EVOLUTIONARY SYMBIOTIC FEATURE SELECTION FOR EMAIL SPAM DETECTION

Paulo Cortez[1], Rui Vaz[2], Miguel Rocha[3], Miguel Rio[4] and Pedro Sousa[5]
[1]*Centro Algoritmi, Dep. Information Systems, Universidade do Minho, Guimarães, Portugal*
[2]*Dep. Information Systems, Universidade do Minho, Guimarães, Portugal*
[3]*CCTC, Dep. of Informatics, Universidade do Minho, Braga, Portugal*
[4]*Dep. Electric and Electronic Engineering, University College London, Torrington Place, London, U.K.*
[5]*Centro Algoritmi, Dep. of Informatics, Universidade do Minho, Braga, Portugal*
pcortez@dsi.uminho.pt, a48052@alunos.uminho.pt, mrocha@di.uminho.pt, m.rio@ee.ucl.ac.uk , pns@di.uminho.pt

Keywords: Collaborative Filtering, Content-Based Filtering, Evolutionary Algorithms, Feature Selection, Naive Bayes, Spam email, Symbiotic Filtering, Text Classification.

Abstract: This work presents a symbiotic filtering approach enabling the exchange of relevant word features among different users in order to improve local anti-spam filters. The local spam filtering is based on a Content-Based Filtering strategy, where word frequencies are fed into a Naive Bayes learner. Several Evolutionary Algorithms are explored for feature selection, including the proposed symbiotic exchange of the most relevant features among different users. The experiments were conducted using a novel corpus based on the well known Enron datasets mixed with recent spam. The obtained results show that the symbiotic approach is competitive.

1 INTRODUCTION

Spam messages are an intrusion of privacy, with problematic content, such as online fraud, phishing attacks or viruses. Due to its tiny cost to reach a high number of potential consumers, spam is widely spread. Currently, there are two main approaches to fight spam (Méndez et al., 2008): Collaborative Filtering (CF) and Content-Based Filtering (CBF). CF strategies are based in sharing information about spam messages (spam message hash, spammer IP address, etc.) in a community of users. CBF filters uses a data mining classifier to analyse content (e.g., word frequencies) extracted from email messages. However, CF often suffers from sparsity of data, when users classify very few messages, and first-rater problem, where an e-mail cannot be classified unless a user has rated it before. Also, people have personal views of what is spam and CF often discards this issue (Gray and Haahr, 2004). On the other hand, CBF requires several representative training examples and poor performances are often achieved for new users.

Recently, a novel Symbiotic Filtering (SF) approach was proposed, which makes use of useful capabilities from both CF and CBF (Lopes et al., 2011). Under the Web 2.0 paradigm, the idea is to use the Internet to gather distinct users interested on similar but not identical goals, i.e., improve the spam detection at a personalized level. The aim of SF is to foster mutual relationships, where all or most members benefit. Rather than exchanging data that may be sensitive (e.g., normal ham messages), the goal of SF is to share information about what each local CBF has learned. Within SF there are two interesting sharing possibilities: CBF models or relevant features. The former approach was addressed in (Lopes et al., 2011). This paper focuses on the latter approach, which is less sensitive, since no spam/ham probability is associated with a particular feature, and requires less communication overhead. For feature selection methods we propose the use of Evolutionary Algorithms (EAs) (De Jong, 2006), since they perform a global multi-point search, quickly locating areas of high quality, even when the search space is very complex. We compared the proposed evolutionary SF approach with other non sharing EA variants, as well with a CBF filter that uses a simpler information gain feature selection method.

The related work of this paper is presented in Section 2. Section 3 presents the e-mail data, local and symbiotic filtering methods, and evaluation metrics. Next, the results are presented and discussed (Section 4). Finally, closing conclusions are drawn (Section 5).

2 RELATED WORK

Within the CBF approach to fight spam, the Naïve Bayes (NB) classifier is the most popular learning algorithm, since it is very fast while often achieving high detection accuracies (Garriss et al., 2006). Most NB solutions are based on textual content (i.e. word frequencies) of email messages. This popular approach has the advantage of being generalizable to wider contexts, such as spam instant messaging (spim) detection. However, the CBF performance is dependent of the type of feature selection method used. In (Méndez et al., 2008) five well-known filter feature selection methods were combined with four types of Naïve Bayes classifiers, with the results showing that the choice of the correct feature selection method is a key issue to improve spam detection.

In (Lopez-Herrera et al., 2008), multi-objective EAs were used to achieve a set of filtering rules with different profiles. The filtering rules were encoded as expression syntax trees and the Non-dominated Sorting Genetic Algorithm (NSGA-II) was applied to maximize two evaluation criteria, i.e., precision and recall. In the same year, (Dudley et al., 2008) proposed an EA to analyse different configurations for SpamAssassin, a widely-used open source spam filter. Their approach consisted in using an EA to achieve an optimal setup, at a personalized level, for the set of weights that is used to infer if a given message is spam. In this case, the EA minimized the number of false positives and false negatives. (Zhang et al., 2008) describes a genetic programming approach to feature extraction for a cost-sensitive classification task of spam. The fitness used comprised three objectives: an approximation to the bayes error, misclassification cost and number of tree nodes used to encode a particular solution. The solution proposed in (Zhang et al., 2008) is the most analogous to the one presented in this paper, since an EA is used for the feature selection. However, our approach makes use of a novel collaborative approach, i.e., sharing relevant features among multiple users.

3 MATERIALS AND METHODS

Spam Data: While there are several public benchmark datasets created to evaluate anti-spam filters, most of these datasets are not fitted for personalized filtering (Metsis et al., 2006). To evaluate SF, ideally there should be real mailboxes collected from distinct users (possibly from a social network) during a given time period. Yet, due to logistic and privacy issues, it is quite difficult to obtain such data. Therefore, we created a novel corpus based on a realist and synthetic mixture of real ham and spam messages. The ham messages are from the popular Enron email collection, related with the year of 2001. From the total of 158 Enron users, we selected the five users that had an higher time overlap: **mar**tin-t, **pla**tter-p, **sai**bi-e, **scho**ltes-d and **smi**th-m. Since these employees worked at the same organization, it is reasonable to assume that they could be somehow connected in the context of a professional social network. The spam set consists in 19196 messages that were retrieved from the Bruce Guenter collection (`http://untroubled.org/spam/`), which is based in fake emails published in the Web, during the year of 2010 (our dataset was built in 2011). Only messages with Latin character sets were selected, because the ham messages use this type of character coding and non-Latin mails would be easy to detect. As proposed in (Lopes et al., 2011), the mixture of spam and ham is based on the time each message was received. First, 9 years were added to the date field of all ham emails. Then, for each user, a random spam/ham ratio, uniform within $[0.5, 3]$, was initially set. Next, the corresponding amount of spam messages were randomly selected, within the same time period as defined by the ham data, from the whole spam set and mixed with the ham data. While the overall spam/ham ratio is set by a random and fixed value, it should be noted that under the proposed time ordered mixture, the spam/ham ratios fluctuate through time. Table 1 shows a summary of the adopted corpus.

Table 1: Summary of the SF corpus

user	size	features	time period	spam/ham
mar	888	5057	[3/10,12/10]	1.51
pla	672	2303	[4/10,12/10]	2.15
sai	1688	4476	[3/10,12/10]	1.38
sch	765	2833	[4/10,12/10]	1.50
smi	941	3460	[3/10,12/10]	0.94

Evaluation: Since spam detection evolves through time (i.e. there is a concept drift), we will adopt the more realistic incremental retraining evaluation procedure, where a mailbox is split into batches $b_1, ..., b_n$ of K adjacent messages ($|b_n|$ may be less than K) (Metsis et al., 2006). For $i \in \{2, ..., n-1\}$, the spam filter is trained with $\mathcal{D}_u = b_1 \cup \ldots \cup b_i$ and tested with the messages from b_{i+1}, where $\mathcal{D}_u$ denotes the training data for user u (Fig. 1). It should be noted that the minimum $\mathcal{D}_u$ size is set to $2K$, since the EA algorithms use the last batch of the training data as the validation set, to compute the fitness value. For a given probabilistic filter, the predicted class for message $\mathbf{x}_j$ is given by: spam if $p(\text{spam}|\mathbf{x}_j, \mathcal{D}_u) > D$,

where $D \in [0.0, 1.0]$ is a decision threshold. For a given D and test set, it is possible to compute the true (TPR) and false (FPR) positive rates: $TPR = TP/(TP+FN)$ and $FPR = FP/(TN+FP)$, where TP, FP, TN and FN denote the number of true positives, false positives, true negatives and false negatives, respectively. The receiver operating characteristic (ROC) curve shows the performance of a two class classifier across the range of possible threshold (D) values, plotting FPR (x-axis) versus TPR (y-axis) (Fawcett, 2006). The global accuracy is given by the area under the curve ($AUC = \int_0^1 ROCdD$) metric, which is adopted in this paper for the evaluation of the distinct spam detection methods. With the incremental retraining procedure, one ROC will be computed for each b_{i+1} batch and the overall results will be presented by adopting an average of the AUC values computed for each batch. In case of the EA algorithms, several runs are executed for each method. Confidence intervals are given by the t-student test at the 95% confidence level, while statistical significance is measured using non-parametric Mann-Whitney paired tests (Flexer, 1996).

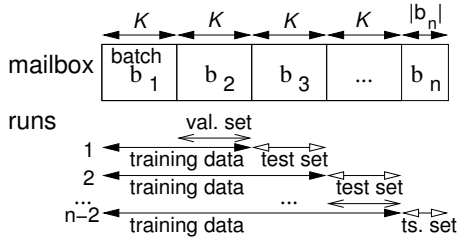

Figure 1: Example of the incremental retraining procedure.

Content-Based Filtering: The CBF filter adopted uses only textual content, i.e. word frequencies of email messages, and is based on the popular NB classifier. The preprocessing follows the steps proposed in (Metsis et al., 2006). The word frequencies were extracted from the subject and body of the message. All HTML tags and non numeric or alphabetic characters were removed. Then, all capital characters were converted into lowercase letters. Next, words with two ore less characters were removed from the text. Each message j was then encoded into a vector $\mathbf{x}_j = (x_{1j}, \ldots, x_{mj})$, where x_{ij} is the number of occurrences of token X_i in the text. As an initial feature selection, any words with very small frequency ($x_{ij} < 5$) in the whole mailbox were removed. In Table 1, the column features denotes the total number of distinct words present in each mailbox of the analyzed corpus.

For the simpler CBF, the feature selection method is based on the information gain criterion (Méndez et al., 2008), which is applied to the training set in order to select the F_{IG} most relevant features. Given its popularity for spam filtering, there are several NB versions that have been successfully applied within this domain (Metsis et al., 2006). In this paper, we adopt the Multinomial NB variant, as implemented in the popular open source RapidMiner tool (Mierswa et al., 2006) and when using a sparse representation, which heavily reduces the computational memory requirements. In (Metsis et al., 2006), such variant obtained a high quality spam detection accuracy, outperforming other NB versions, such as the Multivariate Gaussian.

Evolutionary Feature Selection: Generally there are two main methods for feature selection: filters and wrappers (Guyon and Elisseeff, 2003). Filters methods are independent of the learning algorithm and are applied in the preprocessing stage (e.g., Information Gain). Wrappers test several combinations of features and each testing requires the training of a given classifier. Wrapper methods tend to be more accurate than filters, although they require more computation and the results are specific to a particular classifier. The evolutionary approach for feature selection adopts the same CBF filter previously described. In order to reduce the search space to a reasonable size, the information gain filter is first applied to the training data, in order to select the F_{IG} most relevant features. Then, an EA is applied as a wrapper method, requiring the training of several NB classifiers. Each EA individual is represented as a variable-sized set of strings, which allows the definition of a maximum and minimum number of words. This representation is a more natural form that is closer to the problem to be solved and has the advantage of not requiring a mapping function, when compared with the popular binary representation, since each individual contains the explicit words used by the CBF. The EAs are set to maximize the AUC metric. The computation of the respective fitness is obtained as follows. For a given run of the incremental training (Fig. 1), the training data is divided into training (with all cases except the validation samples) and validation sets (with the last K emails). The features that appear in a given chromosome are fed into the CBF model, which is fit using all training samples. The NB predictions over the validation set are then used to compute the AUC value. After the EA termination criteria, the best individual is selected and the respective features are used to feed a new NB that is fit by using all training data.

For the EA engine, we adopted a general EA, as implemented in the JECoLi Java library (Evangelista et al., 2009). First, there is an initial population with P individuals. New solutions are bred through the use of random respectful recombination (Radcliffe, 1993)

and random mutation operators. The recombination method creates two lists of features: first, with common features between the two progenitors and; second, with the remaining features. The descendants contain all features from the first set plus a random number of words from the second list. The mutation operator replaces a random number of features from the chromosome. In both operators, the minimum and maximum number of features is always preserved. The genetic operators are used (with 50% probability each) to create a new population of size P. Both the original and new populations are evaluated and then a tournament selection is adopted (with a tournament size of 2) to select the P individuals that will survive to the next generation. Finally, the EA is stopped after G generations. Figure 2 shows the schematic of the EA engine adopted. Each EA is executed $n-2$ times, according to the incremental training approach, where each EA run is applied over the training data available at the i-th iteration of the incremental procedure. When creating a random population, P individuals are generated, such that each individual contains a random size, between the minimum and maximum threshold, with randomly selected words from the set of F_{IG} features. Two local EA variants were explored, which are dependent on the type of initial population used. The EA with reinitialization (EAR) uses a random initial population for each run of the incremental training procedure, thus reseting past optimizations. In contrast, the EA with memory (EAM) only uses a random population in the first iteration of the incremental batch (i.e., when the training set is equal to b_1). When a new batch of messages is included in the analysis, the EA restarts with the last population.

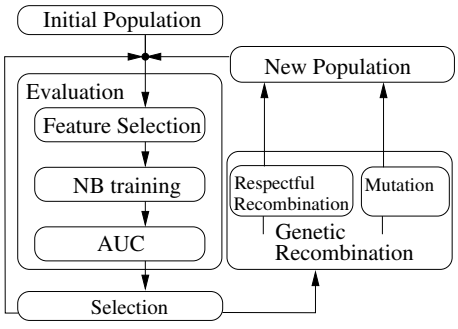

Figure 2: Schematic of the EA engine.

Symbiotic Filtering: The EA that performs a SF (EAS) assumes a symbiotic collaboration within a group of distinct users, which share the most relevant features among the group. Given that no spam or ham probability is assigned to these features, this sharing does not arise privacy concerns. Still, if needed, an anonymous distribution of features can be set, under the use of a trustable application or secure server, as described in (Lopes et al., 2011). The EAS works similarly to EAM except that the initial population includes a percentage of p_s individuals, with features shared from other users, and $1-p_s$ of the best individuals from the previous EAS batch. It is assumed that the symbiotic group has a size of n and each user runs a EAS and during the same time period. To reduce communication costs and computational effort, the exchange of features is asynchronous and occurs only when a new CBF is trained. In this paper, this occurs every time a new batch of messages is analyzed. It should be noted that while the same batch size of K messages is used for all users, the messages included in each batch may be related with distinct dates.

To respect the chronological order of the distinct EAS, the last message date of the training set (t) is used to synchronize the exchange of features. Thus, the sharing is performed among the best individuals from the distinct EAS that were available at time t. For each iteration of the incremental retraining, a given user receives a total of $S = p_s \times P$ individuals, such that the S solutions are equitably retrieved from the other members of the symbiotic group (i.e. each user shares $S/(n-1)$ individuals). To simulate the distributed execution of the EAS, the JECoLi library was adapted to include a different thread for each user. The distinct threads were synchronized, in order to preserve the temporal order. In some situations, user A may receive external features from user B that are not included in the mailbox of A (i.e., mapped in the matrix of word frequencies of A). To increase the diversity of the shared features, we opted for searching for additional features that are extracted from the the best individuals from user B and that appear in the mailbox of A. This procedure is executed until the number of exchanged features is equivalent to the ones contained in the desired $S/(n-1)$ individuals exchange. For demonstration purposes, Fig. 3 plots an example of the symbiotic exchange of individuals. In the example, two users (A, B) share $S/2$ individuals each with user C. It should be noted that while the distinct EAS are run within the same time period, the exchange of individuals is performed using different EAS evolution stages. In the example, the best individuals from user A were searched using all data until batch 3, while the exchange from B was performed over a EA that included only batches 1 and 2.

4 EXPERIMENTS AND RESULTS

All experiments were conducted in Java program-

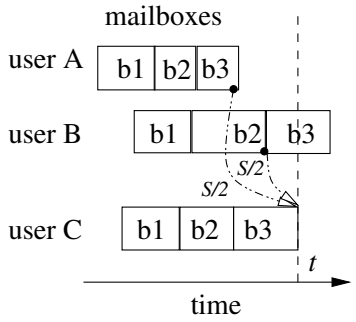

Figure 3: Example of time ordered exchange of individuals.

ming enviroments and we set $K = 100$ for all users, a reasonable value also adopted in (Metsis et al., 2006)(Lopes et al., 2011). For each iteration of the incremental training, the number of information gain selected features was set $F_{IG} = 500$. The configuration parameters used by the EA versions are listed in Table 2. The values related with the last two rows are only used by the EAS. Each EA algorithm was executed a total of 10 runs and results are presented as the average of these runs. We start the analysis by considering the user mar. The results obtained for each batch are presented in Table 3. In general, the obtained results favor the symbiotic approach (EAS), which outperforms the non sharing EA variants (EAR and EAM) and the local NB filter (CBF). In effect, the last row of Table 3 presents the average AUC value over all batches and the higher average AUC value is achieved for EAS. For this user, the remaining methods (EAR, EAM, CBF) achieve considerable worst performances for two of the analyzed batches (5, 9).

Table 2: Parameters set for the EA methods

parameter	**value**
population size (P)	20
minimum individual size (#features)	300
maximum individual size (#features)	400
elitism value	2
stopping criterion (G)	100
shared percentage (p_s)	0.6
symbiotic group size (n)	5

The global results are measured using two criteria: average AUC value, over of all batches ($\bar{b_i}$), shown in Table 4; and percentage of test set batches where the method returns the best AUC value (Table 5). For each user, the last criterion is computed using the formula w/n_{ts}, where w denotes the number of wins of the method and n_{ts} the number of test set batches. When two methods produce the same best AUC value (e.g., batch 3 for user **mar** as shown in Table 3), the

Table 3: Results for user **mar** (AUC values, best in **bold**)

b_i	**CBF**	**EAR**	**EAM**	**EAS**
3	0.960	**0.971**$\pm$0.006	0.969$\pm$0.006	**0.971**$\pm$0.005
4	0.955	**0.957**$\pm$0.008	0.955$\pm$0.011	0.956$\pm$0.019
5	0.919	0.921$\pm$0.006	0.923$\pm$0.005	**0.953**$\pm$0.012*
6	0.973	**0.980**$\pm$0.004	0.974$\pm$0.006	**0.980**$\pm$0.004
7	0.974	0.985$\pm$0.004	**0.986**$\pm$0.001	0.980$\pm$0.005
8	0.963	0.958$\pm$0.008	0.953$\pm$0.008	**0.976**$\pm$0.010*
9	0.877	0.919$\pm$0.009	0.935$\pm$0.007	**0.971**$\pm$0.007*
$\bar{b_i}$	0.946	0.956	0.956	**0.970**

$\star$ - statistically significant when compared with EAM, EAR and CBF.

value of w is increased with 0.5, for each tie and both methods. Overall, the best method is the symbiotic EA (EAS). In terms of the average AUC value, it is the best option for four users and obtains the higher mean value (over all users), as observed in Table 4. Moreover, EAS presents the highest percentage of wins for three users and the best mean value (last row of Table 5). Regarding the non sharing EAs, EAR and EAM obtain a similar performance, in terms of the mean (over all users) AUC value. Nevertheless, EAM presents the second best mean percentage of wins.

Table 4: Results for all users (AUC values, best in **bold**)

user	**CBF**	**EAR**	**EAM**	**EAS**
mar	0.946	0.956$\pm$0.006	0.956$\pm$0.006	**0.970**$\pm$0.009†
pla	0.950	0.949$\pm$0.007	0.947$\pm$0.007	**0.953**$\pm$0.010
sai	**0.983**	0.975$\pm$0.006	0.980$\pm$0.005	0.974$\pm$0.011
sch	0.961	0.967$\pm$0.004	0.964$\pm$0.005	**0.970**$\pm$0.010
smi	0.935	0.938$\pm$0.010	0.942$\pm$0.006	**0.943**$\pm$0.010†
mean	0.955	0.957	0.958	**0.962**

$\dagger$ - statistically significant when compared with CBF.

Table 5: Percentage of batch wins (best value in **bold**)

user	**CBF**	**EAR**	**EAM**	**EAS**
mar	0.0	28.5	14.3	**57.1**
pla	20.0	0.0	20.0	**60.0**
sai	**38.7**	9.7	25.8	25.8
sch	41.7	8.3	0.0	**50.0**
smi	0.0	18.8	**43.8**	37.5
mean	20.1	13.1	20.8	**46.1**

Globally, all spam detection methods achieve a high quality spam detection, with all average AUC values higher than 0.9. The differences between the distinct methods may seem small, with improvements of 0.3 to 2.4 pp of EAS over CBF. Nevertheless, it should be noted that higher improvements may be achieved for a particular batch. For example, the difference between EAS and CBF for user **mar** and

batch 9 is 9.4 pp (Table 3). Also, as shown in Table 5, EAS tends to provide the best AUC values in most of the batches. Moreover, even small improvements may lead to a considerable user added value, since it translates into a better spam email detection probability, which means less time reading unwanted messages and more immunity to virus, worms or phishing attacks. EAS requires more communication and computation when compared with the simpler CBF method. However, the increase in computation is still affordable for a common user and the communication costs are low, around the size of one email message for every batch (e.g. 100 messages). Moreover, the execution of a batch for the EA is not computationally expensive. For example, under the tested computer, the average execution times for 100 generations of the EAS were 11s for user **pla** and 41s for user **mar**.

5 CONCLUSIONS

This paper proposes a novel distributed feature selection approach for spam detection making use of a EA engine for the search of the best features and adopts a SF stategy to share features among distinct users. The goal is to reuse features that were considered relevant for other users in order to improve spam detection at a personalized level. The NB classifier was adopted as the local CBF and tested in a new corpus that performs a realistic mixture of ham messages from five Enron users with recent spam. The performance of EAS was compared with two local EA algorithms (EAR and EAM), as well as the simpler CBF method based on the information gain criterion. The results show that even considering a small simbiotic group (i.e. 5 users), EAS achieves the best spam detection performance, as measured by the AUC metric.

ACKNOWLEDGEMENTS

The work of P. Cortez and P. Sousa was funded by FEDER, through the program COMPETE and Portuguese Foundation for Science and Technology (FCT), by project FCOMP-01-0124-FEDER-022674.

REFERENCES

De Jong, K. (2006). *Evolutionary computation: a Unified Approach*. The MIT Press.

Dudley, J., Barone, L., and While, L. (2008). *Multi-objective spam filtering using an evolutionary algorithm*, pages 123–130. IEEE.

Evangelista, P., Maia, P., and Rocha, M. (2009). Implementing metaheuristic optimization algorithms with jecoli. In *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*, pages 505–510. IEEE.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874.

Flexer, A. (1996). Statistical Evaluation of Neural Networks Experiments: Minimum Requirements and Current Practice. In *Proc. of the 13th European Meeting on Cybernetics and Systems Research*, volume 2, pages 1005–1008, Vienna, Austria.

Garriss, S., Kaminsky, M., Freedman, M., Karp, B., Mazières, D., and Yu, H. (2006). RE: reliable email. In *Proc. of the 3rd conference on Networked Systems Design and Implementation (NSDI)*, pages 297–310, San Jose, CA. USENIX Association Berkeley, USA.

Gray, A. and Haahr, M. (2004). Personalised, Collaborative Spam Filtering. In *1st Conference on E-Mail and Anti-Spam CEAS*.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

Lopes, C., Cortez, P., Sousa, P., Rocha, M., and Rio, M. (2011). Symbiotic filtering for spam email detection. *Expert Systems with Applications*, 38(8):9365–9372.

Lopez-Herrera, A., Herrera-Viedma, E., and Herrera, F. (2008). A multiobjective evolutionary algorithm for spam e-mail filtering. In *Intelligent System and Knowledge Engineering, 2008. ISKE 2008. 3rd International Conference on*, volume 1, pages 366 –371.

Méndez, J., Cid, I., Glez-Peña, D., Rocha, M., and Fdez-Riverola, F. (2008). A Comparative Impact Study of Attribute Selection Techniques on Naive Bayes Spam Filters. In Springer, editor, *8th Industrial Conference on Data Mining*, volume LNAI 5077, pages 213–227.

Metsis, V., Androutsopoulos, I., and Paliouras, G. (2006). Spam filtering with naive bayes – which naive bayes? In *Third Conference on Email and AntiSpam CEAS*, pages 125–134. Citeseer.

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., and Euler, T. (2006). Yale: Rapid prototyping for complex data mining tasks. In *Proc. of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940. ACM.

Radcliffe, N. (1993). Genetic set recombination. *Foundations of Genetic Algorithms*, 2:203–219.

Zhang, Y., Li, H., Niranjan, M., and Rockett, P. (2008). Applying cost-sensitive multiobjective genetic programming to feature extraction for spam e-mail filtering. In *Proc. of the 11th European conference on Genetic programming*, pages 325–336. Springer-Verlag.