Training SpamAssassin with Active Semi-supervised Learning

Jun-Ming Xu[†] Giorgio Fumera[‡] Fabio Roli[‡] Zhi-Hua Zhou[†]
[†]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China
[‡]Dept of Electrical and Electronic Eng., University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy
{xujm, zhouzh}@lamda.nju.edu.cn {fumera, roli}@diee.unica.it

ABSTRACT

Most spam filters include some automatic pattern classifiers based on machine learning and pattern recognition techniques. Such classifiers often require a large training set of labeled emails to attain a good discriminant capability between spam and legitimate emails. In addition, they must be frequently updated because of the changes introduced by spammers to their emails to evade spam filters. To address this issue active learning and semi-supervised learning techniques can be used. Many spam filters allow the user to give a feedback on personal emails automatically labeled during filter operation, and some filters include a self-training mechanism to exploit the large number of unlabeled emails collected during filter operation. However, users are usually willing to label only a few emails, and the benefits of self-training techniques are limited. In this paper we propose an active semi-supervised learning method to better exploit unlabeled emails, which can be easily implemented as a plug-in in real spam filters. Our method is based on clustering unlabeled emails, querying the label of one email per cluster, and propagating such label to the most similar emails of the same cluster. The effectiveness of our method is evaluated using the well known open source SpamAssassin filter, on a large and publicly available corpus of real legitimate and spam emails.

1. INTRODUCTION

Most current commercial and open source spam filters include automatic pattern classifiers based on machine learning and pattern recognition techniques. Such classifiers usually require a representative and large training set of labeled emails to attain a good discriminant capability between spam and ham (legitimate) emails. Moreover, because spammers always devise new tricks to evade spam filters, spam filtering task is a typical *adversarial* classification problem. To deal with ever-changing characteristics of spam emails and keep the filter effectiveness high, classifiers require to be frequently retrained with updated training samples. Such update can be done by the producers of the filter, who can collect large email corpora and analyze them manually or automatically, or by the end users, who can manually confirm or correct the label assigned by the filter to their own emails, collected during filter operation. While the latter method can build user-specific filters by using the ham emails of a given user, it suffers from the drawback that users are usually not willing to label too many emails. This is true especially for server side filters, which usually require users to provide their feedback through web interfaces.

To relieve the burden of providing feedback on the emails automatically labeled by a spam filter, several authors proposed to exploit active learning (AL) techniques like uncertainty sampling, or semi-supervised learning (SSL) techniques [2]. SSL and AL techniques have been studied in the machine learning field to address a common issue to many supervised learning tasks, namely the fact that collecting a sufficiently large set of labeled examples is often difficult or costly, while unlabeled examples are usually much easier and cheaper to obtain. SSL methods aim at exploiting a relatively small set of labeled examples together with a larger set of unlabeled examples to train a classifier without human intervention. Instead, AL techniques aim at selecting a small but informative subset of unlabeled examples, whose label is queried to the user. Such newly labeled examples are used to train the classifier together with previously labeled ones. Some SSL techniques are currently used also in commercial and open source spam filters. For instance, the well known SpamAssassin open source filter includes a simple SSL component based on self-training, called *auto-learning*, which basically enhance the training set with the emails classified with high confidence, using the labels predicted by the filter.

SSL and AL techniques exhibit however some drawbacks in the spam filtering task. A disadvantage of SSL techniques, especially for an adversarial learning task like spam filtering, is that the unlabeled examples added to the training set are likely to be the less informative ones, since they are often correctly classified by the filter. In other words, the lack of feedback from users may confine the capability of SSL methods to capture the variability of spam characteristics. A deficiency of AL techniques based on uncertainty sampling is that the spamminess score is not necessarily the best criterion to select the most informative emails.

In this paper we propose a method, Active Semi-supervised learning based on Clustering Analysis (ASCA), aimed at exploiting the advantages of both SSL and AL in spam filtering tasks, overcoming their drawbacks mentioned above. ASCA is a hybrid of SSL and AL, and is devised to be easily implemented as a plug-in of existing spam filters. Our method is inspired by the work of Nguyen et al. [7], who proposed an AL method with pre-clustering and showed its effectiveness on image classification tasks, and the work of Zhou et al. [15, 16], who proposed a method combining AL with SSL and showed its effectiveness on image retrieval tasks. ASCA works as follows. First, unlabeled emails are clustered into a given number of clusters which depends on the number of emails a user would like to manually label and on the number of available unlabeled emails. The label of one email in each cluster which is deemed to contain only spam or only ham emails is asked to the user, and such label is propagated to the most similar emails in the same cluster. Emails labeled by the user and by label propagation are used to re-train the spam filter. The remaining ones can be clustered again to repeat the whole process, until the user is not willing to label any more emails. The effectiveness of our method is experimentally investigated using the SpamAssassin filter and a large and publicly available corpus of real ham and spam emails (TREC-07).

This paper is structured as follows. In the next section, we briefly review previous works on SSL and AL techniques applied to spam filtering, and give an overview of SpamAssassin, especially on its auto-learning SSL component. Our method is described in section 3. Its experimental investigation is reported in section 4.

2. PREVIOUS WORKS

In this section we give a brief overview on SSL and AL techniques proposed so far for the spam filtering task. We also describe how these techniques have been implemented in the open source Spam Assassin filter, which will be used to experimentally investigate the active semi-supervised learning method proposed in this paper.

2.1 Semi-Supervised Learning and Active Learning for Spam filtering

Many SSL algorithms have been proposed so far in the machine learning field. A comprehensive overview can be found in [18]. Most of them have been applied to tasks related to text classification, in which labeled examples are usually costly to obtain, while unlabeled examples are much cheaper to collect. Roughly speaking, SSL methods fall into four categories: generative methods [8], semi-supervised support vector machines (S3VMs) [1, 3], graph-based methods [14, 19] and disagreement-based methods [17]. Several SSL methods have been applied to spam detection [4, 9].

The key of AL techniques is to identify the most informative unlabeled examples. Algorithms belonging to the uncertainty sampling scheme train a single learner and then query the unlabeled example on which the learner is least confident. Segal et al. [11] developed an approximate, but faster, uncertainty sampling algorithms for labeling large email corpora. Sculley [10] investigated several online active learning algorithms based on different uncertainty sampling criteria. Other AL methods are based on committee-based sampling, which consists in constructing multiple learners and in querying the unlabeled examples on which the learners mostly disagree. The main deficiency of these methods in the spam filtering task is their computational cost. Several AL methods exploit instead clustering to select the unlabeled samples (see for instance [7] and references therein), but they have not been proposed for spam filtering tasks.

Some theoretical and practical works have been proposed to combine the advantageous of SSL and AL techniques. For instance, McCallum and Nigam [5] combined committee-based AL with EM-based SSL algorithm for text classification. Minton et al. [6] developed an algorithm to combine SSL and AL under multi-view learning framework. Zhou et al. [15, 16] proposed a method to combine disagreement-based SSL and AL into one framework, and applied it successfully to content-based image retrieval. Later, Wang and Zhou [12] provided a theoretical justification to the effectiveness of the combination of disagreement-based SSL and AL. To the best of our knowledge, however, the combination of SSL and AL has not been considered for the specific spam filtering task.

2.2 SpamAssassin

SpamAssassin[1] is a well-known and widely used open source spam filter. It is made up of about nine hundred binary-valued "tests", each of which is related to a characteristic of spam or ham emails like the presence of a given keyword in the emails text or some kind of malformed email headers. Each test is associated with a score: the score is zero if the related characteristic is not present, while it is non-zero if such characteristic is present. In particular, scores associated with spam and ham features are respectively positive and negative. For a given email, the score of all tests are summed up. If their sum is greater than a predefined threshold the email is labeled as spam, otherwise it is labeled as ham. The default threshold is 5.0 (note that the scores are not normalised to any interval like $[0, 1]$). Among the tests, nine are associated to disjoint intervals of the continuous-valued output of a text classifier ("naive bayes" classifier). The remaining tests are simple feature detectors which look at the emails' body and header, including DNS block-lists and collaborative filtering databases. Default values are provided for the score of each test and for the decision threshold. These values can be modified by the user, either automatically, using a tool named "mass check" and a training set of labeled emails, or manually. Note that the naive bayes classifier and the mass-check tool are the only components of SpamAssassin based on machine learning techniques. This structure makes it easy to update SpamAssassin by adding or removing tests, as spam email characteristics change.

SpamAssassin provides a SSL mechanism, called "auto-learning", which automatically re-trains the naive bayes classifier using some unlabeled emails collected during filter operation. It is based on the simplest SSL method known as self-training, and consists in constructing a training set made up of emails classified with high confidence, using the labels automatically assigned by the filter. In particular, SpamAssassin uses emails for which the sum of scores associated with some subsets of tests is much greater (for emails labeled as spam) or much lower (for emails labeled as ham) than given

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

thresholds. As explained in section 1, self-training is simple to implement but its effectiveness is likely to be limited, since the selected emails are the less informative ones. Besides this, SpamAssassin can update the naive bayes classifier when user gives his feedback on the label of any email. However it just passively waits for labeled emails when users are willing to do that. No AL approach is used to select emails to query the user. In this paper, we focus on how to better exploit unlabeled emails to improve the naive bayes classifier of SpamAssassin.

3. ACTIVE SEMI-SUPERVISED LEARNING FOR SPAM FILTERING

As explained in the above sections, spam filters need to be frequently updated to keep their effectiveness high against changes in spam emails characteristics introduced by spammers. This requires re-training the components based on machine learning techniques, using updated training sets of labeled spam and ham emails. Unlabeled emails collected during filter operation for any given user can be very useful to this aim. SSL and AL techniques can be used to exploit unlabeled emails, without (SSL) or with limited (AL) user intervention. So far, works on spam filtering considered only one kind of technique, either SSL or AL. However, as suggested by other authors for different tasks, we argue that properly combining SSL and AL can boost the benefits of both approaches and avoid some of the respective drawbacks pointed out in the previous sections, also in the spam filtering task.

In the following we propose a hybrid SSL and AL method, Active Semi-supervised learning based on Clustering Analysis (ASCA). Our main goal was to devise a simple method tuned to spam filtering tasks, which can also be easily implemented as a plug-in of existing spam filters. ASCA is based on the following rationale. Given a set of unlabeled emails collected during filter operation, self-training can be firstly used to exploit emails classified with high confidence. However, the remaining emails classified with low confidence are likely to be more informative to update a classifier. AL can thus be used to exploit these emails by asking the user their correct label. To reduce the number of queries to the user, one can exploit the fact that spam emails are often constructed by introducing random variations to a given template email. Such emails are likely to be more similar to each other than to unrelated ones. To this aim, as suggested in [7] for tasks different than spam filtering, clustering can provide useful information to guide the choice of few *representative* emails to query the user. In particular, an AL algorithm should prefer selecting emails from *different* clusters. Finally, to some extent emails in the same cluster can be expected to share the same label. This suggests that the classifier can be further updated by using a kind of SSL method, namely by propagating the label provided by the user to other similar emails in the same cluster.

The scheme of the algorithm is given in Algorithm 1 below. In detail, the algorithm works as follows. We denote with L the initial training set consisting of labeled emails, where the label is either *spam* or *ham*. The available classification algorithm $\mathcal{F}$ is initially trained on L. Then a set of unlabeled emails U is collected during filter operation. A self-training algorithm, if available, is applied to emails in U to update the classifier $\mathcal{F}$. In this case, emails used by self-training are removed from U. We assume now that the user is willing to label at most q emails among the available unlabeled ones. The task is to select at most q unlabeled emails from U to update the classifier $\mathcal{F}$. To this aim, first all emails in U are clustered into c clusters, whose set is denoted as $C = \{C_1, C_2, \ldots, C_c\}$ ($U = \bigcup_{i=1,\ldots,c} C_i$), where c is a predefined parameter. Based on the labels assigned to emails in U by the classifier $\mathcal{F}$, clusters are categorized into *pure* and *non-pure*, where a pure one is defined as a cluster made up of emails which are given an identical label (either *spam* or *ham*) by the classifier. Emails in pure clusters are likely to share the same *correct* label, though it could not coincide with the one assigned by the classifier. Accordingly, until the user is willing to label new emails, one email in each pure cluster is selected, denoted as *query*, and its label is asked to the user. The selected email is the one closest to the cluster centroid. Then, the label given by the user to the *query* email could be propagated to other similar emails in the same cluster. To avoid labeling errors, the label is propagated only to a predefined fraction $p \in [0, 1]$ of emails closest to *query*, where p is named *propagation rate*. If the number of pure cluster is greater than the number of queries, larger clusters are selected first. Finally, the email closest to the centroid of each non-pure cluster is selected and its label is asked to the user. No label propagation is carried out for non-pure clusters, since labeling errors are more likely than in pure ones. At the end of the above steps it can happen that the number of queries made to the user is lower than the allowed number of queries q. In this case, the whole process above is repeated on all the emails in U not labeled by the user nor by label propagation, until the number of queries reaches q or all emails in U have been labeled.

We point out that the number c of clusters and the propagation ratio p have to be chosen in advance in the ASCA algorithm: the issue of how they affect the performance of ASCA and how their values can be chosen will be addressed in the experimental investigation in the next section. It is also worth noting that the clustering algorithm and the corresponding distance measure can affect the performance of ASCA.

4. EXPERIMENTS

In this section we report the results of experiments carried out to investigate the effectiveness of the proposed ASCA method. The experiments have been carried out using the SpamAssassin filter, and the TREC07p corpus of real ham and spam emails. As explained above, we point out that our method can be easily implemented as a plug-in of real spam filters. In particular, our experiments were carried out without modifying the original SpamAssassin filter. In the following we will first describe the experimental setup and the TREC07p data set. Then we report the obtained results and discuss the issues mentioned in section 3 about the choice of the parameter values.

4.1 Experimental setup

The experiments were carried out using the latest available version of SpamAssassin (3.2.5), with all rules enabled. The default settings were used, including scores and decision threshold values, except for the ones explained below. We remind the reader that SpamAssassin is made

Algorithm 1: Active Semi-supervised learning based on Clustering Analysis (ASCA)

```
Input: $L$: a set of labeled e-mails
    $U$: a set of unlabeled e-mails
    $\mathcal{F}$: a classification algorithm
    $q$: number of e-mails user is willing to manually label
    $p$: propagation ratio
    $c$: number of clusters
if self-training is available then
    Update $\mathcal{F}$ with $U$;
    $U \leftarrow U -$ e-mails used by self-training ;
$queries \leftarrow 0$ ;
while $queries < q$ and $U \neq \varnothing$ do
    $L' \leftarrow \varnothing$; Cluster $U$ into $c$ clusters $C = \{C_1, C_2, \ldots, C_c\}$;
    $C_{\mathrm{pure}} \leftarrow$ set of pure clusters in $C$ ;
    Sort clusters in $C_{\mathrm{pure}}$ and $C - C_{pure}$ in descending order $|C_i|$
    foreach $C_i \in C_{\mathrm{pure}}$ do
        if $queries \geq q$ then
            break;
        $query \leftarrow$ centroid of $C_i$;
        $y \leftarrow$ label of $query$ given by the user; $L' \leftarrow L' \bigcup (query, y)$;
        $queries \leftarrow queries + 1$; $U \leftarrow U - query$;
        $D \leftarrow$nearest $|C_i| \cdot p$ neighbors to $query$ in $C_i$;
        add to $L'$ all e-mails in $D$ with label $y$; $U \leftarrow U - C_i$;
    foreach $C_i \in C - C_{pure}$ do
        if $queries \geq q$ then
            break;
        $query \leftarrow$ centroid of $C_i$;
        $y \leftarrow$ label of $query$ given by the user; $L' \leftarrow L' \bigcup (query, y)$;
        $queries \leftarrow queries + 1$; $U \leftarrow U - \{query\}$;
    Update $\mathcal{F}$ with $L'$;
return $\mathcal{F}$
```

up of about nine hundred binary tests, nine of which are associated to the output of a naive bayes classifier. The naive bayes is the only classifier used in SpamAssassin, and the only component updated by its self-training algorithm. When the number of spam and ham training emails is below two predefined thresholds, 'bayes_min_ham_num' and 'bayes_min_spam_num', SpamAssassin does not take the naive bayes classifier's output into account, because its performance may be not good. We set both thresholds to 1, to be able to investigate the performance improvement that can be attained by our method even when the initial training set is small. Furthermore, to take into account the possible changes in spam emails during time SpamAssassin provides an 'auto-expire' mechanism which disregards the oldest terms in the naive bayes classifier. However this may introduce some randomness in the experiments. To avoid this, we disabled 'auto-expire' by setting the 'bayes_auto_expire' parameter to 0. Moreover, some network rules such as 'Pyzor' query network servers to obtain scores, which makes it possible for the score of one same email to change over time. To avoid this, the scores of all emails were computed only once and stored in order to be used whenever needed without modifications. The only exception was obviously the scores of rules associated to the naive bayes classifier, which were the only ones modified by our algorithm.

The clustering step of ASCA was implemented as follows. Unlabeled emails were represented as feature vectors in which each feature is associated to a term (word) in the email's body. Standard stemming and stop-word removal were used to reduce the term set. The chosen feature representation was the well known term frequency - inverted document frequency (TF-IDF). Then, the simple and well known k-means algorithm was used to cluster unlabeled emails, using the Euclidean distance as distance measure in the feature space. We used the Weka [13] implementation of the k-means algorithm.

The experiments were carried on the large, publicly available TREC07p[2] corpus of real emails. It is made up of 75,419 emails, comprising 25,220 ham and 50,199 spam emails. All emails were arranged in chronological order to simulate a real spam filtering scenario in which training emails are older than testing ones. As the initial training set for the naive bayes classifier we randomly selected 50 emails among the first 10,000 ones. We chose such a small training set to take into account that labeling emails is a time consuming task. Then we randomly chose 10,000 emails from the next 30,000 ones, and used them as the unlabeled emails to apply the auto-learning procedure and our algorithm. The above process was repeated ten times, to carry out ten runs of all the experiments. The remaining 35,419 emails were used as test set.

Each run of the experiments was carried out as follows. First, the naive bayes classifier of Spam Assassin was trained on the 50 emails of the initial training set. Then the 10,000 unlabeled emails were classified by SpamAssassin, with the auto-learning feature enabled. After this we disabled auto-learning and applied ASCA to the emails which remain unlabeled after auto-learning. The performances of SpamAssassin before and after running auto-learning, and after running both auto-learning and ASCA, were then evaluated on the testing emails. All the results reported below refer to average values over the ten runs of the experiments.

The performance measure is the standard receiver operating characteristic (ROC) curve, i.e. the true positive rate (TP, the fraction of correctly recognized spam emails) vs the false positive rate (FP, the fraction of ham emails erroneously labeled as spam), obtained as functions of the decision threshold of SpamAssassin. Since false positive errors are much more harmful than false negative ones, our evaluation was focused on the low-FP part of the ROC curve only.

4.2 Results

We first compare the performance of ASCA to the baseline performance of SpamAssassin, attained by using only the default auto-learning procedure. For comparison we also consider two simpler criteria to select unlabeled emails to query the user: random selection and uncertainty sampling. The former is implemented simply by randomly selecting q emails among the unlabeled ones not used by auto-learning, to be used as a baseline for comparison. The latter consists in repeating the following procedure for q times: query the label of the most uncertain email and re-train the classifier. As confidence measure we used the absolute value of the difference between the score given by SpamAssassin and the default decision threshold of 5.0. The two criteria above and the default SpamAssassin version were compared with

[2] `http://plg.uwaterloo.ca/~gvcormac/treccorpus07`

ASCA for three different values of the number of queries q: 10, 50, 100. For this comparison, the other two parameters of ASCA were fixed as follows: the number of clusters c was set to 50, and the propagation rate p was set to 0.1. The effect of these two parameters on the behavior of ASCA will be investigated in next subsection. The results are shown in figures 1–3 (note that the curve corresponding to auto-learning only is the same in all figures). From these results, we can see that ASCA always outperformed the standard SpamAssassin for low FP rates, as well as the two alternative criteria for email selection during the active learning step. In particular, an unexpected result is that random sampling outperformed uncertainty sampling. The reason seems that in uncertainty sampling the distribution of the selected emails is biased towards scores close to the decision threshold, and is thus not representative of the testing set. In particular, these emails may be very similar, and thus they could provide limited information. Another possible explanation for poor performance of uncertainty sampling is that the naive bayes classifier is only one of the SpamAssassin components. Therefore, if the emails are selected based on the output of SpamAssassin, it is possible to select similar emails even if the classifier is re-trained after each query. Moreover, ASCA does not need to re-train the classifier after each user query and re-scan all unlabeled emails, as uncertainty sampling. The ROC curves of ASCA for the three different q values are shown also in figure 4, together with the ones of SpamAssassin with and without auto-learning, to make the comparison easier. It can be seen that the advantage of ASCA over SpamAssassin, for low FP rates, increases as the number of users queries increases.

4.3 Parameter Selection

We carried out some experiments to study the behavior of ASCA with respect to its parameters: the number of queries q, the number of clusters c, and the propagation rate p. As in the above experiments, our evaluation was focused on the low-FP rate part of the ROC curve. The number of queries q depends on how many emails the user would like to label. As shown by the above results, the performance of ASCA seems to improve for increasing values of q, which is reasonable.

The number of clusters c specifies into how many clusters ASCA will subdivide the emails which remain unlabeled after auto-learning. Three possible values were investigated in the experiments: $c = 10$, 50, 100, and compared under nine different combinations of values of q: 10, 50, 100, and of p: 0.1, 0.5 ,1.0. For the sake of brevity, we report only results on two of the nine combinations above (figure 5 and 6). Under seven combinations out of nine, the results were similar to the ones of figure 5. The results of the other two combinations ($q = 50$, $p = 0.5, 1.0$) are show in Fig 6. In principle, the choice of the number of clusters can affect ASCA performance. However no significant differences were observed when c was in the considered range, between 10 and 100. The possible reason is that for the purpose of this application it is not necessary to identify the "best" emails clustering nor to obtain only pure clusters, namely clusters in which all emails are given the same label by the spam filter. The clustering results just convey some useful information to guide the selection of queries and to identify neighboring emails to propagate the labels given by the user. Even if clustering results are not so good with respect

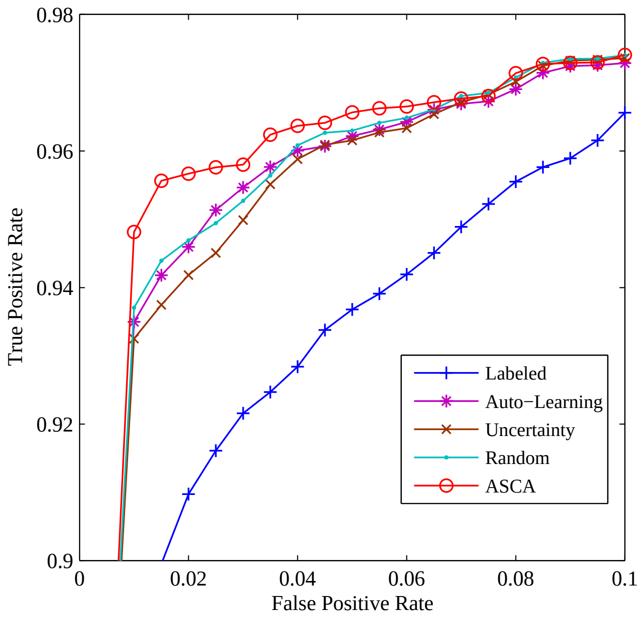

Figure 1: Average test set ROC curves of the proposed method (ASCA) with $q = 10$ user queries, $c = 50$ clusters and propagation rate $p = 0.1$. Also shown the ROC curves of the same method with two alternative email selection criteria (random selection and uncertainty sampling), and of the standard SpamAssassin filter, using only the original training set of labeled emails (Labeled) and using also auto-learning (Auto-Learning).

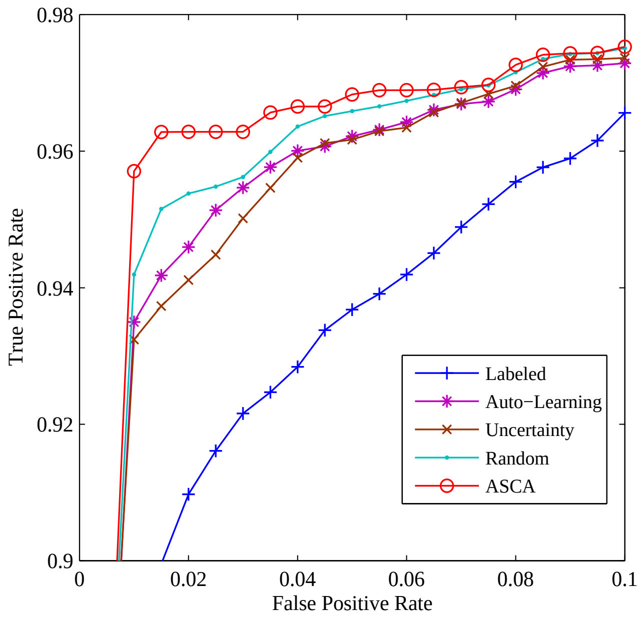

Figure 2: The same ROC curves as in figure 1, with $q = 50$ user queries.

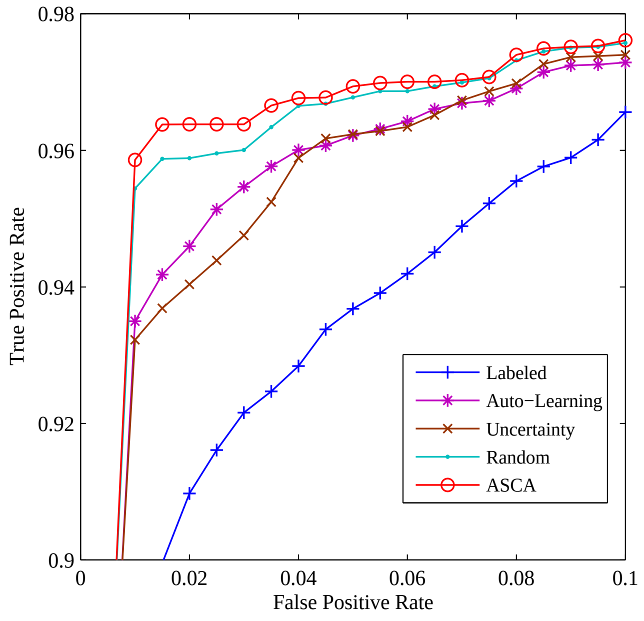

Figure 3: The same ROC curves as in figure 1, with $q = 100$ user queries.

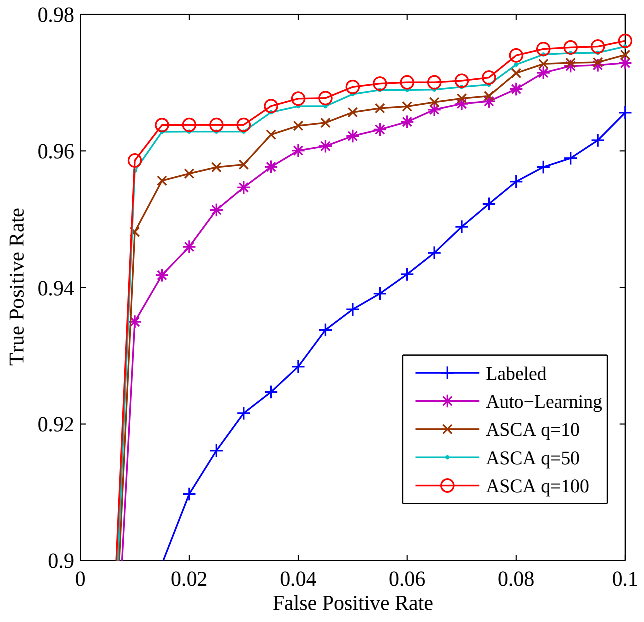

Figure 4: Average test set ROC curves of our method for the three different q values considered (10, 50, 100) and of SpamAssassin (see captions of figures 1–2).

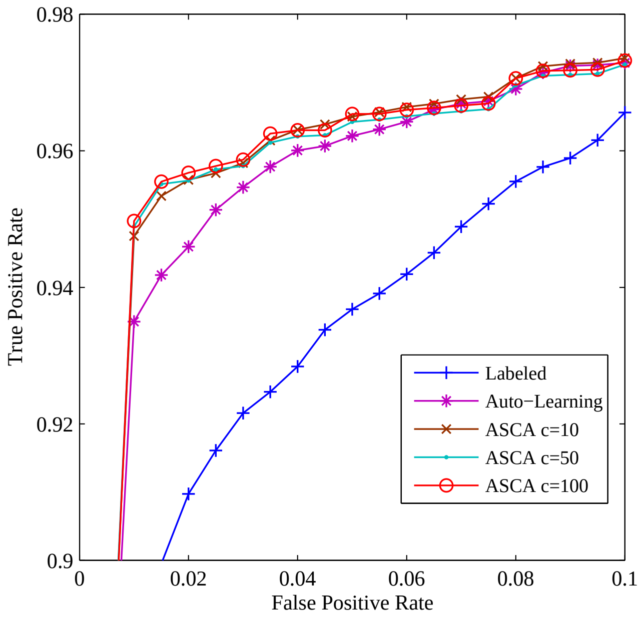

Figure 5: Average test set ROC curves of SpamAssassin (see caption of figure 1), and of our method for $q = 10$ user queries and propagation rate $p = 0.5$, for different number of clusters c.

to a clustering perspective, they nevertheless allow to identify enough informative emails, which is more important for ASCA. Taking into account that in our experiments around 3,700 unlabeled emails were used by ASCA, and the three considered c values (10, 50, 100) did not exhibit significant differences, we can suggest that setting c as one percent of the number of unlabeled emails might be a reasonable choice.

The propagation rate p is the ratio of emails in a given pure cluster to which the label of the query email is propagated by ASCA. Three possible values, 0.1, 0.5, 1.0, were investigated in our experiments. When $p = 1.0$, all emails in a pure cluster are given the same label as the query email. The performance of ASCA for these three p values are evaluated under nine different combinations of values of q (10, 50, 100) and c (10, 50, 100). Again, for the sake of brevity only two representative results are reported in figures 7 and 8. It can be seen that for most parameters values the best performance was attained for $p = 0.1$, while propagating labels to all emails in a cluster does not seem to be a good choice. This raises the question whether label propagation is really useful. To answer this question, we evaluated ASCA performance with $p = 0$ (no label propagation). The performance of ASCA without label propagation is shown in figures 7 and 8. When c and q were set to {10,50}, {10,100}, {50,50}, {50,100}, i.e. $c \leq q$, the performance was close to the one attained with $p = 0.1$. Instead, when c and q were set to {10,10}, {50,10}, {100,10}, {100,50}, {50,100}), i.e. $c \geq q$ (except for the last pair of values), the performance was worse than using label propagation, and sometimes it was even worse than the baseline performance of SpamAssassin with auto-learning only. These results suggest that,

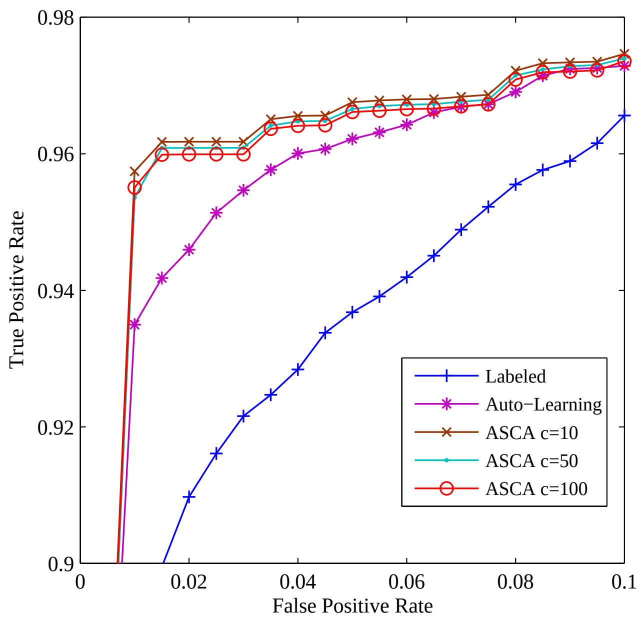

Figure 6: The same ROC curves of figure 5, for $q = 50$ user queries and propagation rate $p = 0.5$.

when the number of queries q is very small, around ten in our experiments, a limited label propagation turns out to be useful to enlarge the training set without introducing labeling errors. Instead, when the number of queries is large enough, at least 50 in our experiments, active learning can provide enough labeled emails and label propagation does not provide a significant improvement.

5. CONCLUSION

In this paper we addressed the problem of updating machine learning-based spam filters components exploiting the usually large amount of unlabeled emails processed during filter operation, and asking the user to manually label only a small number of such emails. In particular, we addressed this problem under the viewpoint of providing an algorithm which can be easily implemented as a plug-in of existing spam filters, without modifying their structures. Methods based either on active learning or on semi-supervised learning techniques were previously proposed to this aim by other authors. We proposed a simple, hybrid active learning and semi-supervised learning method to exploit the strengths of both methods and avoid some of their weaknesses.

In particular, while most active learning methods for spam filtering are based on uncertainty sampling, our method is based on clustering unlabeled emails and then selecting a representative email from each cluster. The rationale is that spam emails are often constructed by adding random changes to a given template. Moreover, a moderate label propagation brings further benefit when the user is willing to manually label only a small number of emails, as it often happens in practice.

The proposed method can be implemented in real spam fil-

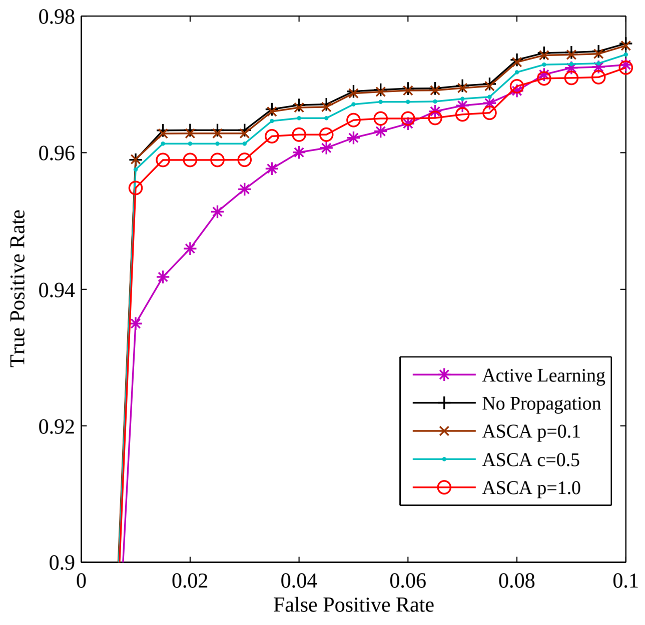

Figure 7: Average test set ROC curves of SpamAssassin (see caption of figure 1), and of our method for $q = 100$ user queries and $c = 10$ clusters, for different values of the propagation rate p.

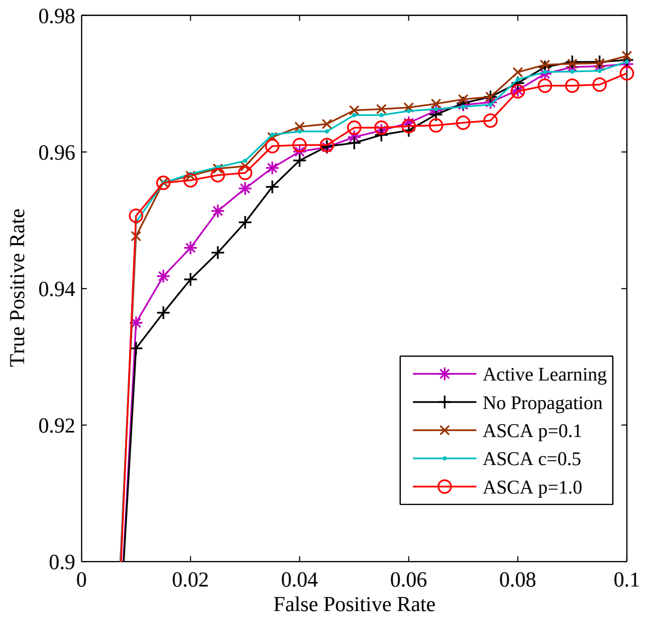

Figure 8: The same ROC curves of figure 7, for $q = 10$ user queries and $c = 100$ clusters.

ters without modifying their original structures. We investigated its performance using the well known and widespread open source SpamAssassin filter, on a large and publicly available collection of real ham and spam emails. Experimental results showed that our method can improve the performance of SpamAssassin with respect to the use of the standard self-training procedure only. We also discussed the behavior of the proposed method with respect to the values of its three parameters, namely the allowed number of queries, the number of clusters and the label propagation ratio, and suggested some guidelines for the choice of the latter two.

Acknowledgements

The authors would like to thank the anonymous reviewers for helpful suggestions, and Battista Biggio for helpful discussion. This research was partially supported by the National Science Foundation of China (60635030, 60721002), Jiangsu Science Foundation (BK2008018) and the Foundation for the Author of National Excellent Doctoral Dissertation of China (200343).

6. REFERENCES

[1] O. Chapelle, V. Sindhwani, and S. S. Keerthi. Branch and bound for semi-supervised support vector machines. In B. Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*, pages 217–224. MIT Press, Cambridge, MA, 2007.

[2] G. V. Cormack. Harnessing unlabeled examples through iterative application of dynamic markov modeling. In *Proceedings of the ECML-PKDD 2006 Discovery Challenge Workshop*, pages 10–15, Berlin, Germany, 2006.

[3] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209, Bled, Slovenia, 1999.

[4] D. Mavroeidis, K. Chaidos, S. Pirillos, and M. Vazirgiannis. Using tri-training and support vector machines for addressing the ecml-pkdd 2006 discovery challenge. In *Proceedings of the ECML-PKDD 2006 Discovery Challenge Workshop*, pages 39–47, Berlin, Germany, 2006.

[5] A. McCallum and K. Nigam. Employing EM and pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning*, pages 350–358, Madison, WI, 1998.

[6] S. Minton and C. A. Knoblock. Active + semi-supervised learning = robust multi-view learning. In *Proceedings of the 19th International Conference on Machine Learning*, pages 435–442, Sydney, Australia, 2002.

[7] H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *Proceedings of the 21st International Conference on Machine learning*, pages 623–630, Banff, Canada, 2004.

[8] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2-3):103–134, 2000.

[9] B. Pfahringer. A semi-supervised spam mail detector. In *Proceedings of the ECML-PKDD 2006 Discovery Challenge Workshop*, pages 48–52, Berlin, Germany, 2006.

[10] D. Sculley. Online active learning methods for fast label-efficient spam filtering. In *Proceedings of the 4th Conference on Email and Anti-Spam*, Mountain View, CA, 2007.

[11] R. Segal, T. Markowitz, and W. Arnold. Fast uncertainty sampling for labeling large e-mail corpora. In *Proceedings of the 3rd Conference on Email and Anti-Spam*, pages 84–89, Mountain View, CA, 2006.

[12] W. Wang and Z.-H. Zhou. On multi-view active learning and the combination with semi-supervised learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1152–1159, Helsinki, Finland, 2008.

[13] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[14] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16.* MIT Press, Cambridge, MA, 2004.

[15] Z.-H. Zhou, K.-J. Chen, and H.-B. Dai. Enhancing relevance feedback in image retrieval using unlabeled data. *ACM Transactions on Information Systems*, 24(2):219–244, 2006.

[16] Z.-H. Zhou, K.-J. Chen, and Y. Jiang. Exploiting unlabeled data in content-based image retrieval. In *Proceedings of the 15th European Conference on Machine Learning*, pages 525–536, Pisa, Italy, 2004.

[17] Z.-H. Zhou and M. Li. Semi-supervised learning by disagreement. *Knowledge and Information Systems*, in press.

[18] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin at Madison, Madison, WI, 2006.

[19] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912–919, Washington, DC, 2003.