Preface

The 2006 ECML-PKDD Discovery Challenge deals with personalized spam filtering and generalization across related learning tasks. The problem setting is motivated by server-sided spam filtering in the case when the users are not prepared to tag their received messages as spam/non-spam and only public sources can be used for training a spam filter. The challenge is organized in cooperation with Strato Rechenzentrum AG.

For the challenge one set of labeled training data collected from publicly available sources and unlabeled inboxes of several users were provided. The inboxes served as evaluation data. The goal was to construct a spam filter for each single user that correctly classifies the emails in its inbox as spam or non-spam. There were two different tasks that differ in the number of inboxes and the proportion of labeled to unlabeled data.

57 teams participated in the challenge and 26 submitted results for the evaluation. The teams approached the problem in very different ways but most of the participants used variants of semi-supervised learning techniques. The following are the winners.

Spam Filtering Performance Award - Task A, three teams share the first rank:

- Khurram Junejo, Mirza Yousaf, Asim Karim
  Lahore University of Management Sciences, Pakistan,
- Bernhard Pfahringer
  University of Waikato, New Zealand,
- Kushagra Gupta, Vikrant Chaudhary, Nikhil Marwah, Chirag Taneja
  Inductis India Pvt Ltd.

Spam Filtering Performance Award - Task B:

- Gordon Cormack
  University of Waterloo, Canada.

Spam Filtering Creativity Award - Task A/B:

- Bernhard Pfahringer
  University of Waikato, New Zealand.

The Discovery Challenge workshop at the ECML-PKDD 2006 conference in Berlin serves as platform to discuss the results, different approaches, and other issues related to the problem setting.

We wish to express our gratitude to

- the participants of the challenge,
- the authors of the submitted papers,

– the invited speaker Alexander Zien (Max Planck Institute for Biological Cybernetics),
– Rolf Schimpfky for his help with preparing the datasets,
– the members of the knowledge management group at Humboldt Universität zu Berlin for their help in developing the idea for the challenge and their expertise for selecting the winner of the creativity award,
– and Strato Rechenzentrum AG for technical and financial support.

August 2006

Steffen Bickel
Discovery Challenge 2006 Chair

Table of Contents

ECML-PKDD Discovery Challenge 2006 Overview

Steffen Bickel

Humboldt-Universität zu Berlin, School of Computer Science
Unter den Linden 6, 10099 Berlin, Germany
bickel@informatik.hu-berlin.de

Abstract. The Discovery Challenge 2006 deals with personalized spam filtering and generalization across related learning tasks. In this overview of the challenge we motivate and describe the problem setting and the evaluation measure. We give details on the construction of the data sets and discuss the results.

1 Introduction

The Discovery Challenge 2006 is about personalized spam filtering and generalization across related learning tasks. People spend an increasing amount of time for reading messages and deciding whether they are spam or non-spam. Some users spend additional time to label their received spam messages for training local spam filters running on their desktop machines. Email service providers want to relieve users from this burden by installing server-based spam filters. Training such filters cannot rely on labeled messages from the individual users, but on publicly available sources, such as newsgroup messages or emails received through "spam traps" (spam traps are email addresses published visually invisible for humans but get collected by the web crawlers of spammers).

This combined source of training data is different from the distributions of the emails received by individual users. When learning spam filters for individual users from this type of data one needs to cope with a discrepancy between the distributions governing training and test data and one needs a balance between generalization and adaptation. The generalization/adaptation can rely on large amounts of unlabeled emails in the user's inboxes that are accessible for server-based spam filters. Utilizing this unlabeled data a spam filter can be adapted to the properties of specific user's inboxes but when little unlabeled data for a user are available a generalization over multiple users is advised.

The Discovery Challenge 2006 covers this setting, labeled training data collected from publicly available sources are provided. The unlabeled inboxes of several users serve as test data. The inboxes differ in the distribution of emails. The goal is to construct a spam filter for each single user that correctly classifies its emails as spam or non-spam. A clever way of utilizing the available sets of unlabeled emails from different users is required.

This overview is organized as follows. In Section 2, we discuss the problem setting and define the evaluation measure. We describe the data sets in Section

3. Section 4 gives an overview of the participants and summarizes the results. In Section 5 we discuss the different approaches and Section 6 concludes.

2 Problem Setting and Evaluation Measure

In the problem setting of the challenge the inboxes of several users are given and the goal is to correctly classify the messages in each inbox as spam or non-spam. No labeled training examples from the inboxes are available, instead, one common set of labeled data is given. The labeled data and the inboxes are governed by different distributions. A learning algorithm cannot rely only on the labeled data because the bias between training data and inboxes hinders learning of a correct classification model for the inboxes. The unlabeled data in the inboxes need to be used to adapt to their distributions.

The individual distributions of the inboxes are neither independent (identical spam messages are sent to many users), nor are they likely to be identical: distributions of inbound messages vary greatly between (professional, recreational, American, Chinese, . . .) email users. A learning algorithm can exploit the similarity of the inboxes.

There are two different tasks that differ in the number of inboxes and the proportion of labeled to unlabeled data (see Section 3).

Usually, cross-validation is used for tuning parameters of a classification model. In our case, cross-validation cannot be used because the emails in the inboxes are unlabeled. We provide a second set of labeled training data and inboxes for parameter tuning. The difference between the tuning set and the evaluation set is that the emails in the inboxes of the tuning set are labeled. The feature representation of the tuning data differs from the evaluation data (different dictionary). This means, the tuning data can not be used to augment the training data.

The problem setting differs from the standard setting of semi-supervised learning in three ways,

- there is a bias between training and evaluation data, the training and test data are governed by different distributions,
- several distinct but similar unlabeled inboxes are given, a multi-task learning or a transfer learning approach can be used for modeling and exploiting the similarity between inboxes,
- the number of labeled emails is larger than the number of unlabeled examples for a single inbox (task A).

The evaluation criterion for the challenge is the AUC value. The AUC value is the area under the ROC curve (Receiver Operating Characteristic curve). A ROC curve is a plot of true positive rate vs. false positive rate as the prediction threshold sweeps through all the possible values. The area under this curve has the nice property that it specifies the probability that, when we draw one positive and one negative example at random, the decision function assigns a higher value to the positive than to the negative example.

We compute AUC values for each inbox separately and average over all inboxes of the task. The winner for each task is the participant with the highest average AUC value. There is an additional creativity award for each task for the most interesting solutions in terms of non-straightforward approaches, innovative ideas, and assumed high impact.

3 Data Sets

The composition of the labeled training set is the same for both tasks, they differ in number of emails. 50% of the labeled training data contain spam emails sent by blacklisted servers of the Spamhaus project (www.spamhaus.org). 40% are non-spam emails from the SpamAssassin corpus and 10% are non-spam emails sent from about 100 different subscribed English and German newsletters. Table 1 summarizes the composition of the labeled training data for both tasks. The labeled data of the tuning set has the same size and composition as the actual training data but with different emails.

	task A	task B
emails sent from blacklisted servers	2000	50
SpamAssassin emails	1600	40
newsletters	400	10
total	4000	100

Table 1. Composition of labeled training data.

Evaluating the filters with respect to the personal distributions of messages requires labeled emails from distinct users. We construct different inboxes using real but disclosed messages. As non-spam part of the inboxes we use messages received by distinct Enron employees from the Enron corpus [9] cleaned from spam. Each inbox is augmented with spam messages from distinct spam sources. Some spam sources are used for multiple inboxes, in those cases all available emails from this source were sorted by date and split into different consecutive subsets. Because of the topic drift the distribution of the emails in the different parts differs.

The two tasks differ in the number and size of inboxes, task A has 3 and task B 15 evaluation inboxes. The size of the inboxes in task A is 2500 and in task B 400. Tables 2 and 3 summarize the composition of the evaluation and the tuning inboxes for task A and B. Each inbox consists of 50% spam and 50% non-spam emails.

The messages are preprocessed and transformed into a bag-of-words representation. We provide feature vectors with term frequencies. Our preprocessing uses charset-, MIME-, base64-, URL- (RFC 1738), and subject line-decoding (RFC 2047). Our tokenization takes care of HTML tags, following the X-tokenizer proposed by Siefkes et al. [3].

inbox ID	evaluation/ tuning	non-spam/ Enron user	spam source
0	eval	Farmer	Dornbos spam trap, part 1 (www.dornbos.com)
1	eval	Lokay	Dornbos spam trap, part 2 (www.dornbos.com)
2	eval	Sanders	spam trap of Bruce Guenter, part 1 (www.em.ca/∼bruceg/spam)
3	eval	Bass	personal spam of Richard Jones, part 1 (www.annexia.org/spam)
4	eval	Campbell	personal spam of Tobias Scheffer, part 1
5	eval	Dasovich	spam collection of SpamArchive.org, part 1
6	eval	Germany	spam collection of SpamArchive.org, part 2
7	eval	Kean	personal spam of Paul Wouters, part 1 (www.xtdnet.nl/paul/spam)
8	eval	Mann	Dornbos spam trap, part 3 (www.dornbos.com)
9	eval	Nemec	Dornbos spam trap, part 4 (www.dornbos.com)
10	eval	Rogers	spam trap of Bruce Guenter, part 2 (www.em.ca/∼bruceg/spam)
11	eval	Scott	spam trap of Bruce Guenter, part 3 (www.em.ca/∼bruceg/spam)
12	eval	Shackleton	personal spam of Richard Jones, part 2 (www.annexia.org/spam)
13	eval	Shapiro	personal spam of Tobias Scheffer, part 2
14	eval	Symes	spam collection of SpamArchive.org, part 3
0	tune	Lay	personal spam of Paul Wouters, part 2 (www.xtdnet.nl/paul/spam)
1	tune	Taylor	spam trap of Bruce Guenter, part 4 (www.em.ca/∼bruceg/spam)

Table 2. Composition of the evaluation and tuning inboxes for task A.

4 Participation and Results

57 teams from 19 different countries participated in the challenge. 26 participants submitted their results for evaluation, 20 teams have an academic and 6 teams a commercial background. Not all teams submitted results for both tasks. We averaged the AUC values for all inboxes as described above and determined the ranking. We conducted significance tests using a significance level of 5% to test the null hypothesis that the second rank has a higher AUC value than the first. The test statistic is computed as described in Hanley and McNeil [7]. For task A

inbox ID	evaluation/ tuning	non-spam/ Enron user	spam source
0	eval	Beck	spam trap of Bruce Guenter (www.em.ca/∼bruceg/spam)
1	eval	Kaminski	spam collection of SpamArchive.org
2	eval	Kitchen	personal spam of Tobias Scheffer (www.em.ca/∼bruceg/spam)
0	tune	Williams	Dornbos spam trap, part 3 (www.dornbos.com)

Table 3. Composition of the evaluation and tuning inboxes for task B.

we could not reject the null hypothesis for rank two and three, this means there is no statistically significant difference between them and they are all ranked first. For task B we could reject the null hypothesis for the second rank, this means there is one winner.

Table 4 and 5 show the first five ranks for task A and task B, respectively. Figure 1 displays the distribution of AUC over all ranks. Some participants report higher results in their workshop paper because they improved their algorithms after the submission deadline.

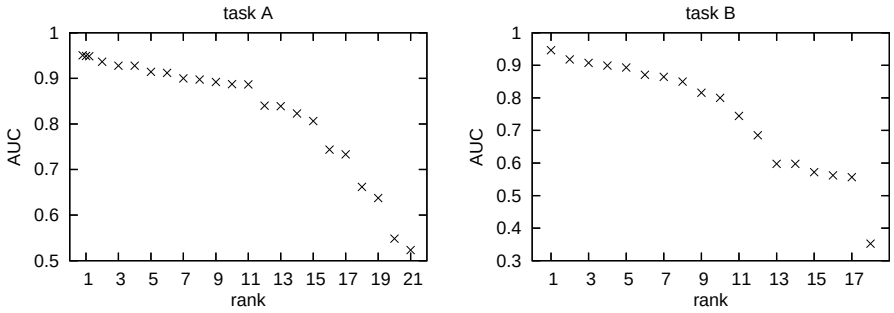


Fig. 1. Distribution of AUC performance dependent on rank over all participants for task A (left) and task B (right).

We selected the solution of Bernhard Pfahringer (University of Waikato, New Zealand) for the Spam Filtering Creativity Award - task A/B, we decided to award one team for both tasks instead of one for each task because most teams used the same algorithm for both tasks. Details on his algorithm are given in the next section.

5 Discussion

The teams approached the problem in very different ways but most of the participants used variants of semi-supervised learning techniques. Among the semi-supervised algorithms were graph-based algorithms [2] , large-margin-based

rank	avg. AUC	team
1	0.9507	Khurram Junejo, Mirza Yousaf, Asim Karim *Lahore University of Management Sciences, Pakistan*
1	0.9491	Bernhard Pfahringer *University of Waikato, New Zealand*
1	0.9487	Kushagra Gupta, Vikrant Chaudhary, Nikhil Marwah, Chirag Taneja *Inductis India Pvt Ltd*
2	0.9365	Nikolaos Trogkanis *National Technical University of Athens, Greece* Georgios Paliouras *National Center of Scientific Research "Demokritos" Greece*
3	0.9278	Chao Xu, Yiming Zhou *School of Computer Science and Engineering,* *Beijing University, China*
4	0.9277	Lalit Wangikar, Mansi Khanna, Ankush Talwar Nikhil Marwah, Chirag Taneja *Inductis India Pvt Ltd*
5	0.9144	Dimitrios Mavroeidis, Konstantinos Chaidos, Stefanos Pirillos, Dimosthenis Christopoulos, Michalis Vazirgiannis *DB-NET Lab, Informatics Dept., Athens University EB, Greece*

Table 4. First five ranks for task A.

rank	avg. AUC	team
1	0.9465	Gordon Cormack *University of Waterloo, Canada*
2	0.9183	Nikolaos Trogkanis *National Technical University of Athens, Greece* Georgios Paliouras *National Center of Scientific Research "Demokritos" Greece*
3	0.9074	Kushagra Gupta, Vikrant Chaudhary, Nikhil Marwah, Chirag Taneja *Inductis India Pvt Ltd*
4	0.8992	Dyakonov Alexander *Moscow State University, Russia*
5	0.8933	Wenyuan Dai *Apex Data & Knowledge Management Lab,* *Shanghai Jiao Tong University*

Table 5. First five ranks for task B.

methods [1, 4, 10], self-training approaches [6, 8], positive-only learning [10], and multi-view learning methods [4]. The assumption in most of those algorithms is that the unlabeled data is drawn from the same distribution as the labeled data. This assumption is violated in our case, but nevertheless semi-supervised learning reduces the error compared to methods that do not utilize the unlabeled data.

Bernhard Pfahringer the winner of the creativity award accounts for the bias between training and evaluation data in two ways [2]. Firstly, whenever a pre-

diction for some evaluation email is needed, his algorithm transforms the whole training set by only selecting those features which are actually present in the evaluation email (i.e. have a non-zero value). A classification model is trained using this transformed training set and that model's prediction is used for the evaluation example in question. This procedure forces the learner to concentrate on the features that are actually present in the evaluation example. This idea of filtering non-existent features is similar to the approach of Steinberg and Golovnya [11]. Secondly, Pfahringer uses a learning algorithm by Zhou et al. [5] that is one of the best known graph-based semi-supervised learning algorithms. The algorithm of Zhou et al. originally suffers from a cubic runtime complexity in the number of examples. Pfahringer develops a variant of this algorithm with linear complexity. The tremendous reduction in runtime and memory requirements make the algorithm applicable for large data sets.

Trogkanis and Paliouras [10], ranked second in both tasks, are very cautious when transferring knowledge from labeled to biased unlabeled data. Their approach is almost unsupervised. A classifier trained an the labeled data is allowed to label only a very few unlabeled emails with high confidence. In the subsequent step the labeled data is ignored and a semi-supervised algorithm is applied only to the inbox emails.

Two teams developed models that account for the similarity of inboxes with transfer learning. Participant Mohammad Al-Hasan (Rensselaer Polytechnic Institute) first measures the pairwise cosine similarity of all emails between all inboxes. In a second step a self-training-like learning algorithm learns separate classifiers for all inboxes in parallel. In each self-training iteration the most confident previously unlabeled email for each inbox is labeled together with the most similar email from one other inbox. With this approach confident decisions from one inbox are transfered to other inboxes. Trogkanis and Paliouras [10] use semi-supervised learning and augment the unlabeled data of one inbox by a weighted set of the unlabeled emails of all other inboxes. Gordon Cormack, ranked first in task B, even ignores the separation of emails into inboxes and pools all inboxes into one unlabeled set for semi-supervised training.

6 Conclusion

Most of the participants obtained lower classification errors by utilizing the data from the unlabeled inboxes in addition to the labeled data. Those results indicate that server-sided spam-filtering can be improved by personalization using unlabeled inboxes.

The results of the participants show that a wide range of semi-supervised learning algorithms can improve the classification performance for the problem setting of the challenge. Most semi-supervised learning algorithms make the implicit assumption that the training and test data are drawn from the same distribution. This assumption is violated in our case. It is an open problem to develop semi-supervised learning methods that account for a bias between

training and test data. We assume that such methods could further improve the benefit of spam filter personalization.

Some participants used transfer learning to account for the similarity between the inboxes. In their experiments the knowledge transfer between the inboxes improved the classification performance. Algorithms that did not exploit the similarity between the inboxes but used more sophisticated semi-supervised methods received higher scores in the overall ranking. This raises the question whether the best semi-supervised approaches can be integrated into a transfer or multi-task learning framework and whether this further improves the classification performance.

To our knowledge there is no spam filtering software for practical settings that utilizes unlabeled examples within a learning framework, also, personalization in server-sided spam filtering algorithms is widely disregarded. In this respect the results of the Discovery Challenge are encouraging. A real application in a server-sided setting poses additional challenges regarding the scalability of the methods.

Acknowledgment

The Discovery Challenge 2006 has been supported by Strato Rechenzentrum AG and by the German Science Foundation DFG under grant SCHE540/10-2.

References

1. Kyriakopoulou A. and Kalamboukis T. Text classification using clustering. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2006.
2. Pfahringer B. A semi-supervised spam mail detector. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2006.
3. Siefkes C., Assis F., Chhabra S., and Yerazunis W. Combining winnow and orthogonal sparse bigrams for incremental spam filtering. In *Proceedings of the European Conference on Principle and Practice of Knowledge Discovery in Databases*, 2004.
4. Mavroeidis D., Chaidos K., Pirillos S., Christopoulos D., and Vazirgiannis M. Using tri-training and support vector machines for addressing the ecml-pkdd 2006 discovery challenge. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2006.
5. Zhou D., O. Bousquet, Lal T., Weston J., and Schölkopf B. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, 2003.
6. Cormack G. Harnessing unlabeled examples through iterative application of dynamic markov modeling. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2006.
7. Hanley J. and McNeil B. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148:839–843, 1983.
8. Junejo K., Yousaf M., and Karim A. A two-pass statistical approach for automatic personalized spam filtering. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2006.

9. B. Klimt and Y. Yang. The Enron corpus: A new dataset for email classification research. In *Proceedings of the European Conference on Machine Learning*, 2004.
10. Trogkanis N. and Paliouras G. TPN^2: Using positive-only learning to deal with the heterogeneity of labeled and unlabeled data. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2006.
11. Dan Steinberg and Mikhaylo Golovnya. Identifying spam with predictive models. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2006.

Harnessing Unlabeled Examples through Iterative Application of Dynamic Markov Modeling

Gordon V. Cormack

University of Waterloo, Waterloo, ON, N2L 3G1, Canada
gvcormac@uwaterloo.ca
cormack.uwaterloo.ca/cormack

Abstract. We describe the application of dynamic Markov modeling – a sequential bit-wise prediction technique – to labeling email corpora for the 2006 ECML/PKDD Discovery Challenge. Our technique involves: (1) converting the corpora's bag-of-words representation to a sequence of bits; (2) using logistic regression on the training data to induce an initial maximum likelihood classifier; (2) combining all test sets into one; (3) ordering the combined set by decreasing magnitude of the log-likelihood ratio; (4) iteratively applying dynamic Markov modeling (DMC) to compute successive log-likelihood estimates; (5) averaging successive estimates to form an overall estimate; (6) partitioning the combined estimates into separate results for each test set. Post-hoc experiments showed that: (a) the iterative process improved on the initial classifier in almost all cases; (b) treating each test set separately yielded nearly indistinguishable results.

1 Dynamic Markov Modeling

Recently we have shown that sequential adaptive data compression methods work well for classifying email messages into spam and non-spam [1, 2]. Our approach uses the DMC data compression model [3] to estimate the conditional likelihood of each successive bit in a message, under two separate prior assumptions: that the message is spam and that the message is non-spam. The log of the ratio of these two likelihoods is computed and the message is classified as spam if the average of these values is positive; otherwise the message is classified as non-spam. The average log-likelihood-ratio itself is used as the *decision function value* required by the 2006 ECML/PKDD Discovery Challenge[4].

The DMC model is a finite state machine with a binary label and a frequency count on each edge. As the message is processed – one bit at a time from left-to-right – the prior probability distribution for each bit is estimated using the ratio of the frequencies of the edges leaving the current state. Then the model is updated to take into account the bit's observed value. The update has three phases: first, the frequency of the edge labeled with the observed value is incremented; second, the target of this edge may be *cloned*; third, the edge is

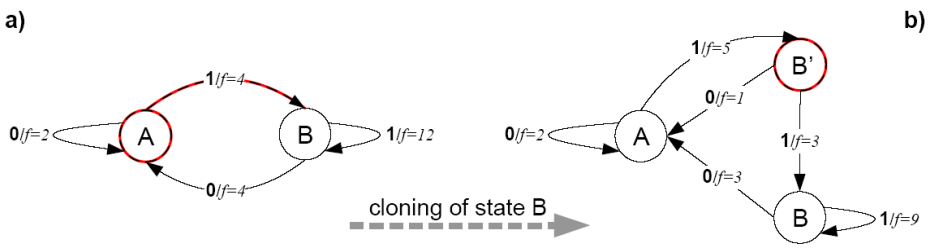


Fig. 1. DMC Cloning

followed to yield a new current state. Consider figure 1. The current state is A and the next bit has the value 1. The prior probability for this value is estimated to be $\frac{2}{3}$. Because B has been visited previously by this edge and also by some other edge,[1] it is cloned, resulting a new state B' to which the current edge is redirected. The outgoing edge frequencies of B and B' are divided in proportion to the number of times each has been visited (3:1 in this example). Finally the edge is followed resulting in a new current state of B'.

The precise criteria for cloning are as follows. Let f_1 be the frequency count (prior to incrementing) of the edge used to reach B. Let f_2 be the sum of the frequency counts of the edges leaving B. Cloning takes place if $f_1 > t_1$ and $f_2 - f_1 > t_2$ for two arbitrary threshold parameters t_1 and t_2. f_1 is the number of times that B has been visited within the current context; f_2 is the number of times it has been visited from other contexts. We used $t_1 = t_2 = 2$, default values which were known to work well in other applications.

To classify messages, we construct two DMC models – one for spam and one for non-spam. For each message in turn, we twice apply the adaptive process detailed above – once using the spam model and once using the non-spam model – to compute the likelihood of the message for each class. After each message is classified, updates to the model corresponding to the *incorrect* class (as determined by the label) are discarded, whereas updates to the model corresponding to the *correct* class are preserved.

The DMC model continuously grows, using more context for predicting common bit sequences. It is thus sensitive to bit, character, and word frequencies, as well as intra- and inter-word patterns as well as punctuation and formatting. Also, since the model is dynamic, it is sensitive to inter-message patterns and the order in which training examples are presented to it. These sensitivities appear to be advantageous in classifying real sequences of email messages.

It is not obvious that the DMC model should be applicable to the vector-space representation of messages in the ECML/PKDD Challenge. This representation discards lexical and formatting information, the order of words, and the order of messages in the test and training sets. Only the frequency of word occurrences is explicitly preserved. We used this information to create a sequential rendering of

[1] We know the total number of visits to B by summing the frequencies of its two outgoing edges.

each message in the following manner: we represented each feature in the vector space as a 16-bit[2] integer. A feature occurring k times in a message was rendered as k adjacent occurrences of the corresponding 16-bit integer. The renderings for all features were concatenated to render the message. This representation aptly captures word frequencies, but not intra- or inter-word patterns[3].

Experiments with the tuning data showed that the DMC model classified the training data very well, and classified the test data fairly well, but with high sensitivity to factors such as the order of the training data. Based on these experiments, we chose a hybrid approach, using logistic regression to build an initial classifier from the labeled training data, followed by several iterations of DMC to harness the unlabeled test data.

2 Iterative Classification

We applied Goodman's adaptive logistic regression implementation [5] to the training and test messages to yield initial log-likelihood-ratio estimates for each of the test messages. Based on these estimates we computed tentative labels for the test messages: those with a positive log-likelihood ratio were labeled as spam; the rest as non-spam.

A sequence of labeled messages was created by concatenating

- the training messages and labels, in the order given
- the test messages, labeled as described above, in decreasing order by the magnitude of the log-likelihood-ratio estimate (i.e. by diminishing confidence in the label).

Our DMC classifier was applied on-line to this sequence, classifying each message before training on its label, to yield a new sequence of log-likelihood-ratio estimates. A new sequence of labeled messages was created from these estimates as described above, and the process repeated five times.

Finally, the six log-likelihood-ratio estimates for each message (initial plus five iterations of DMC) were averaged to yield the final classifier output.

In summary, we use the output from one stage of classification to synthetically label training examples for the next. Since the training examples are ordered by confidence, we have reason to believe that early examples are likely to be correctly labeled, so that the adaptive DMC model will grow to include patterns from correctly classified messages that may not be present in the initial training

[2] There are fewer than 2^{16} distinct features in the ECML/PKDD Challenge data, but they are numbered discontiguously resulting in values greater than 2^{16}. We renumbered the features – preserving the original order but eliminating unused values – to achieve a 16-bit representation.

[3] We note that the feature numbers were assigned by ECML in order of first occurrence. We expect if two common features frequently occur adjacent, there is a good chance that their first occurrences are adjacent. Therefore, some small reflection of the inter-word patterns may be preserved in our rendering.

data. Similarly, by combining together the separate tests, we have reason to believe that the model may grow to incorporate other features representative of spam or non-spam. Finally, prior experience gives us reason to believe that combining classifiers – even combining weak and strong classifiers – by summing log-odds-ratios would provide accurate and reliable results [6].

3 Results

Test Set	Initial LR	Combined mailboxes	Separate mailboxes
task_a_u00	15.1	19.6	19.7
task_a_u01	12.4	11.7	11.2
task_a_u02	7.1	2.6	1.3
all task_a	12.0	11.4	10.9
task_b_u00	23.5	1.7	3.3
task_b_u01	21.3	2.5	2.9
task_b_u02	7.1	3.2	2.1
task_b_u03	3.8	1.2	1.0
task_b_u04	11.3	3.2	2.9
task_b_u05	12.0	10.1	11.9
task_b_u06	20.2	11.7	4.3
task_b_u07	9.5	3.6	4.3
task_b_u08	12.4	2.4	1.5
task_b_u09	16.6	4.0	4.7
task_b_u10	11.5	6.2	7.1
task_b_u11	9.5	5.3	5.1
task_b_u12	17.4	11.1	12.8
task_b_u13	12.7	7.4	8.0
task_b_u14	18.9	6.7	7.3
all task_b	13.7	5.2	5.1

Table 1. Per-mailbox error rates [1-AUC (%)]

Test Set	Initial LR	DMC1	DMC2	DMC3	DMC4	DMC5	Final average
all task_a	12.0	17.9	13.6	11.0	9.7	9.2	10.9
all task_b	13.7	10.0	6.1	5.8	5.5	5.6	5.1

Table 2. Iterative results – separate mailboxes [1-AUC (%)]

Table 1 shows the area *above* the ROC curve (as a percentage) achieved by the initial and final classifier for each mailbox (test set) and for all mailboxes combined in Task A and Task B. The first column is the score achieved by the LR classifier using only the labeled training data. The second column is the end

Test Set	Initial SVM	DMC1	DMC2	DMC3	DMC4	DMC5	Final average
all task_a	14.2	11.4	8.5	7.1	6.5	6.2	7.0
all task_b	49.3	42.4	37.5	38.0	38.2	38.3	24.8

Table 3. SVM initial classifier – separate mailboxes [1-AUC (%)]

result of the iterative process applied to the combined mailboxes. This column corresponds to our official results. The third column shows the result of iterating on each mailbox separately.

We see that in all but one mailbox the iterative process reduced the error rate, in most cases substantially. Unfortunately, the one case was the first mailbox of Task A, with the net result being that the iterative process made an insubstantial improvement overall for this task. For Task B, the iterative process worked remarkably well. For all mailboxes the result was improved; for most, substantially so. The overall effect was that the result on Task B was significantly better than any other.

Table 2 illustrates the convergence of the iterative process. In general, classifier error diminishes with the number of iterations but appears to be near asymptote after the fifth iteration. The final combined score is in general a small further improvement.

We investigated several alternatives in an effort to determine why our method worked well for Task B and not for Task A. We first repeated the iterative process separately for each of the eighteen mailboxes in both tasks. The results – seen in column 3 of table 1 – are nearly indistinguishable from and certainly not significantly different from those derived from combined mailboxes. We know from data compression experiments that DMC is particularly good at modeling heterogeneous data, because it simply "grows" distinct states to handle each category. Therefore we should perhaps not have been as surprised as we were to observe that combining mailboxes has no substantive effect.

We investigated the use of a different initial classifier, namely SVM^{light} with binary features and default parameters. As shown in figure 3 the results were mixed. For Task A, the initial SVM classification was inferior to that yielded by logistic regression, but our iterative process was much better able to improve on it, yielding a dramatic overall improvement. For Task B, the initial SVM classification was very poor, presumably due to over-fitting. The iterative technique was able to improve on it considerably, but the overall result is still not good.

We further investigated the effect of the size of the training and test sets on the result. Using a small sample of the Task A training data had an insubstantial effect – the results were very slightly worse. Similarly, results on a sample of the Task A mailbox were insubstantially different from those on the whole.

4 Conclusion

Our technique for iterative labeling using Dynamic Markov Modeling makes effective use of unlabeled training data to improve on an initial discriminative

classifier derived from labeled data. The technique provided an improvement for seventeen of eighteen mailboxes consisting of unlabeled messages, whether applied to the mailboxes separately or as a common set. For the majority of mailboxes, the improvement was substantial, in some cases reducing the area above the ROC curve by a factor of ten.

We are unable to find a reason for discrepancy between Task A and Task B results. In our pilot experiments the method performed as well on Task A as on Task B. We tried adjusting the training and test set sizes, and the number of combined test sets. We can only conclude that the data used in at least one of the Task A mailboxes is materially different from the rest; perhaps examining the unobfuscated data would yield insight as to the nature of the difference.

References

1. Bratko, A., Cormack, G.V., Filipic, B., Lynam, T.R., Zupan, B.: Spam filtering using statistical data compression. **Journal of Machine Learning Research** (in press; see http://plg.uwaterloo.ca/~gvcormac/jmlr.pdf)
2. Cormack, G.V., Bratko, A.: Batch and on-line spam filter evaluation. In: CEAS 2006 – Third Conference on Email and Anti-Spam, Mountain View (2006)
3. Cormack, G.V., Horspool, R.N.S.: Data compression using dynamic Markov modelling. The Computer Journal **30**(6) (1987) 541–550
4. Bickel, S.: ECML/PKDD 2006 Discovery Challenge http://www.ecmlpkdd2006.org/challenge.html (2006)
5. Goodman, J., Yih, W.T.: Online discriminative spam filter training. In: The Third Conference on Email and Anti-Spam, Mountain View, CA (2006)
6. Lynam, T.R., Cormack, G.V.: On-line spam filter fusion. In: 29th ACM SIGIR Conference on Research and Development on Information Retrieval, Seattle (2006)

A Two-Pass Statistical Approach for Automatic Personalized Spam Filtering

Khurum Nazir Junejo, Mirza Muhammad Yousaf, and Asim Karim

Dept. of Computer Science, Lahore University of Management Sciences
Lahore, Pakistan
{junejo, 05030019, akarim}@lums.edu.pk

Abstract. Typically, spam filters are built on the assumption that the characteristics of e-mails in the training dataset is identical to those in individual users' inboxes on which it will be applied. This assumption is oftentimes incorrect leading to poor performance of the filter. A personalized spam filter is built by taking into account the characteristics of e-mails in individual users' inboxes. We present an automatic approach for personalized spam filtering that does not require users' feedback. The proposed algorithm builds a statistical model of spam and non-spam words from the labeled training dataset and then updates it in two passes over the unlabeled individual user's inbox. The personalization of the model leads to improved filtering performance. We perform extensive experimentation and report results using several performance measures with a discussion on tuning the two parameters of the algorithm.

1 Introduction

E-mail is an indispensable communication method for most computer users and it plays an essential role in the functioning of most businesses. Globalization has resulted in an exponential increase in the volume of e-mails. Unfortunately, a large chunk of it is in the form of spam or unsolicited e-mails. Last year 40% of all e-mails were spam, which totals up to 12.4 billion messages per day and about 176 messages per user per day [1]. In 2002, spam cost non-corporate Internet users 255 million dollars and resulted in a loss of 8.9 billion dollars to U.S. corporations alone [1]. Spam messages not only waste users' time and money but are also harmful for their computer's security. Commtouch, a security service provider, reported 19 new e-mail borne viruses in the month of January 2006 [2]. E-mail users spend an increasing amount of time reading messages and deciding whether they are spam or non-spam and categorizing them into folders. Some e-mail clients require users to label their received messages for training local (or personalized server-based) spam filters. E-mail service providers would like to relieve users from this burden by installing server-based spam filters that can classify e-mails as spam automatically and accurately without user feedback.

Typically, server-based spam filters are trained on general training datasets and then applied to individual users' inboxes. However, the characteristics of individual

users' inboxes are usually not identical to that of the general e-mail corpus used for training the spam filter, resulting in poor filtering performance. Furthermore, the characteristics of spam e-mails evolve with time making non-adaptive filters less robust to change. Thus, there is a need for personalized spam filters that learn from general training datasets and adapt to the characteristics of individual users' inboxes. This adaptation must be done without asking the users to label their e-mails. Earlier works on personalized spam filtering utilize users input. This approach is clearly not convenient for the e-mail user.

In this paper, we present a statistical approach for classifying individual users' e-mails in accordance with the users' e-mails characteristics without requiring their input. The algorithm learns from a general corpus of labeled e-mails in a single pass over them and then updates this learned model in two passes over the individual users' unlabeled e-mails. This approach allows automatic specialization of the general model to the underlying distribution of e-mails in individual users' inboxes. The statistical model is built from the tokens in the e-mail body and their frequencies. Our initial implementation of the algorithm won the performance award at the Discovery Challenge held in conjunction with ECML-PKDD [3]. This paper presents the detailed results of our implementations using several performance metrics and with varying parameters of the algorithm.

The rest of the paper is organized as follows. In section 2, we provide a brief review of content-based spam filters with specific focus on personalized spam filtering. Section 3 describes our algorithm for automatic personalized spam filtering. Section 4 describes the experiments and their results, and a discussion of the results with specific focus on parameter tuning is given in section 5. We conclude in section 6.

2 Personalized Content-based Spam Filtering

Many approaches are used in practice to control the menace of spam including global and local blacklists, global and local whitelists, IP blocking, legislation, and content-based filtering. Content-based filters employ machine learning techniques to learn to predict spam e-mails given a corpus of training e-mails. Such filters are typically deployed on the mail server that filters e-mails for all users of the server. Researchers have developed content-based spam filters using Bayesian approaches [4-7], support vector machines (SVM) [8, 9], nearest neighbor classifiers [10], rule-based classifiers [11, 12], and case-based reasoning [13]. Among these techniques, Bayesian approaches and SVMs have shown consistently good performances. Sahami et al. present one of the earliest naïve Bayes classifier for the spam classification problem [4]. Since then, numerous variations of the naïve Bayes classifier have been presented for spam filtering [5-7]. The popular Mozilla's e-mail client implements a naïve Bayes classifier for spam filtering [6]. Support vector machine (SVM) is a powerful supervised learning paradigm based on the structured risk minimization principle from computational learning theory. SVMs exhibit good generalization capabilities and have shown good spam classification performance. One of the first SVM for the

spam classification problem is presented in [8]. Since then, several extensions and variations have been presented such as [9].

The majority of the supervised machine learning techniques presented for spam filtering assume that e-mails are drawn independently from a given distribution. That is, the statistical distribution of e-mails in the training dataset is identical to that of the individual user's e-mails on which the trained filter will be applied. This assumption, however, is usually incorrect in practice; the training dataset is typically derived from multiple Internet sources reflecting different distributions of spam and non-spam e-mails that are different from that of the individual user's e-mails. A personalized spam filter is capable of adapting to the distribution of e-mails of each individual user. Previous works on personalized spam filtering have relied upon user feedback in the form of e-mail labels from each individual user [14, 15]. This strategy burdens the e-mail user with the additional task of aiding the adaptation of the spam filter. Recently, the problem of automatic personalized spam filtering is investigated by Bickel and Scheffer [16]. They present a nonparametric hierarchical Bayesian model that generalizes across several users' e-mails by minimizing a loss function. Their experiments indicate performance improvements over classifiers developed by assuming independent and identically distributed data.

We present a simple approach for automatic personalized spam filtering that does not require users' feedback. The approach is based on a statistical model of spam and non-spam words in e-mails similar to that developed in Bayesian approaches. However, unlike many Bayesian approaches presented in the literature, we specialize the model to reflect the distributions of e-mails in individual users' inboxes.

3 Our Algorithm

Our personalized spam filtering algorithm consists of two phases of processing. In the first phase, called the training phase, the algorithm learns a statistical model of spam and non-spam words from the training dataset in a single pass over the training dataset. The second phase, called the specialization phase, consists of two passes over the user's inbox (evaluation dataset). In the first pass, the statistical model developed in the training phase is used to label the e-mails in the individual user's inbox, and to build an updated statistical model of the e-mails. In the second pass, the updated statistical model is used to score and classify the e-mails in the individual user's inbox. The pseudo-code of our algorithm is given in Figure 1.

The statistical model is developed as follows: For each distinct word in the labeled (i.e. training or after first pass of evaluation) dataset, count its occurrences in spam and non-spam e-mails. Then, find the difference of these two values for each word. Now choose the significant words by selecting only those words for which the absolute difference between their spam and non-spam occurrences is greater than some integer threshold t. This approach also categorizes the significant words as either a spam word or a non-spam word. Each spam and non-spam word is assigned a weight based on the ratio of its occurrences in the spam and non-spam e-mails. This statistical model of words can then be used to classify a given e-mail by computing its spam score and non-spam score values, where the spam score (non-spam score) of an

N = Total number of e-mails
N_S = Number of spam e-mails
N_N = Number of non-spam e-mails
D = number of words in dictionary (indexed from 1 to D)
C_{Si} = count of word i in all spam e-mails
C_{Ni} = count of word i in all non-spam e-mails
Z_S = set of significant spam words
Z_N = set of significant non-spam words
t = threshold (e.g. $C_{Si} - C_{Ni} > t$ for it to be included in significant spam words)
s = scale factor
W_{Si} = weight associated with significant spam word i
W_{Ni} = weight associated with significant non-spam word i
T_i = word i

Training Phase (Phase 1 on Training Dataset)
Build_Statistical_Model Procedure
-For each distinct word i in dataset find C_{Si} and C_{Ni}
-Find the significant spam words Z_S such that for each word T_i in Z_S, $C_{Si} - C_{Ni} > t$
-Find the significant non-spam words Z_N such that for each word T_i in Z_N, $C_{Ni} - C_{Si} > t$
-For each significant spam and non-spam word find their weight as follows:
-$W_{Si} = [C_{Si} / C_{Ni}] * [N_N / N_S]$, for all words in Z_S
-$W_{Ni} = [C_{Ni} / C_{Si}] * [N_N / N_S]$, for all in Z_N

Specialization Phase (Phase 2 on Evaluation Dataset)
First Pass
Score_Emails Procedure
-For each e-mail in the evaluation dataset
-spam_score = $\sum W_{Si}$ (sum is over all significant spam words in e-mail)
- nonspam_score = $\sum W_{Ni}$ (sum is over all significant non-spam words in e-mail)
-If (s * spam_score > nonspam_score) then classify as spam and output spam_score; other wise classify as non-spam and output -nonspam_score.

-Build statistical model concurrently with scoring e-mails (procedure is identical to Build_Statistical_Model given above)

Second Pass
-Score and classify e-mails using the updated statistical model (procedure is identical to the Score_Emails procedure given above)

Fig. 1. Our automatic personalized spam filtering algorithm

e-mail is the weighted sum of the words of that e-mail that belong to the significant spam (non-spam) words set. If the spam score multiplied by a scaling factor (*s*) is greater than the non-spam score then the e-mail is labeled as spam; otherwise, it is labeled as non-spam. This statistical model is developed in the training phase as well

as in the first pass of the specialization phase. In the second pass of the specialization phase, the final scores and classifications of e-mails are output.

The motivation for using significant words that have differences of their counts in spam and non-spam e-mails greater than a specified threshold is: (1) a word that occurs much more frequently in spam e-mails (or non-spam e-mails) will be a better feature in distinguishing spam and non-spam e-mails than a word that occurs frequently in the dataset but its occurrence within spam and non-spam e-mails is almost similar, and (2) this approach greatly reduces the number of words that are of interest, simplifying the model and its computation. The scale factor is used to cater for the fact that the number of non-spam words, and their weighted sum in a given e-mail, is usually greater than the number of spam words and their weighted sum.

The purpose of the weighting scheme for the significant words is to give an advantage to words for which either the spam count or the non-spam count is proportionally greater than the other. For example, if the word with ID '10' has spam and non-spam counts of 0 and 50, respectively, and the word with ID '11' has spam and non-spam counts of 950 and 1000, respectively, then even though their difference is the same (50) the word with ID '10' gives more information regarding the classification of the e-mail than word with ID '11'.

The specialization phase adapts the general statistical model to the characteristics of the individual user's inbox. The model developed from the training phase is used for the initial classification of the user's e-mails. Furthermore, the statistical model is updated to incorporate the characteristics of the user's inbox. This updated model is then used to finally score and classify the e-mails in the user's inbox.

4 Experiments

In this section, we present the results of our experimental evaluation of the automatic personalized spam filtering algorithm. The algorithm is implemented in Java. The code uses special built-in data structures of Java such as Hash Maps that provide an efficient way of retrieving word objects by avoiding the cost of searching through an array list of word IDs.

4.1 Datasets

We use the datasets available from the ECML-PKDD Discovery Challenge website [3]. The training dataset is a general corpus containing 4000 e-mails collected from several users' inboxes. The evaluation datasets consist of three different users' inboxes each containing 2500 e-mails. These evaluation datasets are identified as Eval-00, Eval-01, and Eval-02. In addition to these datasets, a training and a test (labeled evaluation) dataset containing 4000 and 2500 e-mails, respectively, is provided for parameter tuning. The ratio of spam and non-spam e-mails in all the datasets is 50-50. The distribution of e-mails in the training corpus which is a combined source of training data is different from the distributions of the e-mails received by individual users. An additional evaluation dataset is created from the

Table 1. Characteristics of datasets

	Distinct Words	**Total Words**
Training Corpus	41675	2761246
Eval-00	26580	842998
Eval-01	27523	843634
Eval-02	20227	494536
Eval-Combined	39962	2181168
Tune-Training	39967	2915199
Tune-Test	22991	1197283

above datasets. Dataset Eval-Combined is the collection of e-mails in datasets Eval-00, Eval-01, and Eval-02.

Each e-mail in the datasets is represented by a word (term) frequency vector. Each word in an e-mail is identified by an ID and its frequency count in the e-mail. An additional attribute identifies the label of the e-mail as either spam or non-spam.

4.2 Evaluation Criteria

We evaluate our algorithm using the following performance measures:

1. True positive rate (TP): fraction of spam e-mails correctly classified as spam
2. False positive rate (FP): fraction of non-spam e-mails incorrectly classified as spam
3. Accuracy: fraction of all e-mails that are correctly classified
4. Precision: TP / (TP + FP)
5. Recall: TP / (TP + FN), where false negative rate (FN) is the fraction of spam e-mails that are incorrectly classified as non-spam
6. AUC: area under the receiver operating characteristics (ROC) curve

The first five measures are calculated by taking zero as the discriminating threshold in the scores output by a given filter. The AUC is computed from the ROC generated by sweeping through all possible discriminating thresholds in the scores output by a given filter. The AUC is considered to be a better measure of the overall performance of a filter [17].

4.3 Results

We ran our algorithm by iterating over several values of the threshold (t) and scale factor (s) parameters to find the parameter combination that produces the best performance. The parameters that yield the highest AUC value on the tuning datasets are $t = 8$ and $s = 13$. With these parameters, the performance of our algorithm on the evaluation datasets (users' inboxes) is given in Table 2. The average AUC value for these 3 inboxes is 0.9875. This value is significantly better than 0.9539 obtained by the filter submitted for the Discovery Challenge. The submitted filter used the

Table 2. Performance results of our algorithm with parameters $t = 8$ and $s = 13$ tuned on the tuning datasets

	AUC	Precision (%)	Recall (%)	Accuracy (%)	FP (%)	TP (%)
Eval-00	0.9832	85.53	98.88	91.08	16.72	98.88
Eval-01	0.9896	91.87	98.56	94.92	7.76	98.08
Eval-02	0.9898	98.68	90.24	94.52	1.2	90.24
Average	0.9875	92.02	95.89	93.50	8.56	95.73

parameters $t = 400$ and $s = 8$. These parameters were selected after a few iterations over the tuning datasets. Furthermore, at that time we preferred a large value for t to keep the size of the significant words set small.

We have now experimented with hundreds of parameter combinations by tuning them on the individual evaluation datasets. The performance results of some selected parameter combinations are given in Table 3. The highest AUC value (the 'optimal' filter) for each evaluation dataset is highlighted. It is seen that whenever the AUC value is the highest the corresponding accuracy is also the highest. The filter with threshold value of 400 is the one that was submitted to the Discovery Challenge. In general, the AUC value increases slightly as the threshold increases from zero and then starts decreasing with further increase in the threshold. This is because as the threshold increases the number of significant words decreases to an 'optimal' feature set and then, with further increase in the threshold, the significant words become less discriminatory. For the three evaluation datasets we suggest a threshold of around 10 as this produces a reduction of the total words by more than half at the expense of about 0.3% loss in accuracy. The tradeoff between the accuracy and threshold is discussed in more detail in Section 5.

The key part of our algorithm is the specialization phase in which the model learned from the general corpus is updated in accordance to the characteristics of each individual user's inbox. The second pass of this phase is the step that significantly differentiates our algorithm from the conventional techniques that assume that e-mails are drawn independently from a given distribution. To highlight this personalization characteristic of our algorithm, we ran our algorithm with and without the second pass of the specialization phase. The results of this experiment are shown in Table 4 for the 'optimal' filters highlighted in Table 3. It is seen that the second pass produces a significant improvement in the AUC value because now the model is specialized for each user's inbox. Similarly, the second pass results in an increase of at least 10% in the accuracy on all evaluation datasets, which is a significant amount.

4.4 Results of Some Variations of the Algorithm

We also experiment with some variations of the algorithm given in Figure 1. In the first variation, instead of maintaining the frequencies of words in e-mails we use only their occurrences (i.e. count 1 if a word occurs in an e-mail and 0 otherwise).

Table 3. Performance results of our algorithm with various parameter combinations. Average AUC of the highlighted rows is 0.9902. Average AUC of the submitted filter (with $t = 400$) is 0.9539

	Threshold (*t*)	**Scale Factor (*s*)**	**AUC**	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**FP (%)**	**TP (%)**
Eval-00	0	9	0.9844	97.42	94	95.76	2.48	94
Eval-00	10	9.5	0.9845	96.80	94.64	95.76	3.12	94.64
Eval-00	**11**	**9.5**	**0.9848**	**96.81**	**94.88**	**95.88**	**3.12**	**94.88**
Eval-00	100	7.5	0.9816	95.55	91.2	93.44	4.24	91.12
Eval-00	400	8	0.955	86.94	93.2	89.6	14	93.2
Eval-01	0	11	0.993	94.96	96.56	95.72	5.12	96.56
Eval-01	**4**	**11.5**	**0.993**	**94.76**	**97.04**	**95.84**	**5.36**	**2.96**
Eval-01	10	11.5	0.9929	95.51	95.44	95.48	3.92	94.72
Eval-01	100	9	0.9821	95.60	90.48	93.16	4.16	90.48
Eval-01	400	8	0.9717	91.36	88.88	90.24	8.4	88.88
Eval-02	0	14.5	0.9924	97.02	96.56	96.8	2.96	96.56
Eval-02	**2**	**16**	**0.9928**	**96.29**	**97.68**	**96.96**	**3.76**	**97.68**
Eval-02	10	16.5	0.9918	96.19	97.12	96.64	3.84	97.12
Eval-02	100	14	0.9751	89.73	94.4	91.8	10.18	94.4
Eval-02	400	8	0.935	85.86	83.12	84.72	13.68	83.12

Surprisingly, this variation resulted in a significant increase in the AUC value for the datasets as shown in Table 5. It is seen from Table 5 that the 'optimal' average AUC value for this variation is 0.9932 as compared to 0.9902 for the original algorithm. In particular, the increase in the AUC value of dataset Eval-00 is quite substantial.

To compare the performance of a single classifier/filter for all users to that of personalized filters for each individual user, we experiment with a second variation of the algorithm. In this variation, we combine the 3 evaluation datasets into a single dataset Eval-Combined for which a single filter is built. This results in an increase in the AUC value. The average AUC value of the three evaluation datasets using the frequency counts comes out to be 0.9902 while the AUC value with the combined dataset comes out to be 0.9937. This improvement can be attributable to: (1) the larger size of the combined evaluation dataset, and (2) the fact that the general corpus is a collection of individual users' inboxes and thus resembles the dataset Eval-Combined. Table 5 also shows the performance of the filter based on word occurrences when applied to dataset Eval-Combined. This variation outperformed the rest with an AUC value of 0.9942. Thus, the best AUC value is obtained if we merge the evaluation datasets and use word occurrences rather frequencies for building the statistical model.

5 Parameter Tuning

Our algorithm uses two parameters: threshold (t) and scale factor (s). We ran hundreds of experiments with varying values for these parameters. The performance

Table 4. Performance results of the algorithm after the first and second pass over the evaluation dataset

	After First Pass				After Second Pass			
	AUC	Precision (%)	Recall (%)	Accuracy (%)	AUC	Precision (%)	Recall (%)	Accuracy (%)
Eval-00	0.9090	94.45	70.8	83.32	0.9848	96.81	94.88	95.88
Eval-01	0.9220	93.61	77.36	86.04	0.993	94.76	97.04	95.84
Eval-02	0.9346	89.38	82.88	86.52	0.9928	96.29	97.68	96.96
Average	0.9218	92.48	77.01	85.29	0.9902	95.95	96.53	96.22
Eval-Combined	0.9283	93.10	77.01	56.96	0.9937	96.79	96.66	96.73

Table 5. Performance comparison of algorithm using word frequecies and occurrences

	Based on Frequency Count	Based on Occurrence Count
Eval-00	0.9848	0.9920
Eval-01	0.9930	0.9941
Eval-02	0.9928	0.9936
Average	0.9902	0.9932
Eval-Combined	0.9937	0.9942

spread is seen in the ROC curve for dataset Eval-00 shown in Figure 2. The purpose of the threshold parameter is to find significant spam and non-spam words by filtering out the words that are not discriminatory enough. We experiment with several thresholds (t) to find the t that produces the highest accuracy and AUC value. For the three datasets, the best values for t are 11, 4, and 2, respectively. Figure 3 shows the variation of the threshold t with the number of significant words and the accuracy of the filter. It is observed that for small values of t there is little change in the accuracy of the filter, but the number of significant words in the model is greatly reduced. For example, by increasing t from 10 to 20 does not increase the accuracy significantly, but the number of significant words in the model is reduced by more than a factor of 2. This trade off between the number of significant words and the accuracy suggests a value for t that lies between 10 and 20.

The scale factor (s) is used to counter the effect of greater weight of non-spam words as compared to spam words in e-mails. For example, for dataset Eval-01, the average weight of a spam word is 4.0726, the average weight of a non-spam word is 13.1045, the average number of spam words in spam e-mails is 5.7192, and the average number of non-spam words in spam e-mails is 2.3296. Thus, for this dataset the average spam score of spam e-mails is 4.0726*5.7192 = 23.32 and the average non-spam score of spam e-mails is 13.1045*2.32 = 30.40, resulting in a large number of miss classifications. The scale factor is used to counter this by increasing the spam score by a factor s.

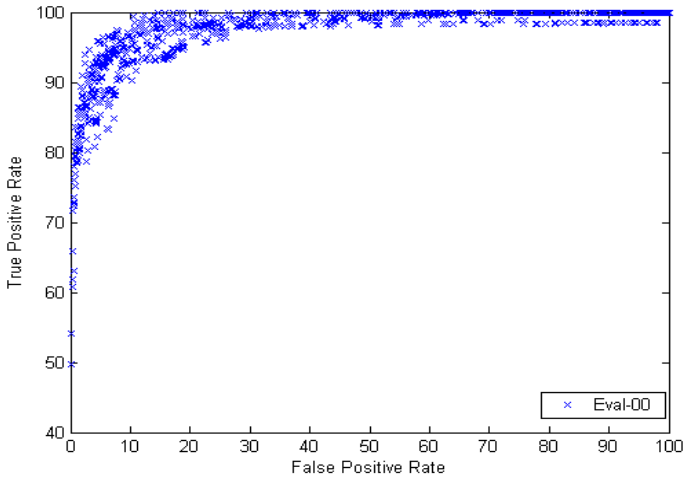


Fig. 2. ROC curve for dataset Eval-00

Based on the above observation, a reasonable way to estimate the scale factor for a dataset for which the ratio of spam-to-non-spam e-mails is known is as follows: adjust the scale factor until the ratio of spam-to-non-spam classification of the filter is equal to the known ratio. This approach can be applied to each individual user's inbox if the ratio of spam-to-non-spam e-mails is known.

6 Conclusion

We present a simple statistical algorithm for automatic personalized spam filtering that does not require users to provide feedback regarding the classifications of e-mails in their inboxes. The algorithm builds a statistical model of words from a training corpus and then adapts it to the distribution of words and e-mails in each individual user's inbox. Overall, the algorithm requires one pass over e-mails in the training corpus and two-passes over e-mails in the individual user's inbox. Our experiments confirm the benefit of personalization with significant performance gains over a filter that assumes that training and evaluation (users' inboxes) datasets follow the same distribution. We present extensive results of our algorithm including a discussion on the estimation of its two parameters.

The problem of automatic personalized spam filtering has generated much interest recently. It is a technically challenging problem that promises significant benefit to e-mail users and e-mail service providers.

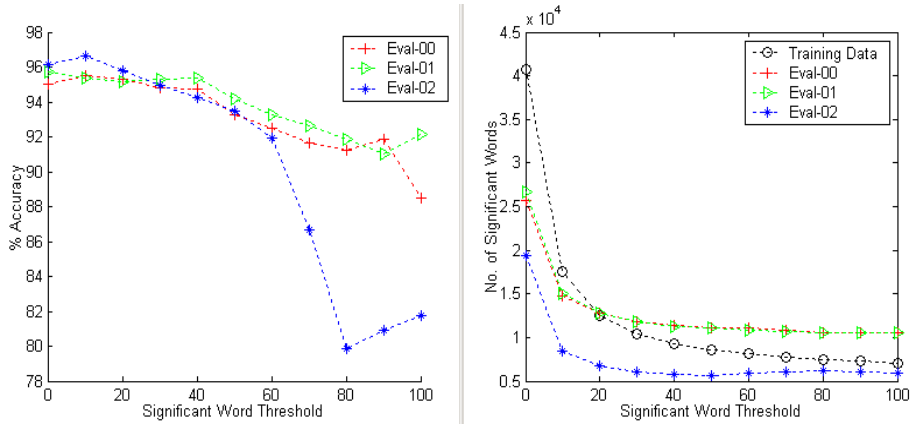


Fig. 3. Impact of threshold on accuracy and number of significant words

References

1. D. Evett: Spam statistics 2006. TopTenREVIEWS http://spam-filter-review.toptenreviews.com/spam-statistics.html (2006)
2. Commtouch: January virus and spam statistics: 2006 starts with a bang. Commtouch Press Release http://www.commtouch.com/Site/News_Events/pr_content.asp?news_id=602&cat_id=1 (2006)
3. ECML-PKDD: Discovery challenge. http://www.ecmlpkdd2006.org/challenge.html (2006)
4. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz: A bayesian approach to filtering junk E-mail. In Proc. of AAAI Workshop on Learning for Text Categorization. AAAI Technical Report WS-98-05 (1998)
5. P. Graham: Better Bayesian filtering. In Proc. of 2003 Spam Conference http://www.paulgraham.com/better.html (2003)
6. E. Michelakis, I. Androutsopoulos, G. Paliouras, G. Sakkis, and P. Stamatopoulos: Filtron: a learning-based anti-spam filter. In Proc. 1st Conf. on Email and Anti-Spam (CEAS 2004) (2004)
7. A.K. Seewald: An evaluation of naive Bayes variants in content-based learning for spam filtering. Kluwer Academic Publsihing (2005)
8. H. Drucker, D. Wu, and V.N. Vapnik: Support vector machine for spam categorization. IEEE Transactions on Neural Networks 10(5) (1999) 1048–1054
9. A. Kolcz and J. Alspector: SVM-based filtering of e-mail spam with content-specific misclassification costs. In Proc. of the TextDM'01 Workshop on Text Mining (2001)

10. G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos: A memory-based approach to anti-spam filtering for mailing lists. Information Retrieval, Springer 6(1) (2003) 49-73
11. W.W. Cohen: Learning rules that classify e-mail. In Proc. of 1996 AAAI Spring Symposium in Information Access (1996)
12. E.C.J. Kay and E. McCreath: Automatic induction of rules for e-mail classification. In Proc. of the Sixth Australasian Document Computing Symposium (2001) 13–20
13. S.J. Delany, P. Cunningham, and L. Coyle: An assessment of case-based reasoning for spam filtering. Artificial Intelligence Review, Springer 24(3-4) (2005) 359-378
14. R. Segal. J. Crawford, J. Kephart, and B. Leiba: SpamGuru: an enterprise anti-spam filtering system. In Proc. of Conference on Email and Anti-Spam (CEAS '04) (2004)
15. A. Gray and M. Haahr: Personalised collaborative spam filtering. In Proc. of Conference on Email and Anti-Spam (2004)
16. S. Bickel and T. Scheffer: Dirichlet-enhanced spam filtering based on biased samples. In Proc. of the Workshop on Learning with Nonparametric Bayesian Methods (with ICML'06) (2006)
17. C. Cortes and M. Mohri: AUC optimization vs. error rate minimization. Advances in Neural Information Processing Systems, NIPS (2004)

Text Classification Using Clustering

Antonia Kyriakopoulou and Theodore Kalamboukis

Department of Informatics, Athens University of Economics and Business
76 Patission St., Athens, GR 104.34
{tonia, tzk}@aueb.gr

Abstract. This paper addresses the problem of learning to classify texts by exploiting information derived from both training and testing sets. To accomplish this, clustering is used as a complementary step to text classification, and is applied not only to the training set but also to the testing set. This approach allows us to estimate the location of the testing examples and the structure of the whole dataset, which is not possible for an inductive learner. The incorporation of the knowledge resulting from clustering to the simple BOW representation of the texts is expected to boost the performance of a classifier. Experiments conducted on tasks and datasets provided in the framework of the ECDL/PKDD 2006 Challenge Discovery on personalized spam filtering, demonstrate the effectiveness of the proposed approach. The experiments show substantial improvements on classification performance especially for small training sets.

1 Introduction

Text classification is one of the first applications of machine learning, that applies to the general problem of supervised inductive learning: given a set of training documents, classified to one or more predefined categories, learn to automatically classify new documents. Automated text classification has been used in a number of different applications: automatic indexing, content management, filtering, and routing, word sense disambiguation, and Yahoo!-style search space categorization [1, 20]. A plethora of techniques have been developed for text classification, including Nearest-Neighbours [29], Regression [30], Neural Networks [28, 14], Naive Bayes [12], Decision Trees [27], and Support Vector Machines [26, 8]. In most cases, the classification algorithms require sufficient training data in order to generalize well on unseen documents. However, the generalization using labeled examples is an extremely costly and time-consuming activity. The need for classifiers that can learn from small training samples is imperative. This is an area of active research and several experiments have been conducted to boost conventional classifiers' performance, by combining supervised learning with semi-supervised or unsupervised, using techniques such as co-training [5, 13], active learning [25, 19], and transductive SVMs [10].

In traditional supervised classification an inductive learner is first trained on a training set, and then is called to classify a testing set, about which it

has no prior knowledge. An ideal situation would be for the classifier to have information about the distribution of the testing examples before it classifies them. This remark motivates the work included in this paper. In this vein, the goal of this paper is to deal with the problem of learning from training sets of different sizes, by exploiting information derived from clustering the whole dataset (both training and testing examples), and embodied in it in the form of *meta-information*.

Clustering has been used in the literature of text classification either as an alternative approach to term selection for dimensionality reduction or as a technique to enhance the training set. In the second case, clustering is used to discover a kind of "structure" in the training examples and expand the feature vectors with new attributes extracted from clusters. Also, it is used to augment a small number of labeled examples with unlabeled examples by propagating label information to the unlabeled data according to clustering results on both labeled and unlabeled data. Several approaches of clustering have been proposed in these areas.

In [2], *class-distributional clustering* [15] is applied as a feature selection method in a text classification context using a Naive Bayes classifier. Words are clustered into groups based on the distribution of class labels associated with each word. In [21], the *information bottleneck (IB)* method [24] is used to find word-clusters that preserve the information about the categories as much as possible. These clusters are used to represent the documents in a new, low dimensional feature space and a Naive Bayes classifier is applied. Accordingly, [3, 4] also use the IB framework to generate a document representation in a word cluster space instead of word space, where words are viewed as distributions over document categories. [6, 7] propose an information-theoretic divisive algorithm for word clustering and apply it to text classification. Classification is done using word clusters instead of simple words for document representation. [22, 23] adopt a more complicated approach, applying *two-dimensional clustering* to text classification. They cluster the training examples in order to deal with problems where the texts in a category are not generated from an identical probability distribution, and also they cluster the words/features of these examples in order to avoid the data sparseness problem. The evaluation is done using Naive Bayes and SVM classifiers on Reuters-21578 corpus and shows a superiority of their algorithm over class-distributional clustering and word clustering.

The idea of using clustering as a technique to enhance the training set is pursued in many works too. In [16], an algorithm is described that uses unlabeled data, independent from the testing set, to improve text classification performance. The algorithm applies clustering to labeled and unlabeled data, and introduces new features extracted from those clusters to the patterns in the labeled and unlabeled data. They evaluate the method using SVM classifiers on Reuters-21578, 20Newsgroups, and WebKB corpora, and find significant improvements in their classification performance. In [17], the technique presented above is combined with co-training. The algorithm trains two predictors in parallel, with one predictor labeling the unlabeled data for training the other in the

next round. The predictors are SVMs, one trained using data from the original feature space, and the other trained with new features that are derived from clustering both labeled and unlabeled data. This new input feature space creates an alternative view of the data, which is used for co-training, using the same supervised learning algorithm that is used for the original feature space. The evaluation of the method using SVM classifiers on Reuters-21578, 20Newsgroups, and WebKB corpora confirm previous findings. The *clustering based text classification (CBC)* approach [31], adopts a different way of exploiting the unlabeled data. According to this approach, labeled training data and unlabeled data are first clustered. Some of the unlabeled data are then labeled based on the clusters obtained, i.e. the labels of the labeled data are propagated to the unlabeled data that are closest to the cluster centroids. Discriminative classifiers are subsequently trained with the expanded labeled dataset. Their experimental results on 20Newsgroups, Reuters-21578 and Open Directory Project (ODP), demonstrated that CBC outperforms existing algorithms, such as TSVMs and co-training, especially when the size of the labeled dataset is very small.

The works of [16, 31] could be considered most relevant ones to our approach. However, they use clustering in order to create a better training set, without looking into the testing set as we do.

In this article, an algorithm that combines supervised and unsupervised classification is proposed. In the unsupervised case, the aim is to extract a kind of "structure" from a given sample of objects, or to rephrase it better to learn a concise representation of these data. The reasoning behind this is that if some structure exists in the objects, it is possible to take advantage of this information and find a short description of the data. In our approach, given a classification problem, the training and testing examples are both clustered before the classification step, in order to extract the "structure" of the whole dataset, exploiting the dependence or association between index terms and documents. The structure extracted from the dataset is "translated" in such a way that each cluster is represented by one representative. This concise representation of the whole dataset is incorporated in the existing data representation; each object is assigned the corresponding cluster id using appropriate artificial *meta*-features. It is expected that the use of prior knowledge about the nature of the testing set will help in building a more efficient classifier for this set.

The paper is organized as follows. Sections 2 and 3 describe the proposed algorithm. In section 4, the experimental settings, i.e. the datasets, the evaluation metrics and the experimental results are presented. Finally, section 5 concludes with a summary of the work and future research.

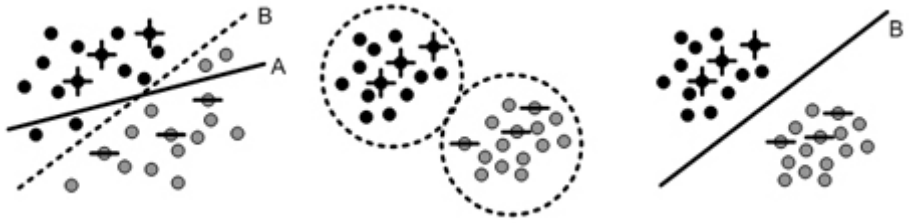
2 Classification with Clustering

In this section, we give the intuition of the proposed algorithm in order to understand how clustering prepares the ground for classification. The algorithm consists of the following steps:

1. Clustering step: to cluster both the training and testing set.
2. Expansion step: to augment the dataset with *meta*-features originated from the clustering step.
3. Classification step: to train a classifier with the expanded dataset.

Figure 1 gives an insight into our approach. The labeled examples of the training set are denoted with + and − signs, while the unlabeled examples of the testing set are denoted with dots.

A classifier trained with the given training examples will probably find hyperplane A instead of the desirable hyperplane B, as shown in Fig. 1(a). In Fig. 1(b), both datasets (training and testing) are clustered into two non-overlapping clusters. In the ideal case, the two clusters contain the positive and negative examples of the whole data set respectively. Then, corresponding *meta*-features are propagated to the existing feature vectors, and all feature vectors inside the same cluster are augmented with the same *meta*-feature. The dataset is transformed into a new coordinate system. Since feature vectors inside the same cluster are augmented with the same attribute-value pair, these vectors are now closer to one another resulting to an increase in the dataset's density, illustrated in Fig. 1(b). Increasing the inter-cluster distance consequently leads to the maximal margin hyperplane B, as shown in Fig. 1(c). In a way, the classifier is tuned to the testing set and the classification efficiency is expected to improve. Intuitively, the classifier with the largest margin will give lower expected risk, i.e. better generalization.



(a) Classification with the original data (b) Expanded dataset after clustering (c) Classification with the new expanded dataset

Fig. 1. An insight into the algorithm proposed

3 The Algorithm

In this section, we present our algorithm in more details. Following the traditional IR approach, we consider a k-class categorization problem, ($k = 1$ in the

case of the spam filtering problem), with a labeled l-sample $\{(\vec{x_1}, y_1), \ldots, (\vec{x_l}, y_l)\}$ of feature vectors $\vec{x_i} \in \mathcal{R}^n$, and corresponding labels $y_i \in \{1, \ldots, l\}$, and an unlabeled m-sample $\left\{\vec{x_1^*}, \ldots, \vec{x_m^*}\right\}$ of feature vectors, where $m \gg l$. In their original representation – that given in the framework of the Challenge – the datasets consist of feature vectors (documents) whose terms (features) are valued with their *term frequency*, $TF(w_i, \vec{x})$, i.e. the number of times term w_i occurs in the document $\vec{x}$. However, in our implementation the *TFIDF* model [18] is used, defined by

$$W(w_i) = TF(w_i, \vec{x}) * IDF(w_i) \tag{1}$$

where

$$IDF(w_i) = \log_2 \left(\frac{|X|}{DF(w_i)} \right) \tag{2}$$

is the *inverse document frequency* $IDF(w_i)$ with $|X|$ total number of documents in the training set, and *document frequency*, $DF(w_i)$, the number of documents that contain the term w_i. All feature vectors are normalized to unit length. For the classification step, the terms that appear only in the testing set but not in the training set are discarded.

The CLUTO^{TM} Clustering Toolkit [11] is used, and a divisive clustering algorithm with repeated bisections is applied. In this method, the disired k-way clustering solution is computed by performing a sequence of $k-1$ repeated bisections. The dataset is first clustered into two groups, then one of these groups is selected and disected further. This process continuous until the desired k number of clusters is found. During each step, the cluster is bisected so that the resulting 2-way clustering solution optimizes the internal criterion function

$$\max \sum_{i=1}^{k} \sqrt{\sum_{\vec{x}_v, \vec{x}_u \in S_i} sim(\vec{x}_v, \vec{x}_u)} \tag{3}$$

where S_i is the set of documents assigned to the i^{th} cluster, and $sim(\vec{x}_v, \vec{x}_u)$ is the similarity between documents $\vec{x}_v$ and $\vec{x}_u$. The generated clusters are non-overlapping.

After the clustering step, each cluster contributes one *meta*-feature to the feature space of the training and testing sets: given the total n features that are used in the representation of the $l+m$ feature vectors, and the k clusters derived from the clustering step, create *meta*-features $x_{n+1}, \ldots, x_{n+k}$. A document $\vec{x}$ that belongs to cluster C_j is characterized by the *meta*-feature x_{n+j}. The weight of that *meta*-feature is computed applying the *TFIDF* model to the clusters. Considering that each document in the cluster contains this *meta*-feature, its *term frequency* $TF(x_{n+j}, \vec{x}) = 1$ and its *inverse document frequency* is defined accordingly

$$IDF(x_{n+j}) = \log_2 \left(\frac{|X|}{|C_j|} \right) \tag{4}$$

Finally, on the classification step the SVM^{light} implementation of SVMs and TSVMs is used [9, 10]. A binary classifier is constructed for each user's *expanded* dataset, a linear kernel is used and the weight C of the slack variables is set to default.

The proposed algorithm is summarized in Table 1.

Table 1. The proposed algorithm.

Clustering step	
Input:	training examples $(\vec{x_1}, y_1), \ldots, (\vec{x_l}, y_l)$ testing examples $\vec{x_1^*}, \ldots, \vec{x_m^*}$ k = desired number of clusters Use a clustering algorithm to cluster all examples
Output:	k cluster ids
Expansion step	
Input:	training examples $(\vec{x_1}, y_1), \ldots, (\vec{x_l}, y_l)$ testing examples $\vec{x_1^*}, \ldots, \vec{x_m^*}$ k cluster ids Create additional *meta*-features for the vectors of *all* the examples. Each cluster corresponds to one new *meta*-feature. Given the total n features that produce the $l + m$ example vectors, and the k clusters derived from the clustering step, create *meta*-features $x_{n+1}, x_{n+2}, \ldots, x_{n+k}$. The values of these new features are defined by $W(x_{n+j}) = \left\{ \begin{array}{cc} \log_2\left(\frac{\lvert X\rvert}{\lvert C_j\rvert}\right) \text{ if } \vec{x} \in C_j \text{ for } j = 1, \ldots, k \\ 0 \qquad\qquad \text{otherwise} \end{array} \right\}$
Output:	expanded training examples $(\vec{x_1'}, y_1), \ldots, (\vec{x_l'}, y_l)$ expanded testing examples $\vec{x_1^*}', \ldots, \vec{x_m^*}'$
Classification step	
Input:	expanded training and testing examples of previous step. Train a SVM/TSVM classifier based on the *expanded* training examples. Classify the *expanded* testing examples.
Output:	predicted labels of the testing examples $y_1^*, \ldots, y_m^*$

A generalization of our algorithm includes the addition of more than one *meta*-features for each cluster. In this case, if f is the desired number of *meta*-features to be added per cluster, then an example $\vec{x}$ that belongs to cluster C_j is expanded with meta-features $x_{n+(j-1)*f+1}, \ldots, x_{n+(j-1)*f+f}$. Experiments that have been conducted show that expanding an example with more than one *meta*-features has a positive effect to classification. However, these experiments are beyond the scope of this paper.

4 A Performance Study

4.1 Experiment Settings

The empirical evaluation is done in two tasks, created and published in the framework of EMLC/PKDD 2006 Challenge Discovery[1]. The goal in both tasks is to make a personalized spam filter for a single user's inbox that correctly classifies its emails as *spam* or *non-spam*. In Task A, each classifier is made using the available training examples and each inbox separately, taking into account the user's inbox specific characteristics. In Task B, the learning algorithm is supposed to generalize over the different users in such a way that data from the other users may be utilized in order to enhance classification performance. In our experiment, however, both tasks are used in the same way, that defined for Task A. The reason for this is that we want to explore the effect of our technique when different sizes of training samples are used; Task A contains 4.000 training examples, whereas Task B only 100 training examples. The evaluation criterion prescribed by the competition is the AUC value. AUC values are computed for each user separately and average over all users.

4.2 Results and Further Discussion

To provide a baseline for comparison, results from the standard SVM and transductive SVM classifiers are also presented.

Preliminary results from experiments conducted on three widely used corpora (Reuters, Ohsumed, and WebKB) have shown an increase of performance of classification when the number of clusters is equal to the number of the predefined classes. In traditional classification tasks it can be assumed that the classes correspond to topics, and there is a one-to-one correspondence between the topic and the class under which the data are classified. Moreover, the examples of a class are clustered together which is logical since they share the same word distribution. So we can assume that there is a one-to-one correspondence between classes, topics and clusters, and use this information to define the desired number of clusters. In spam filtering we can't make such safe assumptions. *Spam* emails can deal with many different topics, there is a one-to-many correspondence between the class *spam* and the topics of the examples that fall under it. The obvious number of clusters to select is two: one cluster with the *spam* emails and one cluster with the *non-spam*. But we loosen this assumption based on the rationale that not all people consider an email as *spam* or *non-spam*. It is suggested that *spam* emails should be similar in both the public domain emails of the training set and the users' inboxes (testing set), while the *non-spam* should differentiate. According to this assumption, the number of clusters is chosen to be equal to *three*: one cluster for the common *spam* emails, one cluster for the common *non-spam* emails and one cluster for the rest.

[1] The task specific number of emails and inboxes, and additional information about the settings of the Challenge Discovery, can be found in http://www.ecmlpkdd2006.org/challenge.html

Table 2 gives the results for Task A. The proposed method leads to an improvement in performance on all users, raising the average AUC by 6.6% when the SVM classifier is used with clustering and by 3.2% when the TSVM classifier is used accordingly.

Table 2. Average AUC for the users of Task A.

Users	**Standard SVM**	**Clustering +SVM**	**Standard TSVM**	**Clustering +TSVM**
user00	84.72	93.26	89.44	95.25
user01	89.10	96.65	94.43	97.28
user02	94.70	96.58	98.92	99.40
Average AUC	**89.51**	**95.50**	**94.26**	**97.31**

The results on the Task B datasets in Table 3 show a 3.2% improvement on AUC over the standard TSVM classifier. Due to the small number of the training and testing examples for this task (100 and 400 examples respectively), no SVMs were used in the classification step. In the last column of Table 3, we present the best results obtained in terms of AUC after several runs of the algorithm with various numbers of clusters and *meta*-features used. These results are for demonstration purposes only and they were possible after the release of the data with their true labels by the Challenge organizers. The numbers of clusters, k, and *meta*-features, f, used in these experiments are mentioned in the parentheses. The results reveal that there is still room for improvements in performance, which is currently under investigation.

One limitation of our algorithm is that with the constant arrival of new emails, the same procedure of clustering, *meta*-feature addition, and classification, should be applied again for the whole dataset, a rather time consuming, and computationally expensive process. A suggestion would be to use incremental clustering instead of the static clustering algorithm used now. Incremental clustering is a method that deals with the problem of updating clusters without frequently performing complete reclustering. This would be a more suitable way for maintaining clusters in the typical, dynamic environment of spam filtering.

Another issue about our algorithm is its rather naive approach to clustering that may not capture all the *meta*-information possible hidden in the dataset. More sophisticated clustering methods have been proposed in the literature that focus on incorporating prior knowledge into the clustering process; conceptual clustering, topic-driven clustering [32], just to name a few. These methods are based in the idea that it is possible to use explicitly available domain knowledge to constrain or guide the clustering process. In our case, the class labels of the training set can constitute the domain knowledge and be used as guidance to a clustering algorithm.

Another issue that needs to be discussed is the representation of the extra knowledge derived from clustering, i.e. the representation of the clusters. The

Table 3. Average AUC for the users of Task B.

Users	Standard TSVM	Clustering +TSVM	Clustering +TSVM (best)
user00	96.76	97.36	98.19 ($k = 2, f = 12$)
user01	91.75	96.67	98.50 ($k = 5, f = 5$)
user02	97.34	97.09	99.93 ($k = 5, f = 5$)
user03	97.11	99.09	99.73 ($k = 5, f = 3$)
user04	94.91	96.44	97.99 ($k = 10, f = 5$)
user05	82.09	95.74	97.31 ($k = 3, f = 3$)
user06	82.91	90.87	93.06 ($k = 4, f = 5$)
user07	98.19	97.65	99.24 ($k = 3, f = 15$)
user08	98.95	99.16	99.77 ($k = 5, f = 1$)
user09	98.18	96.85	99.25 ($k = 2, f = 5$)
user10	92.56	92.99	96.16 ($k = 10, f = 15$)
user11	92.19	94.73	96.73 ($k = 4, f = 5$)
user12	87.77	88.92	92.38 ($k = 4, f = 15$)
user13	92.32	90.14	97.53 ($k = 4, f = 3$)
user14	78.98	92.44	99.28 ($k = 3, f = 15$)
Average AUC	**92.13**	**95.08**	**97.67**

representation schemes where a cluster of points is represented by their centroid or by a set of distant points in the cluster are the most popular ones. It appears that a different cluster representation scheme and its readjustment to our model's terrain might reflect the structure of more complex datasets, in a more efficient way. We hope that this can be further evaluated in future works.

5 Conclusions

This paper has introduced a new way to combine clustering with classification. We presented experimental results on datasets given in the framework of the ECML/PKDD Discovery Challenge 2006 on spam filtering. On all the collections the clustering approach combined with a SVM/TSVM classifier outperformed the standard SVM/TSVM classifier.

Possible extensions and improvements of our model include incremental clustering, and semi-supervised clustering. Other issues that can be further researched include the estimation and statistical basis of the optimum number of clusters and *meta*-features to be used.

References

1. Aas, K., Eikvil, L.: Text Categorization: A Survey. (1999)
2. Baker, L.D., McCallum, A.K.: Distributional clustering of words for text classification. Proceedings of SIGIR98, 21st ACM International Conference on Research and

Development in Information Retrieval, Melbourne, AU, ACM Press, New York, US (1998) 96103
3. Bekkerman, R., El-Yaniv, R., Tishby, N., Winter, Y.: On Feature Distributional Clustering for Text Categorization. Proceedings of SIGIR01, 24th ACM International Conference on Research and Development in Information Retrieval, New Orleans, US, ACM Press, New York, US (2001) 146153
4. Bekkerman, R., El-Yaniv, R., Winter, Y.: Distributional Word Clusters vs. Words for Text Categorization. Journal of Machine Learning Research 3 (2003) 1183-1208.
5. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. Proceedings of the 11th Annual Conference on Computational Learning Theory, (1998) 92-100, ACM Press, NY
6. Dhillon, I., Mallela, S., Kumar, R.: Enhanced word clustering for hierarchical text classification. Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, (2002) 191-200
7. Dhillon, I., Mallela, S., Kumar, R.: A Divisive Information-Theoretic Feature Clustering Algorithm for Text Classification. Journal of Machine Learning Research 3 (2003) 1265-1287
8. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Proceedings of the 10th European Conference on Machine Learning (ECML) (1998)
9. Joachims, T.: Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schlkopf and C. Burges and A. Smola (ed.), MIT-Press, (1999)
10. Joachims, T.: Transductive inference for text classification using support vector machines. Proceedings of 16th International Conference on Machine Learning. San Francisco: Morgan Kaufmann (1999) 200-209
11. Karypis, G.: CLUTO a clustering toolkit. Technical Report 02-017, Dept. of Computer Science, University of Minnesota (2002) Available at http://www.cs.umn.edu/cluto.
12. Lewis, D.D.: Naive (Bayes) at forty: The independence assumption in information retrieval. Proceedings of ECML-98, 10th European Conference on Machine Learning, pages 415, Chemnitz, DE (1998)
13. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning (1999) 1-34
14. Ng, H.T., Goh, W.B., Low, K.L.: Feature selection, perceptron learning, and a usability case study for text categorization. Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval, Philadelphia, US (1997) 6773
15. Pereira, F., Tishby, N., Lee, L.: Distributional clustering of English words. Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (1993) 183-190
16. Raskutti, B., Ferr, H., Kowalczyk, A.: Using unlabeled data for text classification through addition of cluster parameters. Proceedings of the 19th International Conference on Machine Learning ICML (2002)
17. Raskutti, B., Ferr, H., Kowalczyk, A.: Combining clustering and co-training to enhance text classification using unlabeled data. Proceedings of SIGKDD 02, Canada (2002)
18. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. New York: McGraw-Hill (1983)

19. Schohn, G., Cohn, D.: Less is More: Active Learning with Support Vector Machines. Proceedings of ICML-00, 17th International Conference on Machine Learning (2000)
20. Sebastiani, F.: A tutorial on automated text categorization. Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence (1999) 7-35
21. Slonim, N., Tishby, N.: The power of word clustering for text classification. Proceedings of the European Colloquium on IR Research, ECIR (2001)
22. Takamura, H., Matsumoto, Y.: Two-dimensional Clustering for Text Categorization. Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002), Taipei, Taiwan, (2002) 29-35
23. Takamura, H.: Clustering approaches to text categorization. Doctor's thesis (2003)
24. Tishby, N., Pereira, F.C., Bialek, W.: The Information Bottleneck Method. Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing (1999)
25. Tong, S., Koller, D.: Support Vector Machine Active Learning with Applications to Text Classification. Proceedings of ICML-00, 17th International Conference on Machine Learning (2000)
26. Vapnik, V.: The nature of statistical learning theory. Springer, NY (1995)
27. Weiss, S.M., Apte, C., Damerau, F.J., Johnson, D., Oles, F.J., Goetz, T., Hampp, T.: Maximizing text-mining performance. Intelligent Information Retrieval, IEEE (1999)
28. Wiener, E., Pedersen, J.O., Weigend, A.S.: A neural network approach to topic spotting. Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, US (1995) 317332
29. Yang, Y.: Expert network: an effective and efficient learning from human decisions in text categorization and retrieval. Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval, Dublin, IE (1994) 13-22
30. Yang, Y., Chute, C.G.: An example-based mapping method for text categorization and retrieval. ACM Transactions of Information Systems, 12(3) (1994) 252-277
31. Zeng, H.J., Wang, X.H., Chen, Z., Lu, H., Ma, W.Y.: CBC: Clustering based text classification requiring minimal labeled data. Proceedings of the 3rd IEEE International Conference on Data Mining, Melbourne, Florida, USA (2003)
32. Zhao, Y., Karypis, G.: Topic-driven Clustering for Document Datasets. SIAM Data Mining Conference (2005)

Using Tri-Training and Support Vector Machines for addressing the ECML-PKDD 2006 Discovery Challenge

Dimitrios Mavroeidis, Konstantinos Chaidos, Stefanos Pirillos, Dimosthenis Christopoulos and Michalis Vazirgiannis

Department of Informatics, Athens University of Economics and Business, Greece

Abstract. In this paper we present and analyze the methodological approach we have used for addressing the ECML - PKDD Discovery Challenge 2006. The Challenge was concerned with the identification of individual user's spam emails based on a centrally collected training set. The task descriptions of the discovery challenge indicated that we should deviate from the classical supervised classification paradigm and attempt to utilize semi-supervised and transductive approaches. The format of the training data (bag-of-words providing only word IDs), did not allow either for the use of Natural Language Processing (NLP) approaches, or for the use of standard spam-recognition strategies. The submitted model, which achieved 5^{th} place on Task A of the challenge, was derived by Tri-Training, a recent development in Semi-supervised algorithms research. Given a standard classifier, Tri-Training initially uses bagging to produce three diverse training datasets-classifiers, which are used for classifying the unlabeled data and incorporating them into the training set in a theoretically sound way. The classifier we have used within Tri-Training was Support Vector Machines (SVM) and more precisely the Sequential Minimal Optimization (SMO) implementation of WEKA. Moreover, we have used feature normalization and logistic regression models to produce continuous outputs. Apart from a detailed description and a discussion of the submitted model, this paper contains an extensive empirical evaluation of two popular semi-supervised classification algorithms: Transductive Support Vector Machines (TSVM) and Tri-Training.

1 Introduction

The ECML-PKDD discovery challenge 2006, was concerned with the construction of a spam recognition filter, based on previously classified emails. The organizers considered the spam training set to be collected centrally, by server based spam filters that can construct a labeled spam/non-spam training set using publicly available sources and spam traps. Although, the centralized collection of the labeled training set presents several advantages, it is probable that the distribution of the server - collected training set is different than the distribution of the emails received by individual users. This raises the need of deviating from the classical supervised classification paradigm where the goal of classification algorithm is to minimize the expected error over instances taken from the same distribution as the training set, and utilize semi-supervised [1] or transductive [2] approaches that take into account the distribution of the test set (individual user's inboxes), where predictions should be made.

Semi-supervised algorithms take into account both labeled and unlabeled instances and attempt to find a balance between generalization (using the labeled examples) and adaptation (using the unlabeled examples) to construct a model that generalizes well on the whole space of labeled and unlabeled data. A slightly different research paradigm, that is related to semi-supervised learning is presented by transductive learning. The transductive paradigm considers a set of labeled training data and a set of unlabeled test data, and the goal is to perform predictions only on the test data (and not on the whole space of training and test data). Research on semi-supervised algorithms has been receiving increasing attention, and several algorithms have been proposed (i.e. Transductive Support Vector Machines (TSVMs) [3], Tri-Training [4], Spectral Graph Transduction [5]). The Discovery Challenge presents an excellent opportunity for evaluating these algorithms empirically and for exploring possible strategies for tuning their parameters effectively.

In the context of Task A of the Discovery Challenge we have conducted extensive experiments using TSVMs and Tri-Training. The submitted model that yielded the best result on the tuning data and achieved 5^{th} place in Task A of the contest, was derived by the Tri-Training algorithm. Given a classifier and a set of labeled and unlabeled data, Tri-Training initially constructs three diverse datasets-classifiers using bagging [6]. Subsequently, it uses an incremental procedure, where in each round, an unlabeled example is added in the training set of a classifier if the other two classifiers agree on the class label and certain theoretical criteria are met. The classifier used in the context of Tri-Training was the SVM [2] and more precisely the SMO [7] implementation of WEKA [8]. The SVM was parameterized by a linear kernel with complexity parameter $C = 0.015$. Moreover, we have used feature normalization and logistic regression models in order to produce continuous output.

The rest of the paper is organized as follows. Section 2 provides a short description of the Discovery Challenge. Section 3 presents the Data preprocessing strategies we have experimented with. Section 4 analyzes the model evaluation approaches we have used. Section 5 describes the learning algorithms used and presents the experimental results. Section 6 discusses the results and contains the concluding remarks.

2 Discovery Challenge Description

2.1 Task Description

The discovery challenge consisted of two tasks, Task A and Task B. Task A was concerned with the case where the size of the centrally collected training data was larger than the individual users' inboxes. More precisely, the centrally collected training set contained 4000 emails, and the three individual users inboxes, where predictions should be made, contained 2500 emails each. Task B was concerned with the case where the size of the centrally collected training data was small in comparison to the size of the inboxes of the individual users. In task B, the centrally collected training data contained 100 emailes, while predictions should be made on 15 user inboxes, containing 400 emails each. In order to tune the parameters of the algorithms, tuning data was provided for both tasks. Since we have submitted a solution only for task A of the chal-

lenge, in the rest of the document unless otherwise stated, we will refer to Task A of the challenge.

The evaluation of the submissions was performed using the correct class labels for the individual user's inboxes (where the predictions were made), with the Area Under the Receiver Operating Characteristics (ROC) Curve (AUC) as an evaluation measure. The ROC curve was originally introduced in the signal processing community for addressing the problem signal detection, and it has been utilized in various contexts (i.e. model selection [9], introduction of algorithms that optimize the AUC measure [10]) by the machine learning research community.

2.2 Data Description

The methodological approaches that could be utilized in the Discovery Challenge, were determined by the form of the data provided. The discovery challenge datasets were delivered in the bag-of-words representation, where the words were represented by numeric IDs. This prevented the contestants from using any Natural Language Processing (NLP) techniques that could enhance the performance of the learning algorithms. However, taking into account the fact the NLP techniques are receiving increasing attention from the machine learning community (i.e. Word Sense Disambiguation for Text Classification [11]), it would have been interesting if more information were provided, that allowed the use of NLP techniques.

Moreover, the email representations excluded the use of any traditional spam filter methods. Such methods are DNS Black-hole Lists (DNSBLs), which reject the email that come from certain IPs (e.g. dynamic and dial-up IP addresses), keyword-based filtering (e.g. block the emails that contain certain phrases), checksum-based filtering, which takes advantage of the fact that usually the spam emails that are sent by an individual user are almost identical, and several others. Providing such additional information would pose the challenge of deriving decision functions that combine spam recognition rules and the statistical models constructed by the learning algorithms. Although, this is the task that real world spam filters must achieve, this would probably be out of the scope of the ECML-PKDD conference.

3 Data preprocessing

3.1 Feature Selection

Although feature selection has been shown to improve algorithms' performance in several learning tasks and application areas, experimental results have suggested that text classification algorithms should not be expected to benefit from aggressive feature selection [12]. This is because most words (with the exception of stop-words and common terms, issues that can be dealt using stop word lists and term weighting) offer important information for the correct classification of text data. Whatsoever we have investigated possible benefits from using some popular feature selection algorithms.

The feature selection measures we have considered are the Bi-Normal Separation (BNS) metric and the Information Gain (IG). BNS is defined as: $F^{-1}(P(word|+)) -$

$F^{-1}(P(word|-)))$ where F^{-1} is the inverse of the cumulative probability function of the Normal Distribution. For a theoretical analysis of the BNS metric in the context of ROC analysis the interested reader can refer to [13]. The IG is defined as the difference in entropy caused by the existence of a feature. The IG has been used widely in the context of feature selection and machine learning algorithms.

We have conducted experiments using BNS and IG with SVMs and Naive Bayes classifiers on the training and the tuning datasets. The algorithms performed always better with all the features. We have also attempted to use BNS and IG for features selection, prior to using the Tri-Training algorithm (which yielded the best results on the tuning data). In this case also the algorithm performed superiorly when all the features were retained.

3.2 Feature Normalization

In the context of text classification, it has been suggested by many authors (i.e. [14],[15]) that feature normalization can significantly boost the performance of learning algorithms and especially Support Vector Machines. This can be easily understood if we consider that in the unnormalized case, the similarities between the emails will be affected by the size of the emails (longer emails will contain terms with higher frequency of occurrence).

In order to verify the appropriateness of normalization empirically we have experimented using normalized and unnormalized features with Support Vector Machine in the training and tuning data. The experiments have showed that normalization improves the classification results and is thus appropriate in the context of the challenge.

4 Model Evaluation

In order to investigate the possible strategies for model evaluation, we will firstly recall some details concerning the datasets that were provided by the Discovery Challenge. The main training and test data consisted of a labeled training set (*TrainData*) and three individual user's inboxes (*TestDataA,TestDataB,TestDataC*), where the predictions should be made. For tuning the parameters of the algorithms the organizers provided additionally, a labeled training set (*TuneTrainData* and an individual user's inbox where the labels were provided as well (*TuneTestData*).

Using these datasets it is straight forward to evaluate the performance of the supervised learning algoriths using k-fold cross validation on the *TrainData* and the *TuneTrainData* datasets. However, since we are interested in performing predictions on the individual users inboxes, we should investigate possible model evaluation strategies that involve the test data. A straight forward approach would be to simply construct the models on *TuneTrainData* and then use the whole *TuneTestData* to estimate the models' performance. However, in order not to favor models that overfit the *TuneTestData*, we have evaluated the algorithms using cross validation on the combination of the training and the test set. More precisely, we have divided both the *TuneTrainData* and the *TuneTestData* data in k folds. Then, the k fold cross validation result is derived as the average AUC score of the k runs, where in each run we train the model

on *TuneTrainData*-{ Fold i of the *TuneTrainData*} and then measure the AUC of the model on the *TuneTestData*-{ Fold i of the *TuneTestData* }.

5 Learning Algorithms

5.1 Supervised

As we have mentioned in the introductory section, the Discovery Challenge was concerned with the construction of "personalized" spam filters for individual users based on a centrally constructed labeled training set. This implied that semi-supervised and transductive algorithms were appropriate. However since such algorithms are not guaranteed to achieve better performance than standard supervised inductive classifiers, we have initially conducted experiments using two popular classifiers, Naive Bayes (with a kernel density estimator, to address the problem of numeric features) and SVM. For breverity, we do not present here the details of these algorithms. The interested reader can find detailed descriptions in any standard Data Mining textbook (i.e. [16]).

Concerning SVM, we have experimented using a linear kernel, feature normalization and logistic regression models for producing continuous outputs. It has to be mentioned that the use of logistic regression derives proper probabilistic output for the algorithm, however it does not affect the AUC performance (compared to using decision function values). We have explored the effect of using various values for the complexity parameter C. The values we have used were powers of the 2, ranging from 2^{-6} until 2^1. The results of the 10 fold cross validation on the *TrainData* and the *TuneTrainData*, did not vary significantly for the different values of C and the mean AUC was consistently above 0.98. The mean AUC scores of the 10 fold cross validation of the combined *TuneTrainData* and *TuneTestData* (as described in section 4), are reported on Figure 1.

Concerning the Naive Bayes, we have used Kernel Density estimator in order to address the problem of numeric features. The 10 fold cross validation results on the *TrainData* produced and average AUC score of 0.76, and similarly 0.80 on *TuneTrainData*. The average AUC derived by the 10 fold cross validation on the combined *TuneTrainData* and *TuneTestData*, was 0.37. This result signified that the distribution of the individual user's inbox was significantly different than the distribution of the centrally collected training data, and that standard Naive Bayes is highly inappropriate in the context of this challenge.

5.2 Transductive and Semi-Supervised

In our experimental evaluation of transductive and semi-supervised approaches, we have concentrated on two algorithms, TSVMs and Tri-Training. The TSVMs present the transductive version of the popular SVM classifier. The main intuition behind TSVM, is that instead of searching for the separating hyperplane that maximizes the margin between the two classes (as in SVM), it searches for the hyperplane that maximizes the margin between both training (labeled) and test (unlabeled) data.

Concerning TSVM, we have used the SVM-Light implementation available on the web site of T. Joachims (*http://www.cs.cornell.edu/People/tj/*). We have experimented

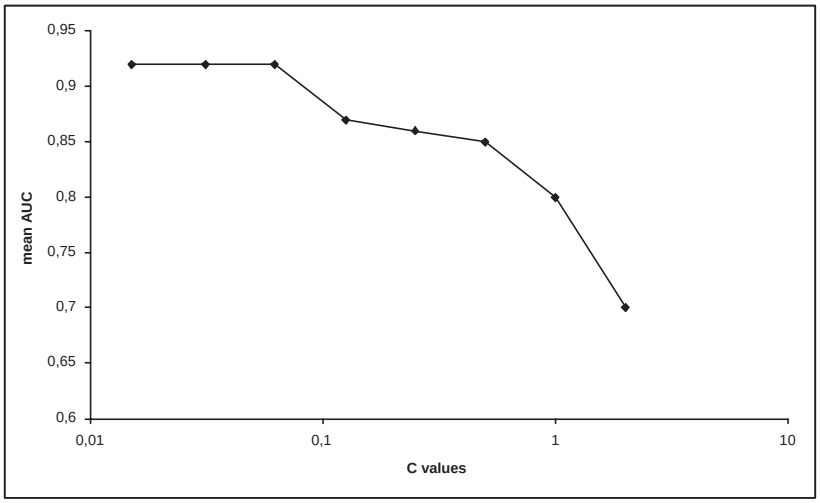


Fig. 1. SVM performance

using both Linear (inner product) and RBF Kernels. Concerning the Linear kernel we have used values that are powers of 2, ranging from 2^{-7} to 2^{12}. The 10 fold cross-validation results (in the fashion described in Section 4) for the various values of C are reported in Figure 2. Concerning the RBF kernel, we have experimented using values of C ranging from 2^5 to 2^{12} and γ values ranging from 2^{-4} to 2^{12}. The average AUC scores for these parameters of the RBF kernel were very low (consistently under 0.55). It has to be noted that in the TSVM experiments we had not normalized the feature space. This is because in the time we have deduced that normalization was appropriate, there was not adequate time for performing the normalized TSVM experiments.

We have also considered the Tri-Training algorithm, which has produced the best result on the cross validated test data. Tri-Training uses as input a supervised learning algorithm and a set of labeled and unlabeled instances. Subsequently, it uses bagging in order to produce three diverse training sets-classifiers. The main Tri-Training algorithm is based on an incremental procedure, where at each step, an instance is added to the training set of a classifier, if the other two classifiers agree on its label, and certain theoretical criteria are met. The criteria used, provide theoretical guarantees that the expected error of the classifier will be reduced when the new labeled example is added. For breverity we do not reproduce here all the details of the Tri-training algorithms, in [4], the interested reader can find the theoretical and empirical evidence for the appropriateness of the Tri-Training algorithm for semi-supervised learning tasks.

In the experimental evaluation we have used the Tri-Training implementation that is available on the web site of Ming Li, (*http://lamda.nju.edu.cn/lim/*), the second author of [4]. Prior to using Tri-Training, we have normalized the feature space. The classifier used for Tri-Training was an SVM and more precisely the SMO [7] implementation of WEKA [8]. Moreover, we have used logistic regression models for producing continuous outputs. It has to be noted that the use of logistic regression derives proper

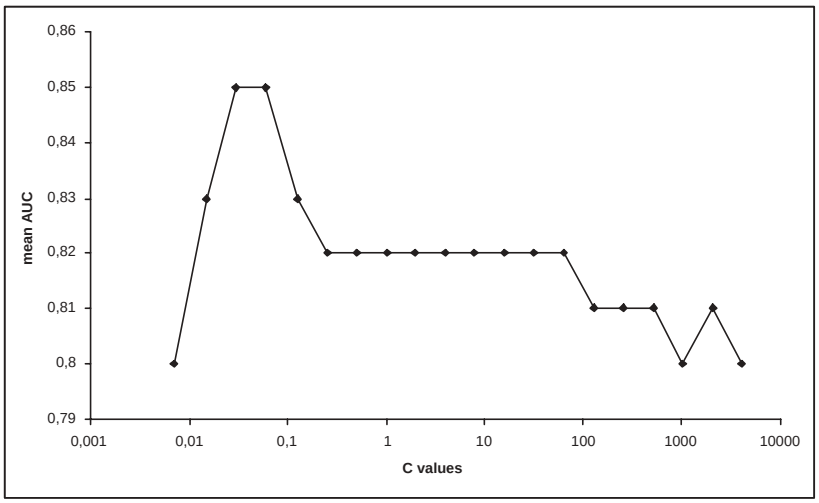


Fig. 2. TSVM performance

probabilistic output for the algorithm, however it does not affect the AUC performance (compared to using decision function values). In the experiments we have used a Linear Kernel (inner product) with various values of C. The values of C were again powers of 2, ranging from 2^{-7} to 2^3. In Figure 3, we report the average AUC scores derived from 10-fold cross validation (in the fashion described in Section 4). The best AUC score: 0.96 was achieved with the value $C = 0.015$.

6 Discussion - Conclusions

Based on the experimental results, an interesting observation that can be made concerns the kernel function, we have used in the submitted model. The Linear Kernel (inner product), yielded the best results on the tuning data among the algorithms we have experimented with and achieved 5^{th} place on Task A of the discovery challenge. Linear kernels are known to suffer from underfitting and it is general appreciated that they are not expressive enough for modeling complex real world data. Our experiments serve as an indication, that with the appropriate choice of C, linear kernels can be successfully applied in real world text classification problems.

Moreover, our experimental results have verified that Normalization can significantly improve classification performance of learning algorithms and that feature selection may not always be appropriate. Although these observations are widely known and have been discussed in various research papers (i.e. in [14, 15, 12]), our experimental results provide additional empirical verification.

Concerning the experimental results of the Tri-Training algorithm, it can be observed that the average AUC is more sensitive with respect to the C parameter than when using SVM and TSVM. An argument that can be used for explaining the sensitivity of Tri-Training, is that since in Tri-Training the results of the classifier are used

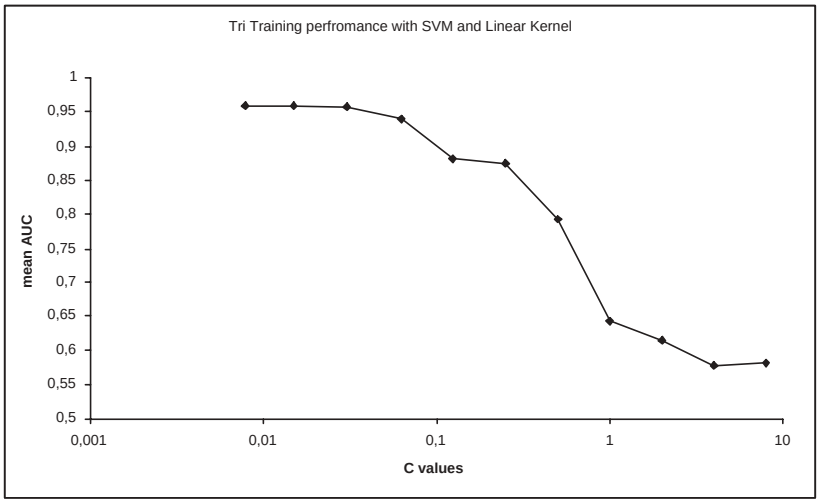


Fig. 3. Tri-Training performance

for adding unlabeled instance into the training set, small changes in the performance of the classifier may result in the addition of noise in the training set. Thus, a small reduction in the performance of the classifier may result in a much larger reduction of the performance of the Tri-Training algorithm. This signifies the importance of parameter tuning for semi-supervised algorithms that work by using a classifier to add the unlabeled instance into the training set (i.e. self-training).

In conclusion, we consider that the Discovery Challenge organized within the ECML - PKDD 2006, provided an excellent opportunity for the empirical evaluation of semi supervised algorithms. The experimental results can be useful both for theoretical and applied research, for understanding the properties of semi-supervised algorithms and identifying situations under which they should be expected to perform well.

References

1. Chapelle, O., Schölkopf, B., Zien, A., eds.: Semi-Supervised Learning. MIT Press, Cambridge (2006)
2. Vapnik, V.: Statistical Learning Theory. Wiley (1998)
3. Joachims, T.: Transductive inference for text classification using support vector machines. In: ICML. (1999) 200–209
4. Zhou, Z.H., Li, M.: Tri-training: Exploiting unlabeled data using three classifiers. IEEE Trans. Knowl. Data Eng. **17**(11) (2005) 1529–1541
5. Joachims, T.: Transductive learning via spectral graph partitioning. In: ICML. (2003) 290–297
6. Breiman, L.: Bagging predictors. Machine Learning **24**(2) (1996) 123–140
7. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In Scholkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning, MIT Press (1998)

8. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2005)
9. Provost, F.J., Fawcett, T.: Robust classification for imprecise environments. Machine Learning **42**(3) (2001) 203–231
10. Ferri, C., Flach, P.A., Hernández-Orallo, J.: Improving the auc of probabilistic estimation trees. In: ECML. (2003) 121–132
11. Mavroeidis, D., Tsatsaronis, G., Vazirgiannis, M., Theobald, M., Weikum, G.: Word sense disambiguation for exploiting hierarchical thesauri in text classification. In: PKDD. (2005) 181–192
12. Forman, G.: An extensive empirical study of feature selection metrics for text classification. Journal of Machine Learning Research **3** (2003) 1289–1305
13. Hanley, J.: The robustness of the binormal assumptions used in fitting roc curves. Medical Decision Making **8** (1998) 197–203
14. Herbrich, R., Graepel, T.: A pac-bayesian margin bound for linear classifiers: Why svms work. In: NIPS. (2000) 224–230
15. A.B.A. Graf, A.S., Borer, S.: Classification in a normalized feature space using support vector machines. IEEE Transactions on Neural Networks **14** (2003) 597–605
16. Hand, D.J., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press (2001)

A semi-supervised Spam mail detector

Bernhard Pfahringer

Department of Computer Science, University of Waikato, Hamilton, New Zealand

Abstract. This document describes a novel semi-supervised approach to spam classification, which was successful at the ECML/PKDD 2006 spam classification challenge. A local learning method based on lazy projections was successfully combined with a variant of a standard semi-supervised learning algorithm.

1 Introduction

This ECML/PKDD 2006 spam classification challenge[1] comprised two different setups, which both were setup in specific way encouraging especially the use of semi-supervised learning methods.

The setup for Task A comprised two mailboxes for tuning, one containing 4000 labeled email messages, the other one containing another 2500 labeled emails. Predictions were sought for another set of three un-labeled mailboxes of 2500 emails each, together with a labeled set of 4000 emails. The tuning set and the evaluation set were also distinct, i.e. they could not be fused to form a larger set of examples to learn from.

The setup for Task B problem comprised three much smaller mailboxes for tuning, one containing 100 labeled emails, the other two containing another 400 labeled emails each. Predictions were sought for another set of 15 un-labeled mailboxes of 400 emails each, together with a labeled set of 100 emails. The tuning sets and the evaluation sets again are distinct, i.e. again they could not be fused to form a larger set of examples to learn from.

Furthermore, all messages were already fully pre-processed and represented in SVMlight format, so no specific additional pre-processing was possibly. As proper pre-processing can be essential for learning success, this setup somewhat limited the possibilities, but also ensured a level playing field for all learning approaches. Generally in text classification, and especially in Spam classification, smart preprocessing can greatly simplify the problem, e.g. one can include attributes representing meta-information like sender names and servers, or other header-fields, or make a distinction between text coming from the subject and text coming from the message body and so on. None of that was possible for the fully pre-processed data given here.

2 Initial experiments

Initial experiments with standard (i.e. non semi-supervised) learning algorithms like multinomial Naive Bayes or support vector machines looked very promising

in cross-validation estimation on the labeled 4000 messages tuning box, but results did not transfer well over to the 2500 messages tuning box. Table 1 shows these somewhat surprising and more importantly disappointing results. A possible explanation might be the huge number of available attributes and potentially quite different usage patterns for these attributes in the two mailboxes. Therefore the learning algorithm could be focusing on a particular subset which works well for one mailbox, but not so for the other.

Table 1. Error rates (in percentages) on the tuning data of Task A for standard support vector machine (SMO) and multinomial NaiveBayes (MNB)

Algorithm	Crossvalidation: 4000 emails	Train/Test split 4000/2500 emails
MNB	3.075	51.20
SMO	0.950	20.56

Generally some form of feature subset selection should be able solve such an overfitting problem, but successful feature subset selection can be a very expensive and time-consuming process. Therefore an alternative lazy method was employed: whenever a prediction for some email is needed, transform the whole training set by only selecting those features which are actually present in the test-example (i.e. have a non-zero value). Essentially the training data is projected onto the subset of those features which are actually present in the testing email at hand. Then some model is trained using this transformed training set and that model's prediction is used for the test example in question. This transformation has some interesting properties: it forces the learner to concentrate on the features that are actually present in the test example. The number of such features is considerably smaller than the total number of features available, ranging from a dozen to about 2000, with the majority being in the low hundreds – instead of the more than a hundred thousand attributes in the full training set. Thus a larger range of classifiers becomes feasible, e.g. logistic regression. Still, linear support vector machines usually were among the best performing classifiers. Generally, using this lazy approach, results in terms of both estimated accuracies and estimated AUC values seemed to improve over the standard non-lazy approach described above. But even these improved estimates still left some clear room for improvement, which is why a proper semi-supervised approach building on this lazy projection idea was developed next.

3 Semi-supervised learning: a LLGC variant

Semi-supervised learning approaches have been shown in general to be able to improve over standard learning approaches when given plenty of unlabeled data. The given challenge problem does not quite fit the standard assumptions of semi-supervised learning, as actually more labeled than unlabeled data is given, at least in Task A: 4000 vs 2500.

A standard semi-supervised learning algorithm is the so-called LLGC algorithm [4], which tries to balance two potentially conflicting goals: locally, similar examples should have similar class labels, and globally, the predicted labels should agree well with the given training labels. We applied a scalable variant [3] of the standard LLGC algorithm to the challenge data. These experiments resulted in rather modest gains for predictive accuracies, but AUC values usually improved much more. As AUC was the criterion of choice for this challenge, this approach looked most promising and was chosen as the final solution. Here is a short summary of the three main modifications to the original LLGC algorithm:

- Allow different similarity measures: LLGC is based on a so-called affinity matrix capturing all pair-wise similarities over both labeled and unlabeled data. Originally, LLGC uses RBF kernels, but for text a cosine-based similarity measure is a much more appropriate, and is consequently used here.
- Allow pre-labeling of the unlabeled data: LLGC starts with all-zero class-distributions for the unlabeled data. We allow pre-labeling by using class-distributions for unlabeled data that have been computed in some way using the training data. The user can shrink the magnitude of these predictions relative to the hard class-labels of the supplied training set by way of a user-settable parameter. This shrinkage seems to improve results considerably when the number of unlabeled examples is much larger than the number of labeled examples. This is not the case for the current problem, and experiments on the tuning mailboxes returned best results when no shrinkage was employed. The pre-labels were computed by the lazy feature subset selection process described in the previous section together with a linear support vector machine as implemented in Weka.
- Reduce complexity by sparsifying the affinity matrix: the standard LLGC algorithm would need to compute the inverse of a dense $6500x6500$ matrix to solve the mailbox classification problem. To save both space and time, we sparsify the affinity matrix by only including the k nearest neighbours in the matrix. We do make sure that the matrix is still symmetric in a post-processing step after the kNN computation. The sparsification allows for a reduction in computational complexity from $O(n^3)$ to $O(n*k*iter)$ where n is the total number of examples, $k << n$ is the number of nearest neighbours, and *iter* is the number of iterations performed by the iterative relaxation algorithm which is used instead of matrix inversion.

4 The final setup

An extensive grid search over the tuning setup was used to find the following suitable values for all user-settable parameters for Task A:

- k = 100, the number of neighbours
- iter = 10, the number of iterations in the LLGC algorithm
- $\alpha = 0.8$, the mixing proportion of original labels to propagated information

– shrinkage = 1.0 (i.e. effectively no shrinkage), the trade-off factor for hard training labels versus estimated pre-labels.

The alpha parameter controls the balance between local and global consistency inside LLGC. Its range is $[0, 1]$, with higher values favouring local consistency over global consistency.

The three mailboxes of Task A were predicted in isolation of each other. It might be possible to improve the final results by actually fusing them all together into one big semi-supervised problem comprising 11500 examples. We did not attempt to do that as only one 2500 box for tuning was available. Therefore it was not possible to assess which of the two setup would perform better, so we chose the safe option of predicting each of the three mailboxes in isolation.

This final setup might seem to be rather expensive to compute, as we use a lazy approach for pre-labeling, and potentially expensive LLGC computation as a kind of post-processing step. In practise the computation is dominated by the pre-labeling effort, and takes about 90 minutes per mailbox, on an average Linux PC from 2005. It should therefore still be feasible in practise to run a similar setup on a client PC (but not a mail server itself) to rank a batch of emails for a particular user according to their degree of spamness.

A similar grid search over the tuning setup for Task B yielded almost identical values for all user-settable parameters:

– k = 100
– iter = 10
– $\alpha = 0.95$
– shrinkage = 1.0 (i.e. effectively no shrinkage)

It is interesting to note that for Task B an even more extreme *alpha* value (0.95 instead 0.8) was selected by this search, putting even more emphasis on the local neighbourhood of each example. Given the scarcity of labeled data in this setup this is probably a reasonable choice. It is also surprising to see that no shrinkage was chosen, even though Task B is more in line with the standard assumption of semi-supervised learning, having an order of magnitude more unlabeled than labeled examples available for learning.

As tests on the tuning setup showed slightly better AUC values for fusing all data into one big problem instead of predicting each mailbox in isolation, we chose this option for the final prediction task as well (again this is different to the Task A setup). So the final model used all 100 labeled training examples plus 15 times 400 unlabeled examples in one go. Even though this is about the same size as for one mailbox of Task A, computation was still much faster, as the computationally dominant pre-labeling step is a lot cheaper in this setting. We have to compute pre-labels for 6000 instead of 2500 examples, but each feature subset-selected training set has only 100 training instances instead of 4000 such instances, which makes for a big difference given the non-linear behaviour of support vector machine learning.

Therefore we conjecture that such joint processing of a number of relatively small mailboxes together with a carefully selected small training set could be feasible on a mail server for a small to medium-sized work group.

5 Conclusion and further research

This paper has introduced a successful combination of two new ideas for the detection of Spam messages: lazy projection of the training data to single messages to combat overfitting of the training data due to an abundance of features; and an efficient semi-supervised learning algorithm variant that can be viewed as a kind of post-processing step for smoothing out potentially conflicting predictions over similar messages. Pragmatically the most important direction for future work is the issue of pre-labeling. The current approach is effective, but also computationally expensive. An alternative to the lazy projection and training of a full classifier per unlabeled message would be projection of the training data to the set of all attributes being used in one full unlabeled mailbox. The next most expensive computational step after pre-labeling is the formation of the sparse affinity matrix, which is currently done in a naive linear full nearest neighbourhood search. More advanced methods like ball or cover trees should be able to speed up this step. More generally, we would like to investigate meta-approaches that would combine the architecture described in this paper with totally different ones like spam filters based on dynamic markov models [2].

Acknowledgments This work has been funded by a Marsden Grant of the Royal Society of New Zealand.

References

1. S. Bickel. Discovery challenge, 2006. http://www.ecmlpkdd2006.org/challenge.html.
2. A. Bratko, G.V. Cormack, B. Filipic, T.R. Lynam, and B. Zupan. Spam filtering using statistical data compression models, 2006. Journal of Machine Learning Research, in press.
3. B. Pfahringer, C. Leschi, and P. Reutemann. Scaling up semi-supervised learning: an efficient and effective llgc variant, 2006. in preparation.
4. D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency, 2003. In 18th Annual Conf. on Neural Information Processing Systems.

Identifying SPAM with Predictive Models

Dan Steinberg and Mikhaylo Golovnya

Salford Systems

1 Introduction

The ECML-PKDD 2006 Discovery Challenge posed a topical problem for predictive modelers: how to separate SPAM from non-SPAM email using classic word count descriptions of email messages. The data for the challenge were released around March 1, 2006 and submissions were due June 7, 2006, allowing entrants to devote as much as three months to preparing and modeling the data. We devoted two calendar weeks and three person weeks to this project, the maximum we could spare given other commitments. We found the project appealing for several reasons. First, we started with the belief that modern data mining methods and specifically boosted trees in the form of Jerome Friedman's MART (Multiple Additive Regression Trees) would perform well. Second, our industrial experience at Salford Systems has to date been focused on the analysis of numeric (non-text) data and were eager to gain more experience in the field of text mining. Third, the project organizers had already completed the initial mapping of the text documents to the word count "term vectors" allowing challenge participants to focus on the use of numerical tools and bypass the messy preprocessing of raw text data. Note that the challenge data contained no information regarding the original text; we do not even know the language of the emails let alone the nature of the triggers that could signal SPAM.
The challenge consisted of two tasks, Task A and Task B. As we addressed only Task A we confine our discussion accordingly.

2 Data

We include a brief description of the data here to allow this paper to be self-contained. Additional information may be found in the companion workshop papers and the challenge website http://www.ecmlpkdd2006.org/challenge.html. As the goal was SPAM detection, every email message in the data sets had been tagged as SPAM or not-SPAM (although the class label was not always made available to the modelers). The main training data set consisted of 4,000 email messages evenly divided between the SPAM and not-SPAM classes. The project documents suggest that the SPAM messages were collected from a public "spam trap" (an email address visible only to crawlers and bots, but invisible to humans) but there is no explicit confirmation of this. The not-SPAM emails were also (apparently) collected from "publicly available sources". Confining training data to public sources was a key component of the stated challenge as automatically generated SPAM filters would

ideally not rely on users to manually label their email messages for the purpose of training a SPAM filter. The challenge required participants to develop predictive models that would perform well on individual user email inboxes, and three such inboxes of 2500 unlabeled messages each were provided. Naturally, the distributions of the words used in the public source training data and the individual email boxes are quite different, and this is a major source of the difficulty for machine generated SPAM filters. We have to allow for the fact that individual users vary considerably in their tastes and interests and so what might look like SPAM in the inbox of user #1 might be an explicitly requested message in the inbox of user #2. In addition to the primary data, some "practice" or tuning data was provided. This data consisted of 4000 labeled training emails from public sources and an additional 2500 labeled emails from a single user inbox. The tuning data was supplied with scrambled word indices so that this data could not be pooled with the primary training data. The scrambling ensures that word number 3 in the tuning data does not correspond to word number 3 in the training data, etc. The tuning data could thus be used only to assess the performance of alternative modeling strategies; no specific detail extracted from a tuning data model could be used to improve the models generating the final challenge predictions. It is worth emphasizing that the tuning email inbox was the only example of labeled user data and that the tuning data provided us the only way to learn anything about adapting public source data to individual user SPAM detection.

3 First Stage Data Preparation

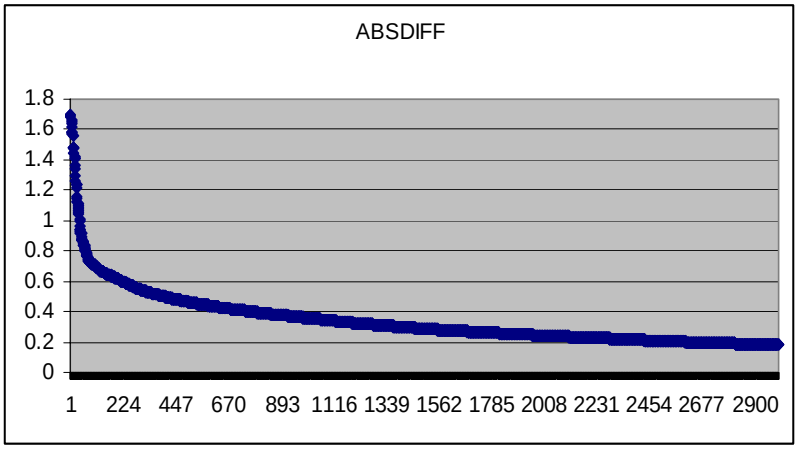
The raw data was delivered in a compressed form often used for sparse matrices. Thus a typical line of data might look like:

```
-1 2:3 9:8 17:3 35:7 71:3 74:2 77:6 85:5 86:1 92:2 94:2 95:6 99:2
```

This record corresponds to not-SPAM as the first element is -1, and it contains 3 occurrences of word #2, 8 occurrences of word #9, ..., and 2 occurrences of word #99. All words in the master vocabulary not listed on the record do not appear in the original email (i.e. they have a frequency or word count of zero). As our analytical and data preparation tools are not equipped to work with such data directly our first task was to expand the data into a complete non-sparse format with zero elements explicitly listed. When the master training data file is so expanded we find a total of more than 150,000 words. The expansion was also performed on all the other data sets. The evaluation data consisted of three user inboxes containing 2500 unlabelled rows of data, with 26580, 27523, and 20227 non-zero word counts respectively. The tuning data consisted of completely labeled data. The public source training data consisted of 4,000 rows and 39967 word counts, and the tuning user inbox consisted of 2,500 rows and 23070 word counts. The next step required combining data from the separate files. For the challenge training and evaluation data one option was to form the union of all words appearing in any file and using this as a master vocabulary. Given the size and cumbersomeness of the resulting files we elected instead to work with intersections. Thus, the intersection of words appearing in both

the training data and a given user inbox formed the master vocabulary for our work on that specific inbox. The number of words in each intersection was 16333, 17791, and 16399 respectively. For the tuning data the word counts were 46219 in the union and 16818 in the intersection of the two files.

Given the large vocabularies some form of attribute selection was advisable and we used a simple statistical test popular in bioinformatics. We first limited the vocabularies separately for each inbox, selecting the subset of words found in both the inbox and the training data. For each attribute so selected we calculated a modified t-test due to Tibshirani et. al (2001) for the difference of means between the SPAM and not-SPAM classes in the training data and then ranked the attributes by the absolute value of this t-statistic. The tests were organized separately for each user inbox and restricted to the vocabulary common to the train data and the inbox in question. An example graph of results appears below for the first user inbox. It plots the statistics against the rank order of the attribute.



This is not our preferred approach to attribute selection as it is based on a myopic univariate discriminant analysis but it was easiest to deploy. The t-tests rank the predictors and it was up to us to select a cut-off for attribute selection. We experimented with several different cutoffs in our initial modeling, trying as few as 1,000 and as many as 8,000 attributes to arrive at a preferred total of 3,000 predictors. All further data preparation and model generation reported below was based on an inbox-specific selection of the top 3,000 attributes.

4 Second Stage Data Preparation

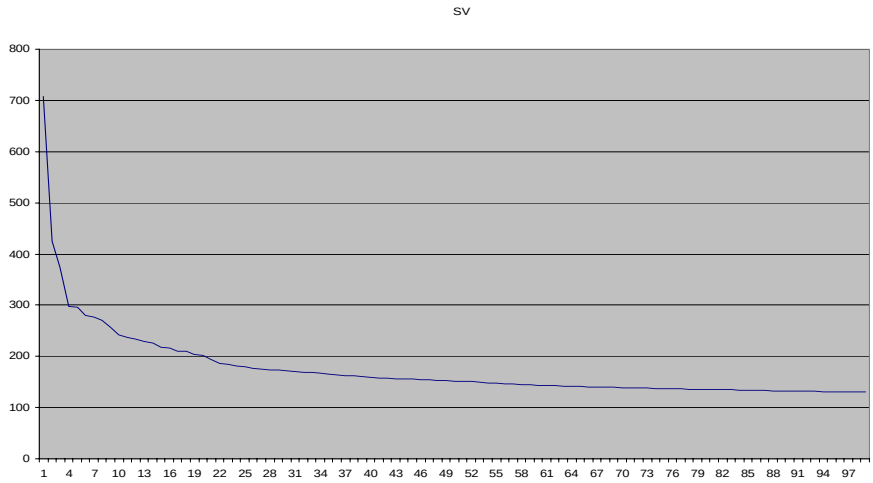
As email messages can vary greatly in length it is difficult to compare or cluster records based on the raw word frequencies. To adjust for the email length we converted the counts to relative frequencies by dividing each term count by the email total word count (so that the sum of relative frequencies is 1 for all records).

Following the text mining literature we also added the following summary document features describing each email:

- LEN: document size, total number of words,
- NNZ: number of unique words (Number NonZero),
- MAXF: count of most often used word,
- ENTR: document entropy, calculated over words,
- KL: distance to the target class training data centroids (Kullback-Leibler statistic)

Each feature was constructed in two versions: on the original document vocabulary (version 1) and on the restricted vocabulary found in the intersection between the training data and the specific inbox (version 2 or version R).

Previous research (Dumais et. al. 1988) has suggested that some form of data compression or summarization is vital for the analysis of word count data and we decided to follow this practice, compressing the data with a Singular Value Decomposition (SVD). The transform was applied to the subset of the best 3,000 relative word counts discussed above. The SVD is similar to Principal Components Analysis (PCA); it creates linear combinations of the original data (whether in raw count or relative frequency form) and effectively captures information on joint patterns of occurrences of individual words. The SVD transformation allows us to capture some types of relationships between words which is precisely what is missing in the raw word count data. One should expect such transformed data to yield better predictive performance than the raw (or normalized) counts. The SVD transformation usually allows a radical reduction in the number of predictors required as much of the information content of the raw data is captured in the leading SVD vectors. We experimented with keeping different numbers of SVD vectors and settled on 20 for most of our modeling work. (We have heard text mining practitioners recommend 40 as a rule of thumb). The screen plot below of the first 100 singular vectors plots the Singular Value against the singular vector number is representative of the results we obtained separately for each user inbox.



Note that the SVD transformation is a second stage of data reduction and follows the first stage of raw attribute selection. The SVD decomposition makes no reference to the target; it is based only on the predictors and can be generated from the training data or from the pooled training and unlabeled user inbox data. Below we report that restricting the SVD to the training data alone yields slightly better results. The SVD can also be computed with or without the summary document features listed in the table above, and the summary document features can also appear as separate predictors in their own right. We experimented to identify the best combination of predictors and inputs into the SVD.

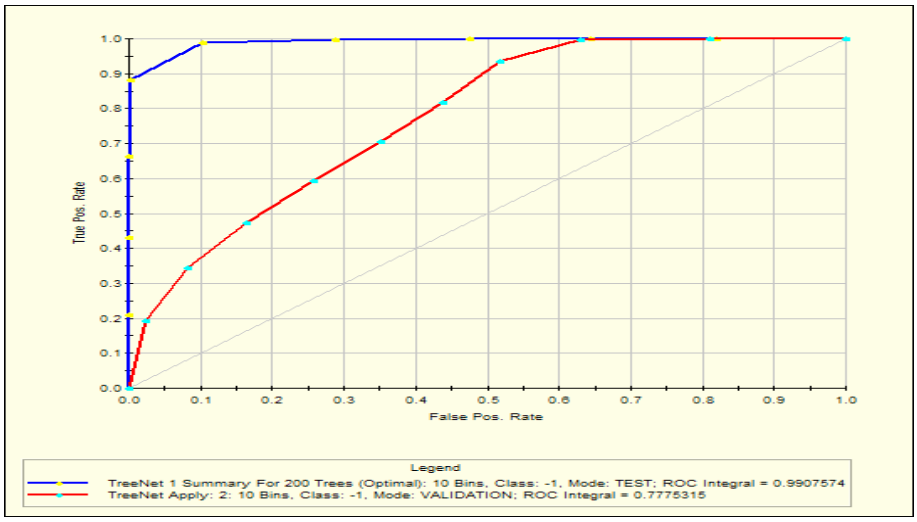
5 General Modeling Methodology

We elected to use TreeNet(tm), the commercial release of Friedman's stochastic gradient boosting as it has performed well in a broad range of real world predictive modeling challenges. Early exploratory runs were based on small ensembles of 200 trees constructed with a moderate learning rate of 0.10 to reduce run times. Our final models were run with a slow learning rate of 0.01, using the binary logistic loss function, and growing 1000 6-node trees. Parameter settings were chosen via experiments on the tuning data described below.

6 Modeling Details

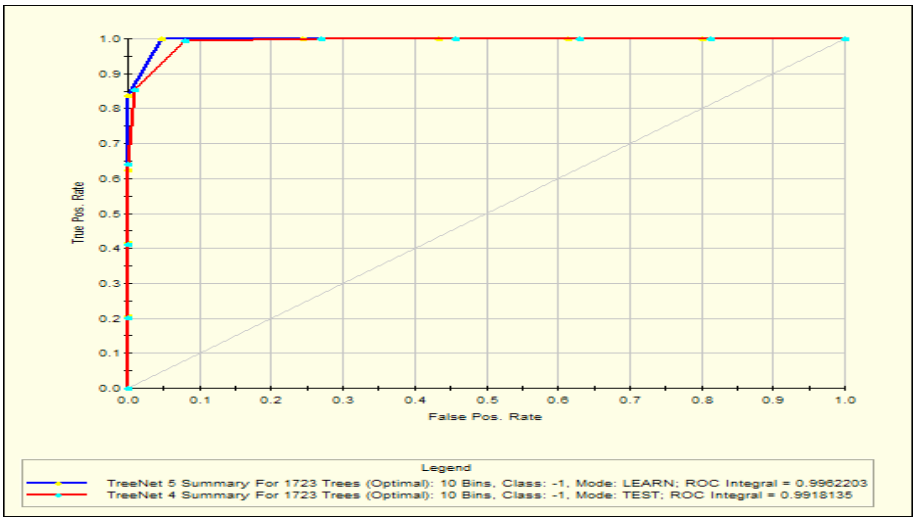
The results we report here are based on what was known to us at the time of the challenge, namely the tuning data. A revised analysis based on the labeled evaluation data would be a desirable undertaking but is not included here. We began with a simple experiment to determine how well a model built entirely on the public source

data would perform on an individual inbox. We divided the training portion of the Tune sample into random halves and built a quick TreeNet model using only the top 3000 ranked raw word counts as predictors and then used the "optimal" model to score the data from the Tune sample inbox. The graph below displays the ROC curves for the two test samples.



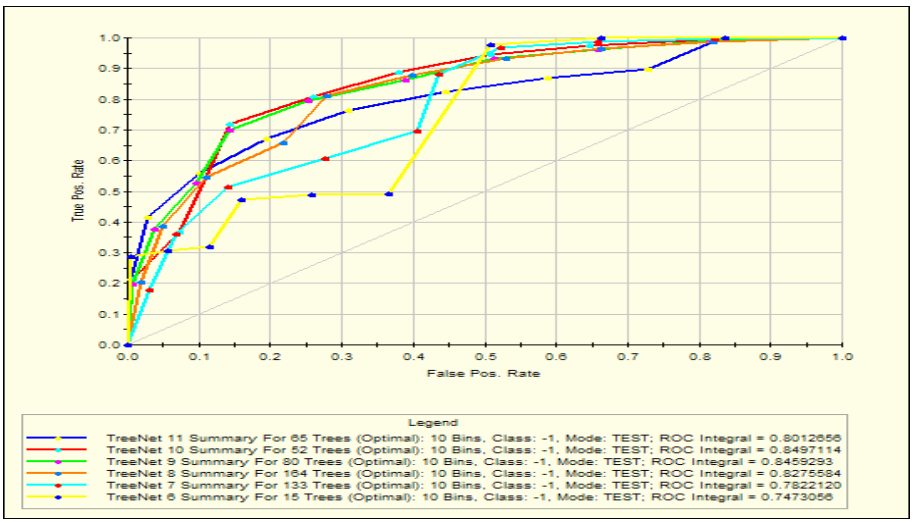
Note that the performance on public source data (the upper curve) is excellent. Even a crude 200-tree ensemble reaches an area under the ROC curve (AUC) of .991 on test data. But when the same model is applied to the rather different distribution of emails and vocabulary found in a real individual's inbox the AUC drops dramatically to .778. While .778 is a value that could be welcomed with enthusiasm in some applications it is far too low for SPAM detection and most email users would find the corresponding classification error rates of such a model annoying.

If we take the 2500 email individual user inbox tuning data, randomly set aside 1/3 of the data for test and develop a model on this individual user's data we again obtain excellent results with an AUC on test data better than .99: In the graph below the two ROC curves correspond to the training and test portions of the individual user's data. This tells us that successful inbox-specific models can be developed using naïve raw word counts as predictors. But the model is completely customized to this user's inbox.



The Challenge requires us to build predictive models from data that are somewhat less relevant than can be found in an individual's inbox, and we turn now to the models built in this way.

The models we report here are all based on the Tune data and were used to guide the parameter settings we used in the final challenge submission. The graph below displays the ROC curves and AUC values achieved on the single user tune inbox.



The lowest ROC curve (over the lower half of the graph) corresponds to the naïve raw word count model. The better models are based on various combinations of SVDs as predictors and document summary features. The best results were obtained when the SVD vectors were based only on the training data instances, but with the document

summary features included in the SVD construction, and with the document summary features also included as separate predictors. These results are very similar to those we obtained on the evaluation data.

7 Comments on Results

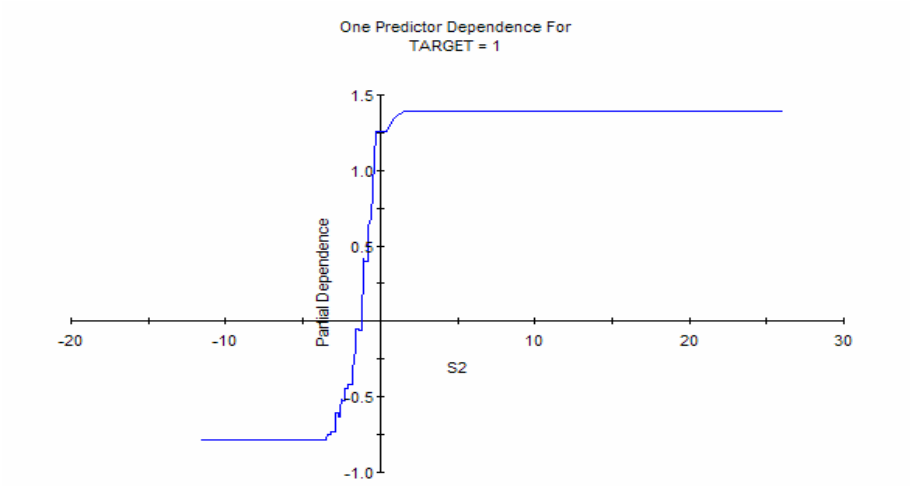
As the details of the data are hidden from us we cannot discuss specific indicators predicting SPAM other than the meaningful features we constructed from the abstract data. We can extract some useful but limited insights and we present those here. Based on the TreeNet variable importance ranking we see that the 2nd SVD vector is ranked most important, and that top 4 variables are all SVD vectors. The document summary features do play a role however, with the 3 such features appearing in the top 10 predictors.

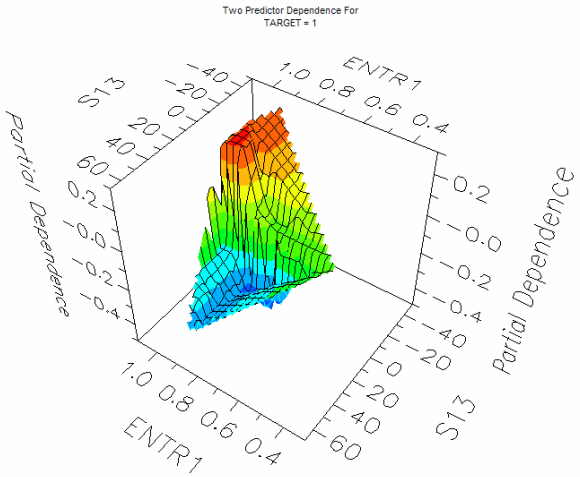
TreeNet Variable Importance

<table>
<tr><th>Variable</th><th>Score</th><th></th></tr>
<tr><td>S2</td><td>100.00</td><td>||||||||||||||||||||||||||||||||||||||||||</td></tr>
<tr><td>S9</td><td>49.11</td><td>||||||||||||||||||||</td></tr>
<tr><td>S3</td><td>33.89</td><td>||||||||||||||</td></tr>
<tr><td>S5</td><td>33.27</td><td>|||||||||||||</td></tr>
<tr><td>ENTR1</td><td>25.53</td><td>||||||||||</td></tr>
<tr><td>S13</td><td>22.85</td><td>|||||||||</td></tr>
<tr><td>S4</td><td>19.85</td><td>||||||||</td></tr>
<tr><td>S29</td><td>19.30</td><td>|||||||</td></tr>
<tr><td>KL2P</td><td>18.64</td><td>|||||||</td></tr>
<tr><td>NNZ1</td><td>16.46</td><td>||||||</td></tr>
<tr><td>S12</td><td>15.52</td><td>||||||</td></tr>
<tr><td>S21</td><td>13.51</td><td>|||||</td></tr>
<tr><td>S27</td><td>12.83</td><td>|||||</td></tr>
<tr><td>S25</td><td>12.03</td><td>||||</td></tr>
<tr><td>LEN1</td><td>11.89</td><td>||||</td></tr>
<tr><td>S10</td><td>10.81</td><td>||||</td></tr>
<tr><td>S17</td><td>9.79</td><td>|||</td></tr>
<tr><td>S23</td><td>9.61</td><td>|||</td></tr>
<tr><td>S1</td><td>8.58</td><td>|||</td></tr>
<tr><td>S26</td><td>8.30</td><td>|||</td></tr>
<tr><td>MAXF2</td><td>8.22</td><td>|||</td></tr>
<tr><td>S14</td><td>8.04</td><td>||</td></tr>
<tr><td>S28</td><td>7.26</td><td>||</td></tr>
<tr><td>KL2M</td><td>7.16</td><td>||</td></tr>
<tr><td>S11</td><td>7.08</td><td>||</td></tr>
<tr><td>S16</td><td>6.96</td><td>||</td></tr>
<tr><td>S18</td><td>6.04</td><td>||</td></tr>
<tr><td>S24</td><td>5.87</td><td>||</td></tr>
<tr><td>S8</td><td>5.37</td><td>|</td></tr>
<tr><td>S20</td><td>5.25</td><td>|</td></tr>
<tr><td>S22</td><td>4.14</td><td>|</td></tr>
<tr><td>S6</td><td>4.13</td><td>|</td></tr>
<tr><td>S15</td><td>2.46</td><td></td></tr>
<tr><td>NNZR</td><td>2.41</td><td></td></tr>
<tr><td>S19</td><td>0.00</td><td></td></tr>
<tr><td>S7</td><td>0.00</td><td></td></tr>
</table>

ENTR2	0.00	
MAXF1	0.00	
LENR	0.00	
S30	0.00	

The TreeNet dependency plots for the top variables are almost all sigmoids or ramp functions, similar to that shown below for the most important predictor S2 (of course some have a negative effect and are thus downward sloping from left to right). There are evidently powerful interactions captured in the model as illustrated in the 3D graph below showing the dependency of the target on ENTR1 and S13. :





8 Improving the Results

During the challenge we conjectured that there were at least two additional ways to improve the models. First, our list of features is rather small and others could be created, in particular features based on the prevalence of words appearing in each target class. Second, we suspect that reweighting the training data to make it more similar in word distribution to any individual inbox might also be helpful.

References

1. Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and Regression Trees. Wadsworth & Brooks/Cole, California.
2. Dumais S. T. , and G. W. Furnas and T. K. Landauer and S. Deerwester and R. Harshman (1988). Using Latent Semantic Analysis To Improve Access To Textual Information. CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems, 281—285, Washington, D.C.
3. Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of Statistics, 29:1189–1232.
4. Tibshirani, Robert, Trevor Hastie, Balasubramanian Narasimhan, Michael Eisen, Gavin Sherlock, Pat Brown, and David Botstein. (2001). Exploratory screening of genes and clusters from microarray experiments. http://www-stat.stanford.edu/~tibs/ftp/samclus.pdf.

TPN^2: Using positive-only learning to deal with the heterogeneity of labeled and unlabeled data

Nikolaos Trogkanis [1], Georgios Paliouras [2]

[1] School of Electrical and Computer Engineering, NTUA, Greece
tronikos@gmail.com
[2] Inst. of Informatics and Telecommunications, NCSR "Demokritos", Greece
paliourg@iit.demokritos.gr

Abstract. This paper introduces TPN^2, the runner up method in both tasks of the ECML-PKDD Discovery Challenge 2006 on personalized spam filtering. TPN^2 is a classifier training method that bootstraps positive-only learning with fully-supervised learning, in order to make the most of labeled and unlabeled data, under the assumption that the two are drawn from significantly different distributions. Furthermore, the unlabeled data themselves are separated into subsets that are assumed to be drawn from multiple distributions. For that reason, TPN^2 trains a different classifier for each subset, making use of all unlabeled data each time.

Keywords: one-class learning, positive-only learning, semi-supervised learning, multi-strategy learning, spam filtering

1 Introduction

The topic of the ECML-PKDD Discovery Challenge 2006 was personalized spam filtering. The goal was to train a personalized spam/ham classifier for each user that correctly classifies the emails in the user's inbox. Despite their personalization, it was assumed that the classifiers will be trained and used on the mailing server. Therefore, training cannot rely on messages labeled by the individual users. An obvious surrogate for training data is the use of publicly available sources, such as mailing lists and newsgroups and emails received through "spam traps"[1]. Such data have been used for benchmarking spam filters in the past, e.g. the Ling-spam[2] corpus. In addition to these labeled data, the personal emails of the users are assumed to be available for training, but without labels. These unlabeled data are available in large volumes and can be used to improve the classifiers in a semi-supervised fashion.

In semi-supervised learning a small set of labeled examples of every class and a large unlabeled set are used for building the classifier. Semi-supervised learning has been shown to be particularly beneficial in training text classifiers, such as spam

[1] Spam traps are email addresses published visually invisible for humans but get collected by the web crawlers of spammers.
[2] Ling-spam is available at: http://www.iit.demokritos.gr/skel/i-config/downloads/

filters, e.g. [1], [3], [9]. However, all of these techniques assume that labeled and unlabeled examples are generated from the same distribution. This assumption may be violated in practice, and when this happens these methods perform poorly.

One such example is the ECML-PKDD 2006 competition, where the labeled public data are very different from the emails received by individual users. Clearly, the unlabeled data are closer to the data expected to be processed by the filter in operation. Therefore, their use is even more important than in the usual semi-supervised learning scenario. At the same time, the use of the labeled data is essential, but has to be done with care, in order to avoid misleading the training process.

Addressing this dual problem, we chose to rely mostly on the large amounts of unlabeled emails in the user's inboxes. We used the labeled training data only to help us label a small part of the unlabeled data, sufficient to bootstrap the semi-supervised learning process. In doing that, we also took into account a natural asymmetry between the spam and the ham classes, namely that spam is much less personal than ham. Therefore, the discrepancy between the labeled and the unlabeled data is expected to be much higher for ham than for spam. Thus, we named spam the positive class and applied a positive-only learning approach on the unlabeled data.

To solve the problem of learning from positive and unlabeled examples, a few algorithms have been proposed in the past few years. One class of algorithms is based on a two-step strategy. This class includes SEM (Spy Expectation Maximization) [6], PEBL (Positive Example Based Learning) [11] and Roc-SVM (Rocchio – Support Vector Machines) [4]. These algorithms aim to iteratively discover the true negative examples, while maintaining correctly-classified the positive ones. It has been shown theoretically that this approach can lead to a good classifier [6]. In addition to these two-step algorithms, there are other methods that aim to estimate the proportion of negative to positive examples in unlabeled data and use that to bias the training process, e.g. PNB (Positive Naive Bayes) [2] and biased-SVM [5].

One final aspect of our approach was the utilization of multiple different subsets of unlabeled data. These subsets correspond to the inboxes of different users. Clearly the inbox of a user should weigh more in the training of that user's personalized filter. However, the inboxes of other users can also provide useful information. For that reason, we performed a weighted aggregation of the inboxes, giving more weight to the inbox of the current user. The inbox of the current user is taken as "foreground" and the other inboxes as "background" data. The weight of foreground data varied according to the dissimilarity of the user's inbox from other inboxes.

In summary the contribution of our TPN^2 method comprises:

- the use of fully-supervised learning to bootstrap positive-only learning on data from a different distribution;
- the weighted aggregation of foreground and background unlabeled data.

The rest of the paper is organized as follows. After presenting in section 2 the algorithms that we used, in section 3 we present the TPN^2 method. Then, in section 4, we empirically evaluate the proposed technique on the two tasks of the ECML-PKDD Discovery Challenge 2006. Finally, we provide our conclusions from this work, together with suggestions for future work in section 5.

2 Description of Existing Algorithms

As explained in section 1, the proposed method combines both fully-supervised and semi-supervised learning. In the fully-supervised stage, the common Naive Bayes algorithm, following the multinomial model was used, while in the semi-supervised stages a version of PNB (Positive Naive Bayes) was combined with PEBL (Positive Example Based Learning) and Roc-SVM (Rocchio – Support Vector Machines). These four algorithms are presented briefly in this section.

2.1 Naive Bayes Multinomial (NBM)

Given a set D of labeled documents, let us denote by PD (respectively ND) the set of positive documents (respectively negative documents) in the set D. Considering bag-of-words representation of the documents, each document is represented as the vector $d = \langle x_t \rangle_{t=1...|V|}$, where $|V|$ the size of the vector of features X_t taking values x_t. Each feature corresponds to a word and the value that it takes in a vector is a function of the number of occurrences of the word in document d. In the simplest case, the feature function indicates only the presence or absence of a word from a document. In that case the document vector contains binary features.

Bayes classifiers assign an unclassified document to the most probable class, using Bayes theorem:

$$\arg\max_j \{p(c_j \mid d)\} = \arg\max \{p(c_j)\, p(d \mid c_j)\} \quad (1)$$

Naive Bayes calculates the required a-posteriori and a-priori probabilities as frequencies on the training data, under the simplifying assumption that the probability of a document given a class can be expressed as the product of the individual probabilities of its feature values x_t given the class. In other words, features are assumed to be independent given the class.

According to [7], there are two models of the Naive Bayes classifier that are mostly used for text classification. These are the multi-variate Bernoulli and the multinomial. Despite its initial use for handling word frequencies, the multinomial model has recently been shown to perform better than the multi-variate Bernoulli even when ignoring frequencies and translating the document vectors into binary ones, e.g. [10] and [8]. For this reason, we have opted for the multinomial model, which elaborates equation 1 as follows:

$$\arg\max_j \left\{ p\left(c_j \mid d\right) \right\} = \arg\max_j \left\{ p\left(c_j\right) p\left(|d|\right) |d|! \prod_{t=1}^{|V|} \frac{p\left(w_t \mid c_j\right)^{x_t}}{x_t!} \right\} \quad (2)$$

where class and word probability estimates were calculated as frequencies on the training data, using Laplace smoothing to avoid zero probabilities. We have tested the method with both types of document vector, i.e., frequencies and binary, and arrived at similar conclusions to what has been reported in the literature, i.e. that binary document vectors lead to better performance. Therefore, we focus on binary vectors in the rest of the paper.

2.2 Naive Bayes Multinomial Positive (Positive Naive Bayes – PNB)

The second algorithm that we used is a representative of the second class of positive-only learning methods mentioned in section 1, i.e. those that estimate the proportion of negative to positive examples in unlabeled data and use that to bias the training process. In particular, we adopt the approach proposed in [2] for PNB (Positive Naive Bayes), using the multinomial model for Naive Bayes.

According to PNB, we assume to be given an estimate $\hat{p}(pos)$ of the positive class probability $p(pos)$, a set *PD* of positive documents together with a set *UD* of unlabeled documents, the Naive Bayes Multinomial Positive classifier classifies a document $d = \langle x_t \rangle_{t=1...|V|}$ as explained in section 2.1, calculating the class probabilities as follows:

$$p(c_0 \equiv pos) = \hat{p}(pos), \; p(c_1 \equiv neg) = 1 - \hat{p}(pos) \quad (\mathbf{3})$$

and the word probability estimates, using Laplace smoothing as follows:

$$p(w_t \mid pos) = \frac{1 + \#(w_t, PD)}{|V| + \#(PD)} \quad (\mathbf{4})$$

$$p(w_t \mid neg) = \frac{\hat{p}(w_t) - p(w_t \mid pos) \cdot pr(pos)}{1 - pr(pos)} \quad (\mathbf{5})$$

where $\hat{p}(w_t) = \frac{\#(w_t, UD)}{\#(UD)}$.

2.3 Positive Example Based Learning (PEBL)

PNB was combined with two positive-only learners belonging in the first class mentioned in section 1, i.e. those that iteratively search for the true negative examples in the unlabeled ones, while maintaining correctly-classified the positive examples. The first of the two algorithms that we tested was PEBL [11], which adopts the following strategy:

1. identify a set of reliable negative documents from the unlabeled set (strong negative);
2. apply a common classifier learning algorithm, such as SVM, on the positive and the strong negative to obtain a classifier;
3. apply the classifier on the unseen data that are not in the strong-negative set;
4. add the new negatives to the strong-negative set and retrain the SVM;
5. stop the iterative process when no new negatives are found by the classifier and consider the remaining examples as positive.

In the first step, i.e. the one that identifies the first set of strong negatives, PEBL uses the 1-DNF method. This method rejects any unlabeled examples that contain words that appear very commonly in the positive examples. Clearly, this is a very strict criterion, but also one that reduces the chances of mislabeling positive examples as negative in the first step.

2.4 Roc-SVM

The second of the two-step algorithms that we used was Roc-SVM [4], which follows a similar approach to PEBL, consisting of two steps: (1) extracting some reliable negative documents from the unlabeled set, (2) applying SVM iteratively to build a classifier.

In the first step Roc-SVM uses a more elaborate method than PEBL, a Rocchio classifier is built on the positive and unlabeled data, assuming that all unlabeled are negative, and then this classifier is applied on the unlabeled data to identify strong negative. The resulting labeled data are used to train the SVM classifier in step 2, following the same iterative procedure as in PEBL.

The second difference of Roc-SVM to PEBL is that it does not trust the final SVM in the iterative process to be the best-trained one, as this classifier is often affected by noise. Instead it chooses either that, or the one produced in the first iteration, depending on how well the final one classifies the positive examples. The SVM produced in the first iteration is often a very good one, due to the way in which the strong negative examples are chosen.

3 Proposed Method

The TPN^2 method addresses the problem of training a classifier in the presence of some labeled and many unlabeled data derived from different distributions. An example of that is the use of labeled public email and personal mailboxes in the Challenge. The method deals with this problem, by training a fully-supervised classifier on the labeled data and using that to select a small number of good positive examples in the unlabeled. These strong positives are used to bootstrap a positive-only learner. An implicit assumption made here is that the positive examples are less different in the labeled and the unlabeled data than the negative ones. This is likely to be the case with spam (positive) vs. ham (negative) emails.

The unlabeled data may also comprise a number of similar but different subsets, such as the mailboxes of different users. In order to make the best use of the unlabeled data, TPN^2 trains a separate classifier for each subset, e.g. each user, using at the same time all unlabeled data. However, it weighs the current subset more, treating that as "foreground" data, while treating the rest of the unlabeled data as "background" data.

Our method consists of four stages: (1) creating a weighted mixture of the different subsets of unlabeled data, (2) training a fully-supervised classifier to select the strongest positive from the unlabeled examples, (3) iteratively extend the positive set with a positive-only learner, and (4) using a two-step positive-only learner to refine the final classifier. The four stages are described in more detail below, while the pseudocode for TPN^2 is presented in table 1.

Stage 1: Weighted mixture of foreground and background unlabeled data
In this stage the set of unlabeled data is weighted. The method focuses on one of the distinct subsets in the dataset, treating that as foreground data and awarding its members with w times more weight than the rest of the unlabeled (background) data.

This is implemented by simply using each example of the foreground data w times, instead of just once. The value of the weight w is user-defined, but we will show a heuristic method for choosing it at the end of this section.

Stage 2: Selection of strong positives from the unlabeled examples
In this stage, fully-supervised learning is used to train a classifier on the labeled data. After experimentation, we chose the Naive Bayes classifier, using the multinomial model (NBM) for this purpose. Once the classifier is trained, it is applied on the unlabeled data, allowing us to choose a small number of strong positive examples. Strong positives are the examples classified as positive by NBM with confidence greater than a user-defined threshold.

Stage 3: Iterative extension of the strong positive set
Given a set of good positive examples and many unlabeled ones, we apply a positive-only learner (PNB) to identify more positive. We assume here that the labeled data have provided more information about the positive class and there are thus very few false positives among the selected set of strong positive examples.[3] Using the initial set of strong positive examples, PNB builds a classifier that is applied on all unlabeled examples. Those examples that are classified as positive will make the new positive set, which is used in turn by PNB to build a new classifier. At the end of stage 3, we will have a positive set containing most of the positive examples of the unlabeled set and very few false positives.

Stage 4: Refine the positive-only trained classifier
Having identified most of the positive examples, we refine the classifier using a different positive-only learner that focuses on finding strong negative examples. We have tested both PEBL and Roc-SVM in that role.

The algorithm presented in table 1, has three parameters that need to be defined by the user:

1. Positive class probability p. This was provided for the challenge and it is p=0.5.
2. Confidence threshold for NBM, above which a positive example is considered strong positive. Given the fact that Naive Bayes tends to push probabilities estimates to 0 and 1, we have opted for a strict value for this threshold, i.e. 0.99.
3. Foreground data weight w. For the selection of this parameter the following heuristic is proposed: Test for increasing values of w and keep the lowest value that leads to the maximum number of identified positive emails in E_i just before entering the final stage. This heuristic pushes the assumption of minimum false positive rate to the extreme.

[3] This was actually proven when we were given the true labels of the challenge data. About half of the messages identified as ham by the initial classifier were false.

Table 1. Pseudocode description of the TPN^2 method.

Input:

- labeled training emails, T
- unlabeled subsets, E_1, …, E_n
- foreground subset, E_i
- foreground weight, w
- positive (spam) class probability, p

Output:

- classifier

Algorithm:

$E := \sum_{j \neq i} E_j + w \cdot E_i$; // ...(1)

```
NBM := construct_NBM (T); // ...(2)
POS := NBM.classify(E); // ...(3)
do {
        POS_OLD := POS;
        U := E – POS; // ...(4)
        PNB := construct_PNB(POS, U, p); // ...(5)
        POS := PNB.classify(E); // ...(6)
} while (POS ≠ POS_OLD); // …(7)
U := E – POS;
PEBL := construct_PEBL(POS, U); // ...(8)
return PEBL;
```

Notes:

…(1) E is a weighted mixture of all unlabeled data (the foreground data E_i is added w times)
…(2) Naive Bayes Multinomial (NBM) learns from T
…(3) NBM is used to extract the strongest positive examples from the unlabeled ones
…(4) remove the strong positives from the unlabeled examples
…(5) train PNB on strong positives and remaining unlabeled, using positive class probability[4]
…(6) use the trained PNB to classify all unlabeled and keep the positive
…(7) continue iteratively, until no more positive can be found
…(8) run PEBL (or Roc-SVM) on positive and remaining unlabeled

[4] In the version of the algorithm that participated in the challenge, we used a more pessimistic estimate of the positive class probability for PNB. The use of p, as shown here, led to considerably better results than all of the reported results in the challenge. We would like to thank the reviewer of the paper for this simplifying suggestion.

4 Experimental Results

4.1 Experimental Set-up

This section uses the Challenge data to study the behavior of TPN^2 under varying conditions and parameter values. In particular we wanted to study:

1. The performance of TPN^2 in the two different tasks of the challenge. Table 2 presents the main properties of the two tasks. In Task A, the size of labeled training data from public corpora is large and so is the size of unlabeled data per user. The aim here is to be able to train a personalized filter from each user's data separately. In contrast, Task B requires the use of information from the unlabeled data of other users. The labeled data is very limited and so is the number of emails available for each user. Presumably, the users share enough common characteristics to be able to utilize unlabeled data from all inboxes when training a personalized filter.
2. The choice of value for the parameter w, i.e. the relative weight of the user's own data (foreground data) to the rest of the unlabeled data (background data). This value is expected to be smaller and closer to 1, the closer the user's data are to the norm.
3. The effect of the various training stages and corresponding algorithms that we used.

Table 2. Number of emails and inboxes for each task of the challenge.

	Task A	Task B
Number of labeled training emails	4000	100
Number of emails within one evaluation inbox	2500	400
Number of inboxes for evaluation	3	15

The data was provided in feature vector format and therefore the use of text analysis or heuristics that are commonly used in spam filtering was not possible. The performance of the methods was assessed by the AUC (Area Under Curve) method, which measures the area under the ROC (Receiver Operating Characteristics) curve. The ROC curve is usually a plot of sensitivity against 1-specifity. In this case it was a plot of the true positive rate (correctly identified spam) against the false positive rate (incorrectly identified spam).

4.2 Choosing the Weight of Foreground Data

In section 3, we presented a heuristic for choosing the value of w, based on the unlabeled data only. This section presents the results of this heuristic (Tables 3 and 4). The tables present the value of w chosen by the heuristic, the optimal, according to the AUC score, choice of w in the range [1,30] if we were given the labels of all unlabeled data, the number of unlabeled examples assigned to the positive class by the method trained with the heuristic w: $|POS|=(TP+FP)$ in E_i just before entering the

final stage, the number of false positive examples: FP in E_i, and the performance of TPN^2 using PEBL with the heuristic and the optimal values of w.

Table 3. Task A performance of the method, using PEBL in stage 4 and setting the value of w with the proposed heuristic vs. the optimal value.

Inbox	w (heur)	w (opt)	|POS|	FP	AUC (heur)	AUC (opt)
task_a_u00	5	27	905	2	0.936654	0.939847
task_a_u01	13	19	992	2	0.948652	0.949384
task_a_u02	21	15	1157	8	0.991288	0.991479
Average					**0.958865**	**0.960237**

Table 4. Task B performance of the method, using PEBL in stage 4 and setting the value of w with the proposed heuristic vs. the optimal value.

Inbox	w (heur)	w (opt)	|POS|	FP	AUC (heur)	AUC (opt)
task_b_u00	1	4	181	2	0.9852	0.9915
task_b_u01	9	25	181	1	0.986375	0.9904
task_b_u02	27	2	183	1	0.9857	0.9876
task_b_u03	1	9	197	16	0.981	0.9945
task_b_u04	1	26	144	9	0.929975	0.94415
task_b_u05	1	1	122	16	0.853175	0.853175
task_b_u06	18	28	122	5	0.87295	0.88445
task_b_u07	1	1	184	4	0.98505	0.98505
task_b_u08	8	5	198	7	0.99365	0.995975
task_b_u09	1	19	187	4	0.976325	0.985975
task_b_u10	1	26	165	13	0.925125	0.970925
task_b_u11	1	20	158	4	0.939425	0.9521
task_b_u12	1	17	188	1	0.9861	0.9921
task_b_u13	1	1	124	5	0.946575	0.946575
task_b_u14	3	3	159	2	0.9404	0.9404
Average					**0.952468**	**0.960992**

The first observation is that the chosen value of w is much more variable in task A, than task B. In task B, in 10 out of the 15 mailboxes the heuristic chooses to give the same weight to foreground and background data. Practically, this means that 10 out of the 15 classifiers in task B are identical. This is an indication of the similarity between the unlabeled data of different users in task B. In contrast, the values of w chosen in task A are high, focusing the training process on the data of the user, rather than the background data.

Another observation is that the choice of w with the heuristic method is quite good in most cases. Although the choices are not so close to the ones we would choose if we were given the labels of unlabeled data, the optimal w does not lead to much better performance.

In order to study the sensitivity of TPN^2 to the choice of w, figure 1 presents the AUC performance using PEBL for varying w in the two tasks. For the sake of comprehensibility, figure 1 presents results for only four indicative datasets of task B.

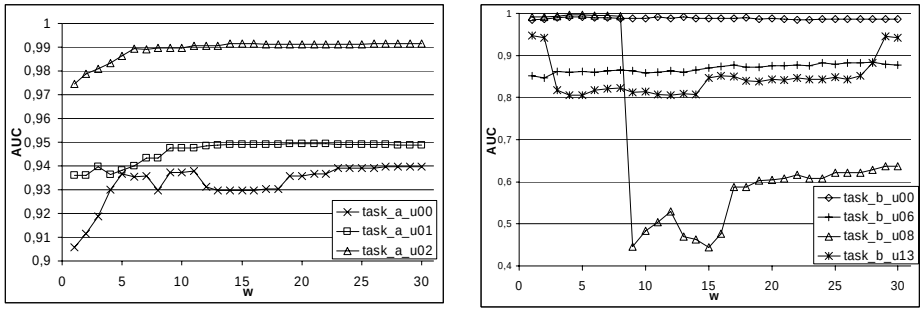


Fig. 1. Performance for inboxes of Task A and Task B varying w.

The figure shows that the method is relatively insensitive to the choice of w in most of the datasets. This is also confirmed by most of task B datasets that are not shown here. Nevertheless, a careful examination of the curves shows that the choice of w is important. One obvious argument for that is the steep and sudden change decrease in performance for dataset task_b_u08 when w changes from 8 to 9. More importantly, the shape of the curves is very different in task A than task B. Choosing a low value for w in task A could hurt the performance seriously.

Finally, it is also not always true that a large value of w is better, as shown by the performance in task_b_u08. Even in task A, where the curves in figure 1 seem to indicate that there is no difference for any big value of w, the performance slightly decreases after the optimal value of w. For w → infinity, which means no influence of the background inboxes ($E=E_i$), we have the following values of AUC for each inbox: 0.895408, 0.909718, and 0.875867, which are much lower than the best we achieved.

Therefore, one needs to choose w carefully, although its exact value can vary without significant loss of performance in most cases. The proposed heuristic works reasonably well, although there could be room for improvement.

4.3 Performance of the Algorithms used in Different Stages

This subsection examines the contribution of each of the three learning stages to the performance of TPN^2. Tables 5 and 6 present the results obtained in each stage for each dataset. All of the results are obtained using the value of w chosen by the heuristic of section 3.

Table 5. Performance in the three learning stages for Task A.

	Stage 2	Stage 3	Stage 4	
Inbox	NBM	PNB	PEBL	Roc-SVM
task_a_u00	0.818971	0.864881	0.936654	0.924884
task_a_u01	0.874001	0.901113	0.948652	0.945581
task_a_u02	0.897548	0.967851	0.991288	0.987226
Average	**0.863507**	**0.911282**	**0.958865**	**0.952564**

Table 6. Performance in the three learning stages for Task B.

	Stage 2	Stage 3	Stage 4	
Inbox	NBM	PNB	PEBL	Roc-SVM
task_b_u00	0.493075	0.948963	0.9852	0.981825
task_b_u01	0.456338	0.952325	0.986375	0.980175
task_b_u02	0.7228	0.954787	0.9857	0.9856
task_b_u03	0.707225	0.984637	0.981	0.980975
task_b_u04	0.77165	0.878	0.929975	0.926025
task_b_u05	0.617887	0.761825	0.853175	0.830475
task_b_u06	0.569925	0.768138	0.87295	0.8687
task_b_u07	0.563175	0.974075	0.98505	0.986025
task_b_u08	0.520763	0.986625	0.99365	0.9943
task_b_u09	0.431412	0.964063	0.976325	0.9776
task_b_u10	0.6655	0.9127	0.925125	0.936025
task_b_u11	0.714988	0.905338	0.939425	0.9368
task_b_u12	0.634975	0.967538	0.9861	0.988375
task_b_u13	0.66315	0.853562	0.946575	0.942425
task_b_u14	0.56955	0.904675	0.9404	0.9378
Average	**0.606828**	**0.914483**	**0.952468**	**0.950208**

As expected, the performance of the fully-supervised classifier (NBM) is much better in task A than task B, since the labeled dataset available in task A is larger. However, with the use of PNB, our method is able to reach approximately the same level of performance in stage 3. Then, the improvement in the fourth stage is essentially the same for both tasks and both of the algorithms that we tested. Thus, the main conclusion is that the proposed method can compensate for the lack of labeled data, by iteratively searching for strong positive examples in the unlabeled data set. Furthermore, the use of a two-step positive-only learner in the last stage is important, when a substantial set of strong positives has been established.

5 Conclusions

In this paper, we introduced the TPN^2 method, which tackles the problem of learning from labeled and unlabeled that are derived from different distributions. The method adopts a four-stage approach combining fully-supervised and positive-only learning methods. The underlying assumption is that the positive examples are more similar in the labeled and unlabeled data than the negative ones. Based on this assumption, the core of the method iteratively selects strong positive examples from the unlabeled data, starting from the ones most confidently identified by a classifier trained on the labeled data. Furthermore, the method handles unlabeled data comprising of different subsets. In that case, the method builds a separate classifier for each subset, using the whole set of unlabeled data, but focusing more on the current subset.

The proposed method participated in the ECML/PKDD Discovery Challenge 2006, the subject of which was the construction of personalized spam filters. The challenge

defined two tasks, in which a set of public email data was given as labeled and a number of personal inboxes as unlabeled data. The two tasks posed a different proportion of labeled and unlabeled data, as well as a different number of personal inboxes. The proposed method obtained the second place in both tasks, as it is particularly suitable for the scenario of the challenge, i.e. spam (positive) email is more homogeneous in the two datasets (public and private) than ham (negative) email.

The paper contains a selection of the results obtained in the various experiments with the parameters of the method, focusing particularly on the choice of algorithms for the four stages of the method and the choice of weight for the foreground data. Despite the good results, a number of extensions seem interesting, such as the use of different weights for different subsets of the unlabeled background data. Additionally, a different configuration of the positive-only learners could be used to reduce the risk of error amplification by the iterative use of the same search bias. Finally, the method should be tested on other problems, which may violate its underlying assumptions.

References

1. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. Proceedings of the Workshop on Computational Learning Theory, COLT-98, (1998) 92-100.
2. Denis, F., Gilleron, R., Tommasi, M.: Text classification from positive and unlabeled examples. Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU, (2002).
3. Joachims, T.: Transductive inference for text classification using support vector machines. Proceedings of ICML-99, 16th International Conference on Machine Learning, (1999) 200-209.
4. Li, X., Liu, B.: Learning to classify text using positive and unlabeled data. Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI-03, (2003).
5. Liu B., Dai Y., Li X., Lee W., Yu P.: Building text classifiers using positive and unlabeled examples. Proceedings of the Third IEEE International Conference on Data Mining, ICDM-03, (2003).
6. Liu, B., Lee, W. S., Yu, P., Li, X.: Partially supervised classification of text documents. Proceedings of the Nineteenth International Conference on Machine Learning, ICML-02, (2002).
7. McCallum, A., Nigam, K.: A comparison of event models for naïve Bayes text classification. AAAI-98 Workshop on Learning for Text Categorization, (1998).
8. Metsis V., Androutsopoulos I., Paliouras G.: Spam Filtering with Naive Bayes - Which Naive Bayes?. Proceedings of the 3rd Conference on Email and Anti-Spam, CEAS-06, (2006).
9. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning, 39(2/3), (2000) 103-134.
10. Schneider, K.-M.: On Word Frequency Information and Negative Evidence in Naive Bayes Text Classification. España for Natural Language Processing, EsTAL, (2004).
11. Yu, H., Han, J., Chang, K.: PEBL: Positive example based learning for Web page classification using SVM. Proc. ACM SIGKDD International Conference on Knowledge Discovery in Databases, KDD-02, (2002).