Collaborative Email-Spam Filtering with the Hashing-Trick

Joshua Attenberg
Polytechnic Institute of NYU
Five MetroTech Center
Brooklyn NY, 11201
josh@cis.poly.edu

Kilian Weinberger, Anirban Dasgupta
Alex Smola, Martin Zinkevich
Yahoo! Research
4401 Great America Pkwy.
Santa Clara, CA 95054 USA
{kilian, anirban, smola, maz}@yahoo-inc.com

ABSTRACT

This paper delves into a recently proposed technique for collaborative spam filtering [7] that facilitates personalization with finite-sized memory guarantees. In large scale open membership email systems most users do not label enough messages for an individual local classifier to be effective, while the data is too noisy to be used for a global filter across all users. Our hybrid global/individual classifier is particularly effective at absorbing the influence of users who label emails very differently from the general public – because of strange taste or malicious intent. We can accomplish this while still providing sufficient classifier quality to users with few labeled instances. Our proposed technique can be used with a variety of classifiers and can be implemented in a few lines of code. We verify the efficacy of our proposed technique on a popular web spam benchmark data set.

1. INTRODUCTION

Collaborative spam filtering in open membership systems, such as Yahoo Mail^{TM}, depends on user generated label information. Users provide feedback by labeling emails as spam or not-spam. These labels are then used to train a spam filter. Although the majority of users provide very little data, as a collective the amount of training data is very large (many millions of emails per day). Unfortunately, there is substantial deviation in users' understanding of what constitutes spam and non-spam. As a result, spam filtering based on a global classifier will be sub-optimal. Conversely, there is often insufficient personal information to train an individual classifier for all users.

In this paper we adopt the *hashing-trick* [7, 8] to build a *personalized global* classifier. The hashing-trick maps the global and all personal classifiers into a single low-dimensional feature space, in which we train a *single* weight vector capturing the individual aspects of each user.

The hashing-trick is a natural fit for spam filtering. Nearly all commonly used spam classifiers, such as Naïve Bayes, logistic regression and support-vector machines [3] require emails be represented in the bag-of-words format. A major disadvantage of the bag-of-words representation is the necessity of a dictionary data structure for mapping words to vector indices. In collaborative spam filtering, the set of possible word tokens is generally very large. As a result, the dictionary can use up a significant portion of a servers' memory. Furthermore, the dynamic nature of language in both spam and not-spam requires that the dictionary adapt to new words, essentially growing over time. The hashing-trick overcomes this obstacle by rendering the dictionary unnecessary; words are hashed directly to indices. While hash collisions may result in several words being mapped into the same index, and therefore being falsely considered identical, such collisions rarely affect classification results. The decision whether an email is spam and not-spam is rarely based on a single word but on a combination of many slightly indicative words. In fact, without a dictionary much more memory is available to store the weights of the classifier, extra space that may be used to improve classifier performance, as demonstrated empirically [7, 2, 8].

In [7], we used the hashing trick to hash multiple classifiers into the same space, allowing us to obtain a large number of local classifiers with negligible additional computational or memory requirements, and established it empirically with real data. The resulting hybrid classifier overcomes the tough choice between local and global classification. In this extension, we also show (using the TREC data set with synthetic labels) how this hybrid solution can handle malicious noise.

CEAS 2009 - Sixth Conference on Email and Anti-Spam July 16-17, 2009, Mountain View, California USA. A full version of this work is available on the website of the primary author.

2. HASHING-TRICK

The hashing-trick [7, 8] is a method to scale up linear learning algorithms. The main idea is quite simple. Instead of generating bag-of-word feature vectors through a dictionary that maps tokens to word indices, one uses a hash-function that *hashes* words directly into a feature vector. The hash function $h : \{Strings\} \rightarrow [1..m]$ operates directly on strings and should be approximately uniform[1]. In [7] we propose to use a *second* independent hash function $\xi : \{Strings\} \rightarrow \{-1, 1\}$, which determines if the particular hashed-dimension of a token should be incremented or decremented. This causes the hashed feature vectors to be unbiased. Algorithm 2 shows a pseudo-code implementation of the hashing-trick that generates a hashed bag-of-words feature vector for an email.

Personalization:

The above hashing-trick can be effectively used to train spam classifiers with bounded memory requirements [7]. How-

[1]For the experiments in this paper we used the public domain implementation from http://burtleburtle.net/bob/hash/doobs.html

```
hashingtrick([string] $email$)
$\vec{x} = \vec{0}$
for $word$ in $email$ do
    $i = h(word)$
    $\vec{x}_i = \vec{x}_i + \xi(word)$
end for
return $\vec{x}$
```

ever, even more powerful is the extension allowing the formation of multiple classifiers on a data set [7]. In the spam-classification setting, this technique provides a handle to tackle the elusive goal of personalization of spam-filters in the presence of widely uneven amounts of labeled data per user. The main issues in tackling personalization are twofold – the large number of users in a typical email system create an enormous blowup in the number of feature-weights needed to be stored, and the "cold-start" problem, where new or unengaged users have insufficient labeled emails to generate an effective personal spam filter. In practice, both problems become egregious for large-scale email-providers — they have millions of email users and so the resulting storage increases million-fold when trying to personalize, yet most email users are notoriously lazy in labeling emails as "spam" or "not-spam", leaving them forever in the "cold-start" phase.

Using a single global classifier, on the other hand, is both efficient in terms of the space-complexity and in terms of providing new users with a reasonable spam classifier. While providing efficient solutions to these two issues, a single global classifier is problematic for spam-filtering due to the issue of noisy feedback. It is well known that users differ significantly in their view of spam and not-spam, especially in the case of bulk and business emails. Furthermore, it is fairly regular for spammers to infiltrate the user base using many different accounts so as to be able to provide feedback that is designed to confuse the spam-classifier. As our experiments will show, the global classifier is susceptible to even a small fraction of users giving noisy feedback.

We combat the above problems by using the hashing-trick to efficiently create a large instance of multitask learning– a setting where multiple related yet slightly different concepts are learned simultaneously. For the thousands of users, U, who users who participate in the filter's feedback system labeling messages as spam or not-spam, an individualized spam filter, $\vec{w}_u$ is trained. Additionally, a global classifier (w_0) is trained from the aggregate labels compiled from all users. This provides a consensus filter used to overcome the sparsity in training data resulting from the disparity in the amount of labeled content per user. An email can now classified according to the additive scores of w_0 and w_u.

If the total number of features is d, then storing all the above vectors would need $O(d \times (|U| + 1))$ space. However, as words appear in accordance with Zipf's law, most users will only encounter a small fraction of the total vocabulary. As a result, the $|U| + 1$ classifiers necessary for a hybrid global/local spam filter are extremely compressible. We will compress them by hashing all weight vectors into a single weight vector $\vec{w}$. In order to this, leverage a major strength of the hashing trick– approximate preservation of orthogonal vectors. This allows us to learn many email filters simultaneously in a single hashed space. In the context of personalization, we make use of this property by hashing each token twice: once in it's original form, and once with the labeler's user-id prepended to it. See Algorithm 2 for details. Intuitively, this adds individualized tokens for each particular user. Imagine for example that user "barney" does like emails containing the word "viagra", whereas the majority of users won't. The personalized hashing trick will learn that "viagra" itself is a spammy word, whereas "viagra_barney" is a not-spammy word.

```
personalized_hashingtrick(string $userid$, [string] $email$)
$\vec{x} = \vec{0}$
for $word$ in $email$ do
    $i = h(word)$
    $\vec{x}_i = \vec{x}_i + \xi(word)$
    $j = h(word \circ userid)$
    $\vec{x}_j = \vec{x}_j + \xi(word \circ userid)$
end for
return $\vec{x}$
```

3. EXPERIMENTAL SETUP AND RESULTS

To assess the validity of our proposed techniques, we conduct a series of experiments on the freely distributed `trec07p` data set, a standard benchmark set for spam filter performance evaluation. This dataset contains 75, 419 labeled and chronologically ordered emails taken from a single email server over four months in 2007 and compiled for TREC spam filtering competitions [1]. E-mails are either spam (or positive) or ham (or negative).

We divided the TREC data set temporally, using the first 75% of the e-mails as training, and testing on the last 25% of the e-mails. For all our experiments we used the Vowpal Wabbit [5] (VW) implementation of stochastic gradient descent on a square-loss[2].

In order to appraise the performance of the hash-trick, we tokenize the bodies of the e-mails and create binary features indicating the presence or absence of each feature in the email body. We then use VW on these tokenized e-mails, using a hashtable of a size between 2^5 and 2^{12}. For each of these sets of hashed feature vectors, we train and test a single, global classifier, and then analyze the receiver operating characteristic (ROC) curve [6]. We evaluated classifiers by choosing the point on the curve where the ham misclassification rate (the fraction of ham e-mails classified as spam e-mails, false positive rate, or HMR) was 1%.[3]

All figures presented below display the spam catch rate (the fraction of spam e-mails that were classified as spam e-mails, true positive rate, or SCR) when the HMR is 1%. We believe this metric accurately reflects a realistic email filter setting where misclassifying ham messages as spam creates an extremely negative overall system experience. While this metric is highly correlated with the area under the ROC

[2]We would like to point out that the hashing-trick is independent of the type of classifier and could be applied to almost all commonly used spam filters. Although we are only presenting results obtained with the Vowpal Wabbit, similar trends should be observed with other classifiers. As hash function we use the public-domain implementation by Jenkins [4]

[3]In order to tune the algorithm to get a false positive rate of 1%, it is necessary in practice to reserve a holdout set from the training set.

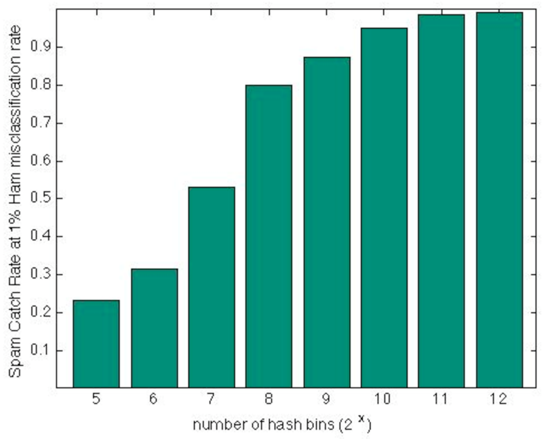


Figure 1: Spam catch rate for TREC07p with the number of hash bins r ranging from 2^5 to 2^{12}.

curve (AUC), the AUC encodes much additional irrelevant information for this particular task.

Figure 1 presents the SCR at 1% HMR. Note using 2^8 hash bins will still result in achieving 80% SCR. This increases to over 90% when using 2^{10} hash bins. With 2^{12} hash bins, the classifier achieves a 99.57% SCR with a 1% HMR, with an area under the ROC curve (AUC) of 0.99829. Given that the full TREC data set has a total of 508, 531 unique terms this means that using a weight vector of 0.8% the size of the full set obtains essentially the same performance as one using all dimensions.

3.1 Simulating Malicious Labeling Activity

In order to replicate the influence of malicious email labelers that abound in a real-world spam filtering setting, we chose to actively modify the label assignments from those provided by human judges in the TREC data set. In order to mimic the effects of this adversarial labeling, a number of "quality" users are initially chosen at random, and their labeled emails are assembled to create a baseline data set. The remaining users are then considered "malicious". The emails of these malicious users are re-labeled, with labels chosen uniformly at random.[4] New data sets can now be created with the inclusion of varying portions, p, of malicious users. For the purpose of this paper, we chose $p \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. By varying the number of malicious users, we see trends in spam filter performance. All modified data sets used in this project are publicly available at `http://cis.poly.edu/~josh/spam/`.

Label modification for our adversarial simulation is performed at the granularity of an individual user — a given user either has their natural labeling assignment or all of their labels are randomly set. By delineating the data set according to users, we simulate the realistic setting where many users label according to their own preference for clean inboxes, while some users are in fact artificial software constructs, programmed to degrade spam filtering systems such that favored emails will escape detection and reach the remaining benevolent user community. Because it is difficult to define what constitutes a *good* label assignment for a user

[4]Note that random labels are nastier to deal with than simply flipped labels-if the labels are flipped, then the classifier "just" learns which users has flipped labels, and then flip them back.

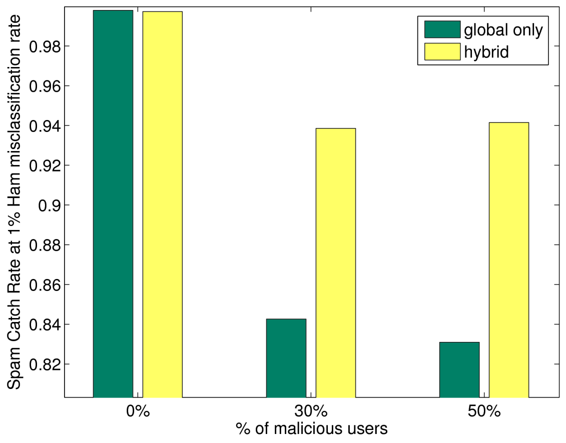


Figure 2: Performance of global and hybrid classifiers at varying degrees of maliciousness. We used 2^{20} hash bins in the experiments.

who is habitually malicious, we exclude these users from the test set. In this way, we are strictly measuring the malicious users' influence on classifier accuracy as seen by legitimate users in various spam filtering scenarios.

With $p = 0.2$, the global classifier with 2^{12} hash bins is reduced to 86% SCR at 1% HMR. With the quantities of spam emails sent daily, this is unacceptable.

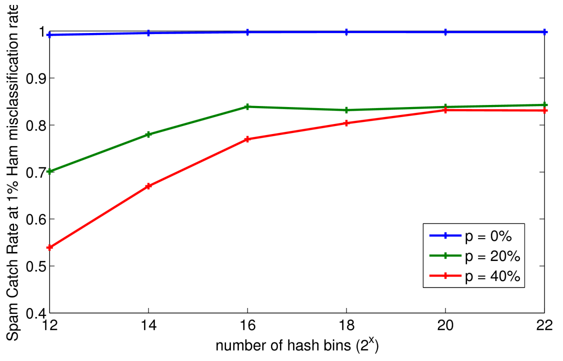
3.2 Mitigating Malicious Users by Hashing

We note that in our experimental framework, as in most reasonable real-world open email systems, the quality of labeling varies from person to person. By utilizing individual classifiers to discriminate spam, consistently malicious or otherwise low-quality labelers can only exert influence on their local classifier, effecting the labeling of spam which they may see. However, most users label one or two e-mails–too few emails to train an effective individual classifier. In order to achieve the benefits of personalized classification while maintaining good general performance for most users, we adopt a hybrid, global + individual classifier. As discussed in Section 2, feature hashing allows simple personalization by projecting multiple classifiers onto a single feature space in $\mathbb{R}^r$ with little interaction.

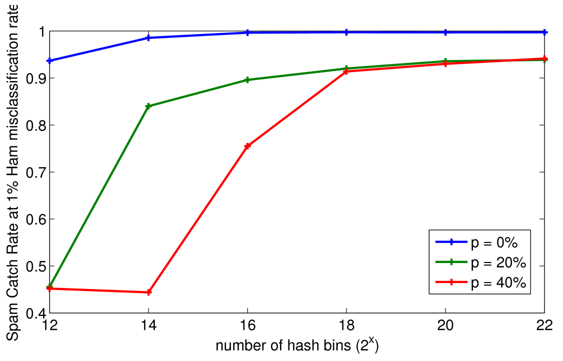
Figure 2 presents a comparison in classifier performance under varying malicious loads by global and hybrid classifiers. On the unmodified TREC data set, with 0% malicious users, global and hybrid classifiers both offer similarly excellent performance. However, as the level of malicious activity rises to 30%, the global classifier degrades more rapidly: with 30% malicious users, the global classifier has 83% accuracy, and the hybrid classifier has 94% accuracy. The purely local classifier performs very badly, mostly because several users have no training data. Note that in this experiment $r = 2^{20}$ was used to cope with the increased information presented by individualized classifiers. While this is larger than the experiment presented in Figure 1, if each weight requires storage as a double precision floating point number, the total space requirement is only 8MB.

3.3 Personalization and Hashing

In Section 2 we have discussed how personalization can be used to mitigate the effect of noisy labels from users. In implementing such a setting, a dimensionality reduction tech-



(a) Global Classifier



(b) Hybrid Classifier

Figure 3: Influence of r on global and hybrid classifier performance with malicious users

nique such as the hashing-trick becomes critical– without such a technique, even for the trec07 dataset, with around 5000 users and over 500,000 tokens, the total number of dimensions needed would be over 2.5 billion. However, the hashing-trick comes with a compromise of its own – one must ensure that the number of hash bins is sufficiently large, and the hash-function is sufficiently random so as not to avoid a lot of collisions. Since the classifiers are now trained on the hash-space, a collision of a spam-indicative feature with one that indicates ham, would be detrimental to the performance. In [7] we show that under certain assumptions on the hashing function and the number of hash bins, we can prove theoretical bounds on the distortion that is caused by the hashing. The number of hash bins that we need in practice is much less than our theoretical bounds. In this section we vary both the number of hash bins and the fraction of malicious users. Again we consider the SCR when the HMR is bounded by 1% – Figure 3 shows our findings.

Figure 3(a) presents the spam catch rate for a global spam filter for different numbers of hash bins, the three colored lines indicate three different levels of malicious activity: the fraction p of malicious users being $\{0\%, 20\%, 40\%\}$. Figure 3(b) represents the same experiment only with the combined hybrid classifier. Focusing on the case of $p = 0\%$, we note that, as before, both classifiers do very well. The hybrid classifier actually does slightly worse than the global classifier with $r = 2^{12}$, this is likely due to the vastly increased number of tokens being mapped into each hash bin in the hybrid filter. This barrage of collisions may result in conflicting signals, and therefore lowered accuracy.

With malicious users, both global and hybrid classifiers require more hash bins to achieve near-optimum performance. Since the hybrid classifier has more tokens, the number of hash-collisions is also correspondingly larger. With 2^{12} hash bins, for instance, the average number of collisions per hash bucket is over $60,000$. Correspondingly, the hybrid classifier does worse than the global one for 2^{12} hash bins. However, the situation quickly is quickly remedied by offering increased storage size; with 2^{18} hash bins, the hybrid classifier achieves approximately 90% SCR, beating the 80% SCR achieved by the global-only classifier. Further increases in the number of hash bins does not improve the classification performance significantly.

4. CONCLUSION

This work demonstrates the *hashing-trick* as an effective method for collaborative spam filtering. It allows spam filtering without the necessity of a memory-consuming dictionary and strictly bounds the overall memory required by the classifier. Further, the hashing-trick allows the compression of many (hundreds of thousands of) classifiers into a single finite-sized weight vector. This allows us to run personalized and global classification together with very little additional computational overhead. We provide strong empirical evidence that the resulting classifier is more robust against noise and absorbs individual preferences that are common in the context of open-membership spam classification.

5. REFERENCES

[1] G. Cormack. TREC 2007 spam track overview. In *The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings*, 2007.

[2] K. Ganchev and M. Dredze. Small statistical models by random feature mixing. In *Workshop on Mobile Language Processing, Annual Meeting of the Association for Computational Linguistics*, 2008.

[3] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani. *The elements of statistical learning*. Springer New York, 2001.

[4] R. Jenkins. Algorithm alley. *Dr. Dobb's Journal*, September 1997. Source code available at `http://burtleburtle.net/bob/hash/doobs.html`.

[5] J. Langford, L. Li, and A. Strehl. Vowpal wabbit online learning project. Technical report, `http://hunch.net/?p=309`, 2007.

[6] K. Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the sixth international workshop on Machine learning table of contents*, pages 160–163. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1989.

[7] K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford, and A. Smola. Feature hashing for large scale multitask learning. In *ICML*, 2009.

[8] W. S. Yerazunis. Sparse binary polynomial hashing and the CRM114 Discriminator. In *MIT Spam Conference*, 2003.