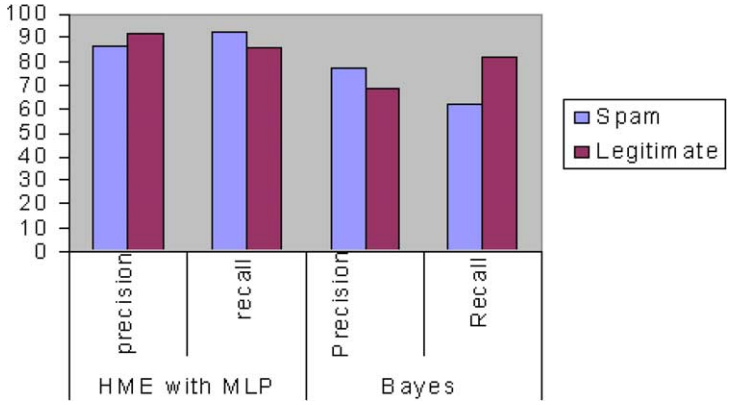

Fig. 6. Visualization of performance results of Table 6. HME's with two layer perceptrons as experts' nodes and Bayes classifier using the Simba feature selection algorithm.

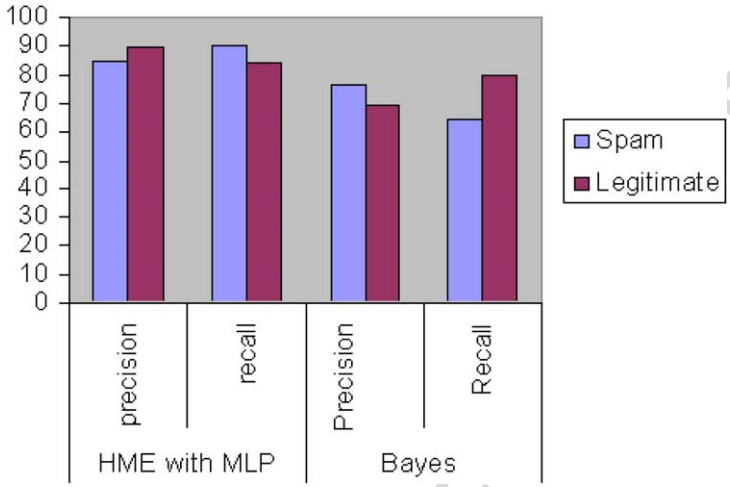

Fig. 7. Visualization of performance results of Table 7. HME with two layer perceptrons as experts' and Bayes classifier using the G-flip feature selection algorithm.

ments and formatting tags were removed in contrast to the hard_ham corpus used for our evaluation purposes.

Summarizing the experiments, we may consider that the two HME-based architectures have been used in a multi-class classification context. In this classification context considering there are f populations of groups, the problem is to decide about the membership or not of an unclassified entity with an f-dimensional feature vector. This membership is defined with an f-dimensional feature vector, where the ith element of the output vector is one or zero depending on whether the entity does or

does not belong to the ith group F_i ($i = 1, \ldots, f$). The collection of e-mails that we used had 1397 spam messages, 250 non-spam messages (the hard_ham corpus) and 2500 non-spam messages (the easy_ham corpus). The total number of discrete features in the corpus after preprocessing reached the number 31.628. By applying the Simba algorithm the 300 most-representative features were used, which consisted of the dimension of the classification vector for the first experiment. In the second case, the G-flip algorithm returned an optimal set of 2105 features. Thus in both cases the number of used features remains adequately low; in the former case the algorithms require less time to perform the tests than in the latter due to the very small dimension of the vector; still both cases perform almost the same by means of accuracy and recall. This fact plus the possibility to update easily the feature set and the possibility to adjust to frequent content alterations consist among the strengths of the proposed method.

Our system presents high degrees of precision, considerably higher than rule-based approaches or even better than these of Bayesian classifiers; it also has the advantage that it demands small training times on the used corpora. The number of representative features can be updated periodically and kept separate from other data. This is very encouraging since most of the filters are susceptible to dedicated word attacks. Against these types of attack, frequent filter retraining proves very effective [44]. Therefore it is essential for a reliable filter to be able to adapt to changes by incorporating a flexible retraining mechanism as in our case.

5. Conclusions

Based on performed experiments with publicly available data-sets, with high similarity between legitimate and unsolicited mail, we drew the following conclusions: Our SF-HME approach proves to be robust and efficient in both means of accuracy and required training times. Furthermore, it does not suffer from the necessity to reconstruct the training set as it happens with other approaches [31]. In our experiments we incorporated in the test set more features than the ones reported by other approaches (which removed attachments, HTML tags and other similar features which simplify the discrimination process). We achieved results that outperform the naïve-Bayesian classifier which has been generally accepted as one of the most efficient ones [14,19,28,30].

The hypothesis that the combination of a powerful feature selection with an accurate classifier such as the HME gives good results and seems to be validated through our experiments; this conclusion was verified using both SF-HME architectures the generalized linear and the perceptron architecture. It is worth mentioning that the triple combination of a good feature selection strategy (which finds the most appropriate features when preprocessing e-mails), a good algorithm that reduces the dimensionality of the data and a powerful classifier scheme seems to be the solution for efficient e-mail classification.

Our system can be used also for authorship identification. In fact given a suitable training set and an appropriate test-set it can discriminate e-mails not on terms of spam or legitimate, but on whether it belongs to a specific group (characterizing the author's style attributes). Given the fact that most of the authors do not change their writing style (such as the use of specific punctuations, or some specific motto's, etc.) and given the fact that large volumes of spam originate from very few sources [27, 46] our method can prove valuable in the following two ways: first, in case where the spammer changes locations and cannot be blocked by black lists or when he adds certain words to trick the filter (assuming that his basic style is the same); second, in cases where from large volumes of messages we want to identify the messages sent by an author (to assure that a case against him in court can stand). O'Brien et al. [27], by applying the Chi by degrees of freedom method – which is often used for authorship identification – claim that if the style attributes are recorded from specific authors and an author specific data-set is formed, then the large volume of spam originating from dedicated spammers can be eliminated.

In these cases, we can use three categories of features (attributes):

(1) "textual" features, e.g. words extracted from e-mails (after stemming, if necessary),
(2) style marker's features (attributes), e.g. average sentence length, number of blank lines, total number of lines and
(3) structural features, e.g. if e-mail contains signature text, number of attachments, has a greeting acknowledgment.

The process is similar as the one explained in Section 3.1 and showed in Table 1, where the labels will represent specific features. The presence of a combination of features will be indication that the e-mail originated from a specific author.

In addition, de Vel et al. [45] focus on the discrimination between authors for the case of both aggregated e-mail topics as well as across different e-mail topics. They used 156 e-mail documents on three topics (movies, food, travels) of three native language English authors in the experimental evaluation of the author-topic categorization. A set of features (attributes) including structural characteristics and linguistic patterns were derived and used for mining the e-mail content. A total of 170 style marker's attributes and 21 structural attributes were used in the experiments. They indicate that in general the Support Vector Machine (SVM) classifier that they used with the style markers and structural features is able to effectively discriminate the authors. They also conclude that 10–20 documents for each author should be sufficient for satisfactory categorization when the authorship identification (discrimination) is attempted from a small set of known candidates.

We are planning to experiment in the future with a broader combination of algorithms and to apply our techniques in identification of e-mails from the same author among a collection of e-mails (for forensic reasons). Unfortunately there is not so far available some specific widely accepted benchmark for testing the ability of a system to capture author-oriented spam and we are thus currently working towards creating a test set so as to continue our experimentation to this direction.

Acknowledgements

We would like to thank all the anonymous reviewers for their insightful comments on the current as well as on earlier versions of this paper, which helped substantially in improving the quality of the manuscript.

This work was co-funded from the EU by 75% and from the Greek Government by 25% under the framework of the Education and Initial Vocational Training Program – Archimedes.

References

[1] W. Cohen, Learning rules that classify e-mail, in: *Proceedings of the 1996 AAAI Spring Symposium on Machine Learning in Information Access*, California, 1996.

[2] M. Sahami, S. Dumais, D. Heckerman and E. Horvitz, A Bayesian approach to filtering junk e-mail, in: *Learning for Text Categorization – Papers from the AAAI Workshop*, Madison, WI, 1998, pp. 55–62; AAAI Technical Report WS-98-05.

[3] I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, G. Paliouras and C.D. Spyropoulos, An evaluation of naïve Bayesian anti-spam filtering, in: *Proc. of the Workshop on Machine Learning in the New Information Age*, 2000.

[4] K. Kira and L. Rendell, A practical approach to feature selection, in: *Proc. of 9th International Workshop on Machine Learning*, 1992, pp. 249–256.

[5] G. Bachrach, A. Navot and N. Tishby, Margin based feature selection – theory and algorithms, in: *Proc of Int. Conference on Machine Learning (ICML) 2004*, Alberta, Canada, 2004.

[6] R.A. Jacobs, M.I. Jordan, S.J. Nowlan and G.E. Hinton, Adaptive mixtures of local experts, *Neural Computation* **3**(1) (1991), 79–87.

[7] M.I. Jordan and R.A. Jacobs, Hierarchical mixtures of experts and the EM algorithm, *Neural Computation* **6**(2) (1994), 181–214.

[8] S.R. Waterhouse and A.J. Robinson, Classification using hierarchical mixtures of experts, in: *Proceedings 1994 IEEE Workshop on Neural Networks for Signal Processing*, Long Beach, CA, IEEE Press, 1994, pp. 177–186.

[9] J. Ioannidis, Fighting spam by encapsulating policy in email addresses, in: *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003*, San Diego, CA, USA, 2003.

[10] H. Drucker, V. Vapnik and D. Wu, Support vector machines for spam categorization, *IEEE Transactions on Neural Networks* **10**(5) (1999).

[11] A.S. Weigend, M. Mangeas and A.N. Srivastava, Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting, *International Journal of Neural Systems* **6** (1995), 373–399.

[12] J.S. Bridle, Probabilistic interpretation of feed forward classification network outputs with relationships to statistical pattern recognition, in: *Neurocomputing: Algorithms, Architectures, and Applications*, F.F. Souli'e and J. H'erault, eds, Springer-Verlag, New York, 1990, pp. 227–236.

[13] J. Fritsch, M. Finke and A. Waibel, Context-dependent hybrid HME/HMM speech recognition using polyphone clustering decision trees, in: *Proceedings of ICASSP-97*, 1997.

[14] I. Androutsopoulos, J. Koutsias, K. Chandrinos and C. Spyropoulos, An experimental comparison of naïve Bayesian and keyword-based anti-spam filtering with 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, 2000, pp. 160–167.

[15] J.D. Brutlag and C. Meek, Challenges of the email domain for text classification, in: *Proc. of the 17th International Conference on Machine Learning*, USA, Stanford University, 2000, pp. 103–110.

[16] L. Cranor and B. Lamachia, Spam!, *Communications of the ACM* **41**(8) (1998), 74–83.

[17] P. Gburzinsky and J. Maitan, Fighting the spam wars: A remailer approach with restrictive aliasing, *ACM Transactions on Internet Technology* **4**(1) (2004), 1–30.

[18] S. Hinde, Spam, scams, chains, hoaxes and other junk mail, *Computers & Security* **21**(7) (2002), 592–606.

[19] J. Hidalgo, Evaluating cost sensitive bulk email categorization, in: *SAC 2002*, Madrid, Spain, 2002, pp. 615–620.

[20] P. Hoffman and D. Crocker, Unsolicited bulk email: Mechanisms for control, Technical Report UBE-SOL, IMCR-008, Internet Mail Cons., 1998.

[21] H. Katirai, Filtering junk e-mail: A performance comparison between genetic programming & naïve Bayes, available at http://members.rogers.com/hoomank/papers/katirai99filtering.pdf, 1999.

[22] T. Nicholas, Using adaboost and decision stumps to identify spam e-mail, available at http://nlp.stanford.edu/courses/cs224n/2003/fp/tyronen/report.pdf, 2003.

[23] D. Lewis, Feature selection and feature extraction for text categorization, San Francisco, Morgan Kaufmann, 1992, pp. 212–217.

[24] D. Koller and M. Sahami, Hierarchically classifying documents using very few words, in: *International Conference on Machine Learning (ICML)*, 1997, pp. 170–178.

[25] D. Mladenic, Feature subset selection in text-learning, in: *Proc. of the 10th European Conference on Machine Learning*, 1998.

[26] S. Kiritchenko and S. Matwin, Email classification with co-training, in: *Proc. Annual IBM Centers for Advanced Studies Conference (CASCON 2001)*, 2001.

[27] O'Brien and C. Vogel, Spam filters: Bayes vs. Chi-squared; letters vs. words, in: *The International Symposium on Information and Communication Technologies*, September 24–26, 2003.

[28] X. Carreras and L. Marquez, Boosting trees for anti-spam email filtering, in: *Proceedings of RANLP-01 International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, BG, 2001.

[29] P. Cunningham, N. Nowlan, S. Delany and J. Haahr, A case-based approach to spam filtering that can track concept drift, in: *The ICCBR'03 Workshop on Long-Lived CBR Systems*, Trondheim, Norway, June 2003.

[30] K. Gee, Using latent semantic indexing to filter spam, in: *Proceedings of ACM Symposium on Applied Computing, SAC 2003*, FA, USA, ACM Press, 2003, pp. 460–464.

[31] R. Shapire and Y. Singer, Improved boosting algorithms using confidence-rated predictions, *Machine Learning* **37**(3) (1999), 297–336.

[32] R. Drewes, An artificial neural network spam classifier, Technical Report, available at project homepage: www.interstice.com/drewes/cs676/spam-nn

[33] M. Woitaszek and M. Shaaban, Identifying junk electronic mail in Microsoft Outlook with a support vector machine, in: *Proc. of the 2003 Symposium on Applications and Internet*, January 2003, Orlando, FL, USA, IEEE Press, 2003.

[34] A. Kolsz, A. Chowdhury and J. Alspector, The impact of feature selection on signature-driven spam detection, in: *Conference on email and Anti-Spam 2004*, CA, USA, 2004.

[35] http://spamassassin.org/publiccorpus

[36] T. Fawcett, "*In vivo*" spam filtering: A challenge for KDD, *SIGKDD Explorations* **5**(2) (2003), 140–149.

[37] K.M. Schneider, Learning to filter Junk email from positive and unlabeled examples, in: *Proceedings of the 1st International Joint Conference on Natural Language Processing (IJCNLP-04)*, Sanya City, Hainan Island, China, 2004, pp. 602–607.

[38] R.E. Schapire, Y. Freund, P. Bartlett and W. Sun Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, in: *Proc. 14th International Conference on Machine Learning*, Morgan Kaufmann, 1997, pp. 322–330.

[39] K. Crammer, R. Gilad-Bachrach, A. Navot and N. Tishby, Margin analysis of the lvq algorithm, in: *Proc. of the 17th Conference on Neural Information Processing Systems*, 2002.

[40] C. Zhao, Towards better accuracy for spam predictions, Technical Report, University of Toronto, 2004.

[41] T. Eggendorfer, Comparing SMTP and HTTP tar pits in their efficiency as an anti-spam measure, in: *2006 MIT's Spam Conference*, MA, USA, March 2006.

[42] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/spambase/

[43] D. Geer, Will new standards help curb spam? *IEEE Computer* **2** (2004), 13–16.

[44] D. Lowd and C. Meek, Good word attacks on statistical spam filters, in: *Proceedings of 2005 Anti-Spam Conference*, Stanford, USA, 2005.

[45] O. de Vel, A. Anderson, M. Corney and G. Mohay, Mining e-mail content for author identification forensics, *SIGMOD Record* **30**(4) (2001), 55–65.

[46] M. Ludlow, Just 150 'spammers' blamed for e-mail woe, *The Sunday Times*, 1st December, 2002.

[47] J.R. Levine, Experiences with greylisting, in: *Proceedings of 2005 Anti-Spam Conference*, Stanford, USA, 2005.

[48] G.J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, Wiley, New York, 1997.

[49] K. Murphy, Open source Matlab implementation of Bayes Net ToolBox BNT, available at http://bnt.sourceforge.net/

[50] J.S. Bridle, Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, in: *Neuro-Computing: Algorithms, Architectures and Applications*, F.F. Soulié and J. Hérault, eds, Springer, Berlin, Germany, 1990, pp. 227–236.

[51] S.-K. Ng and G.J. McLachlan, Using the EM algorithm to train neural networks: misconceptions and a new algorithm for multiclass classification, *IEEE Transactions on Neural Networks* **17** (2004), 738–749.

[52] M. Aitkin and R. Foxall, Statistical modelling of artificial neural networks using the multi-layer perceptron, *Statistics and Computing* **13** (2003), 227–239.