Web spam Identification Through Content and Hyperlinks

Jacob Abernethy
Dept. of Computer Science
University of California
Berkeley, CA, USA
jake@cs.berkeley.edu

Olivier Chapelle
Yahoo! Research
Santa Clara, CA, USA
chap@yahoo-inc.com

Carlos Castillo
Yahoo! Research
Barcelona, Spain
chato@yahoo-inc.com

ABSTRACT

We present an algorithm, WITCH, that learns to detect spam hosts or pages on the Web. Unlike most other approaches, it *simultaneously* exploits the structure of the Web graph as well as page contents and features. The method is efficient, scalable, and provides state-of-the-art accuracy on a standard Web spam benchmark.

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous; I.2.6 [**Learning**]; I.5 [**Pattern Recognition**]

Keywords

Web spam, graph regularization, Support Vector Machines

1. INTRODUCTION

Web spam manifests itself as web content generated deliberately for the purpose of triggering unjustifiably favorable relevance or importance of some Web page or pages [9]. It has been observed that spam and non-spam pages exhibit different statistical properties which can be exploited for building automatic classifiers [13].

From a machine learning perspective, the spam detection task differs from a typical classification task since not only do we have standard features available for every page/host, but we are also given a directed hyperlink structure on our data as well. A hyperlink often reflects some degree of similarity [7, 16] among pages. Complex patterns can be observed in hyperlinks; for instance, in the particular case of spam it has been observed that non-spam hosts rarely link to spam hosts, even though spam hosts do link to non-spam hosts. Several techniques based on the propagation of (dis)trust along the hyperlinks have exploited this idea [10, 12, 17].

In this paper we present a learning algorithm that we call WITCH, for Web spam Identification Through Content and Hyperlinks, that directly uses the hyperlink structure during the learning process in addition to page features. Specifically, we learn a linear classifier on a feature space using an SVM-like objective function. The hyperlink data is exploited by way of *graph regularization*, which produces a predictor that varies smoothly between linked pages. Our results suggest that this method of SVM with graph regularization is highly effective at detecting Web spam, outperforming all other state-of-the-art methods that we implemented. Our method also performs well even with little training data.

It is, as far as we know, the first technique for spam detection that simultaneously uses features and the hyperlink graph for training. Note that these features can be a combination of any type of features such as content-based and link-based features.

2. ALGORITHMS

For the remainder of this paper, we will discuss classification of *hosts* as spam or non-spam. A host is a group of Web pages sharing the same "host" component in their URLs. All techniques can be similarly applied to individual pages as well. Assume we are given the following: (i) a set of l labeled examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$, where $\mathbf{x}_i$ denotes the feature vector associated with the i-th host and y_i is its label: $+1$ for spam and -1 for non-spam; (ii) a set of u unlabeled examples, $\mathbf{x}_{l+1}, \ldots, \mathbf{x}_n$, with $n = l + u$; and (iii) a weighted directed graph whose nodes are $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Let E be the sets of pairs (i, j) whenever node i is connected to node j, and let a_{ij} be the weight of the link from $\mathbf{x}_i$ to $\mathbf{x}_j$.

2.1 Learning with Graph Regularization

Suppose we want to learn a linear classifier $f(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}$. A familiar approach is to train a linear *Support Vector Machine* (SVM) [15]. In this case, $\mathbf{w}$ is found as the minimizer of the following objective function:

$$\Omega(\mathbf{w}) \ = \ \frac{1}{l}\sum_{i=1}^{l} R(\mathbf{w}\cdot\mathbf{x}_i, y_i) + \lambda \mathbf{w}\cdot\mathbf{w}, \qquad (1)$$

where λ is a parameter of the algorithm. The above objective function captures the necessary trade-off between fitness and complexity, for we would choose $\mathbf{w}$ to correctly classify our training data while maintaining a large margin. Here we use the hinge function $R(u, y) = \max(0, 1 - uy)$ to represent the loss on the training data, but any convex loss function may be used. The quantity $\mathbf{w}\cdot\mathbf{w}$ represents the size of the margin and is often referred to as the *regularization* term.

For the special case of classification tasks on the Web, one has the additional advantage of the hyperlinks between

AIRWeb '08, April 22, 2008 Beijing, China.

```
Algorithm 1 WITCH
  Params: $\lambda_1, \lambda_2, \gamma$, convex function $\Phi(\cdot,\cdot)$
  Input: labeled training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$
  Input: unlabeled set $\mathbf{x}_{l+1}, \ldots, \mathbf{x}_n$
  Input: hyperlink graph $E$ with edge weights $\{a_{ij}\}_{(i,j)\in E}$
  Solve:
      $\mathbf{w}, \mathbf{z} \leftarrow \arg\min_{\mathbf{w},\mathbf{z}} \Omega(\mathbf{w}, \mathbf{z})$,
  with $\Omega(\cdot)$ defined in (3).
  Predict: label node $i$ as $\text{sign}(\mathbf{w}\cdot\mathbf{x}_i + z_i)$.
```

nodes. Hyperlinks can be represented as a directed graph with edge set E. Hyperlinks are not placed at random, and it has been shown empirically that they imply some degree of similarity between the source and the target node of the hyperlink [11, 7]. Based on this observation, it is natural to add an additional regularizer to the objective function:

$$\begin{aligned}\Omega(\mathbf{w}) \;=\; & \frac{1}{l}\sum_{i=1}^{l} R(\mathbf{w}\cdot\mathbf{x}_i, y_i) + \lambda\mathbf{w}\cdot\mathbf{w} \\ & + \gamma \sum_{(i,j)\in E} a_{ij}\Phi(\mathbf{w}\cdot\mathbf{x}_i, \mathbf{w}\cdot\mathbf{x}_j), \qquad (2)\end{aligned}$$

where a_{ij} is a weight associated with the link from node i to node j. The first two terms correspond to a standard linear SVM described above. The third term enforces the desired graph regularization described above. The function Φ represents any distortion measure, and is chosen according to the problem at hand.

The objective (2) was proposed in [4, 16], where Φ was chosen to be $\Phi(u, v) := (u - v)^2$, which implicitly encodes the expectation that hyperlinked neighbors should have similar predicted values. One novelty of our proposed method is that, contrary to [4, 16], we utilize *asymmetric* graph metrics tuned to the particular task of Web spam classification. With spam, incorporating hyperlink direction is crucial: spam hosts frequently link to genuine hosts but rarely vice versa. This has been empirically confirmed in [6, 8]. We exploit hyperlink direction through the alternative metric $\Phi(u, v) = \max(0, v - u)^2$, and provide a much more detailed discussion in Section 3.1.

2.2 Additional Slack Variables

In the case where the feature space is not rich enough, a simple linear classifier $\mathbf{w}\cdot\mathbf{x}$ might not be flexible enough. But as in [16], one can introduce a parameter z_i for every node i and learn a classifier of the form $f(\mathbf{x}_i) = \mathbf{w}\cdot\mathbf{x}_i + z_i$. This extra term can be seen as an additional slack variable that gives more freedom to the learned classifier.

Our new objective becomes:

$$\begin{aligned}\Omega(\mathbf{w}, \mathbf{z}) \;=\; & \frac{1}{l}\sum_{i=1}^{l} R(\mathbf{w}\cdot\mathbf{x}_i + z_i, y_i) + \lambda_1\mathbf{w}\cdot\mathbf{w} + \lambda_2\mathbf{z}\cdot\mathbf{z} \\ & + \gamma \sum_{(i,j)\in E} a_{ij}\Phi(\mathbf{w}\cdot\mathbf{x}_i + z_i, \mathbf{w}\cdot\mathbf{x}_j + z_j). \qquad (3)\end{aligned}$$

Here we introduce two regularization parameters λ_1 and λ_2 for controlling the values of both $\mathbf{w}$ and $\mathbf{z}$.

This last objective function is the basis for WITCH, which we now summarize in Algorithm 1.

2.3 Optimization

Since the objective function is convex and differentiable, one can simply use nonlinear conjugate gradient [14] to optimize it. This is a standard and very efficient method for nonlinear optimization and it only requires the computation of the gradient. It is also much faster than the dual algorithm of [16]. Details can be found in [3].

Another way of optimizing (3) is to perform an alternate optimization on $\mathbf{w}$ and $\mathbf{z}$. The advantage of this approach is that both steps can be performed using standard algorithms: the minimization on $\mathbf{w}$ can be done using a regression algorithm, while the minimization on $\mathbf{z}$ boils down to a standard iterative propagation algorithm on the graph. We refer the reader to [3] for a detailed description of this method.

3. IMPLEMENTATION AND RESULTS

All experimental results from this section are based on a data set that we discuss in Section 4. In order to compare performance of each method, we chose the metric Area Under the ROC curve (AUC). AUC provides a natural measure of accuracy of a predicted ranking, and requires only that the algorithm outputs an ordering of the test set. Further details about the experimental setup can be found in [3].

3.1 Design Elements

We proceed to analyze the performance of our spam classification technique, WITCH (Web spam Identification Through Content and Hyperlinks). We start by describing in detail the different algorithm settings that we consider.

Graph weights. For each pair of nodes i, j, we are given $n_{i,j}$, the number of links from host i to host j. To utilize graph regularization we must specify edge weights, and we tried a variety of different weighting schemes. Absolute weights $a_{i,j} := n_{i,j}$ tended to over-regularize hosts with abundant hyperlinks, which was improved by using binary weights, $a_{i,j} := \mathbf{1}[n_{i,j} > 0]$, and further by "square-root weights", $a_{i,j} := \sqrt{n_{i,j}}$. The best performance, however, was obtained with logarithmic weights, $a_{i,j} := \log(1 + n_{i,j})$, and we report all results with this latter choice.

Graph Regularizer. Since we expect a host's "spamicity" (or similarly, "authenticity") to be preserved locally within the web, the graph regularization function $\Phi(\cdot,\cdot)$ ought to encode how we enforce this locality in our predictions. If node i links to node j, then $\Phi(f_i, f_j)$ should measure how "unnatural" it is that a node with spam score f_i links to a node with spam score f_j. We have already defined two possible regularization functions:

$$\begin{aligned}\Phi_{sqr}(f_i, f_j) \;&\triangleq\; (f_i - f_j)^2 \\ \Phi^{+}_{sqr}(f_i, f_j) \;&\triangleq\; \max(0, f_j - f_i)^2 = [f_j - f_i]^2_+\end{aligned}$$

The first of these penalizes the square of any deviation between the predicted values of i and j. The second function, on the other hand, only penalizes the predicted spam scores when the node creating the link has a lower predicted spam value than the link's destination. The function Φ^{+}_{sqr} encodes our assumption that, while the spamicity value can reasonably decrease through a link (i.e. when bad links to good), it should not increase (i.e. when good links to bad).

For the task at hand, the latter choice would appear more appropriate. In general, while bad nodes may link to good nodes (perhaps to appear good), good nodes typically have

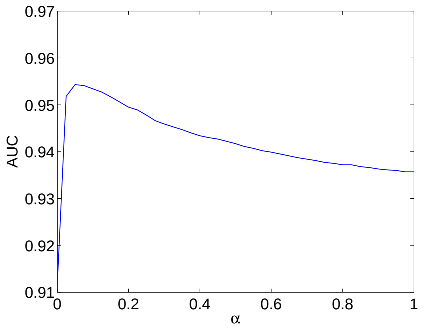

Figure 1: Effect of mixed regularization. $\alpha = 0$ implies $\Phi := \Phi^{+}_{sqr}$, while $\alpha = 1$ implies $\Phi := \Phi_{sqr}$.

no incentive to link to bad nodes (unless perhaps they are paid to do this), and thus we expect only rarely to observe good-to-bad pairs. We have also observed this empirically: among the labeled nodes within the WEBSPAM-UK2006 dataset described above, only 1.8% of the out-links from non-spam hosts point to spam hosts, while 14.7% of out-links from spam hosts point to non-spam hosts.

Interestingly, we have found that the best choice or regularization is neither of the above but rather a *mixture of the two*. For any $\alpha \in [0, 1]$, define:

$$\Phi_\alpha(a, b) \quad \triangleq \quad \alpha\Phi_{sqr}(a, b) + (1 - \alpha)\Phi_{sqr+}(a, b)$$

We found this mixed regularization to be very effective, and surprisingly a great improvement over either using only Φ_{sqr} or only Φ^{+}_{sqr}. In Figure 1, we plot performance, measured by AUC, as a function of the choice of α. When $\alpha = 0$, only the nonspam→spam links are penalized. When $\alpha = 1$ any deviation in the predicted spam value between linked hosts incurs a cost. For the remainder of the results in the paper, we report results using $\Phi_{0.1}$, i.e. where $\alpha = 0.1$, as the regularization function. The value of α can be tuned more carefully, but performance is relatively stable around this value.

Model Selection. WITCH requires the choice of three hyperparameters, $\lambda_1, \lambda_2, \gamma$. We maintained a hold-out set consisting of a random 20% sample of the training data. On a $7 \times 7 \times 7$ grid of parameters, we trained WITCH for each combination and chose the triple $(\lambda_1, \lambda_2, \gamma)$ that returned the best test performance on these data. The final results we report in Table 1 were obtained after validation.

3.2 Comparison with Variant Algorithms

We now present the performance of WITCH on the dataset discussed in Section 4.

Recall that WITCH takes advantage of three primary tools in training: host features, slack variables, and regularization along the hyperlink graph. Each of these elements plays a different role in the algorithm and contributes to performance at varying levels. To see the relative importance of each, we now consider alternative approaches that involve various subsets of the above.

Only Features. We train a linear classifier on the given feature space with no graph regularization. That is, we set $\lambda_2 = 0$ and $\gamma = 0$ in (3). This is a standard SVM.

Features + Graph Regularization (GR). We train a linear classifier on the provided feature space but we additionally regularize according to the hyperlink graph structure. This corresponds to minimizing (2).

Slack Variables + GR. We ignore features and directly learn the label using graph regularization. This is equivalent to optimizing (3) under constraints $\mathbf{w} = 0$.

Features + Slack + GR (WITCH). We now utilize all tools available. We simultaneously train a linear classifier and slack variables, and we regularize the predicted values along the graph. This is algorithm 1.

In Table 1 we report performance for each of the above four methods. We observe that the greatest boost appears to be due to the addition of slack variables. This is likely the result of underfitting: there may not be a single linear predictor $\mathbf{w}$ on the available feature space that can accurately detect spam, thus the slack introduces an additional level of freedom to the model for accurately classifying spam.

To see how performance depends on subset size, in Figure 2 we also compare each of the above algorithms for seven different training-set sizes.

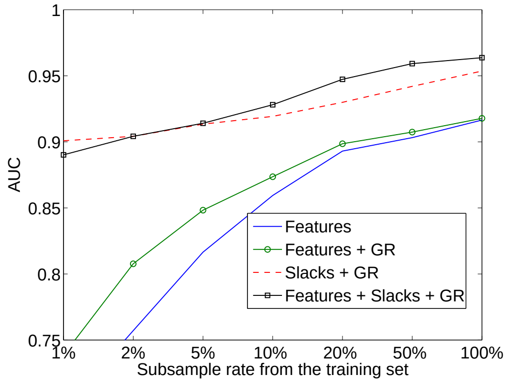

Figure 2: Performance of WITCH and variants (from Section 3.2). Experiments repeated 10 times with random subsets and median is plotted.

4. COMPARISON WITH OTHER METHODS

To compare methods for detecting spam on the World Wide Web, a group of researchers recently organized the Web Spam Challenge [2] based on the WEBSPAM-UK2006 spam collection [5], a public Web Spam dataset annotated at the level of hosts. The collection represents a graph of 11,402 hosts in the .uk domain, out of which 7,473 are labeled. We used the set of 236 features proposed for the challenge. The training/testing split was fixed and is the same that was used in the Web Spam Challenge Track I.

The first track of the competition, in which we did not take part, ended in April of 2007; the best performance in terms of AUC[1] was obtained by Gordon Cormack with 0.956. On the same dataset, WITCH outperforms all submissions to the

[1] This track in fact had several winners, as the competition consider F-Measure as well as AUC for a performance metric.

first track of the challenge, obtaining an AUC performance of 0.963[2].

The challenge included a Track II that ended in July 2007, and for this track we submitted predictions using the methods discussed herein. WITCH obtained the highest AUC against the 10 other submissions [1]. The data in Track II was generated from the data in the first track, but with a modified feature set, a new training/test set split, and it omitted page contents and host addresses.

Table 1 summarizes the performances of the methods discussed in this paper. It also includes 2 recent state-of-the-art methods for web spam detection, stacked graphical learning [6] and transductive link spam proposed in [17], which uses only hyperlinks and not content based features. This method outperforms other well-known graph-based methods based on label propagation such as TrustRank [10] and Anti-Trust Rank [12]. Further experimental results using these two competing methods can be found in [3].

Table 1: Results summary with two training sets.

Training Algorithm	AUC 10%	AUC 100%
SVM + stacked g.l.	0.919	0.953
Link based (no features)	0.906	0.948
Challenge winner	–	0.956
Only Features	0.859	0.917
Features + GR	0.874	0.917
Slack + GR	0.919	0.954
WITCH (Feat. + Slack + GR)	**0.928**	**0.963**

5. CONCLUSIONS

In this paper we have presented a novel algorithm, WITCH, for the task of detecting Web spam. We have compared WITCH to several proposed algorithms and we have found that it outperforms all such techniques. Finally, WITCH obtains the highest AUC performance score on an independent Web spam detection challenge.

We attribute these positive results to a few key observations. First, best results are achieved when both content features *and* the hyperlink structure are used. Second, simply training a graph-regularized linear predictor is insufficient, as the addition of slack variables provides a very significant improvement. Third, one needs to choose the right graph regularizer, as we have observed that penalizing both spam→nonspam links and nonspam→spam links is important, yet the tradeoff should be much heavier on the latter. Lastly, we have observed that the form of the graph weights contributes significantly to performance, where using the logarithm of the number of links worked best.

6. REFERENCES

[1] Graph Labeling Workshop. http://graphlab.lip6.fr/, 2007.

[2] Web Spam Challenge. http://webspam.lip6.fr/, 2007.

[3] J. Abernethy, O. Chapelle, and C. Castillo. WITCH: A new approach to web spam detection. Technical Report 2008-001, Yahoo! Research, 2008.

[4] M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2005.

[5] C. Castillo, D. Donato, L. Becchetti, P. Boldi, S. Leonardi, M. Santini, and S. Vigna. A reference collection for web spam. *SIGIR Forum*, 40(2):11–24, December 2006.

[6] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: Web spam detection using the web topology. In *Proceedings of SIGIR*, Amsterdam, Netherlands, July 2007. ACM.

[7] B. D. Davison. Topical locality in the web. In *Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval*, pages 272–279, Athens, Greece, 2000. ACM Press.

[8] Q. Gan and T. Suel. Improving web spam classifiers using link structure. In *AIRWeb '07: Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, pages 17–20, New York, NY, USA, 2007. ACM.

[9] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *First International Workshop on Adversarial Information Retrieval on the Web*, pages 39–47, Chiba, Japan, 2005.

[10] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating Web spam with TrustRank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 576–587, Toronto, Canada, August 2004. Morgan Kaufmann.

[11] S. W. Haas and E. S. Grams. Page and link classifications: connecting diverse resources. In *DL '98: Proceedings of the third ACM conference on Digital libraries*, pages 99–107, New York, NY, USA, 1998. ACM Press.

[12] V. Krishnan and R. Raj. Web spam detection with anti-trust rank. In *ACM SIGIR workshop on Adversarial Information Retrieval on the Web*, 2006.

[13] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In *Proceedings of the World Wide Web conference*, pages 83–92, Edinburgh, Scotland, May 2006.

[14] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, 1994.

[15] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons Inc, 1998.

[16] T. Zhang, A. Popescul, and B. Dom. Linear prediction models with graph regularization for web-page categorization. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 821–826, New York, NY, USA, 2006. ACM Press.

[17] D. Zhou, C. J. C. Burges, and T. Tao. Transductive link spam detection. In *AIRWeb '07: Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, pages 21–28, New York, NY, USA, 2007. ACM Press.

[2] The hyperparameters $\lambda_1, \lambda_2, \gamma$ were chosen on a validation set as we describe in Section 3.1.