Know your Neighbors: Web Spam Detection using the Web Topology

Carlos Castillo[1] chato@yahoo-inc.com
Debora Donato[1] debora@yahoo-inc.com
Aristides Gionis[1] gionis@yahoo-inc.com
Vanessa Murdock[1] vmurdock@yahoo-inc.com
Fabrizio Silvestri[2] f.silvestri@isti.cnr.it

[1]Yahoo! Research Barcelona
C/Ocata 1, 08003 Barcelona
Catalunya, SPAIN

[2]ISTI – CNR
Via G. Moruzzi 1, 56124 Pisa
ITALY

ABSTRACT

Web spam can significantly deteriorate the quality of search engine results. Thus there is a large incentive for commercial search engines to detect spam pages efficiently and accurately. In this paper we present a spam detection system that combines link-based and content-based features, and uses the topology of the Web graph by exploiting the link dependencies among the Web pages. We find that linked hosts tend to belong to the same class: either both are spam or both are non-spam. We demonstrate three methods of incorporating the Web graph topology into the predictions obtained by our base classifier: (*i*) clustering the host graph, and assigning the label of all hosts in the cluster by majority vote, (*ii*) propagating the predicted labels to neighboring hosts, and (*iii*) using the predicted labels of neighboring hosts as new features and retraining the classifier. The result is an accurate system for detecting Web spam, tested on a large and public dataset, using algorithms that can be applied in practice to large-scale Web data.

Categories and Subject Descriptors: H.4.m [Information Systems Applications]: Miscellaneous

General Terms: Algorithms, Measurement.

Keywords: Link spam, Content spam, Web spam

1. INTRODUCTION

Traditional information retrieval algorithms were developed for relatively small and coherent document collections such as newspaper articles or book catalogs in a library. Very little, if any, of the content in such systems could be described as "spam." In comparison to these collections, the Web is massive, changes more rapidly, and is spread over geographically distributed computers [2]. Distinguishing between desirable and undesirable content in such a system presents a significant challenge, as every day more people are using search engines more often.

Search engine spamming, also known as *spamdexing*, encompasses malicious attempts to influence the outcome of ranking algorithms, for the purpose of getting an undeservedly high rank. Obtaining a higher rank is strongly correlated with traffic, and a higher rank often translates to more revenue. Thus, there is an economic incentive for Web site owners to invest on spamming search engines, instead of investing on improving their Web sites. Spamming the Web is cheap, and in many cases, successful.

Web spam is not a new problem, and is not likely to be solved in the near future. According to Henzinger et al. [17] *"Spamming has become so prevalent that every commercial search engine has had to take measures to identify and remove spam. Without such measures, the quality of the rankings suffers severely"*. Web spam damages the reputation of search engines and it weakens the trust of its users [16]. For instance, Eiron et al. [12] ranked 100 million pages using PageRank [23] and found that 11 out of the top 20 were pornographic pages, which achieved such high ranking through link manipulation, indicating that the PageRank algorithm is highly susceptible to spam. Spamming techniques are so widely known that there have been even spamming competitions (e.g., the contest to rank highest for the query "nigritude ultramarine" [11] among others).

From the perspective of the search engine, even if the spam pages are not ranked sufficiently high to annoy users, there is a cost to crawling, indexing and storing spam pages. Ideally search engines would like to avoid spam pages altogether before they use resources that might be used for storing, indexing and ranking legitimate content.

Overview of our approach. We start by building an automatic classifier that combines a set of link-based and content-based features. In general, traditional machine learning methods assume that the data instances are independent. In the case of the Web there are dependencies among pages and hosts. One such dependency is that links are not placed at random and in general, similar pages tend to be linked together more frequently than dissimilar ones [10].

Such a dependency holds also for spam pages and hosts: spam tends to be clustered on the Web. One explanation for this behavior is that spam pages often adopt link-based rank-boosting techniques such as link-farming. These techniques can be as simple as creating a pool of pages linking

SIGIR'07, July 23–27, 2007, Amsterdam, The Netherlands.

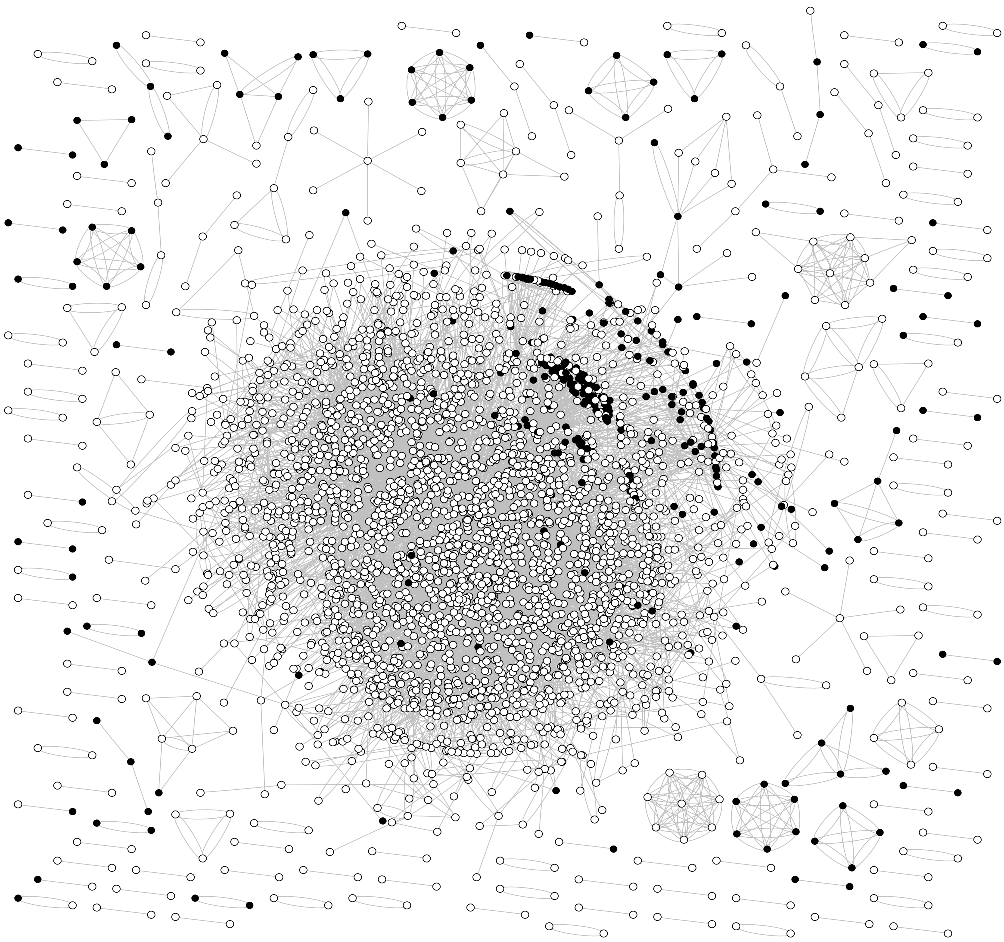

Figure 1: Graphical depiction of the hostgraph (undirected), prunned to include only labeled nodes with a connection of over 100 links between them. Black nodes are spam, white nodes are non-spam. Most of the spammers in the larger connected component are clustered together (upper-right end of the center portion). Most of the other connected components are single-class (either only spam nodes, or only non-spam nodes).

to a page whose rank is to be raised. In practice spammers use sophisticated structures that are difficult to detect.

We investigate techniques that exploit the connections between spam hosts in order to improve the accuracy of our classifiers. We assume that hosts that are well-linked together are likely to have the same class label (spam or non-spam). More generally, we can assume that two hosts in the same class should be connected by short paths going mostly through hosts in the same class.

Figure 1 shows a visualization of the host graph in the Web spam collection we are using (described in Section 3). An edge between two hosts is shown only if there are at least 100 links between the two hosts. In the figure, black nodes are spam and white nodes are non-spam. The layout of the nodes in the figure was computed using a spring model. For the larger connected component of the graph, we can see that spam nodes tend to be clustered together (in the upper right corner of the central group of nodes of Figure 1). For the nodes that are not connected to this larger connected component (or are connected by links below the threshold), we can see that most of the groups are either exclusively spam, or exclusively non-spam.

Main contributions:

– To the best of our knowledge this is the first paper that integrates link and content features for building a system to detect Web spam.

– We investigate the use of a cost sensitive classifier to exploit the inherent inbalance of labels: in the dataset we use, most of the Web content is not spam.

– We demonstrate improvements in the classification accuracy using dependencies among labels of neighboring hosts in the Web graph. We incorporate these dependencies by means of clustering and random walks.

– We apply stacked graphical learning [8] to improve the classification accuracy, exploiting the link structure among hosts in an efficient and scalable way.

The rest of this paper is organized as follows. Section 2 describes the previous work on Web Spam Detection. In Section 3 we discuss the dataset and experimental framework. Section 4 describes the features we extract. In Section 5 we present the classification algorithms and propose methods to improve their accuracy. Section 6 shows how to improve the classification accuracy by exploiting the graph topology.

Finally, Section 7 presents our conclusions and discusses future work on spam detection.

2. PREVIOUS WORK

Previous work on Web spam detection has focused mostly on the detection of three types of Web spam: link spam, content spam, and cloaking.

Link spam consists of the creation of a link structure, usually a tightly knit community of links, aimed at affecting the outcome of a link-based ranking algorithm. Methods for the detection of link-based spam rely on automatic classifiers (e.g., [4]), propagating trust or distrust through links (e.g., [13]), detecting anomalous behavior of link-based ranking algorithms [30], removing links that look suspicious for some reason (e.g., [9]), or using "bursts" of linking activity as a suspicious signal [25].

Content spam is done by maliciously crafting the content of Web pages [14], for instance, by inserting keywords that are more related to popular query terms than to the actual content of the pages. Methods for detecting this type of spam use classifiers [22] or look at language model disagreement [21]. To some extent, these techniques overlap with some of the methods used in e-mail spam filtering.

Cloaking consists of sending different content to a search engine than to the regular visitors of a web site (e.g., [27]). The version of the page that is sent to the search engine usually includes content spam, and can be detected using content-spam detection methods, or by comparing the indexed version of a page to the page that users actually see.

The link connections among spam pages has been used for propagating the "non-spam" label in TrustRank [13], or for propagating the "spam" label in BadRank [28], or for propagating both [29, 6]. In contrast, the detection of Web spam presented in this paper is based on smoothing the predictions obtained by a classification system, and not on propagating the labels themselves.

In the machine-learning literature there are many papers on using label-propagation methods (e.g., [19, 20]). Recently, some researchers have applied regularization methods for improving the accuracy of webpage classification tasks, [24, 31, 1]. For instance, Angelova and Weikum [1] improve a text classifier by exploiting link dependencies and weighting links according to text similarity.

Label-propagation methods have been tested in collections in the order of thousands of nodes, but they cannot be used directly to Web-scale data, because they need to keep the graph in main memory during the smoothing phase, which is not feasible in practice. We only use smoothing methods that can work with the graph represented as a stream in secondary memory, and can be used with Web datasets of any size. We show that even when we avoid using sophisticated regularization techniques for the sake of scalability, we still can obtain significant improvements in the classification accuracy of the base system.

3. DATASET AND FRAMEWORK

3.1 Data set

We use the `WEBSPAM-UK2006` dataset, a publicly available Web spam collection [7]. It is based on a crawl of the `.uk` domain done in May 2006, including 77.9 million pages and over 3 billion links in about 11,400 hosts.

This reference collection is tagged at the host level by a group of volunteers. The assessors labeled hosts as "normal", "borderline" or "spam", and were paired so that each sampled host was labeled by two persons independently. For the ground truth, we used only hosts for which the assessors agreed, plus the hosts in the collection marked as non-spam because they belong to special domains such as `police.uk` or `.gov.uk`.

The benefit of labeling hosts instead of individual pages is that a large coverage can be obtained, meaning that the sample includes several types of Web spam, and the useful link information among them. Since about 2,725 hosts were evaluated by at least two assessors, a tagging with the same resources at the level of pages would have been either completely disconnected (if pages were sampled uniformly at random), or would have had a much smaller coverage (if a sub set of sites were picked at the beginning for sampling pages). On the other hand, the drawback of the host-level tagging is that a few hosts contain a mix of spam/non-spam content, which increases the classification errors. Domain-level tagging would have been another option and should be explored in future work.

For the content analysis, a summary of the content of each host was obtained by taking the first 400 pages reachable by breadth-first search. The summarized sample contains 3.3 million pages. All of the content data used in the rest of this paper were extracted from a summarized version of the crawl. Note that the assessors spent on average 5 minutes per host, so the vast majority of the pages they inspected were contained in the summarized sample.

3.2 Framework

The foundation of our spam detection system is a cost-sensitive decision tree. We found this classification algorithm to work better than all other methods we tried. The features used to learn the tree were derived from a combined approach based on link and content analysis to detect different types of Web spam pages. Most of features we used were previously presented in [4, 22], but we believe that ours is the first attempt to combine both link-based and content-based features.

The features used to build the classifiers are presented in Section 4. After the initial classification, discussed in Section 5 we applied several smoothing techniques, presented in Section 6.

Evaluation. The evaluation of the overall process is based on a set of measures commonly used in Machine Learning and Information Retrieval [3]. Given a classification algorithm $\mathcal{C}$, we consider its confusion matrix:

		Prediction	
		Non-spam	Spam
True Label	Non-spam	a	b
	Spam	c	d

where a represents the number of non-spam examples that were correctly classified, b represents the number of non-spam examples that were falsely classified as spam, c represents the spam examples that were falsely classified as non-spam, and d represents the number of spam examples that were correctly classified. We consider the following measures: true positive rate (or recall), false positive rate and F-measure. The recall R is $\frac{d}{c+d}$. The false positive rate is defined as: $\frac{b}{b+a}$. The F-measure is defined as $F = 2\frac{PR}{P+R}$, where P is the precision $P = \frac{d}{b+d}$.

For evaluating the classification algorithms, we focus on the F-measure F as it is a standard way of summarizing both precision and recall. We also report the true positive rate and false positive rate as they have a direct interpretation in practice. The true positive rate R is the amount of spam that is detected (and thus deleted or demoted) by the search engine. The false positive rate is the fraction of non-spam objects that are mistakenly considered to be spam by the automatic classifier.

Cross-validation. All the predictions reported in the paper were computed using tenfold cross validation. For each classifier we report the true positives, false positives and F-measure. A classifier whose prediction we want to estimate, is trained 10 times, each time using the 9 out of the 10 partitions as training data and computing the confusion matrix using the tenth partition as test data. We then average the resulting ten confusion matrices and estimate the evaluation metrics on the average confusion matrix.

The clustering and propagation algorithms in Sections 6.2 and 6.3 operate on labels assigned by a base classifier. To avoid providing these algorithms with an oracle, by passing along the true labels we do the following: (1) For each *unlabeled data instance* we pass to the clustering and propagation algorithms the label predicted by the base classifier. (2) For each *labeled data instance* we pass the label predicted by the baseline classifier when the data instance was in the test partition.

4. FEATURES

We extract link-based features from the Web graph and host graph, and content-based features from individual pages. For the link-based features we follow Becchetti et al. [4], and for the content-based features, Ntoulas et al. [22].

4.1 Link-based features

Most of the link-based features are computed for the home page and the page in each host with the maximum PageRank. The remainder of link features are computed directly over the graph between hosts (obtained by collapsing together pages of the same host).

Degree-related measures. We compute a number of measures related to the *in-degree* and *out-degree* of the hosts and their neighbors. In addition, we consider various other measures, such as the *edge-reciprocity* (the number of links that are reciprocal) and the *assortativity* (the ratio between the degree of a particular page and the average degree of its neighbors). We obtain a total of 16 degree-related features.

PageRank. PageRank [23] is a well known link-based ranking algorithm that computes a score for each page. We compute various measures related to the PageRank of a page and the PageRank of its in-link neighbors. We obtained a total of 11 PageRank-based features.

TrustRank. Gyöngyi et al. [15] introduced the idea that if a page has high PageRank, but it does not have any relationship with a set of known trusted pages then it is likely to be a spam page. TrustRank [15] is an algorithm that, starting from a subset of hand-picked trusted nodes and propagating their labels through the Web graph, estimates a *TrustRank score* for each page. Using TrustRank we can also estimate the *spam mass* of a page, i.e., the amount of PageRank received from a spammer. The performance of TrustRank depends on the seed set, in our case we used 3,800 nodes chosen at random from the Open Directory Project, excluding those that were labeled as spam. We observed that the relative non-spam mass for the home page of each host (the ratio between the TrustRank score and the PageRank score) is a very effective measure for separating spam from non-spam hosts. However, using this measure alone is not sufficient for building an automatic classifier because it would yield a high number of false positives (around the 25%).

Truncated PageRank. Becchetti et al. [5] described Truncated PageRank, a variant of PageRank that diminishes the influence of a page to the PageRank score of its neighbors.

Estimation of supporters. Given two nodes x and y, we say that x is a *d-supporter* of y, if the shortest path from x to y has length d. Let $N_d(x)$ be the set of the d-supporters of page x. An algorithm for estimating the set $N_d(x)$ for all pages x based on probabilistic counting is described in [5]. For each page x, the cardinality of the set $N_d(x)$ is an increasing function with respect to d. A measure of interest is the *bottleneck number* $b_d(x)$ of page x, which we define to be $b_d(x) = \min_{j \le d}\{|N_j(x)|/|N_{j-1}(x)|\}$. This measure indicates the minimum rate of growth of the neighbors of x up to a certain distance. We expect that spam pages form clusters that are somehow isolated from the rest of the Web graph and they have smaller bottleneck numbers than non-spam pages. Figure 2 shows a histogram of $b_4(x)$ for spam and non-spam pages. For most of the non-spam pages, the bottleneck number is around 2.2, while for many of the spam pages it is between 1.3 and 1.7.

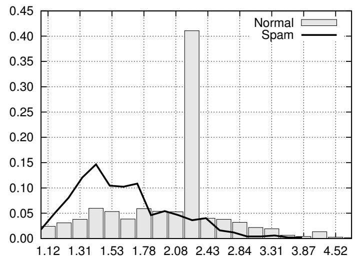

Figure 2: Histogram of the minimum ratio change of the # of neighbors from distance i to distance $i-1$

4.2 Content-based features

For each web page in the data set we extract a number of features based on the content of the pages. We use most of the features reported by Ntoulas et al. [22], with the addition of new ones such as the *entropy* (see below), which is meant to capture the compressibility of the page. Ntoulas et al. [22] use a set of features that measures the precision and recall of the words in a page with respect to the set of the most popular terms in the whole web collection. Motivated by this idea, we add a new set of features that measures the precision and recall of the words in a page with respect to the q most frequent terms from a query log, where $q = 100, 200, 500, 1000$.

Number of words in the page, number of words in the title, average word length. For these features we count only the words in the visible text of a page, and we consider words consisting only of alphanumeric characters.

Fraction of anchor text. Fraction of the number of words in the anchor text to the total number of words in the page.

Fraction of visible text. Fraction of the number of words in the visible text to the total number of words in the page, include `html` tags and other invisible text.

Compression rate. We compress the visible text of the page using `bzip`. Compression rate is the ratio of the size of the compressed text to the size of the uncompressed text.

Corpus precision and corpus recall. We find the k most frequent words in the dataset, excluding stopwords. We call *corpus precision* the fraction of words in a page that appear in the set of popular terms. We define *corpus recall* to be the fraction of popular terms that appear in the page. For both corpus precision and recall we extract 4 features, for $k = 100, 200, 500$ and 1000.

Query precision and query recall. We consider the set of q most popular terms in a query log, and query precision and recall are analogous to corpus precision and recall. Our intuition is that spammers might use terms that make up popular queries. As with corpus precision and recall, we extract eight features, for $q = 100, 200, 500$ and 1000.

Independent trigram likelihood. A trigram is three consecutive words. Let $\{p_w\}$ be the probability distribution of trigrams in a page. Let $T = \{w\}$ be the set of all trigrams in a page and $k = |T(p)|$ be the number of distinct trigrams. Then the independent trigram likelihood is a measure of the independence of the distribution of trigrams. It is defined in Ntoulas et al. [22] to be $-\frac{1}{k}\sum_{w\in T}\log p_w$.

Entropy of trigrams. The entropy is another measure of the compressibility of a page, in this case more macroscopic than the compressibility ratio feature because it is computed on the distribution of trigrams. The entropy of the distribution of trigrams, $\{p_w\}$, is defined as $H = -\sum_{w\in T} p_w \log p_w$.

The above list gives a total of 24 features for each page. In general we found that, for this dataset, the content-based features do not provide as good separation between spam and non-spam pages as for the data set used in Ntoulas et al. [22]. For example, we found that in the dataset we are using, the distribution of average word length in spam and non-spam pages were almost identical. In contrast, for the data set of Ntoulas et al. [22] that particular feature provides very good separation. The same is true for many of the other content features. Some of the best features (judging only from the histograms) are the corpus precision and query precision, which is shown in Figure 3.

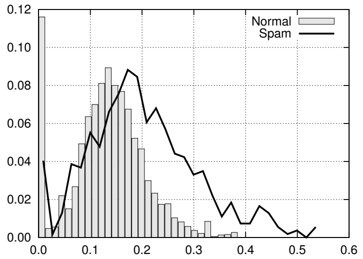

Figure 3: Histogram of the query precision in non-spam vs. spam pages for $k = 500$.

4.3 From page features to host features

In total, we extract 140 features for each host (described in Section 4.1) and 24 features for each page (described in Section 4.2). We then aggregate the content-based features of pages in order to obtain content-based features for hosts.

Let h be a host containing m web pages, denoted by the set $P = \{p_1, \ldots, p_m\}$. Let $\widehat{p}$ denote the home page of host h and p^* denote the page with the largest PageRank among all pages in P. Let $c(p)$ be the 24-dimensional content feature vector of page p. For each host h we form the content-based feature vector $c(h)$ of h as follows

$$c(h) = \langle c(\widehat{p}), c(p^*), \mathbf{E}[c(p)], \mathbf{Var}[c(p)]\rangle.$$

Here $\mathbf{E}[c(p)]$ is the average of all vectors $c(p)$, $p \in P$, and $\mathbf{Var}[c(p)]$ is the variance of $c(p)$, $p \in P$. Therefore, for each host we have $4 \times 24 = 96$ content features. In total, we have $140 + 96 = 236$ link and content features.

In the process of aggregating page features, we ignore hosts h for which the home page $\widehat{p}$ or the maxPR page p^* is not present in our summary sample. This leaves us with a total of 8,944 hosts, out of which 5,622 are labeled; from them, 12% are spam hosts.

5. CLASSIFIERS

We used as the base classifier the implementation of C4.5 (decision trees) given in Weka [26]. Using both link and content features, the resulting tree used 45 unique features, of which 18 are content features.

In the data we use, the non-spam examples outnumber the spam examples to such an extent that the classifier accuracy improves by misclassifying a disproportionate number of spam examples. At the same time, intuitively, the penalty for misclassifying spam as normal is not equal to the penalty for misclassifying normal examples as spam. To minimize the misclassification error we used a cost-sensitive decision tree. We imposed a cost of zero for correctly classifying the instances, and set the cost of misclassifying a spam host as normal to be R times more costly than misclassifying a normal host as spam. Table 1 shows the results for different values of R. The value of R becomes a parameter that can be tuned to balance the true positive rate and the false positive rate. In our case, we wish to maximize the F-measure. Incidentally note that $R = 1$ is equivalent to having no cost matrix, and is the baseline classifier.

Table 1: Cost-sensitive decision tree

Cost ratio (R)	1	10	20	30	50
True positive rate	64.0%	68.0%	75.6%	80.1%	87.0%
False positive rate	5.6%	6.8%	8.5%	10.7%	15.4%
F-Measure	0.632	0.633	**0.646**	0.642	0.594

We then try to improve the results of the baseline classifier using bagging. Bagging is a technique that creates an ensemble of classifiers by sampling with replacement from the training set to create N classifiers whose training sets contain the same number of examples as the original training set, but may contain duplicates. The labels of the test set are determined by a majority vote of the classifier ensemble. In general, any classifier can be used as a base classifier, and in our case we used the cost-sensitive decision trees described above. Bagging improved our results by reducing the false-positive rate, as shown in Table 2. The decision tree created by bagging was roughly the same size as the tree created without bagging, and used 49 unique features, of which 21 were content features.

Table 2: Bagging with a cost-sensitive decision tree

Cost ratio (R)	1	10	20	30	50
True positive rate	65.8%	66.7%	71.1%	78.7%	84.1%
False positive rate	2.8%	3.4%	4.5%	5.7%	8.6%
F-Measure	0.712	0.703	0.704	**0.723**	0.692

The results of classification reported in Tables 1 and 2 use both link and content features. Table 3 shows the contribution of each type of feature to the classification. The content features serve to reduce the false-positive rate, without diminishing the true positive result, and thus improve the overall performance of the classifier. The classifier that serves as the foundation for future experiments paper uses bagging with a cost-sensitive decision tree, where $R = 30$.

Table 3: Comparing link and content features

	Both	Link-only	Content-only
True positive rate	78.7%	79.4%	64.9%
False positive rate	5.7%	9.0%	3.7%
F-Measure	**0.723**	0.659	0.683

6. SMOOTHING

In this section we describe a different usage of the link structure of the graph than the one presented in Section 4. During the extraction of link-based features, all nodes in the network were anonymous, while in this regularization phase, the identity (the predicted label) of each node is known, and important to the algorithm.

6.1 Topological dependencies of spam nodes

Before describing how to use the aggregation of spam hosts to improve the accuracy of spam detection, we provide experimental evidence for the following two hypotheses:

> Non-spam nodes tend to be linked by very few spam nodes, and usually link to no spam nodes.
>
> Spam nodes are mainly linked by spam nodes.

Examining the out-link and the in-link graphs separately, we count the number of spam hosts contained in the adjacency list of each one of the hosts.

Let $S_{\mathrm{OUT}}(x)$ be the fraction of spam hosts linked by host x out of all labeled hosts linked by host x. Figure 4(a) shows the histogram of S_{OUT} for spam and non-spam hosts. We see that almost all non-spam hosts link mostly to non-spam hosts. The same is not true for spam hosts, which tend to link both spam and non-spam hosts.

Similarly, let $S_{\mathrm{IN}}(x)$ be the fraction of spam hosts that link to host x out of all labeled hosts that link to x. Figure 4(b) shows the histograms of S_{IN} for spam and non-spam hosts. In this case there is a clear separation between spam and non-spam hosts. The general trend is that spam hosts are linked mostly by other spam hosts. More than 85% of the hosts have an S_{IN} value of more than 0.7. On the other hand, the opposite is true for non-spam hosts; more than 75% of the non-spam hosts have an S_{IN} value of less than 0.2.

(a) Spam nodes in out-links **(b)** Spam nodes in in-links

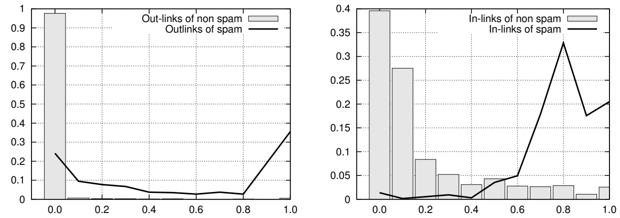

Figure 4: Histogram of the fraction of spam hosts in the links of non-spam or spam hosts.

6.2 Clustering

We use the result of a graph clustering algorithm to improve the prediction obtained from the classification algorithm. Intuitively, if the majority of a cluster is predicted to be spam then we change the prediction for *all* hosts in the cluster to spam. Similarly if the majority of a cluster is predicted to be non-spam then we predict that *all* hosts in this cluster are non-spam.

We consider the undirected graph $G = (V, E, w)$ where V is the set of hosts, w is a weighting function from $V \times V$ to integers so that the weight $w(u, v)$ is equal to the the number of links between any page in host u and any page in host v, and E is the set of edges with non-zero weight. Ignoring the direction of the links may result in a loss of information for detecting spam, but it drastically simplifies the graph clustering algorithm.

We cluster the graph G using the METIS graph clustering algorithm algorithm [18].[1] We partition the 11400 hosts of the graph into 1000 clusters, so as to split the graph into many small clusters. We found that the number of clusters is not crucial, and we obtained similar results for partitioning the graph in 500 and 2000 clusters.

The clustering algorithm can be described as follows. Let the clustering of G consist of m clusters $C_1, \ldots, C_m$, which form a disjoint partition of V. Let $p(h) \in [0..1]$ be the prediction of a particular classification algorithm $\mathcal{C}$ so that for each host h a value of $p(h)$ equal to 0 indicates non-spam, while a value of 1 indicates spam. (Informally, we call $p(h)$ the *predicted spamicity* of host h) For each cluster C_j, $j = 1, \ldots, m$, we compute its *average spamicity* $p(C_j) = \frac{1}{|C_j|}\sum_{h \in C_j} p(h)$. Our algorithm uses two thresholds, a lower threshold t_l and an upper threshold t_u. For each cluster C_j if $p(C_j) \leq t_l$ then all hosts in C_j are marked as non-spam, and $p(h)$ is set to 0 for all $h \in C_j$. Similarly, if $p(C_j) \geq t_u$ then all hosts in C_j are marked as spam, and $p(h)$ is set to 1. The results of the clustering algorithm are shown in Table 4. The improvement of the F-measure obtained over classifier without bagging is statistically significant at the 0.05 confidence level; the improvement for the classifier with bagging is much smaller. Note that this algorithm never has access to the true labels of the data, but uses only predicted labels, as explained at the end of section 3.2.

We considered variations of the above algorithm that change the labels of only clusters larger than a given size threshold, or of clusters of a small relative *cut* (the ratio of the weight of edges going out of the cluster with respect to weight of edges inside the cluster). However, none of these variations

[1] `http://glaros.dtc.umn.edu/gkhome/views/metis/`

Table 4: Results of the clustering algorithm

	Baseline	Clustering
Without bagging		
True positive rate	75.6%	74.5%
False positive rate	8.5%	6.8%
F-Measure	0.646	**0.673**
With bagging		
True positive rate	78.7%	76.9%
False positive rate	5.7%	5.0%
F-Measure	0.723	**0.728**

yielded any noticable improvement over the performance of the basic algorithm, so we do not discuss them in more detail. We note that the implementation of the clustering algorithm we use might not be scalable to arbitrarily large web collections. For such data one might want to use more efficient clustering algorithms, for instance, pruning edges below a threshold and finding connected components.

6.3 Propagation

We use the graph topology to smooth the predictions by propagating them using random walks, following Zhou et al. [32]. We simulate a random walk starting from the nodes that the base classifier has marked as spam, following a link with probability α, and returning to a spam node with probability $1-\alpha$. When returning to a spam node, the node to return to is picked with probability proportional to its "spamicity" prediction. After this process was run, we used the training part of the data to learn a threshold parameter, and used this threshold to classify the testing part as non-spam or spam.

We tried three forms of this random walk: on the host graph, on the transposed host graph (meaning that the activation goes backwards) and on the undirected host graph. We tried different values of the α parameter and got improvements over the baseline with $\alpha \in [0.1, 0.3]$, implying short chains in the random walk. In Table 5 we report on the results when $\alpha = 0.3$, after 10 iterations (this was enough for convergence in this hostgraph).

Table 5: Result of applying propagation

	Baseline	Fwds.	Backwds.	Both
Classifier without bagging				
True positive rate	75.6%	70.9%	69.4%	71.4%
False positive rate	8.5%	6.1%	5.8%	5.8%
F-Measure	0.646	0.665	0.664	**0.676**
Classifier with bagging				
True positive rate	78.7%	76.5%	75.0%	75.2%
False positive rate	5.7%	5.4%	4.3%	4.7%
F-Measure	0.723	0.716	**0.733**	0.724

As shown in Table 5, the classifier without bagging can be improved (and the improvement is statistically significant at the 0.05 confidence level), but the increase of accuracy for the classifier with bagging is small.

6.4 Stacked graphical learning

Stacked graphical learning is a meta-learning scheme described recently by Cohen and Kou [8]. It uses a base learning scheme $\mathcal{C}$ to derive initial predictions for all the objects in the dataset. Then it generates a set of extra features for each object, by combining the predictions for the related objects in the graph. Finally, it adds this extra feature to the input of $\mathcal{C}$, and runs the algorithm again to get new, hopefully better, predictions for the data.

Let $p(h) \in [0..1]$ be the prediction of a particular classification algorithm $\mathcal{C}$ as described above. Let $r(h)$ be the set of pages related to h in some way. We compute:

$$f(h) = \frac{\sum_{g \in r(h)} p(g)}{|r(h)|}$$

Next, we add $f(h)$ as an extra feature for instance h in the classification algorithm $\mathcal{C}$, and run the algorithm again. This process can be repeated many times, but most of the improvement is obtained with the first iteration.

Table 6 reports the results of applying stacked graphical learning, by including one extra feature with the average predicted spamicity of $r(h)$. For the set $r(h)$ of pages related to h we use either the in-links, the out-links or both.

Table 6: Results with stacked graphical learning

	Baseline	Avg. of in	Avg. of out	Avg. of both
True positive rate	78.7%	84.4%	78.3%	85.2%
False positive rate	5.7%	6.7%	4.8%	6.1%
F-Measure	0.723	0.733	0.742	**0.750**

We observe that there is an improvement over the baseline, and the improvement is more noticeable when using the entire neighborhood of the host as an input. The improvement is statistically significant at the 0.05 confidence level. In comparison with the other techniques we studied, this method is able to significantly improve even the classifier with bagging.

A second pass of stacked graphical learning yields an even better performance; the false positive rate increases slightly but the true positive rate increases by almost 3%, compensating for it and yielding a higher F-measure. The feature with the highest information gain is the added feature, and so serves as a type of summary of other features. With the added feature, the resulting decision tree is smaller, and uses fewer link features; the tree uses 40 features, of which 20 are content features. Consistently with [8], doing more iterations does not improve the accuracy of the classifier significantly.

Table 7: Second pass of stacked graphical learning

	Baseline	First pass	Second pass
True positive rate	78.7%	85.2%	88.4%
False positive rate	5.7%	6.1%	6.3%
F-Measure	0.723	0.750	**0.763**

The improvement of the F-Measure from 0.723 to 0.763 is of about 5.5%, and this actually translates to a large improvement in the accuracy of the classifier. The smoothing techniques we use improve the detection rate from 78.7% to 88.4%, while the error rate grows by less than one percentage point, from 5.7% to 6.3%.

7. CONCLUSIONS

There is a clear tendency of spammers to be linked together and this tendency can be exploited by search engines to detect spam more accurately.

There is a lot of related work on spam detection, however, we can compare our results with previous results only indirectly. The reason is that the majority of prior research on Web spam has been done on data sets that are not public. With respect to using content only, we note that our set of content-based features includes the the set described in Ntoulas et al. [22]. In their paper, they report an F-Measure of 0.877 [22, Table 2] using a classifier with bagging. Using essentially the same technique, we obtain a performance that is much lower than theirs (our F-Measure is 0.683 in Table 3 compared to their 0.877). This is consistent with the observation presented at the end of Section 4.2, that is, the content of the spam pages in the dataset we use resembles much more closely the content of non-spam pages.

Similarly, our link-based features were extracted from a public dataset used previously by Becchetti et al. [4], and again, the accuracy of the same classifier is much lower on their new dataset (our F-Measure is 0.683 compared to the previous F-Measure of 0.879), supporting the conclusion that distinguishing between spam and non-spam is inherently more difficult in the new data. Nevertheless, our best classifier detects 88.4% of the spam hosts with 6.3% false positives. If the error rate is too high for the application or search engine policies at hand, it can be adjusted by adjusting the cost matrix, for more conservative spam filtering (at the expense of a lower detection rate).

The graph-based algorithms we use operate over the graph on a stream way, using a small number of passes over the Web graph that is kept on secondary memory, and keeping only a small amount of data in main memory. This means that the system we have proposed can be used in large Web datasets of any size.

The connected nature of spam on the Web graph suggests that the use of regularization is a promising area of future work. While we explored such regularization methods in isolation and we assess their performance independently, an interesting direction for future work is to combine the regularization methods at hand in order to improve the overall accuracy.

Acknowledgements

We thank the support of the EU within the 6th Framework Programme under contract 001907 "Dynamically Evolving, Large Scale Information Systems" (DELIS). This work has been done while Fabrizio Silvestri was visiting the Yahoo! Research Laboratories in Barcelona.

8. REFERENCES

[1] R. Angelova and G. Weikum. Graph-based text classification: learn from your neighbors. In *ACM SIGIR*, pages 485–492, 2006.

[2] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. Searching the web. *ACM Transactions on Internet Technology (TOIT)*, 1(1):2–43, 2001.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999.

[4] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Link-based characterization and detection of Web Spam. In *AIRWeb*, 2006.

[5] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Using rank propagation and probabilistic counting for link-based spam detection. In *ACM WebKDD*, Pennsylvania, USA, August 2006.

[6] A. Benczúr, K. Csalogány, and T. Sarlós. Link-based similarity search to fight web spam. In *AIRWeb*, 2006.

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

[8] W. W. Cohen and Z. Kou. Stacked graphical learning: approximating learning in markov random fields using very short inhomogeneous markov chains. Technical report, 2006.

[9] A. L. da Costa-Carvalho, P.-A. Chirita, E. S. de Moura, P. Calado, and W. Nejdl. Site level noise removal for search engines. In *WWW*, pages 73–82, New York, NY, USA, 2006.

[10] B. D. Davison. Topical locality in the web. In *ACM SIGIR*, pages 272–279, Athens, Greece, 2000.

[11] I. Drost and T. Scheffer. Thwarting the nigritude ultramarine: learning to identify link spam. In *ECML*, volume 3720 of *LNAI*, pages 233–243, Porto, Portugal, 2005.

[12] N. Eiron, K. S. Curley, and J. A. Tomlin. Ranking the web frontier. In *WWW*, pages 309–318, New York, NY, USA, 2004.

[13] Z. Gyöngyi and H. Garcia-Molina. Link spam alliances. In *VLDB*, 2005.

[14] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *AIRWeb*, 2005.

[15] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating Web spam with TrustRank. In *VLDB*, 2004.

[16] Z. Gyöngyi and H. G. Molina. Spam: It's not just for inboxes anymore. *IEEE Computer Magazine*, 38(10):28–34, 2005.

[17] M. R. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. *ACM SIGIR Forum*, 36(2):11–22, 2002.

[18] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.

[19] Q. Lu and L. Getoor. Link-based classification using labeled and unlabeled data. In *ICML Workshop on The Continuum from Labeled to Unlabeled Data*, Washington, DC, 2003.

[20] S. A. Macskassy and F. Provost. Suspicion scoring based on guilt-by-association, collective inference, and focused data access. In *International Conference on Intelligence Analysis*, 2005.

[21] G. Mishne, D. Carmel, and R. Lempel. Blocking blog spam with language model disagreement. In *AIRWeb*, 2005.

[22] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In *WWW*, pages 83–92, Edinburgh, Scotland, May 2006.

[23] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[24] X. Qi and B. D. Davison. Knowing a web page by the company it keeps. In *CIKM*, pages 228–237, Arlington, VA, USA, November 2006.

[25] G. Shen, B. Gao, T.-Y. Liu, G. Feng, S. Song, and H. Li. Detecting link spam using temporal information. In *ICDM*, Hong Kong, December 2006.

[26] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.

[27] B. Wu and B. D. Davison. Cloaking and redirection: A preliminary study. In *AIRWeb*, 2005.

[28] B. Wu and B. D. Davison. Identifying link farm spam pages. In *WWW*, pages 820–829, New York, NY, USA, 2005.

[29] B. Wu, V. Goel, and B. D. Davison. Propagating trust and distrust to demote web spam. In *MTW*, May 2006.

[30] H. Zhang, A. Goel, R. Govindan, K. Mason, and B. Van Roy. Making eigenvector-based reputation systems robust to collusion. In *WAW*, volume 3243 of *LNCS*, pages 92–104, Rome, Italy, 2004. Springer.

[31] T. Zhang, A. Popescul, and B. Dom. Linear prediction models with graph regularization for web-page categorization. In *ACM KDD*, pages 821–826, New York, NY, USA, 2006.

[32] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16:321–328, 2004.