A Semi-Supervised Approach for Web Spam Detection using Combinatorial Feature-Fusion

Ye Tian, Gary M. Weiss, Qiang Ma

Department of Computer and Information Science
Fordham University
441 East Fordham Road
Bronx, NY 10458
{tian,gweiss,ma}@cis.fordham.edu

Abstract: This paper describes a machine learning approach for detecting web spam. Each example in this classification task corresponds to 100 web pages from a host and the task is to predict whether this collection of pages represents spam or not. This task is part of the 2007 ECML/PKDD Graph Labeling Workshop's Web Spam Challenge (track 2). Our approach begins by adding several human-engineered features constructed from the raw data. We then construct a rough classifier and use semi-supervised learning to classify the unlabelled examples provided to us. We then construct additional link-based features and incorporate them into the training process. We also employ a *combinatorial feature-fusion method* for "compressing" the enormous number of word-based features that are available, so that conventional machine learning algorithms can be used. Our results demonstrate the effectiveness of semi-supervised learning and the combinatorial feature-fusion method.

Keywords: feature construction, classification, link mining, information fusion, class imbalance.

1 Introduction

Search engines perform a vital role in permitting users to quickly and efficiently find information on the World Wide Web. Because web pages that are pointed to by many other web pages are favored by most search engines, spam web pages, which are created for the sole purpose of influencing search engine results, have become quite common. If the quality of search engine results is to be preserved, automated means of identifying these spam pages must be developed. The ECML/PKDD Graph Labeling Workshop's Web Spam Challenge supports the development of these methods, by providing a forum for researchers to develop, evaluate, and compare the effectiveness of a variety of methods. The second track of this challenge focuses on machine learning methods for identifying web spam, and in this paper we describe our machine learning entry for that challenge.

Our machine learning approach starts by supplementing the raw data provided by the Web Spam Challenge with additional, human-engineered content-based [1] and link-based [2] features. We then build a rough classifier using the raw data and these supplemental features, and use this classifier to label all of the unlabeled graph nodes provided by the Web Spam Challenge. Thus, we perform semi-supervised learning. We use the actual labels for the training data and the predicted labels for the other nodes to generate additional link-based features. In addition to the use of semi-supervised learning, we also use a combinatorial feature-fusion method, described in previous work [3,4], to “compress” the enormous number of content-based features, so that conventional classifier induction algorithms, which do not handle sparse features, can be used. Our results indicate that both semi-supervised learning and combinatorial feature fusion are effective at improving web spam detection.

This paper is organized as follows. In Section 2 we provide background on the combinatorial feature-fusion method. Then in Section 3 we describe the Web Spam Challenge data and the features that we manually engineer in order to improve the performance of our system. Section 4 describes our experimental methodology, including the feature-fusion method and use of semi-supervised learning. Our results from the Web Spam Challenge are then presented in Section 5. Finally, we describe our conclusions and areas for future work in Section 6.

2 Background on the Combinatorial Feature Fusion Method

In this section we describe the terminology and concepts associated with the combinatorial feature-fusion method, described in detail in previous work [3,4]. We use a simple example to describe the method. Our feature fusion method currently only handles numeric features, but this is not an issue for the Web Spam Challenge.

A data set is made up of examples, or records, each of which has a fixed number of features. Consistent with previous work on information fusion [3], we view the value of a feature as a *score*. Table 1 introduces a sample data set, with the score values in Table 1a replaced by rank values in Table 1b. The ranks in Table 1b have been computed in the straightforward manner, where a low score yields a low rank. This will not always be the case. As we will shortly see, the higher scores will receive lower ranks if this ranking scheme yields improved predictive performance.

Table 1. A sample data set with the data (a) unmodified and (b) with score values replaced by rank. The original data set contains eight examples, labeled A-H, with five numeric features and a binary class variable. In this example class 1 is the minority class and accounts for 3/8, or 37.5%, of the examples.

	F1	F2	F3	F4	F5	Class
A	1	4	3	2	8	1
B	3	3	5	5	4	0
C	5	5	2	6	7	1
D	7	6	15	3	2	0
E	11	13	16	7	14	0
F	15	16	4	13	11	0
G	9	7	14	1	18	1
H	17	15	9	8	3	0

(a)

	F1	F2	F3	F4	F5
A	1	2	2	2	5
B	2	1	4	4	3
C	3	3	1	5	4
D	4	4	7	3	1
E	6	6	8	6	7
F	7	8	3	8	6
G	5	5	6	1	8
H	8	7	5	7	2

(b)

Next we show how to compute the performance of a feature, using feature F2 from the sample data set as an example. For our purposes, the performance of a feature indicates how well its rank performs at predicting minority-class examples. The records in the data set are sorted by the rank value of F2 and the results are shown in Table 2a. The performance of F2 is then computed as the fraction of the records at the "top" of the table that belong to the minority class. The number of records considered is based on the percentage of minority-class examples in the training data, so in this case we look at the top 37.5%, or 3, records. In this case the performance of F2 is 2/3. Table 2b shows the performance values for all of the features.

Table 2. The results of ordering the examples by the rank of F2 is shown in Table 2a. The top three records are then used to compute the performance of F2, which in this case is 2/3, since 2 of the 3 records are associated with the minority class. The performance of all five feataures is provided in Table 2b.

	F2 Rank	Class
B	1	0
A	2	**1**
C	3	**1**
D	4	0
G	5	1
E	6	0
H	7	0
F	8	0

(a)

Feature	Performance
F1	0.67
F2	0.67
F3	0.67
F4	0.67
F5	0

(b)

This method is also used to compute the performance of combined (i.e., fused) features. However, to do this we need to determine the rank of a fused feature, so we can sort the examples by this rank. We compute this using a *rank combination function*, which averages the ranks of the features to be combined. This is done for each record. As an example, if we want to fuse features F1–F5 and create a new feature, then the rank of this fused feature for record A is computed as: (rank(F1) + rank(F2) + rank(F3) + rank(F4) + rank(F5))/5 = (1+2+2+2+5)/5 = 2.4. Once the rank values are computed, the performance value can be computed as before.

3 Data and Data Engineering

In this section we describe the raw data provided to us as part of the Web Spam Challenge and then describe the features that we design/engineer based on this raw data. We do not discuss the features automatically generated by our feature-fusion method or the link-based features generated as part of semi-supervised learning until Section 4.

The data utilized in this paper was provided as part of the Web Spam Challenge (track II, Corpus #1), held in conjunction with the 2007 ECML/PKDD Graph Labeling workshop. The data describes the content and link structure of about 9,000 examples, where each example describes 100 web pages from an Internet host. Each example can be viewed as a node in a graph, where nodes are connected if there are hyperlinks between them. The data is physically distributed over the following four data sets:

1. Feature Vectors: these vectors correspond to the TF-IDF vectors over the 100 web pages for a host. Thus this data set contains word frequency information. These are sparse vectors in that if a word does not occur, then it is not represented in the vector.
2. Link Matrix: each non-zero entry in this data set represents an edge in the graph and thus determines which nodes are connected by hyperlinks.
3. Training Labels: identifies the class label (spam or normal) for each node in the training set.
4. Validation Labels: identifies the class label (spam or normal) for each node in the "validation" set.

The raw data from the first two data sets are used to generate the examples for our classifier (we discuss this shortly). The training labels data set determines which examples are used for training while the validation labels data set determines which examples are used to evaluate the classifier to generate our preliminary results. The training data consists of 907 examples (hosts) and the validation data set contains 1800 examples. The test set labels are maintained by the Web Spam Challenge organizers, who use these labels to score the classifier results that are submitted by the challenge competitors. The class distribution of data, for both the training and validation set, is approximately 80% normal and 20% spam. This data is skewed, which can lead to some difficulties when learning a predictive model [5]. Since the training and validation data sets contain a total of 2,707 examples, 6,365 of the 9,072 examples are left unclassified. These examples could be ignored, but we use semi-supervised learning [6] to exploit them and improve our predictive model.

We manually engineered (i.e., constructed) seven features from the raw Web Spam Challenge data. Each feature is associated with a node/host and is computed based on information associated with that node/host. Table 3 summarizes the engineered features used in this study. Note hotwords are the words (i.e., content-based features) that appear in the greatest percentage of the 9,072 nodes provided in the Web Challenge Data. In order to limit the number of features for consideration we track only the top 500 hotwords (this may be too restrictive and should be increased in future work).

Table 3. Summary of Engineered Features. The first three features are content-based features and the remaining featurs are link-based features.

1	%HotwordsCovered	Percentage of the 500 hotwords found in the node
2	%Hotwords	Percentage of unique words in a node that are hotwords
3	TFIDF-Above-0.2	1 if any of the TF-IDF values is above 0.2; 0 otherwise
4	InboundLinks	The number of inbound links to this host
5	OutboundLinks	The number of outbound links from this host
6	InboundFraction	The fraction of total links that are inbound links
7	OutboundFraction	The fraction of total links that are outbound links

4 Experimental Methodology

We describe our experiments in this section. In Section 4.1 we describe the learning algorithms that we employ. Then in Section 4.2 we describe the basic procedure for encoding the examples for our learning problem. In Section 4.3 we describe how we generate new link-based features by using semi-supervised learning and in Section 4.4 we describe how we use combinatorial feature-fusion to construct new features.

4.1 Learning Algorithm

In this paper we evaluate three classifier induction algorithms, which are part of the Weka data mining package [7]. These algorithms are ADTree, an alternating decision tree algorithm [8], SMO, an implementation of a support vector machine, and Bayes, an implementation of a naïve Bayes classifier. Note that for alternating decision trees the ultimate classification is determined by multiple paths through the tree rather than a single path through the tree. Since our preliminary results showed that ADTree performed best, only classifiers induced using this algorithm were used as part of the official Web Spam Challenge.

4.2 Example Generation

The examples for the training and validation data sets start with the features included in the feature vector data set described in Section 3. These include the TF-IDF values for the content-based features. Since this information is provided using a sparse representation and our machine learning methods do not handle sparse representations of features, we insert null entries for the missing features. Because our learning algorithms can not handle the enormous number of resulting content-based features, we prune all but the top 200 such features based on the performance values produced by our feature-fusion method on the training data, as described in Section 2. This should keep the features that are best able to predict web spam, since the performance metric is based on the ability to predict the minority-class examples. Next, we join these 200 features with the seven engineered features listed in Table 3. We then add the class labels for the training and validation data using the class information provided as part of the Web Spam Challenge.

4.3 Semi-Supervised Learning

Once the examples are generated as described in Section 4.2, we use the training data to build a "rough" classifier. This is then used to classify the nodes *not* in the training set. We then use the predicted class labels for these non-training set nodes and the actual class labels for the training set nodes to determine, for each link in the link matrix, whether the inbound or outbound side is spam or normal. From this we construct four new link-based features, which indicate the prevalence of spam for the neighbors of a node. These features represent, for each node, the number of inbound (outbound) spam links and the percentage of inbound (outbound) links that are spam.

We could use these four newly constructed features to build our final model, but instead choose to include an additional round of semi-supervised learning, on the assumption that it will yield superior results. That is, we train a classifier using all previous features plus these four new ones and then *again* predict the class labels for the nodes that are not in the training set. We then recompute these four link-based features using the updated labels. These link-based feature values are then included for use in building our final classification model, once the combinatorial feature-fusion method described in Section 4.4 is used to construct some additional features.

4.4 Use of Combinatorial Feature-Fusion

Next we use our combinatorial feature-fusion strategy to introduce new "fused" features. Section 2 described how to fuse features and evaluate their performance, but did not discuss how we decide which features to fuse and how to determine which of these fused features to keep. In previous work we tried a variety of fusion strategies [4], but in this paper we simply fuse all pairs of features. Given that we have a total of 211 features, this leads to C(211,2), or 22,155, possible pairings. While restricting the fusion to only two features is limiting, we will see that it still improves classifier performance.

We next decide which of the fused features to keep. We order the fused features by their performance value and then tentatively add them in one by one. After adding each feature, we regenerate the classifier from the training data and compare the performance of the classifier on a hold-out set (selected from the training data) with the performance prior to adding the feature. If the feature yields an improvement in performance with a t-test confidence of at least .9, then we keep the feature; otherwise we discard it. Since we measure classifier performance using AUC and the F-measure, for the purpose of deciding whether to keep a feature or not we measure performance using the average value of AUC and the F-measure.

5 Results

In this section we report our preliminary results on the validation data provided to us, using three classifiers, and then present the official Web Spam Challenge results, calculated by the competition organizers. Our preliminary results are summarized in Table 4. These results are reported with and without the various enhancements so the impact of these enhancements can be evaluated. In all cases the seven basic engineered features from Table 3 are included. The results in Table 4 show that when neither semi-supervised learning nor feature-fusion is used, the results are not very good, and that both of these enhancements yield consistent improvements in all measures of classifier performance. The results also demonstrate that the classifier generated by ADTree yields the best overall performance, and hence we submit only the classifiers induced using this algorithm for the Web Spam Challenge.

Table 4. Summary of Preliminary Classifier Performance Results. The table shows the results without any enhancements, then using semi-supervised learning, then using semi-supervised learning and the feature-fusion method.

Classifier	Metric	Enhancements		
		Initial	Semi-supervised learning	Semi-supervised learning + fusion
ADTree	AUC	.753	.824	.931
	F-Measure	.291	.523	.716
	Precision	.553	.611	.797
	Recall	.370	.457	.649
SMO	AUC	.581	.613	.628
	F-Measure	.301	.373	.410
	Precision	.320	.368	.392
	Recall	.285	.247	.284
Bayes	AUC	.703	.762	.772
	F-Measure	.451	.495	.517
	Precision	.345	.368	.392
	Recall	.651	.758	.763

The results for the official Web Spam Challenge are provided in Table 5. These results are presented separately for the validation data and the test data even though the validation data was not used to train the classifier (the class labels for the test data were not provided to the competitors). Two sets of results are reported since the competition permitted two classifiers to be submitted. In our case the differences were minor in that a few features were ommitted for the second classifier.

Table 5. Summary of our Classifier Perrformance for the Official Competition. All results are based on the ADTree classifier, using both the semi-supervised and fusion enhancements.

Evaluated Data	AUC	Precision	Recall	Accuracy (thresh=0.5)	Accuracy (thresh=optimal)
Validation	.889	.766	.438	.839	.850
Validation	.884	.794	.338	.821	.838
Test	.864	.725	.406	.825	.842
Test	.854	.738	.318	.803	.832

The results in Table 5 show reasonable values for AUC, but show relatively low values for recall. In particular, the values for the validation data in Table 5 are worse than those for the same data in Table 4. Although some last minute changes that were made for the competition may have negatively impacted our results, the only change that we are aware of that should have had a negative impact is that for our preliminary results the validation data was inadvertently used to calculate the performance values for the feature fusion method—but *not* used to train the actual classifier. It is not clear to us that this should cause a substantial difference in overall performance.

6 Conclusion and Future Work

In this paper we describe a machine learning approach for identifying web spam. Our approach involves adding human-engineered features and then using semi-supervised learning to exploit the unlabeled examples that are provided as part of the Web Spam Challenge data. We also use our combinatorial feature-fusion method in order to reduce the number of TF-IDF content-based features and to construct new features that are combinations of these features. We feel that our final results are reasonable and, perhaps more significantly, we show that both our feature-fusion strategy and use of semi-supervised learning lead to dramatically improved classification performance.

We see many opportunities for future work. Many of these opportunities relate to our feature fusion method. First, we would like to employ a heuristic version of the feature fusion method so that we can handle more than 200 features and also generate more complex feature combinations. Also, the rank combination used in our feature fusion method assigns equal weight to each feature and we would like to learn the optimal weights for combining these ranks. Since we assign a rank based only on the positive or negative magnitude of the score value, our method will not handle the case well where the most predictive value occurs in the middle. We could address this by binning the numerical values and then ranking the bins based on their predictive value. Using a similar idea we could also use the feature fusion method to handle non-numerical values.

We would also like to further study the semi-supervised learning method that we employed. We would like to try to generate more sophisticated link-based features, determine how the number of iterations of semi-supervised learning impacts classifier performance, and determine if the values of the link-based features that we construct converge after several iterations of semi-supervised learning.

References

1. Ntoulas, A., Najork, M., Manasse, M., Fetterly, D. Detecting spam web pages through content analysis, Proceedings of the 15th International World Wide Web Conference (2006)
2. Becchetti, L., Castillo, C., Donato, D., Leonardi, S., Baeza-Yates, R. Link Based Characterization and Detection of Web Spam, Workshop on Adversarial Information Retrieval on the Web (2006)
3. Hsu, D.F., Chung, Y., Kristal, B. Combinatorial fusion analysis: methods and practices of combining multiple scoring systems. Advanced Data Mining Technologies in Bioinformatics. Hershey, PA: Idea Group Publishing; 32–62 (2006)
4. Tian, Y., Weiss, G., Hsu, D.F., Ma, Q. A Combinatorial Fusion Method for Feature Mining, Proceedings of KDD'07 Workshop on Mining Multiple Information Sources (2007)
5. Weiss, G. M. Mining with rarity: A unifying framework. SIGKDD Explorations, 6(1): 7-19.
6. Chapelle, O., Scholkopf, B., Zien, A. Semi-Supervised Learning. MIT Press, Cambridge, MA. (2006).
7. Freund, Y., Mason, L. The alternating decision tree learning algorithm. Proceedings of the Sixteenth International Conference on Machine Learning, 124-133 (1999).
8. Markov, Z., Russell, I. An introduction to the WEKA data mining system. In Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education. 367–368 (2006)