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 feature machine-learning algorithms trained using statistical representations of the terms that usually appear in the e-mails. Still, these methods require a training step with labelled data. Dealing with the situation where the availability of labelled training instances is limited slows down the progress of filtering systems and offers advantages to spammers. Currently, many approaches direct their efforts into Semi-Supervised Learning (SSL). SSL is a halfway method between supervised and unsupervised learning, which, in addition to unlabelled data, receives some supervision information such as the association of the targets with some of the examples. Collective Classification for Text Classification poses as an interesting method for optimising the classification of partially-labelled data. In this way, we propose here, for the first time, 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

Flooding inboxes with annoying and time-consuming messages, more than 85% of received e-mails are spam[1].

Several approaches have been proposed by the academic community to solve the spam problem [1–4]. Among them, the termed as *statistical approaches* [5] use machine-learning techniques to classify e-mails. These approaches have proved their efficiency detecting spam and are the most extended technique to fight it. In particular, the use of the Bayes' theorem is widely used by the anti-spam filters (e.g., SpamAssasin [6], Bogofilter [7], and Spamprobe [8]).

These statistical approaches are usually supervised, i.e., they need a training set of previously labelled samples. These techniques perform better as more training instances are available. It means that a significant amount of previous labelling work is needed to increase the accuracy of the models. This work includes a gathering phase in which as many e-mails as possible are collected. Then,

[1] http://www.junk-o-meter.com/stats/index.php

each e-mail has to be classified as spam or legitimate. Finally, machine-learning models are generated based upon the labelled data.

This task is usually performed for text categorisation. Since text classification mostly uses the content of the documents and external sources to build accurate document classifiers, there is a great effort in the scientific community [9–11] directed towards the link structure among documents, to improve the performance of document classification.

The connections that can be found within documents vary from the most common citation graph, such as papers citing other papers or websites linking other websites, to links constructed from relationships including: co-author, co-citation, appearance at a conference venue, and others. The combination of these connections leads to the creation of an interlinked collection of text documents.

In some cases, it is interesting to determine the topic of not just a single document, but to infer it for a collection of unlabelled documents. Collective classification tries to collectively optimise the problem taking into account the connections present among the documents. This is a semi-supervised technique, i.e., uses both labelled and unlabelled data – typically a small amount of labelled data and a large amount of unlabelled data – that reduces the labelling work.

Given this background, we propose the first spam filtering system that uses collective classification to optimise the classification performance. Through this approach, we minimise the necessity of labelled e-mails without a significant penalisation of the accuracy of detection.

Summarising, our main findings are the following: (i) we describe how to adopt collective classification for spam filtering, (ii) we try to determine which is the optimal size of the labelled dataset for collective-classification-based spam filtering, and (iii) we show that this approach can reduce the efforts of labelling e-mails while maintaining a high accuracy rate.

The reminder of this paper is organised as follows. Section 2 describes the process of using collective classification applied to the spam filtering problem. Section 3 details the experiments performed and presents the results. Finally, Section 4 concludes and outlines avenues for future work.

2 Collective Classification for Spam Filtering

Collective classification is a combinatorial optimization 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 [12]. Being $\mathcal{D}$ a random collection of documents, it is divided into two sets $\mathcal{X}$ and $\mathcal{Y}$ where $\mathcal{X}$ corresponds to the documents for which we know the correct values and $\mathcal{Y}$ are the documents whose 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, $\mathcal{L} = \{l_1, ..., l_q\}$.

Since the spam problem can be tackled as a text classification problem, we use the *Waikato Environment for Knowledge Analysis* (WEKA) [13] and its Semi-

Supervised Learning and Collective Classification plugin[2]. In the remainder of this section we review the collective algorithms used in the empirical evaluation.

2.1 CollectiveIBk

It uses internally WEKA's classic IBk algorithm, implementation of the *K-Nearest Neighbour* (KNN), to determine the best k on the training set and builds then, for all instances from the test set, a neighbourhood consisting of k instances from the pool of train and test set (either a naïve search over 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 distance to the test instance they belong to. The neighbourhoods are sorted according to their 'rank', where 'rank' means the different occurrences of the two classes in the neighbourhood.

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

2.2 CollectiveForest

It uses WEKA's implementation of RandomTree as base classifier to divide the test set into folds containing 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 in a fold).

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

2.3 CollectiveWoods & CollectiveTree

CollectiveWoods works like CollectiveForest using CollectiveTree instead of RandomTree.

Collective tree is similar to WEKA's original RandomTree classifier, it splits the attribute at that position that divides the current subset of instances (training and test instances) into two halves. The process finishes if one of the following conditions is met:

- Only training instances would be covered (the labels for these instances are already known).
- Only test instances in the leaf, case in which distribution from the parent node is taken.
- Only training instances of one class, case in which all test instances are considered to have this class.

[2] http://www.scms.waikato.ac.nz/~fracpete/projects/collectiveclassification

To calculate the class distribution of a complete set or a subset, the weights are summed up 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 and, for the numeric attribute, distribution of the binary split based on median is calculated and then the weights are summed up for the two bins and finally normalised.

2.4 RandomWoods

It works like WEKA's classic RandomForest but using CollectiveBagging (classic Bagging, a machine learning ensemble meta-algorithm to improve stability and classification accuracy, extended to make it available to collective classifiers) in combination with CollectiveTree in contrast to RandomForest, which uses Bagging and RandomTree.

3 Empirical Evaluation

To evaluate the collective algorithms we used the *Ling Spam*[3] and *SpamAssassin*[4] datasets. Ling Spam consists of a mixture of both spam and legitimate messages retrieved from the *Linguistic list*, an e-mail distribution list about *linguistics*. It 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 (for a more detailed description of the corpus please refer to [14, 15]). From the 4 different datasets provided in this corpus, each of one with different pre-process steps, we choose 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. Unfortunatelly, due to computational restrictions we were obliged to reduce the dataset to a 50%, so the final used dataset comprises 3,023 e-mails, of which 964 are spam e-mails and 2,059 are legitimate messages.

In addition, we performed for both datasets a *Stop Word Removal* [16] based on an external stop-word list[5] and removed any non alpha-numeric character.

We then used the *Vector Space Model* (VSM) [17], an algebraic approach for *Information Filtering* (IF), *Information Retrieval* (IR), indexing and ranking, to create the model. This model represents natural language documents in a mathematical manner through vectors in a multidimensional space.

We extracted the top 1,000 attributes using *Information Gain* [18], an algorithm that evaluates the relevance of an attribute by measuring the information gain with respect to the class: $IG(j) = \sum_{v_j \in R} \sum_{C_i} P(v_j, C_i) \cdot (P(v_j, C_i)/(P(v_j) \cdot P(C_i))$ where C_i is the i-th class, v_j is the value of the j-th interpretation,

[3] http://nlp.cs.aueb.gr/software_and_datasets/lingspam_public.tar.gz
[4] http://spamassassin.apache.org/publiccorpus/
[5] http://www.webconfs.com/stop-words.php

$P(v_j, C_i)$ is the probability that the j-th attribute has the value v_j in the class C_i, $P(v_j)$ is the probability that the j-th interpretation has the value v_j in the training data, and $P(C_i)$ is the probability of the training dataset belonging to the class C_i.

We constructed an *ARFF* file [19] (i.e., Attribute Relation File Format) with the resultant vector representations of the e-mails to build the aforementioned WEKA's classifiers.

To evaluate the results, we measured the most frequently used for spam: precision, recall and Area Under the ROC Curve (AUC). We measured the precision of the spam identification as the number of correctly classified spam e-mails divided by the number of correctly classified spam e-mails 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.

Additionally, we measured the recall of the spam e-mail messages, which is the number of correctly classified spam e-mails divided by the number of correctly classified spam e-mails and 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 *Area Under the ROC Curve* (AUC), which establishes the relation between false negatives and false positives [20]. The ROC curve is represented by plotting the rate of true positives (TPR) against the rate of false positives (FPR). Where 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, $FPR = FP/(FP + TN)$.

For our experiments we tested the different configurations of the collective algorithms with sizes for the $\mathcal{X}$ set of known instances, varying from a 10% to a 90% of the instances used for training (i.e., instances known during the test).

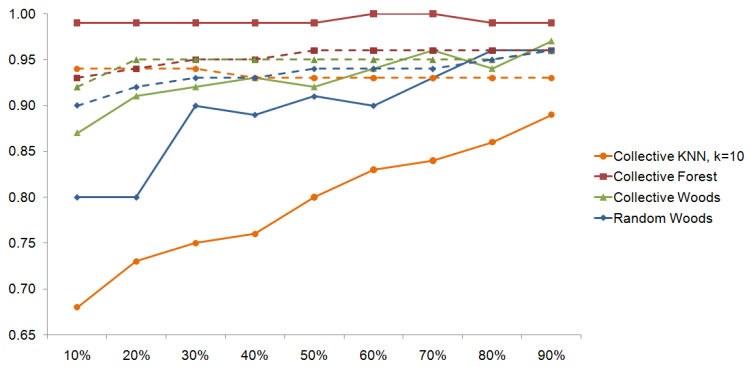

Fig. 1. Precision of the evaluation of collective algorithms for spam filtering with different sizes for the $\mathcal{X}$ set of known instances. Solid lines correspond to Ling Spam and dashed lines correspond to SpamAssassin.

Fig. 1 shows the precision of the different algorithms. Collective KNN shows significant improvements with Ling Spam when the number of known instances increases (from 0.68 with 10% to 0.89 with 90%), but remains constant with SpamAssassin (between 0.93 and 0.94). Collective Forest was the best collective algorithm when evaluating the precision achieving between 0.99 and 1.00 for Ling Spam and no less than 0.93 for SpamAssassin. Finally, Collective Woods and Random Woods experience some improvements when increasing the number of known instances when testing with both datasets.

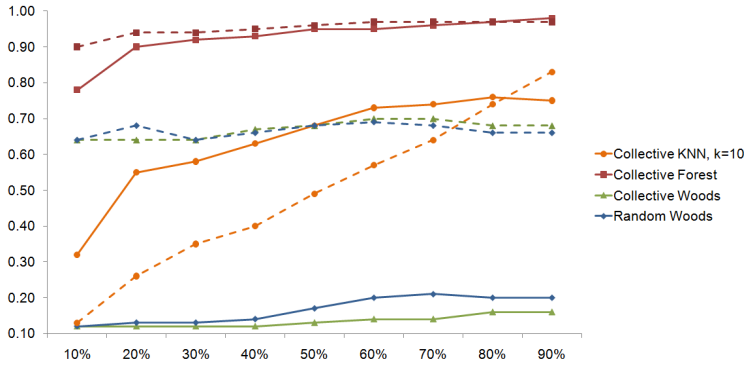

Fig. 2. Recall of the evaluation of collective algorithms for spam filtering with different sizes for the $\mathcal{X}$ set of known instances. Solid lines correspond to Ling Spam and dashed lines correspond to SpamAssassin.

Fig. 2 shows the recall of the different algorithms. Again, Collective KNN shows better results, although not good enough, when the number of known instances increases: from a 0.32 with 10% to 0.75 with 90% for Ling Spam and from 0.13 with 10% to 0.83 with 90%. Collective Forest presents a poor 0.78 for 10% with Ling Spam but behaves better with the rest of configurations in both datasets: a minimum of 90% and a maximum of 0.97. Finally, Collective Woods and Random Woods behave similar, with very poor recall, achieving maximums with 90% of 0.16 and 0.20 respectively for Ling Spam and 0.68 and 0.66 for SpamAssassin.

Finally, Fig. 3 shows the Area under de ROC curve (AUC) of the different algorithms. Once more, the performance of Collective KNN increases with more known instances: from 0.64 with 10% to 0.87 with 90% for Ling Spam and from 0.56 to 0.90 for SpamAssassin. Collective Forest offers a perfect 1.00 for every configuration with Ling Spam and a minimum of 0.99 with SpamAssassin posing as a suitable choice for collective classification. Finally, Collective Woods and Random Woods offer similar results, increasing from 0.86 both to 0.92 and 0.90 respectively with Ling Spam and from 0.93 and 0.92 to 0.94 both with SpamAssassin.

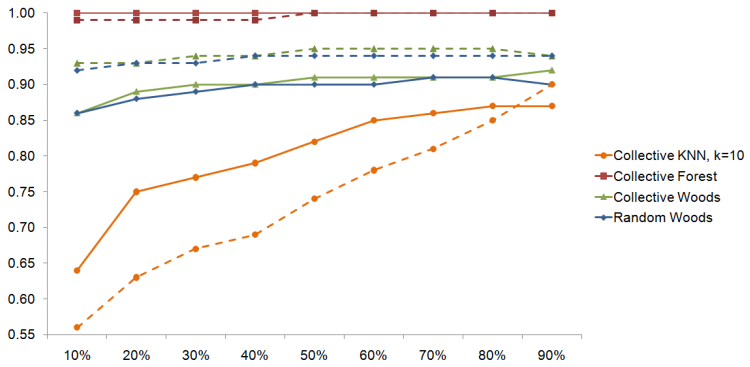

Fig. 3. Area under de ROC curve (AUC) evaluation of collective algorithms for spam filtering with different sizes for the $\mathcal{X}$ set of known instances. Solid lines correspond to Ling Spam and dashed lines correspond to SpamAssassin.

4 Discussion and Concluding Remarks

Collective Classification algorithms for spam filtering pose as a suitable approach for optimising the classification of partially-labelled data and, therefore, overcome the amount of unclassified spam e-mails that are created every day.

In particular, Collective Forest shows great results for every configuration of known instances (i.e., different sizes for the $\mathcal{X}$ set of known instances), with values above 0.93 of precision, above 0.90 of recall (only offering a poor recall of 0.78 with a 10% of $\mathcal{X}$) and almost 1.00 for all configurations of AUC.

Since precision and AUC are slightly affected with the variation of known instances, values of $\mathcal{X}$, to determine the optimal size of labelled data, and assuming that Collective Forest is the chosen algorithm, the recall should be the factor to take into account. For a value of $\mathcal{X} = 60\%$, CollectiveForest achieves its maximums, only experiencing a loss of 0.03 of recall for Ling Spam.

As the number of unsolicited bulk messages increases, the classification and labelling steps, that commonly supervised methods make use of, become more unattainable. To revert this situation, we propose the first spam filtering system that uses collective classification to optimise classification performance. Through the algorithms introduced, the necessity of labelled e-mails is minimised, by a 40%, without a significant penalisation in the detection capabilities.

Future work will be focused on three main directions. First, we plan to extend our study of collective classification by applying more 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 on the effects of the labelled degree of the data.

References

1. Robinson, G.: A statistical approach to the spam problem. Linux J. **2003** (March 2003) 3
2. Chirita, P., Diederich, J., Nejdl, W.: MailRank: using ranking for spam detection. In: Proceedings of the 14th ACM international conference on Information and knowledge management, ACM (2005) 373–380
3. Schryen, G.: A formal approach towards assessing the effectiveness of anti-spam procedures. In: System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on. Volume 6., IEEE (2006) 129–138
4. Chiu, Y., Chen, C., Jeng, B., Lin, H.: An Alliance-Based Anti-spam Approach. In: Natural Computation, 2007. ICNC 2007. Third International Conference on. Volume 4., IEEE (2007) 203–207
5. 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
6. Mason, J.: Filtering spam with spamassassin. In: HEANet Annual Conference. (2002)
7. Raymond, E.: Bogofilter: A fast open source bayesian spam filters (2005)
8. Burton, B.: Spamprobe-bayesian spam filtering tweaks. In: Proceedings of the Spam Conference. (2003)
9. Dengel, A., Dubiel, F.: Clustering and classification of document structure-a machine learning approach. Document Analysis and Recognition, International Conference on **2** (1995) 587
10. Fujisawa, H., Nakano, Y., Kurino, K.: Segmentation methods for character recognition: from segmentation to document structure analysis. Proceedings of the IEEE **80**(7) (2002) 1079–1092
11. Denoyer, L., Gallinari, P.: Bayesian network model for semi-structured document classification. Information Processing & Management **40**(5) (2004) 807–827
12. Namata, G., Sen, P., Bilgic, M., Getoor, L.: Collective classification for text classification. Text Mining (2009) 51–69
13. Garner, S.: Weka: The Waikato environment for knowledge analysis. In: Proceedings of the New Zealand Computer Science Research Students Conference. (1995) 57–64
14. 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
15. Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C., Stamatopoulos, P.: A memory-based approach to anti-spam filtering for mailing lists. Information Retrieval **6**(1) (2003) 49–73
16. Wilbur, W., Sirotkin, K.: The automatic identification of stop words. Journal of information science **18**(1) (1992) 45–55
17. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Communications of the ACM **18**(11) (1975) 613–620
18. Kent, J.: Information gain and a general measure of correlation. Biometrika **70**(1) (1983) 163–173
19. Holmes, G., Donkin, A., Witten, I.H.: Weka: a machine learning workbench. (August 1994) 357–361
20. 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