Using old Spam and Ham Samples to Train Email Filters

Jose-Marcio Martins da Cruz
Centre de Calcul et Systemes d'Information
Ecole des Mines de Paris
Paris, France
Jose-Marcio.Martins@mines-paristech.fr

Gordon V. Cormack
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
gvcormac@uwaterloo.ca

ABSTRACT

Email spam filters are commonly trained on a sample of recent spam and ham (non-spam) messages. We investigate the effect on filter performance of using samples of spam and ham messages sent months before those to be filtered. Our results show that filter performance deteriorates with the overall age of spam and ham samples, but at different rates. Spam and ham samples of different ages may be mixed to advantage, provided temporal cues are elided.

1. INTRODUCTION

Spam filters are commonly trained on historical collections of messages, each labeled as spam or ham (non-spam). Their theory of operation assumes that these training messages are a random sample of those to be filtered; an assumption that is clearly not true because, when the filter is trained, the set of messages to be filtered exists only in the future. It is known that future messages are best approximated by recent messages [5].

Under the assumption that that models built with recent samples best represent what future objects will be, models should be continuously updated, including new samples and forgetting old ones. Over time, models may become inadequate for several reasons : class priors can change over time or underlining changes can happen on the nature of objects being studied.

However, acquiring and labeling recent messages may be impractical, and they may not be plentiful enough for adequate training. Sometimes, it may be more practical to acquire recent examples of one class than the other; for example, spam from a spam trap or ham from the client interface. Also, when filtering messages for a group of recipients, it may be difficult to continuously update the ham data set and it could be desirable to train the filter with a convenient initial data set of hams and spams and, over time, to preserve the initial ham data set and renew only the spam data set.

In this paper we present some experiments we've done in order to understand what happens to the effectiveness of a filter being used in conditions which aren't ideal: less-than recent training samples or samples having different ages. Also, we're more interested in understanding qualitative behaviours than absolute values of effectiveness, as the latter depends heavily on the kind of classifier being used and on the context, than in proposing solutions to overcome concept drift.

This paper is organized as follows. Section 2 recalls some relevant research on how to update models used by classifiers to solve the concept drift problem in spam filters. Section 3 presents the objectives of our this paper. Section 4 presents the environment in which we've done our experiments. In section 5 we describe our experiments and present results. And finally sections 6 and 7 presents some discussion and conclusions.

CEAS 2009 - Sixth Conference on Email and Anti-Spam July 16-17, 2009, Mountain View, California USA
.

2. RELATED WORK

It's usually accepted that the characteristics of email change over time [12]. While it can be assumed that people don't change frequently the way they write legitimate messages (hams), spam evolves for different reasons. The amount of spam changes to reflect spam activity[12], so class priors change accordingly. Spam filtering can be seen as an adversarial game [18] where the strategy of each part changes over the time : spam content change as spammers want to deceive spam filters (generating false negatives) and spam filters change to adapt to the changes in spam content.

It has been shown that spam filter evaluation is more accurate when using an on-line model, where messages are submitted in chronological order, than with a batch model where order is irrelevant [5]. This reflects the time dependency of email.

Changes can be in class (target) priors, which may be simpler to solve, as this change may be observed directly. Changes may appear in hidden concepts [22], which may be much harder to detect and may even be confounded with noise, as these changes can't be correlated to some directly measurable parameter. Changes in these hidden concepts is usually refered as *concept drift* [22].

Machine learning techniques are applied to many domains handling non-stationary streams of data. Research in these domains is very active and probably even more than in the spam filtering domain. Data Mining is intended to identify and acquire information from streams of data [14] or even simply to detect when changes occur and the speed of change [1][3]. Clustering [2] and Online Classification [23] of non-stationary data streams are domains nearer our spam filtering problem.

The canonical solution implies all past relevant samples shall be available and some method which will integrate new samples to the current model and forget the most old or less usefull ones. The two hard points of this approach are the storage space needed to save past samples and the heuristic used to select which old samples can be removed, for which the computational cost may be high. This can be even harder when the amount of data being handled is large as well as the rate at which changes occur.

In the literature, techniques employed to handle concept drift (and to update the models) are commonly studied in categories specific to the domains or the kind of classifiers to which they will be applied. We consider two categories, based on how they are applied to spam filters.

The first kind of solution, which we will refer as *instance-based update,* are those requiring that old samples, or at least a summary of each sample shall be available when the model will be updated. Forgetting old samples consists in removing them from the training data set. A trivial example of this kind of solution is the use of a fixed size sliding time window over some time just before the current date.

The second type of solution, which we will refer as *incremental update*, are those for which each new sample is used to update the parameters of the model and is discard just after. This is the kind of solution usually found in spam classifiers, even if not all classifiers are well suited for this.

Although, in theory, all new examples should be used to update the model, in practice it's common that only some examples are used. Most of the time it's impossible to have the correct label for each example, and sometimes it shall be considered that this feedback will be available for only one class. This happens in both situations described below.

2.1 Instance-based update

The first approach raises naturally from instance-based classifiers (sometimes referred to as "lazy learners"), where all examples may explicitly be used during the classification process [22] or to update the model when it was detected that the model changed. Within this approach, the learning process maintains all samples (or a summary of them) inside a time window of fixed or variable size. The learning process evolves moving the window forward, adding new messages to the front and removing old ones from the tail. Although this approach isn't limited to *instance-based classifiers*, we will refer to it as *instance-based training*, as it shall store and individually access each sample in order to remove older ones from the training set.

Using this approach, Cunningham [10] retrain a nearest neighbour classifier when the accuracy falls below some predefined level. Fernandes-Riverola et all [13] use feature selection to select which samples to remove or to maintain and to update the window size. Hsiao [17] detects changes inside clusters to decide when to update them. Delany et all [11] uses a case based classifier (lazy learner) : only misclassified messages are added to the data set and a periodic retraining is done to remove less relevant samples.

One advantage usually mentioned by defenders of *instance-based training* approach is the ability to detect and adapt to local changes inside classes. On the other hand, the storage place needed to save all examples may be important. Also, if adding new samples may be trivial, deciding which old samples can be removed may not be the same, mainly on large training sets.

2.2 Incremental update

In the opposite approach, incremental *update*, samples are presented sequentially for training and are discarded just after use. The goal is to eliminate the need of saving all past examples.

When doing active learning, the classifier is allowed to ask the label a sub set of the messages submited to the classifier. Messages for which the real label will be asked may be choosen randonly or based on some criteria, e.g., messages for which the assigned score is close to the classification threshold [21][20].

Most open-source filters use some variant of "Train on Errors", "Train Until No Errors" or "Train on Everything". Bogofilter[19], an open source "Bayesian" spam filter, expires features (not examples containing these features) which weren't seen after some time. It's not clear that models updated with these approaches don't degenerate after some time, as it's not to demonstrated that these approaches converge to the real models. Sculley[20]showed that some classifiers, which model is updated with the "Train Until No Error" approach, present over-fitting and may be easily broken by noise.

Goodman and Yih [15] use gradient descent in an adaptive logistic regression classifier to do sequential learning, eliminating some drawbacks of previous solutions.

2.3 Effects of concept drift

Although there has been many research work to find efficient ways to solve the concept drift problem, at our knowledge, very few research were done to evaluate the consequences of the drift itself, when models aren't updated. Some limited results can be found in [11], but the methodology seems specific to the kind of classifiers being evaluated.

Examples of questions which remain without answer are : how does concept drift affects classification errors, at which class drifts are more important or which characteristics of messages, other than the content itself, can mitigate these effects.

3. OBJECTIVES

The global objective of this research is to investigate the influence of concept drift in spam filter effectiveness. So, we're interested to verify if the spams class changes more than the ham class, if results from the two approaches to handle concept drift result in completely different effectiveness results. A secondary objective is to understand at what extent a spam filter can be used, without being updated, to filter recent messages with an "acceptable" effectiveness.

The first two series of experiments we've done try to simulate, at some extent, the two kind of solutions, described in the previous section, to update the email stream models used. With these experiments we will also try to identify some points which can help to mitigate the effect of concept drift in spam filtering. The last series of experiments was done to investigate at which part of email (headers or body), concept drift is more important. These objectives are described below.

- *Temporal References* - temporal references are always present in email messages : most of them inside headers and sometimes in the body. Cormack and Lynam [8] suggested how some of them appear inside messages. In fact, the big picture here is the identification of features which will deviate the classifier from the originally intended targets : *"old and recent messages"* instead of *"hams and spams"*. If their effect isn't negligeable, it become necessary to identify these features and elide them to minimize their influence on the remaining experiments. This is something we discovered during our preliminary experiments and we decide to included them in the whole picture, given its importance.

- *Influence of age and relative age of examples* - effectiveness of mail filtering is supposed to degenerate with the age of examples used to train the filter. With this series of experiments we examine the situation where, after an initial training, the filter is employed during some time without updating the training data. This situation is compared to one side training with replacement : only one class of training data sets is updated using a constant size time window (recent messages are added while old ones are removed) and is equivalent to *instance-based update*, described above.

- *Incremental update* - Investigate the effect of samples age when doing one-side or both sides incremental update. While in the previous objective old messages are replaced by new ones, we'll just add new messages to the training set. We're interested in the comparison of situations where both classes are updated, only one is updated and none of them are updated.

- *Whole message, headers and body* - The information present in the headers and in the body of messages aren't of the same nature. Information inside headers are mostly related to the way the message was created and to its path from the sender to the recipient. The body of the message contains, most of the time, the real message payload, but may also include some meta-information, like attached files, or HTML code. Given this difference, it's interesting to investigate if the influence of the age of samples is the same in both parts.

4. EXPERIMENTAL SET UP

4.1 Data Sets

To realize these experiments we used two private corpus of messages.

The first data set (MrX) contains 160,000 messages (8017 hams; 151.983 spams) addressed to one email recipient over the course of about eight months. This data set was mainly used as a guide and to confirm behaviours found in the second data set. We split the messages by delivery date into 8 equal sets, numbered from the most recent (0) to oldest (7), each representing about a month. 0 was used as the test set. MrX is the private corpus used at TREC experiments.

The second data set (MrJ) contains messages addressed to another different recipient over the course of 11 months (exactly 330 days). Messages from the last 30 days (22645 messages : 17250 spams and 5395 hams) was used as test set and numbered as set 0. Messages from the first 300 days (184539 messages : 118275 spams and 66264 hams) were split in 20 equal sets covering 15 days each and used to train the filter and numbered from 1 to 20.

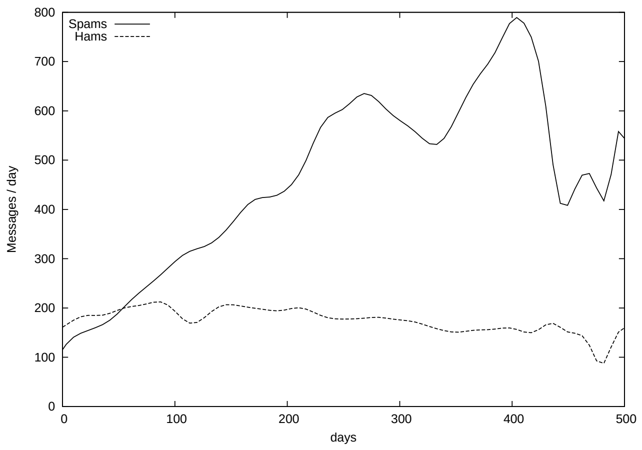

Figure 1: Count of hams and spams a day in MrJ corpus. This work tooks messages from day 0 to 330.

Figure 1 presents the number of messages (spams and hams) each day. Messages utilised in this experiment are those from day 0 to day 330. The abrupt drop in the number of spams corresponds to the the McColo shutdown [1]. This figure already presents how the kind of change, over the time, we can expect in amount of messages : and so deduce how class priors can change.

MrX and MrJ corpus come from different continents, from countries with different official languages (english for MrX and french for MrJ). Also, the ratio ham/spam isn't the same in both corpus. MrJ subscribes to a lot of newsletters and discussion lists.

4.2 Manipulating messages

For these experiments, we used an elementary window of size 1 month. While MrX corpus was splited in 1 month size sets, MrJ was splited in sets of size 15 days : 2 sets shall be combined to create a month of messages, but time references can be moved in 15 days steps instead of a month, allowing us to gather results in more intermediate points.

The basic window size (1 month) was choosen empirically. This size should be big enough so the classifier effectiveness will be not too far from its nominal value. On the other hand, when using small sizes results may be cluttered by high frequency noise and when using big sizes results can be oversmooth, hiding interesting behaviours.

The test set is always the same - the most recent month : set 0 for both MrX and MrJ corpora. Doing our experiments this way allowed us to compare results evaluated on the same test set, when experimental parameters are changed. In a more understandable way, we can say : *"instead of investigating what could happen in the future if we do this now, we're investigating what could it be now if we've done that in the past"*.

In each run, the most recent month just before the test set.- is used as baseline training set, corresponding to new ham and new spam. Remaining older sets are used to measure the effect of using progressively older training examples. The way they are combined in the training set depends on the experiment being done.

[1]McColo - http://www.washingtonpost.com/wp-dyn/content/article/2008/11/12/AR2008111200658.html

Messages were submitted, for training and classification, in the same chronological order they were delivered to the recipient mailbox.

4.3 Classifiers

All experiments were done with an adaptive logistic regression filter [15]. Tokenization uses character 4-grams features extracted from the first 3500 characters of messages. This is the same filter which participated at TREC 2007 spam track [4], but it was modified to take into account the unbalance between the amount of hams and spams. This is a soft classifier which outputs a real number score (the odds of being a spam). This score is used to draw the ROC (Receiver Operating Characteristic) of the classifier. Spam/ham classification is done using the neutral value as threshold.

4.4 Evaluation methodology and effectiveness figures

We basically employed the same methodology of TREC Spam Tracks [9], with its companion toolkit[7].

The TREC toolkit was designed to evaluate the effectiveness of a filter on a single run. In our research, we basically needed to evaluate and compare multiple runs of the classifier applied to the same test set but with different training sets. Training sets are built from the same corpus of messages after some temporal criteria (start and end time). The logic behind this criteria can be different for each class and for each experiment. Results from each run shall be saved in such a way that it can be feed into the TREC evaluation toolbox [7]. All this logic can be easily programmed and a controller (a perl script) was developed to manage each experiment, in order to assure homogeneity and reproducibility of experiments.

TREC toolkit outputs a number of figures of merit allowing one to evaluate and compare classifiers. Two metrics usually employed to evaluate filters are the spam and ham misclassification rates (*smr* and *hmr*, respectively) [9], which are the fraction of spam and hams that are misclassified. Both values are interesting, but they are specific to each class and allows us to evaluate the filter at some specific situation (at some predefined threshold). Logistic Average Misclassification (LAM), is the mean of this values evaluated using the log odds of misclassification rates, bit it's also tied to threshold used to evaluate *hmr* and *smr*.

The main effectiveness metrics which interests us is the 1-AUC% (Area Under Curve) [9], as it gives a global measure of the classifier effectiveness independently of some predefined threshold. This metrics is derived from ROC (Receiver Operating Characteristics). The smaller the value of 1-AUC, the better the classifier effectiveness. Typical values for modern spam classifiers are inferior to 0.1 %.

Although we're using the 1-AUC metrics to globally evaluate the classifier, *hmr* and *smr* are control parameters we use to verify that, for each class, the specific filter effectiveness isn't deviating too much from its operating point. Other than *hmr* and *smr*, the TREC toolkit outputs the confidence intervals for each metrics, allowing us to evaluate how good the value is.

5. EXPERIMENTS AND RESULTS

To achieve our objectives, we organized our experiments in three parts. The first one is intended to investigate the effect of using old samples to classify recent messages and to identify and evaluate the effect of temporal references found in messages. The second part is intended to examinate the effectiveness of incrementally updating the training data. The last part is intended to verify which part of messages are more sensitive to the age of samples used to classify recent messages.

Table 1: Temporal cues of the sort illustrated here were identified and elided using the spam filter to distinguish new messages from old, instead of spam from ham.

cue	example
header date	**Mon, 4 Dec 2006 13:21:34**
reply date	On Tue, 31 Mar 2009, Joe Denver wrote:
daylight time	**-0400 (EDT)**
server hostname	by **mail1**.institution.net
server config.	**(8.13.1/8.13.1)**
generated ID	**01C7178F**.000D1CD0
seasonal reference	thanks for making **2006** a great year

5.1 Effect of age and of relative age of hams and spams samples

The effect of samples age and samples relative age was investigated with three series of experiments, reproducing, at some extent, the *instance-based* training, with a fixed size sliding window being used to select samples to train the classifier.

1. *Old Ham - Old Spam* : The most recent month of sample messages (sets 1 and 2 of MrJ corpus and set 1 of MrX corpus) was used as baseline as training set to classify messages of the test set. This experiment was repeated progressively aging the samples of both classes, in steps of 15 days for MrJ and 1 month for MrX (using a sliding window of size one month). This experiment approximates the situation where the filter is initially trained and used to classify messages during some time without retraining.

2. *Old ham - New spam* : This series of experiments is similar to the above one, but instead of aging both classes, the spam class is held constant (recent samples) and the ham class is progressively aged. This is the situation where the filter is initially trained with hams and spams and and, during some time, only *spam* samples are renewed (with replacement).

3. *New Ham - Old spam* : This series of experiments is similar to the above one, but aging the spam class instead of the ham class.

Preliminary experiments [6] with MrX corpus presented two unexpected behaviours : the filter error (1-AUC%) increased faster when aging only one class (no matter which one) than when aging both classes (figure 2) and the error rate presented an unexpected discontinuity around the year boundary (figure3), exploding after that point.

Figure 2 shows the filter error, expressed as $1 - AUC$ (the area above the receiver operating characteristic curve), for all combinations of training sets. Baseline error is $1 - AUC = 0.02\%$, increasing to 0.2% and 0.1% for sets 6 and 7, respectively. Substituting new ham or new spam substantially degrades performance, a result that is on the surface surprising as the average age of the training examples is

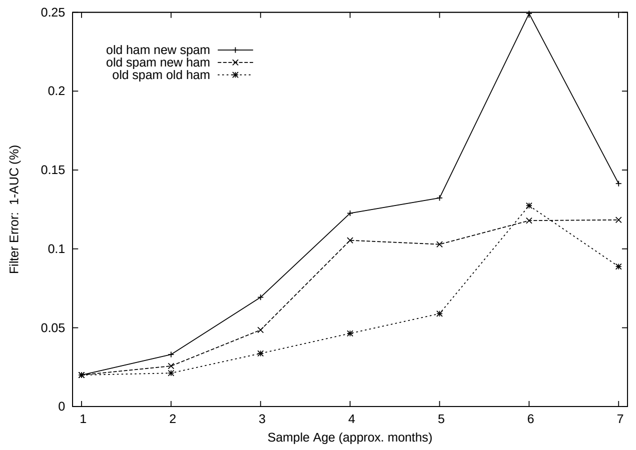

Figure 2: Effect of old and new training samples on filter error (MrX). The origin of each curve represents training on the most recent ham and spam available. The three curves represent: substituting progressively older ham, progressively older spam, and progressively older ham and spam of the same age.

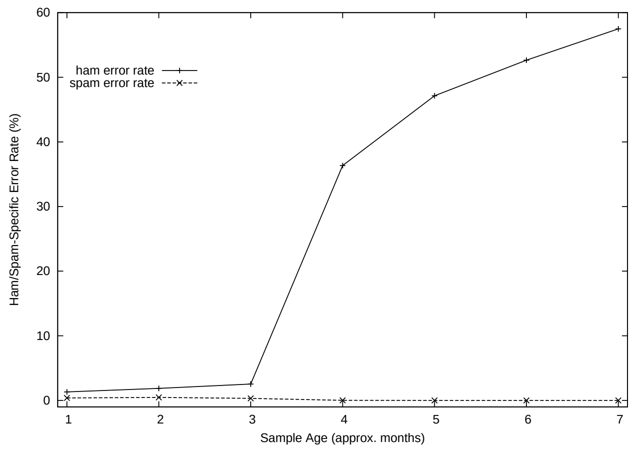

Figure 3: Separate ham and spam error rates training with progressively older ham and new spam. Spam error rate vanishes while ham error rate increases dramatically, even for 1- and 2-month-old ham.

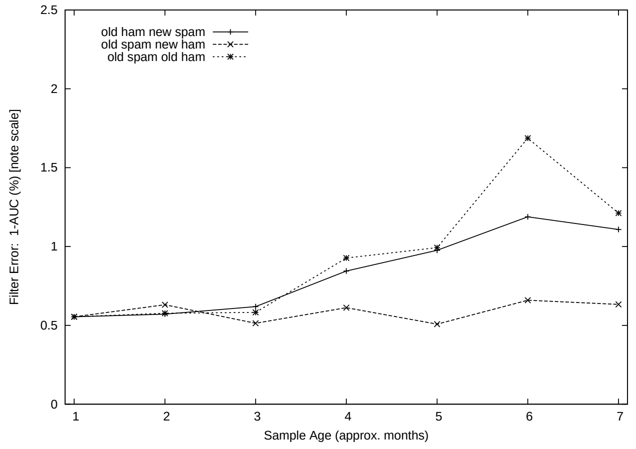

Figure 4: Effect of removing email headers (MrX). Overall error is increased tenfold but the effect of age disparity between training examples disappears.

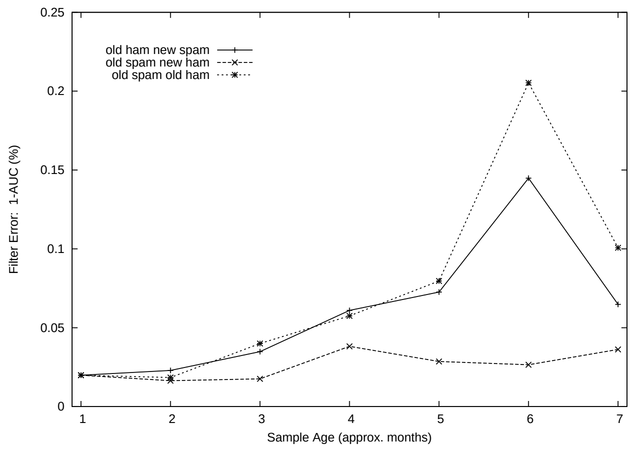

Figure 5: Effect of eliding features to mitigate temporal effects (MrX). Effectiveness on new training sample is restored to that of Figure 2 while the effect of age disparity disappears.

decreased. Figure 3 provides further insight into this phenomenon: as progressively older ham is combined with new spam, the ham error rate explodes, while the spam error rate vanishes. The complementary effect (not shown) is observed when older spam is combined with new ham. The filter is learning to recognize new messages, not spam.

We posit that the features used to recognize new messages are contained largely in the message headers, which contain explicit timestamp information. The results obtained from removing the headers altogether, shown in figure 4, support this theory by virtue of the fact that the mixtures of new and older training messages outperform strictly older messages. But overall performance is degraded by nearly a factor of ten. Clearly the header is of critical importance to the filter and removing it is not a step toward improved effectiveness.

We therefore investigate the approach of eliding only date-specific information in the header. Eliding explicit dates alone, as shown in the first line of table 1, yields no measurable benefit. But when the other cues shown in 1 are elided, filter effectiveness on new training data is as good as the baseline and on mixed-age training data is improved dramatically (figure 5). In particular, old spam and new ham works nearly as well as new spam and new ham, and much better than old spam and old ham.

The temporal cues were discovered with the aid of the spam filter itself, trained to classify messages as *new* (belonging to set 1) or *old* (belonging to set 7) rather than as spam or ham. Once the most discriminative features were identified, it was not difficult to write ad hoc scripts to eliminate them from the header. Table 1 is a complete list of the sorts of cues we found: inappropriate use of daylight saving time, server hostnames and software that were reconfigured over time, timestamp-derived message IDs and MIME delimiters and dates found in the body of replies.

We repeated these experiments with MrJ messages using smaller time steps (15 days instead of 1 month). Previous temporal cues found in MrX corpus were removed, but we found another temporal cue : dates in the body of replies (*"On Tue, 31 Mar 2009, Joe Denver wrote:"*).

Results of these series of runs are shown in figures 6, 7 and 8. Global error rate (1-AUC% - figure 6) degenerates

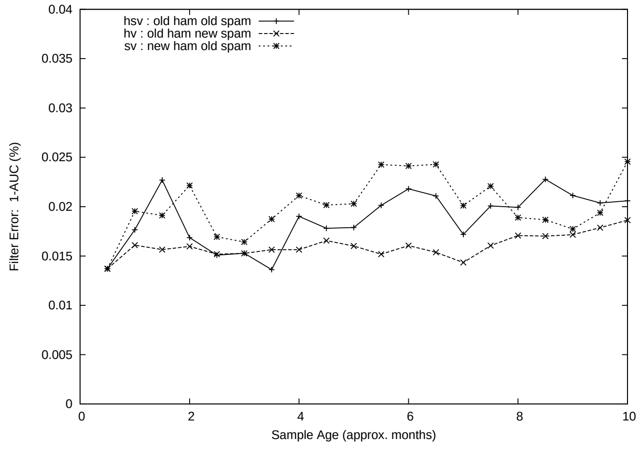

Figure 6: Effect of mixing old and new training samples on filter error (MrJ)

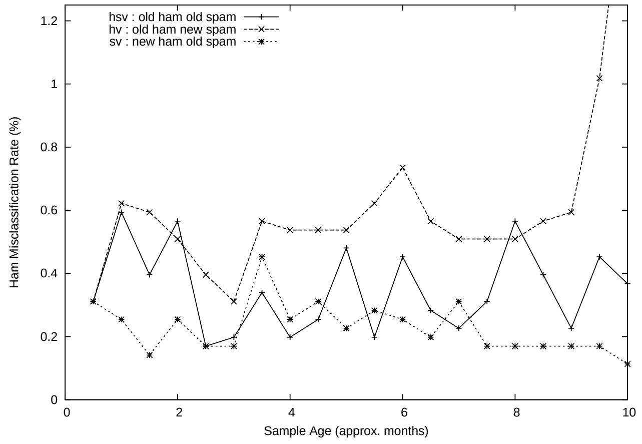

Figure 7: Effect of mixing old and new training samples on Ham Misclassification Rate (MrJ)

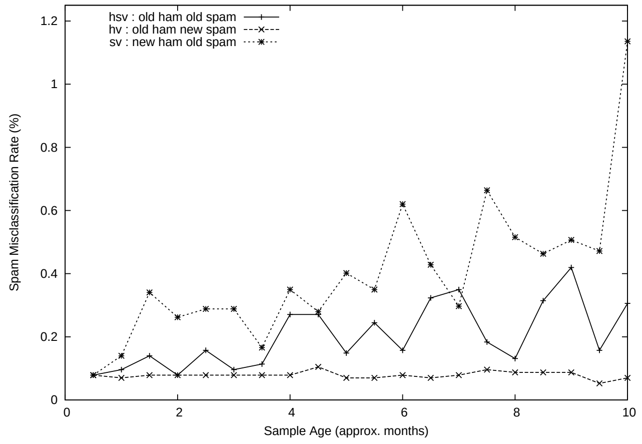

Figure 8: Effect of mixing old and new training samples on Spam Misclassification Rate (MrJ)

from ˜ 0.015 % to 0.025% with 0.95% confidence interval almost constant with lower and higher limits at 0.005% and 0.05%. This loss is much less important than for MrX corpus. When aging only one class, the misclassification rate (figures 7 and 8) degenerates in the class being aged. Interpreting this result as *instance-based update* gives : if one retrain only one class, the misclassification rate of the opposite class degenerates.

However, global error (1-AUC%) results from MrX and MrJ corpus present one qualitative difference : for MrX corpus aging both hams and spams is the worst situation (figure 5), while for MrJ, old spam with new ham is worse during a large time interval (figure 6). One hypothesis to explain this is the possible existence of some residual temporal reference, which can merely be virtual instead of quite explicit references as found before. An example of this are spams mentioning events widely known and appearing only during some particular dates, such as the recent U.S. presidential election[2] being used as the object of a spam campaign. As long as the effect of this references on the filter effectiveness is limited, it may be difficult to identify and remove them. We shall accept that completely removing all temporal references may be unrealistic and this may be integrated into results as a noise source.

Another abnormal behaviour we can notice is the abrupt increase in the ham misclassification rate when only the spam class is aged (figure 7). Investigating the reason of the errors in the last run, we've found that 26 of the 60 misclassified messages come from a number or newsletters (New York Times, CBS, CNN and Foxnews). This set corresponds to the month when the recipient subscribed to those newsletters, and these newsletters are atypical with respect to other messages in MrJ corpus. If these errors are discarded, the error rate falls down to a more plausible value. This is another kind of virtual temporal reference which can appear on corpus of messages.

5.2 Incremental update

The goal of this series of experiments is to investigate the effect of incremental update : the classifier is initially trained with recent messages of both classes and used, during some time, to classify messages. During this period it is incrementally updated. We ran four series of experiments :

1. *No training* - This is the baseline to be compared with the three other in this series. The classifier is initially trained with one month of recent messages and used to classify recent messages for some months without training. This is the same as the "old ham - old spam" experiment above.

2. *Full training* - This is the same as above, but the classifier training data is incrementally updated (without removing old messages) with data from both classes. The size of the sliding windows grow with time - pushing back the start date and helding constant the ending date.

3. *One side spam training* - only spams are used to incrementally update the classifier training data - the sliding window of spam class is updated as above, and

[2]http://www.networkworld.com/news/2008/110508-obama-spam.html?fsrc=netflash-rss

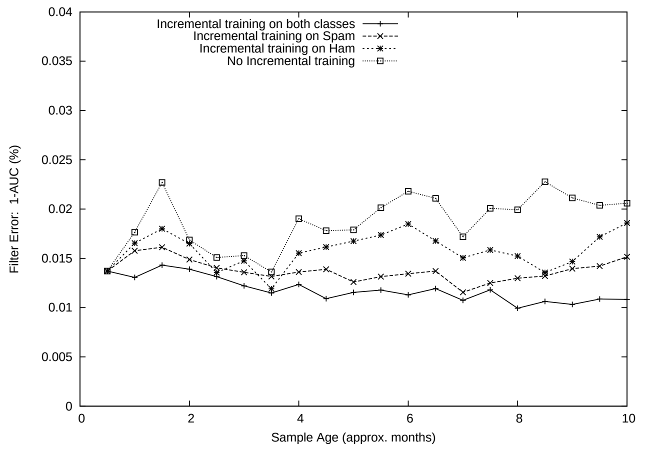

Figure 9: Effect of incremental update on the filter error

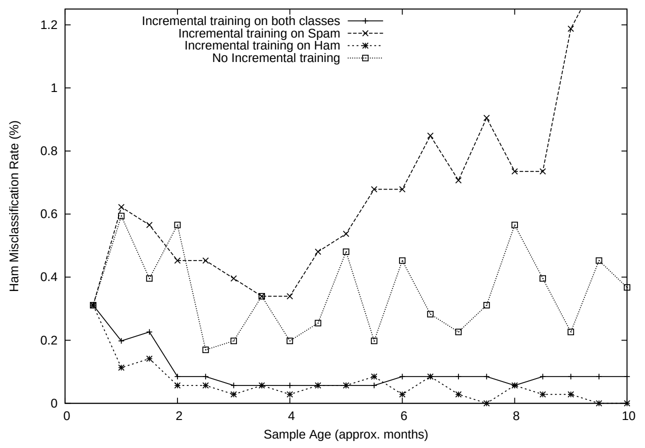

Figure 10: Effect of incremental update on ham misclassification rate - one-side training on spams is the less favorable

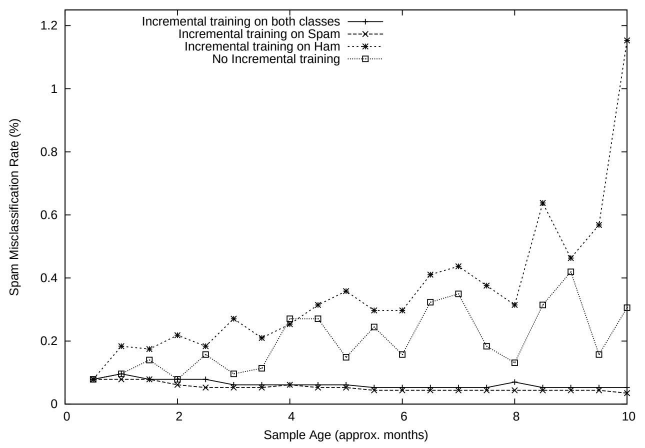

Figure 11: Effect of incremental update on spam misclassification rate - one-side training on hams is the less favorable

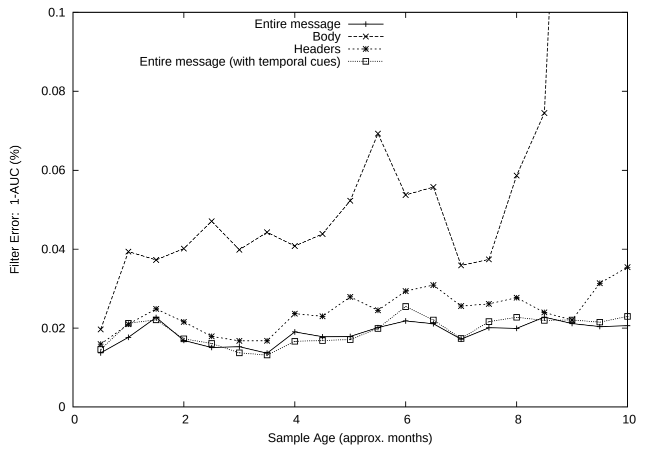

Figure 12: Effect on filter effectiveness when using the whole message, headers only or body only. Effectiveness is worse when using body only

the sliding window of ham class is moved back with constant size.

4. *One side ham training* - only hams are used to incrementally update the classifier training data - same as above, but classes are interchanged.

This experiment is similar to the previous one. But model update is done simply by adding new samples, without removing old ones as it was done before.

Global filter error results (1-AUC%), displayed in figure 9, show that filter effectiveness is better when incremental update is done on both classes than when done on a single class, which is better than no training at all. But one-side updating on spam class is better on ham class. The expected result for specific class errors (figures 10 and 11) is that incremental update on both classes improve error rate on both classes. On the other hand, one side training improves the error rate of the class being trained but degenerates that one of the other class.

5.3 Whole message, headers only or body only

To investigate for which parts of messages, their age has stronger influence on filter effectiveness, we simply run a series of experiments, like the one above to investigate the effect of message age, but on four types of objects : original whole message without removing temporal cues, whole messages with temporal cues removed, headers only and body only.

Filter errors, expressed as 1-AUC% (figure 12) shows that using both body and headers is the situation where the filter is most effective, using headers only comes just behind. And using body only is the very worst.

This result suggests that message body changes faster than headers. The reason is out of the scope of this paper, but a possible explanation is related to their nature. Message body is the container of useful information, while headers is the container for metadata, and most of them are related to message relaying - the way the message was created and distributed. Although spammers change the visual and semantics of messages to elude content filters, the way these messages are usually distributed (network of botnets) doesn't change too much. This is an interesting point to be investigated in a separate research.

When the classifier uses only the body of messages, results are much worse for MrX corpus (figure 4 : 1-AUC% jumps up from 0.6% to 1.7%) than for MrJ corpus (figure 12 : 1-AUC% jumps up from 0.02% to 0.1%). Using recent messages, MrX global error (1-AUC%) increases from 0.02% to 0.6%, while global error for MrJ increases just from 0.015% to 0.02%. The explanation we found, which needs to be confirmed with more research, come from the fact that dominant language in MrX ham corpus is english, while it's a mix of english and french for MrJ. Most of the spams are in english, so it's possible that the dominant language in legitimate messages is relevant to the filter efficiency when using only the body contents for classification. Headers content is neutral with a very weak dependency on the language used in the body of the message. Also, this is a vast research subject out of the scope of this paper.

6. DISCUSSION

Although this research is about temporal behaviour of email classification, all along this work we've implicitly supposed that the message generation is, at some extent, a stationary random process, which obviously isn't true. So, we shall try to understand the limitations of our work.

The first point of importance is the duration of our experiment : between 8 and 12 months. We've shown that, during the dates of our experiments it was plausible to utilise old samples to classify recent messages, provided some necessary preprocessing was done (removal of temporal references). In other situations, this period may be longer or shorter, provided qualitative changes in the nature of messages are limited. The origin of these changes may be internal (e.g., organizational) or external (a big spammer ceasing its activity).

Another point of interest is the size of the sliding window used in our experiments : one month, choosen empirically. It shall be big enough to make the classifier work near the region of optimal effectiveness. On the other hand, its size shouldn't be nor too small (to avoid spurious high frequency noise) nor too big (to avoid oversmooth hiding interesting behaviours).

In MrJ corpus, we can see that the ratio ham/spam increased from, roughly 1:1 to 1:8 (figure 1) during the period of our experiment. To compensate this, we had two options : using a variable size window or using the natural capability of the classifier to compensate unbalanced class priors. The latter seemed less complex to implement and there is no evidence that the former was more realistic.

We've done all experiments only with a logistic regression classifier, even if other kind of classifiers were available. Different classifiers have different learning rates, so it's possible that the dimension time may be adapted using some scale factor. Also, it seems to us that generative classifiers like Naive Bayes may behave differently. It may be interesting to redo all experiments we've done with other kind of classifiers.

Another interesting research direction concerns a corpus with samples from many recipients instead of a single one. This is a step forward sharing classifiers models to filter spams for a group of users.

The natural choice for experimenting with a group of recipients could be the Enron data created by TREC spam track. We've done some preliminary experiments with it, but it seemed to us that although this corpus was fine for evaluation of classifiers, it was quite atypical for the evaluation of the temporal effects we were interested. In particular, an important virtual temporal reference appears in messages from October 2001 (see table 2), represented, at the same moment, by the bankrupt and the huge increase in the amount of messages.

Table 2: Distribution of messages in Enron Corpus

	# hams	# spams
Aug 2001	1881	5055
Sep 2001	2630	4550
Oct 2001	10413	3820
Nov 2001	8359	4419
Dec 2001	3778	3731
Jan 2002	5687	4392

7. CONCLUSIONS

It has been widely assumed that caracteristics of email, and mainly spam, change substantially over time and spam filters shall absolutely be continuously retrained. It's intuitive to accept that people doesn't usually change the way they compose messages, nor the vocabulary employed, and that spams change both from visual and semantics point of view.

The use of old training data degrades performance, but not nearly so much as the use of raw training data in which the ham and spam have different ages.

If age cues are removed, training data of mixed age may provide improved performance in the situation where only new ham or new spam is available. Header removal is too radical as it dramatically compromises overall performance. If a few tell-tale temporal cues are identified and elided, substituting newer training data for one class of messages appears to yield improved effectiveness over using old for both.

Our approach to identifying the training cues was not entirely automatic, and not entirely blind to the training data (but definitely blind to the test data). We believe it is a good candidate to be automated. And even if effected manually, it is much more efficient than labeling a new training set. The cues we discovered closely match those mentioned by the authors of the TREC 2005 Spam Corpus [8].

Experiments with MrX corpus (figure 5) had shown that, for this particular corpus, using old ham is worse than using old spam, while the same experiment with MrJ corpus (figure 6) doesn't present a so noticeable difference. On the other hand, for both corpora, when both hams and spams are aged together, effectiveness doesn't change too much.

We can't generalize our results, as we tested only one kind of classifier (logistic regression), and only two corpora (MrX and MrJ), it seems that, in some circonstances, provided some conditions are met, it may be possible to have a filter trained with not so recent messages for which the effectiveness remains acceptable. So, this widely accepted belief that only recent messages can be used to train a filter isn't allways true.

We also have shown that incremental update, on both classes ham and spam, may improve effectiveness and that one side incremental update degrades the misclassification rate on the opposite class. However endless incremental update can eventually generate an overfited model ([16] p.

194), mainly on classifiers unable to forget less recent samples.

References

[1] Aggarwal, C. C. On change diagnosis in evolving data streams. *IEEE Trans. on Knowl. and Data Eng. 17*, 5 (2005), 587–600.

[2] Aggarwal, C. C., Watson, T. J., Ctr, R., Han, J., Wang, J., and Yu, P. S. A framework for clustering evolving data streams. In *In VLDB* (2003), pp. 81–92.

[3] Boettcher, M., Hoeppner, F., and Spiliopoulou, M. On exploiting the power of time in data mining. *SIGKDD Explorations Newsletter 10*, 2 (2008), 3–11.

[4] Cormack, G. V. University of waterloo participation in the trec 2007 spam track. In *Sixteenth Text REtrieval Conference (TREC-2007)* (Gaithersburg, MD, 2007), NIST.

[5] Cormack, G. V., and Bratko, A. Batch and on-line spam filter evaluation. In *CEAS 2006: The Third Conference on Email and Anti-Spam* (2006).

[6] Cormack, G. V., and da Cruz, J. M. M. On the relative age of spam and ham training samples for email filtering. In *SIGIR Conference 2009* (Boston, Massachussets, 2009).

[7] Cormack, G. V., and Lynam, T. R. TREC spam filter evaluation toolkit. http://plg.uwaterloo.ca/~gvcormac/jig/.

[8] Cormack, G. V., and Lynam, T. R. Spam corpus creation for TREC. In *CEAS 2005: The Second Conference on E-mail and Anti-spam* (2005).

[9] Cormack, G. V., and Lynam, T. R. On-line supervised spam filter evaluation. *ACM Transactions on Information Systems 25*, 3 (2007).

[10] Cunningham, P., Nowlan, N., Delany, S. J., and Haahr, M. A case-based approach to spam filtering that can track concept drift. In *In The ICCBR'03 Workshop on Long-Lived CBR Systems* (2003), pp. 03–2003.

[11] Delany, S. J., Cunningham, P., and Tsymbal, A. A comparison of ensemble and case-base maintenance techniques for handling concept drift in spam filtering. In *Proceedings of the 19th International Conference on Artificial Intelligence (FLAIRS 2006)* (2006), G. Sutcliffe and R. Goebel, Eds., AAAI Press, pp. 340–345.

[12] Fawcett, T. 'In Vivo' spam filtering: A challenge problem for data mining. *KDD Explorations 5*, 2 (December 2003).

[13] Fdez-Riverola, F., Iglesias, E., Diaz, F., Mendez, J., and Corchado, J. Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications 33*, 1 (2007), 36 – 48.

[14] Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. Mining data streams: a review. *SIGMOD Rec. 34*, 2 (2005), 18–26.

[15] Goodman, J., and tau Yih, W. Online discriminative spam filter training. In *CEAS 2006: The Third Conference on Email and Anti-Spam* (2006).

[16] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning : Data Mining, Inference and Prediction.* Springer, New York, 2001.

[17] Hsiao, W.-F., and Chang, T.-M. An incremental cluster-based approach to spam filtering. *Expert Syst. Appl. 34*, 3 (2008), 1599–1608.

[18] Lowd, D., and Meek, C. Adversarial learning. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (New York, NY, USA, 2005), ACM, pp. 641–647.

[19] Raymond, E. S., Relson, D., Andree, M., and Louis, G. Bogofilter. http://bogofilter.sourceforge.net/, 2009.

[20] Sculley, D. Advances on online learning-based spam filters. In *PhD Thesis - Tufts University* (2008).

[21] Tong, S., and Koller, D. Support vector machine active learning with applications to text classification. In *Journal of Machine Learning Research* (2000), pp. 999–1006.

[22] Widmer, G. Learning in the presence of concept drift and hidden contexts. In *Machine Learning* (1996), pp. 69–101.

[23] Yang, C., and Zhou, J. Non-stationary data sequence classification using online class priors estimation. *Pattern Recogn. 41*, 8 (2008), 2656–2664.