Collective Classification for Spam Filtering

Carlos Laorden, Borja Sanz, Igor Santos, Patxi Galán-García, and Pablo G. Bringas

DeustoTech Computing - S^{3}Lab, University of Deusto
Avenida de las Universidades 24, 48007 Bilbao, Spain
{claorden,borja.sanz,isantos,patxigg,pablo.garcia.bringas}@deusto.es

Abstract. Spam has become a major issue in computer security because it is a channel for threats such as computer viruses, worms and phishing. Many solutions to the spam problem feature machine-learning algorithms that are trained using statistical representations of terms that often appear in spam e-mails. However, these methods require a training step with labelled data. Dealing with situations in which the availability of labelled training instances is limited slows the filtering systems' progress and offers advantages to spammers. Currently, many approaches direct their efforts at Semi-Supervised Learning (SSL). SSL is a halfway method between supervised and unsupervised learning. In addition to using unlabelled data, SSL receives supervision information such as associations between targets and examples. Collective Classification for Text Classification is an interesting method for optimising the classification of partially-labelled data. Here, we propose for the first time the use of Collective Classification algorithms for spam filtering to overcome the amount of unclassified e-mails that are sent every day.

Keywords: Spam filtering, collective classification, semi-supervised learning

1 Introduction

It has been reported that more than 85% of received e-mails are spam[1]; thus, inboxes are flooded with annoying and time consuming messages.

Several approaches have been proposed by the academic community to solve the spam problem.

Of these, the approaches referred to as *statistical approaches* [1] use machine-learning techniques to classify e-mails. Statistical approaches have been proven efficient at detecting spam and are the most fully developed technique used to fight it. In particular, Bayes' theorem is widely used by anti-spam filters (e.g., SpamAssasin[2], Bogofilter[3], and Spamprobe[4]).

[1] http://www.spam-o-meter.com/stats/
[2] http://spamassassin.apache.org/
[3] http://bogofilter.sourceforge.net/
[4] http://spamprobe.sourceforge.net/

Statistical approaches are usually supervised, i.e., they require a training set of previously labelled samples. Because statistical techniques perform better when more training instances are available, a significant amount of previous labelling effort is needed to increase the models' accuracy. Due to the dimensions of real-world datasets it is necessary to find alternatives to supervised approaches. The area of security has evolved trying to reduce the dependency on labelled samples by applying unsupervised and semi-supervised approaches to different problems such as malware detection [2–4], intrusion detection systems [5, 6] or spam filtering [3, 7]. In a similar vein, other approaches rely on knowledge sharing, such as collaborative spam filtering [8–10]. This paradigm includes a set of e-mail clients sharing their knowledge about recently received spam e-mails, providing a highly effective defence against a substantial fraction of spam attacks, also alleviating the burdens of frequent training stand-alone spam filters [10].

Similarly, in the area of text categorisation, collective classification attempts to collectively optimise the problem of topic determination by taking connections present among the documents into account. Connections between documents vary from the common citation graph formed when papers cite other papers or when websites link to other websites, to the links constructed between relationships (e.g., co-authors, co-citations, or co-appearances at a conference venue). Combinations of these connections can be used to create interlinked collections of text documents. This type of classification is a semi-supervised technique, i.e., it uses both labelled and unlabelled data (typically, a small amount of labelled data and a large amount of unlabelled data); this reduces the work involved in labelling.

Given this background, we propose the first spam filtering system that uses collective classification to optimise classification performance. This approach minimises the necessity of labelled e-mails without compromising the accuracy of the filter.

In this paper, we (i) describe a method for adopting collective classification in spam filtering; (ii) attempt to determine the optimal size of the labelled dataset for collective classification-based spam filtering; (iii) compare our collective approach with commonly used supervised algorithms; and (iv) show that this approach can reduce the effort of labelling e-mails while maintaining a high accuracy rate.

2 Collective Classification for Spam Filtering

Collective classification is a combinatorial optimisation problem, in which we are given a set of documents or nodes, $\mathcal{D} = \{d_1, ..., d_n\}$ and a neighbourhood function N, where $N_i \subseteq \mathcal{D} \setminus \{\mathcal{D}_i\}$, which describes the underlying network structure [11]. $\mathcal{D}$, a random collection of documents, is divided into two sets $\mathcal{X}$ and $\mathcal{Y}$, where $\mathcal{X}$ corresponds to the set of documents for which we know the correct class, and $\mathcal{Y}$ represents the set of documents whose class values need to be determined. Therefore, the task is to label the nodes $\mathcal{Y}_i \in \mathcal{Y}$ with one of a small number of labels $\mathcal{L} = \{l_1, ..., l_q\}$.

Because the spam problem can be tackled as a text classification problem, we use the *Waikato Environment for Knowledge Analysis* (WEKA) [12] and its Semi-Supervised Learning and Collective Classification plugin[5]. In the remainder of this section we review the collective algorithms used in the empirical evaluation of the method.

2.1 Collective IBk

Collective IBk uses internally WEKA's classic IBk algorithm, its implementation of the *K-Nearest Neighbour* (KNN) algorithm, to determine the best k in the training set. It then builds a neighbourhood consisting of k instances for all instances in the test set from the pool of train and test sets. Either a naïve search of the complete set of instances or a k-dimensional tree is used to determine neighbours. All neighbours in such a neighbourhood are sorted according to their distances from the test instances to which they belong. The neighbourhoods are sorted according to 'rank', where 'rank' indicates the different occurrences of the two classes (i.e., spam and legitimate) in the neighbourhood.

For every unlabelled test instance ranked highest, class label is determined by majority vote or, in the case of a tie, by the first class. This is performed until no unlabelled test instances remain. Classification terminates by returning the class label of the instance that is about to be classified.

2.2 Collective Forest

Collective Forest uses WEKA's implementation of Random Tree as a base classifier to divide the test set into folds each of which contain the same number of elements. The first iteration trains using the original training set and generates the distribution for all the instances in the test set. The best instances are then added to the original training set (being the number of instances chosen the same as the number in a fold).

The next iterations train with the new training set and then generate the distributions for the remaining instances in the test set.

2.3 Collective Woods and Collective Tree

Collective Woods works in a similar manner as Collective Forest but uses Collective Tree instead of Random Tree.

Collective Tree is similar to WEKA's original Random Tree classifier. It splits each attribute at the position that divides the current subset of instances (training and test instances) into two halves. The process is finished when one of the following conditions is met:

- Only training instances are covered (the labels for these instances are already known).

[5] `http://www.scms.waikato.ac.nz/~fracpete/projects/collective-classification`

– Only test instances remain in the leaf, in which case distribution is taken from the parent node.
– Training instances of only one class remain. In this case, all test instances are considered to belong to this class.

To calculate the class distribution of a complete set or a subset, weights are summed according to the weights in the training set, and then normalised. The nominal attribute distribution corresponds to the normalised sum of weights for each distinct value. For a numeric attribute, the distribution of the binary split is calculated based on the median. The weights are summed for the two bins and finally normalised.

2.4 Random Woods

Random Woods works like WEKA's classic Random Forest but uses Collective Bagging (a machine learning ensemble meta-algorithm for improving stability and classification accuracy that has been extended so as to be available to collective classifiers) in combination with Collective Tree. Random Forest, in contrast, uses Bagging and Random Tree.

3 Empirical Evaluation

To evaluate the collective algorithms we used the *Ling Spam*[6], *SpamAssassin*[7] and *TREC 2007 Public Corpus*[8] datasets. Ling Spam consists of a mixture of spam and legitimate messages retrieved from the *Linguistic list*, an e-mail distribution list about *linguistics*. The Ling Spam dataset comprises 2,893 different e-mails, of which 2,412 are legitimate e-mails obtained by downloading digests from the list and 481 are spam e-mails retrieved from one of the authors of the corpus [13]. Each of the four datasets provided in Ling Spam makes use of different pre-processing steps. We chose the *Bare* dataset, which has no pre-processing. The SpamAssassin public mail corpus is a selection of 1,897 spam messages and 4,150 legitimate e-mails. Finally, the TREC 2007 Public Corpus [14] contains all e-mail messages delivered to a particular server. The server contained many accounts fallen into disuse and a number of 'honeypot' accounts published on the web, which were used to sign up for a number of services, some legitimate and some not. The TREC dataset contains 75,419 messages, of which 25,220 are legitimate e-mail and 50,199 are junk messages; the messages are divided into three subcorpora [14] and we used the *full* one.

Due to computational restrictions, in our experiments, we randomly extracted 20% of the full subcorpora while maintaining the spam-legitimate ratio. As a result, our TREC dataset comprises 5,063 legitimate e-mails and 10,021 junk messages.

[6] `http://nlp.cs.aueb.gr/software_and_datasets/lingspam_public.tar.gz`
[7] `http://spamassassin.apache.org/publiccorpus/`
[8] `http://plg.uwaterloo.ca/~gvcormac/spam`

We also performed a *Stop Word Removal* [15] for all datasets based on an external stop-word list[9] and removed any non alpha-numeric characters.

We then used the *Vector Space Model* (VSM) [16], an algebraic approach for *Information Filtering* (IF), *Information Retrieval* (IR), and indexing and ranking, to create a model. This model represents natural language documents in a mathematical manner through vectors in a multidimensional space. Then we constructed an Attribute Relation File Format (ARFF) file [17] with the e-mails' resultant vector representations to build the aforementioned WEKA classifiers using the default parameters.

To evaluate the results, we applied the most frequently used measures for spam; these are: precision, recall and Area Under the ROC Curve (AUC). We defined and measured the precision of spam identification as the number of correctly classified spam e-mails divided by the number of correctly classified spam messages and the number of legitimate e-mails misclassified as spam, $S_P = N_{s\to s}/(N_{s\to s} + N_{l\to s})$, where $N_{s\to s}$ is the number of correctly classified spam and $N_{l\to s}$ is the number of legitimate e-mails misclassified as spam.

We also measured the recall of spam e-mail messages; recall was defined as the number of correctly classified spam e-mails divided by the number of correctly classified spam e-mails plus the number of spam e-mails misclassified as legitimate, $S_R = N_{s\to s}/(N_{s\to s} + N_{s\to l})$.

Finally, we measured the AUC, which establishes the relationship between false negatives and false positives [18]. The ROC curve is represented by plotting the rate of true positives (TPR) against the rate of false positives (FPR). The TPR is the number of spam messages correctly detected divided by the total number of junk e-mails, $TPR = TP/(TP + FN)$, and the FPR is the number of legitimate messages misclassified as spam divided by the total number of legitimate e-mails, $FPR = FP/(FP + TN)$.

In our experiments we tested various configurations of the collective algorithms with different sizes of the $\mathcal{X}$ set of known instances; the latter varied from 10% to 90% of the instances used for training (i.e., instances known during the test). It must be noted that, due to unknown issues with the implementation of the algorithms, the TREC dataset could not be tested with Collective Woods and Random Woods.

Fig. 1 shows the precision of identifying spam in individual datasets using the different algorithms. Collective KNN shows significant improvements with Ling Spam when the number of known instances increases (from 0.60 with 10% to 0.84 with 90%), but remains constant with SpamAssassin and TREC. When precision was evaluated, Collective Forest was the best collective algorithm, it achieved a precision of between 0.99 and 1.00 for Ling Spam, of no less than 0.94 for SpamAssassin and of no less than 0.88 for TREC. In testing with Ling Spam and SpamAssassin, Collective Woods and Random Woods showed some improvement when the number of known instances was increased.

Fig. 2 shows the recall of the various algorithms. Again, Collective KNN shows better results (although still not sufficiently accurate) when the number

[9] `http://www.webconfs.com/stop-words.php`

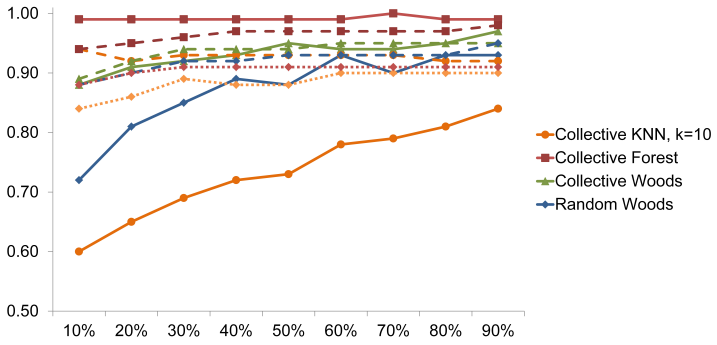

Fig. 1. Precision of the evaluation of collective algorithms of spam filtering with different sizes for the $\mathcal{X}$ set of known instances. Solid lines correspond to Ling Spam, dashed lines correspond to SpamAssassin, and dotted lines correspond to TREC.

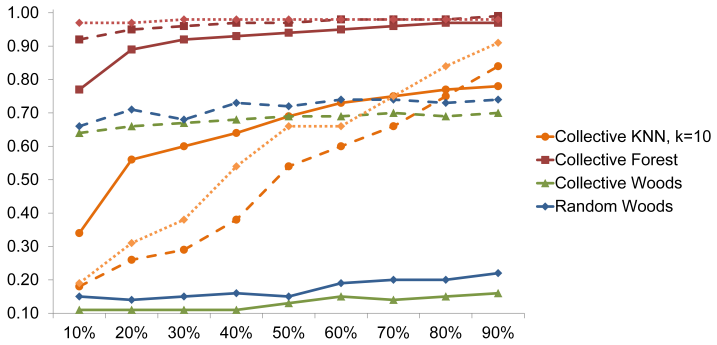

Fig. 2. Recall of the evaluation of collective algorithms for spam filtering with different sizes of the $\mathcal{X}$ set of known instances. Solid lines correspond to Ling Spam, dashed lines correspond to SpamAssassin and dotted lines correspond to TREC.

of known instances increases. With Collective KNN, recall increases from 0.34 with 10% to 0.78 with 90% for Ling Spam, from 0.18 with 10% to 0.84 with 90% for SpamAssassin and from 0.19 to 0.91 for TREC. Collective Forest presents poor recall of 0.77 for 10% with Ling Spam but behaves better with the remaining configurations in all datasets, showing a minimum of 0.89 and a maximum of 0.97 recall for Ling Spam, recall between 0.92 and 0.99 for SpamAssassin and between 0.97 and 0.98 for TREC. Collective Woods and Random Woods demonstrate similarly poor recall, achieving maxima with 90% of 0.16 and 0.22, respectively, for Ling Spam and 0.70 and 0.74 for SpamAssassin.

Finally, Fig. 3 shows the Area Under the ROC Curve (AUC) corresponding to the results obtained with the different algorithms. Once more, the performance of Collective KNN increases with more known instances; the AUC increases from 0.64 with 10% to 0.87 with 90% for Ling Spam, from 0.58 to 0.90 for SpamAs-

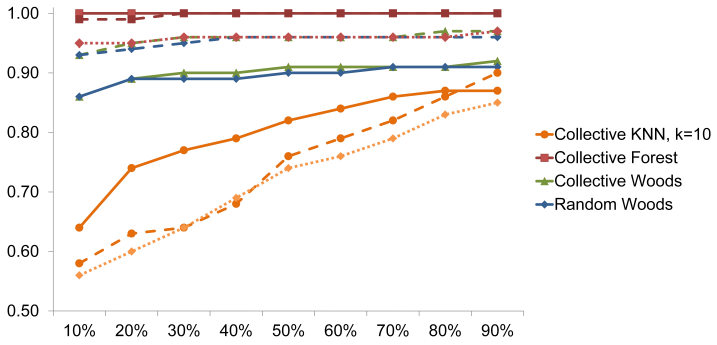

Fig. 3. Area under the ROC curve (AUC) evaluation of collective algorithms for spam filtering with different sizes of the $\mathcal{X}$ set of known instances. Solid lines correspond to Ling Spam, dashed lines correspond to SpamAssassin and dotted lines correspond to TREC.

sassin and from 0.56 to 0.85 for TREC. Collective Forest offers a perfect 1.00 for every configuration with Ling Spam, a minimum of 0.99 with SpamAssassin and a minimum of 0.95 for TREC, supporting Collective Forest as a suitable choice for collective classification. Collective Woods and Random Woods achieve similar results, increasing from 0.86 both to 0.92 and 0.91 respectively with Ling Spam and increasing from 0.93 both to 0.97 and 0.96 respectively with SpamAssassin.

4 Comparison with Supervised Approaches

To evaluate the contribution of Collective Classification to spam filtering, we compare the filtering capabilities of our approach with those of commonly used machine-learning algorithms [19–24].

To assess the machine-learning classifiers, we used the same datasets as for Collective Classification (i.e., Ling Spam, SpamAssassin and TREC) applying the following methodology:

- **Cross validation:** To evaluate the performance of machine-learning classifiers, *k-fold cross validation* [25] is commonly used in machine-learning experiments [26]. For each classifier tested, we performed a k-fold cross validation with $k = 10$. In this way, our dataset was split 10 times into 10 different sets of learning sets (90% of the total dataset) and testing sets (10% of the total data).
- **Learning the model:** For each fold, we perform the learning phase of each algorithm with each training dataset and apply different parameters or learning algorithms, depending on the concrete classifier. We used three different models:
  - *Bayesian Networks:* To train Bayesian Networks, we used *K2* and *Tree Augmented Naïve* (TAN) structural learning algorithms. We also con-

ducted experiments with *Naïve Bayes*, a classifier that has been used widely for spam filtering [21, 22].

- *Decision Trees:* To train decision trees, we used *Random Forest* and *J48* (Weka's *C4.5* implementation).
- *Support Vector Machines:* We used a *Sequential Minimal Optimisation* (SMO) algorithm with a *polynomial kernel*, a *normalised polynomial kernel* and a *Radial Basis Function* (RBF) based kernel. In addition, we used LibSVM[10] for the linear (i.e., hyperplane) and sigmoid kernel implementation.

Table 1. Comparison of results for the best collective algorithm of our approach, Collective Forest, with results obtained applying commonly used supervised algorithms.

	LingSpam			*SpamAssassin*			*TREC*		
Model	**Prec.**	**Rec.**	**AUC**	**Prec.**	**Rec.**	**AUC**	**Prec.**	**Rec.**	**AUC**
BN: K2	0.91	0.98	1.00	0.91	0.89	0.98	1.00	0.40	0.96
BN: TAN	0.92	0.98	1.00	0.94	0.96	0.99	0.90	0.97	0.95
Naïve Bayes	0.97	0.94	1.00	0.83	0.92	0.96	0.92	0.39	0.88
SVM: Polynomial	0.97	0.97	0.98	0.97	0.97	0.98	0.91	0.97	0.90
SVM: Norm Polynom	1.00	0.94	0.97	0.98	0.98	0.99	0.92	0.98	0.90
SVM: RBF	0.99	0.93	0.96	0.98	0.97	0.98	0.81	1.00	0.76
SVM: Lineal	0.97	0.97	0.98	0.97	0.97	0.98	0.91	0.97	0.89
SVM: Sigmoid	1.00	0.47	0.73	0.96	0.89	0.93	0.86	0.99	0.84
DT: J48	0.88	0.84	0.93	0.92	0.93	0.95	0.86	0.98	0.88
DT: RF N=10	0.97	0.95	1.00	0.95	0.98	1.00	0.91	0.98	0.97
Collect. Forest (10%)	0.99	0.77	1.00	0.94	0.92	0.99	0.88	0.97	0.95
Collect. Forest (20%)	0.99	0.89	1.00	0.95	0.95	0.99	0.90	0.97	0.95
Collect. Forest (50%)	0.99	0.94	1.00	0.97	0.97	1.00	0.91	0.98	0.96

Table 1 shows a comparison between results obtained using the best collective algorithm of our approach, Collective Forest, and the most commonly used supervised machine-learning algorithms. The results in Table 1 show that using only 10% of labelled data, Collective Forest offers sound results with the single drawback of a recall of 0.78 for Ling Spam that could be improved. When 20% of the labelled data is used, recall for the Ling Spam dataset is recovered, and the rest of the results improve slightly. Finally, with 50% of the labelled instances, Collective Forest outperforms most of the supervised configurations, while considerably reducing the labelling efforts.

5 Discussion and Concluding Remarks

Collective Classification algorithms for spam filtering present a suitable approach to optimising the classification of partially labelled data, thereby overcoming the massive number of unclassified spam e-mails that are created every day.

Collective Forest shows strong results for every configuration of known instances (i.e., different sizes of the $\mathcal{X}$ set of known instances), with precision values

[10] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

above 0.93, recall values above 0.90 (although offering only a poor recall of 0.78 with 10% of $\mathcal{X}$) and approaching 1.00 of AUC for all configurations.

Because precision and AUC are slightly affected by the variations of known instances, recall should be taken into account in determining the optimal size of labelled data, assuming that Collective Forest is the chosen algorithm. Moreover, knowing that a high recall implies a low false positive rate, and that classification costs in spam filtering are asymmetric (i.e., it is worse to block a legitimate email, than to allow an spam email), reinforces the importance of recall to measure the suitability of the different configurations. In our case, for a value of $\mathcal{X} = 50\%$, Collective Forest achieves the most balanced results between accuracy and amount of labelled data, with only minimal improvements in some configurations when compared to Collective Forest with $\mathcal{X} = 90\%$.

As the number of unsolicited bulk messages increases, the classification and labelling steps used by common supervised methods become increasingly unattainable. To reverse this situation, we propose the first spam filtering system that uses collective classification to optimise classification performance. The algorithms introduced minimise the necessity of using labelled e-mails by 50% without compromising detection capability.

Future work will follow three main directions. First, we plan to extend our study of semi-supervised classification by applying additional algorithms to the spam problem. Second, we will select different features as data to train the models. Finally, we will perform a more complete analysis of the effects of the degree of data labelling.

References

1. Zhang, L., Zhu, J., Yao, T.: An evaluation of statistical spam filtering techniques. ACM Transactions on Asian Language Information Processing (TALIP) **3**(4) (2004) 243–269
2. Santos, I., Nieves, J., Bringas, P.G.: Semi-supervised learning for unknown malware detection. In: Proceedings of the 4^{th} International Symposium on Distributed Computing and Artificial Intelligence (DCAI). 9th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS). (2011) 415–422
3. Santos, I., Laorden, C., Ugarte-Pedrero, X., Sanz, B., Bringas, P.G.: Anomaly-based spam filtering. In: Proceedings of the 6^{th} International Conference on Security and Cryptography (SECRYPT). 5–14
4. Santos, I., Ugarte-Pedrero, X., Sanz, B., Laorden, C., Bringas, P.G.: Collective classification for packed executable identification. In: Proceedings of the 8^{th} Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS). (2011) 23–30
5. Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001, Citeseer (2001)
6. Herrero, Á., Corchado, E., Pellicer, M., Abraham, A.: Movih-ids: A mobile-visualization hybrid intrusion detection system. Neurocomputing **72**(13-15) (2009) 2775–2784

7. Laorden, C., Ugarte-Pedrero, X., Santos, I., Sanz, B., Bringas, P.G.: Enhancing scalability in anomaly-based email spam filtering. In: Proceedings of the 8^{th} Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS). (2011) 13–22
8. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: P2p-based collaborative spam detection and filtering. In: Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on, IEEE (2004) 176–183
9. Gray, A., Haahr, M.: Personalised, collaborative spam filtering. In: Proceedings of 1st conference on email and anti-spam. (2004)
10. Parvathaneni, S.: Collaborative spam filtering. International Journal of Engineering Research and Applications **1**(3) (2011) 427–431
11. Namata, G., Sen, P., Bilgic, M., Getoor, L.: Collective classification for text classification. Text Mining (2009) 51–69
12. Garner, S.: Weka: The Waikato environment for knowledge analysis. In: Proceedings of the New Zealand Computer Science Research Students Conference. (1995) 57–64
13. Androutsopoulos, I., Koutsias, J., Chandrinos, K., Paliouras, G., Spyropoulos, C.: An evaluation of naive bayesian anti-spam filtering. In: Proceedings of the workshop on Machine Learning in the New Information Age. (2000) 9–17
14. Cormack, G.: TREC 2007 spam track overview. In: Sixteenth Text REtrieval Conference (TREC-2007). (2007)
15. Wilbur, W., Sirotkin, K.: The automatic identification of stop words. Journal of information science **18**(1) (1992) 45–55
16. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Communications of the ACM **18**(11) (1975) 613–620
17. Holmes, G., Donkin, A., Witten, I.H.: Weka: a machine learning workbench. (August 1994) 357–361
18. Singh, Y., Kaur, A., Malhotra, R.: Comparative analysis of regression and machine learning methods for predicting fault proneness models. International Journal of Computer Applications in Technology **35**(2) (2009) 183–193
19. Drucker, H., Wu, D., Vapnik, V.: Support vector machines for spam categorization. IEEE Transactions on Neural networks **10**(5) (1999) 1048–1054
20. Carreras, X., Marquez, L.: Boosting trees for anti-spam email filtering. Arxiv preprint cs/0109015 (2001)
21. Schneider, K.: A comparison of event models for Naive Bayes anti-spam e-mail filtering. In: Proceedings of the 10^{th} Conference of the European Chapter of the Association for Computational Linguistics. (2003) 307–314
22. Seewald, A.: An evaluation of naive Bayes variants in content-based learning for spam filtering. Intelligent Data Analysis **11**(5) (2007) 497–524
23. Santos, I., Laorden, C., Sanz, B., Bringas, P.G.: Enhanced topic-based vector space model for semantics-aware spam filtering. Expert Systems With Applications **39**(1) (2012) 437–444 doi:10.1016/j.eswa.2011.07.034.
24. Laorden, C., Santos, I., Sanz, B., Alvarez, G., Bringas, P.G.: Word sense disambiguation for spam filtering. Electronic Commerce Research and Applications (2012) doi:10.1016/j.elerap.2011.11.004.
25. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International Joint Conference on Artificial Intelligence. Volume 14. (1995) 1137–1145
26. Bishop, C.: Pattern recognition and machine learning. Springer New York. (2006)