Multi-View Learning for Web Spam Detection

Ali Hadian* and Behrouz Minaei-Bidgoli
Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
Email: hadian@comp.iust.ac.ir, b_minaei@iust.ac.ir

***Abstract*—Spam pages are designed to maliciously appear among the top search results by excessive usage of popular terms. Therefore, spam pages should be removed using an effective and efficient spam detection system. Previous methods for web spam classification used several features from various information sources (page contents, web graph, access logs, etc.) to detect web spam. In this paper, we follow page-level classification approach to build fast and scalable spam filters. We show that each web page can be classified with satisfactory accuracy using only its own HTML content. In order to design a multi-view classification system, we used state-of-the-art spam classification methods with distinct feature sets (views) as the base classifiers. Then, a fusion model is learned to combine the output of the base classifiers and make final prediction. Results on our Persian web spam dataset show that multi-view learning significantly improves the classification performance, namely AUC by 22%, while providing linear speedup for parallel execution.**

***Index Terms*—Web Spam, Content Spam, Machine Learning, Multi-View Learning**

I. Introduction

Web search engines are the most important tool for finding required information from the web. Processing the excessive amounts of data, even the text contents of web pages in the simplest case, is a very challenging task. Search engines should not only do this process in a few milliseconds for each query, but also have to provide high-quality and relevant results in the best order. As a part of this task, malicious and spam pages should be removed from the results. Web spam can be defined as "an unjustifiably favorable relevance or importance score for some web page, considering the pages' true value" [1]. More specifically, spam pages are defined as the pages that boost the ranking algorithm, and thus appear among the top results of the target queries. They usually deceive ranking algorithms by *keyword stuffing* of popular query terms [2].

Studies on recent web corpora have shown that spam pages, if not nullified effectively, are likely to become the majority of search results. Specifically, For queries that are frequently searched, almost all of the results are likely to be spam [3]. The ideal solution is to use ranking algorithms that are robust to spam. Unfortunately, spam-aware ranking methods are not strong enough to resist against spammers [4]. Therefore, spam pages should be removed before the ranking using complementary spam detection methods, i.e. machine learning methods. This will prevent spam pages from competing with other pages in the ranking phase.

Designing a spam detection system is a continuing process. As the spam filters evolve, new spamming techniques will be invented and new efforts will be required to detect them [4]. Web spam detection methods are still unable to reach satisfactory performance for search engines. Besides, most of the previous works propose features that are computationally expensive, such as temporal or host-level features [5]. Moreover, host-level features requires the search engine to dedicate considerable amount of resources for storing and retrieving host features.

In this work, we implemented a page-level spam detection system. We show that feature sets extracted from web pages, without using host information or any external source, can still reach satisfactory performance. The main advantage of this approach is that each page can be processed independently, which yields a linearly scalable system. In spite of the poor classification results for previous page-level methods, we show that a two stage classification with multi-view learning strategy can significantly improve the performance. Multi-view learning is a classifier fusion method in which the results of the predictive functions learned from different views on the same instances are combined to make final prediction. Three feature sets from the state-of-the art are selected as the base classification models, each one representing a different view for learning the target concept. Then, a fusion model is trained for aggregating the predictions of the base classifiers.

The remainder of the paper is structured as follows. Section 2 describes the previous work on web spam detection. In Section 3 we describe the construction of our real-world dataset. The proposed spam detection framework, including the base and meta classifiers and their results, is described in section 4. Finally, section 5 concludes our work and presents future directions.

II. Related Works

Web spam pages are created to deceive ranking algorithms of a search engine. Even though search engines use complex ranking algorithms and several information sources, ranking of web pages is based a simple rule: A web page is possibly relevant to a query if 1)its contents are similar to the query and 2)the importance of web page in the web, according to web-graph or visit rate, is relatively high. Based on this rule, spam pages are boosted by two general techniques:

doi:10.4304/jetwi.5.4.395-400

1) *Content spamming* by boosting content-based ranking algorithms, e.g. deliberate usage of keywords that are frequently searched. Using several popular keywords in the contents of the HTML pages will increase the chance of becoming relevant to more user queries. Moreover, because content-based ranking algorithms pay considerable attention to frequency of query terms in the result pages, a spam site owner may repeat popular keywords in the pages of his site. In this case, the spam pages are more likely to be relevant with popular queries, and subsequently, gain higher ranking in the query results [1, 6]

2) *Link spamming* through creating large group of pages that link to target (boosted) pages. For example, one can create a large number of automatically created web pages, all of which are linking to a target page. This will induce the link-based ranking algorithms, e.g. PageRank, to consider a high importance for the target page(s) [1].

It would be ideal if the ranking algorithm is robust against spamming. Otherwise, the ranking system should be supported by a spam filter. Link-based methods, i.e. algorithms that process the web-graph, are rich in both spam-aware ranking and spam filtering. Considerable effort been directed towards robust and spam-resilient link-based ranking, such as TrustRank [7], Anti-TrustRank [8], DiffusionRank [9], and many more [10-12]. Most of the these methods are based on trust propagation in graph, which is implemented through iteration over the web-graph. However, trust propagation algorithms are offline, and cannot rapidly investigate newly crawled web pages.

On the other hand, despite few recent efforts[13], content-based ranking methods are still defenceless against content spam [4]. Therefore, spam pages should be detected by standalone spam classification methods. Popular approaches for content-based spam detection are very similar to document classification techniques. Content-spam detection is considered as a hard classification task, because the spammers try to deceive the spam detector by manipulating the features of spam pages to make them similar to normal web pages. Nevertheless, a content-based filter is more straightforward to implement, and can be used in any search engine architecture.

From another point of view, spam detection methods can be classified to host-level and page-level methods, according to their granularity level. During the last decade, various method have been proposed for detecting web spam pages. However, research have been almost biased to host-level approach, mainly because the only publicly available spam datasets, namely *UK2006, Castillo2006* and its newer version *UK-2007* are collected with host labels. Erdelyi et al. suggest that content-based features can be very efficient for spam classification and created a simple classifier fusion system to classify host-level instances on UK2007. [14]. Similar work was done by Cormack et al. on *ClueWeb* dataset [3] with overlapping n-gram features. However, none of the above have investigated several features sets for ensembling.

III. Dataset

A. Choice of Granularity

Comparing to page-based spam classification, designing host-based classifiers is not straightforward, and has higher computational cost and more complexity for feature extraction and system integration. Aggregating page features into host features requires batch processing, which limits its scalability [15]. Moreover, it is difficult to measure the effectiveness of a host-level filter in a search engine, because some hosts, are more likely to appear in the results list, and some other hosts are less retrieved, or their contents are rarely searched by users at all.

In contrast, page-level spam filtering is simpler and more flexible. Page-level filters can be used either in crawl-time, just after fetching each page, without the need to wait for other pages from the same host, or at index-generation time. The labelling process is much easier, because the experts only need to judge a simple page rather than surfing the whole host for finding spam signatures [16]. Few research have been done on page-level filters [3, 16], and each work have followed different approaches for collecting data, labelling instances, learning method, and evaluation function.

B. Dataset Description

Web spam detection is usually considered as a two-class classification task. To create a dataset for this task, a number of instances should be assessed by users. To our knowledge, all of the works following the page-level spam classification used datasets from commercial search engines [16, 17]. The only exception is the ClueWeb09 spam scores [3] for which the original set of training labels was not published. In this work, we adopted a collection of 50 million web pages from commercial Persian search engines.

The set of labelled instances can be used as the dataset for learning and verifying the spam detection model. In order to estimate the performance of the spam detector, the dataset should be ideally an i.i.d sample of the pages retrieved by the system. The sampling strategy is quite simple: If a web page have been more often showed in the results of the queries, we would more likely consider this page in the dataset, because correctly classifying this page is more important for the system. A number of queries should be selected according to their probability of being searched. Therefore, we collected a list of frequently searched queries. This log consists of a batch of query terms searched by the users. By aggregating repeated queries, the log can be represented as the list of unique queries and the frequency of each query, which means how many times a query is searched. Let $Q = q_1,...,q_n$ be the set of unique queries, and f_i the frequency of q_i. Assuming enough size for the query log, the estimated probability of each query is

$$\hat{p}_i = \frac{f_i}{\sum_{q_i \in Q} q_i}. \quad (1)$$

After sampling a number of queries, N pages from the top results of each query are labeled by the evaluators (N=10). A group of human adjudicators evaluated the pages, and the majority vote is used as the final label. The dataset contains 5512 unique labelled instances.

Note that we could use the instance selection based on each single web page, and select each web page according to its probability of being observed among the search results of all queries. However, this will significantly degrade the classification performance, because it is likely that we have a few number of pages for each query. For instance, if we select only a single page from the results of the query "Nicole Kidman" being spam, a context-sensitive classifier will probably learn the rule that "Any page containing the terms Nicole and Kidman is spam". This will yield to biased classification models with poor generalization. Hence we have selected the queries based on their probabilities, and then evaluated all top N results of each query.

In order to quantify the agreement between raters, we used kappa statistic:

$$\kappa = \frac{P - P_e}{1 - P_e}. \quad (2)$$

where P is the agreement probability and P_e is the probability of chance agreement. Cohen's coefficient is a statistical measure for analyzing inter-rater reliability. The key advantage of kappa statistic over simple "percent agreement" is that kappa takes chance agreements into account. In our dataset, we observed $\kappa = 0.57$, which is similar to the *UK-2006* dataset ($\kappa = 0.56$) and much better than the dataset described in [18] ($\kappa = 0.45$)

IV. Spam Detection System

The system is composed of two major components. At first, the pages are classified by several content-based spam detection algorithms. Results of this classifiers will be processed as a second classifier, which makes final prediction according to the results of the base classifiers. Figure 1 shows the overall structure of our spam detection system.

A. Base Classifiers

Three distinct feature sets were used as the base classifiers. These feature sets have been have shown relatively better performance among others, as well as being fast and effective.

1) High Level Features

The classical method for spam filtering is to extract specific features from HTML structure of web pages, initially proposed by Ntoulas et al. [19]. Some of these features have shown to be very effective, namely the number of words in a page, the compression ratio of the page, average word length, title size, fraction of visible text, PageRank, and number of outlinks [19, 20]. As suggested by [14], Computationally expensive features (e.g. most of link-based features) are skipped for the sake of efficiency. Totally 30 features were selected for classification.

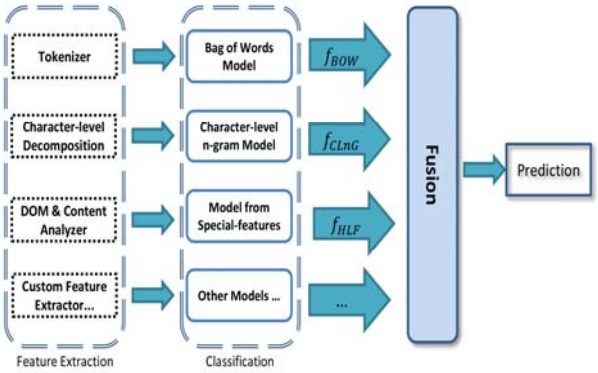

Figure 1: Web spam detection framework

Among several classification algorithms, decision tree, random forest, and KNN are reported to have the best discrimination [14, 19, 20]. Because we prefer classifiers with generative outputs, we selected KNN with K=7, using euclidean distance and inverse distance weighting. In order to accelerate the search for neighbor finding, we used KD-Tree space-partitioning data structure [21].

2) Bag-of-words

The bag-of-words model represents the document as a set of unordered terms. Each page p_i is represented as a term-frequency vector in the vector space model:

$$p_i = (TF(\omega_1), TF(\omega_2), ..., TF(\omega_N)). \quad (3)$$

where $TF_i(\omega_j)$ is the term frequency of the term ω_j in page p_i, and N is the total number of distinct terms in the corpus. To reduce the dimensionality, terms with very low frequencies were pruned, and 39,488 features remained. In the text-classification literature, three classification algorithms are mostly used for this high-dimensional learning task, namely naive-bayes, logistic regression, and linear support vector machines. While naive-bayes and (online) logistic regression are much faster for training, they suffer from the curse of dimensionality when number of training instances is small. SVM, on the other hand, is more consistent with high-dimensional datasets. The discriminating function of SVM is a hyperplane in the feature space which separates positive and negative instances. The margin between hyperplane and classifying instances is maximized by the following criteria:

$$\begin{aligned} &\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i, \\ &s.t. y_i(w \cdot x_i - b) \geq 1 - \xi_i \qquad \forall i \end{aligned}. \quad (4)$$

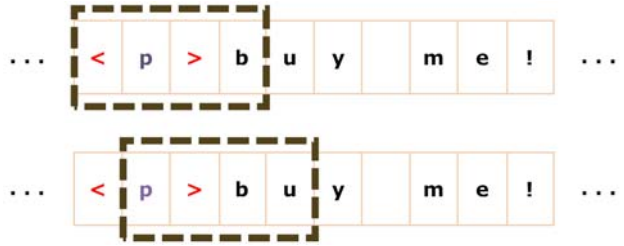

Figure 2: Character-level 4-gram extraction

where w denotes the normal vector of the discriminating hyperplane, b is a scalar vector, and C is a trade-off between accuracy and margin size. The discriminant function is $w \cdot x - b$ for an incoming feature vector x. We used popular SVM implementation from SVMLight[1] project, which is very fast for high-dimensional data [22].

3) Character-level 4-grams

In this model, any sequence of 4 characters represents a feature. It was primarily used for email spam detection task [23], and further proposed for classifying web spam [3, 17]. To extract the features, a 4-character width window starts from the beginning of the page, sliding one character per step. For instance, as visualized in figure 2, the extracted 4-grams for "< p> buy me!" are "< p> b", "p> bu", "> buy", "buy!", "uy! ", "y! m", and "! me". Then, each 4-character string is converted to its equivalent memory representation as a byte sequence, and the byte sequence is treated as a long number. This number can be used as the ID of the feature. However, because of the extremely high dimension of such feature set, features are reduced to a 10^6 dimensional space by simply dividing their ID by 10^6, and using the remainder as the ID in the new feature space [3]. Again, we used linear SVM for classification.

Table 1 shows the performance of the base classifier. To measure our performance, we performed five-fold cross validation for all of the experiments.

B. The Meta-classifier

In this step, outputs of the base classifiers are combined to make the prediction, such that

$$\hat{c} = f_{meta}(f_{HLF}(x), f_{BOW}(x), f_{CL4G}(x)). \quad (5)$$

If all of the three classifiers could generate probability or log-odds estimates, we would be able to use simple fusion operators, such as naive Bayes combination. However, because we used discriminative SVM classifiers for a couple of feature sets, we can not use probabilistic methods for combination. As shown in figure 3 , the prediction of f_{HLF} (KNN) is probabilistic estimate, while the two other classifier output discriminative values. This complex feature space requires a proper model for effective classification. In this case, we use outputs of the three classifiers as a feature set for training the meta-classifier.

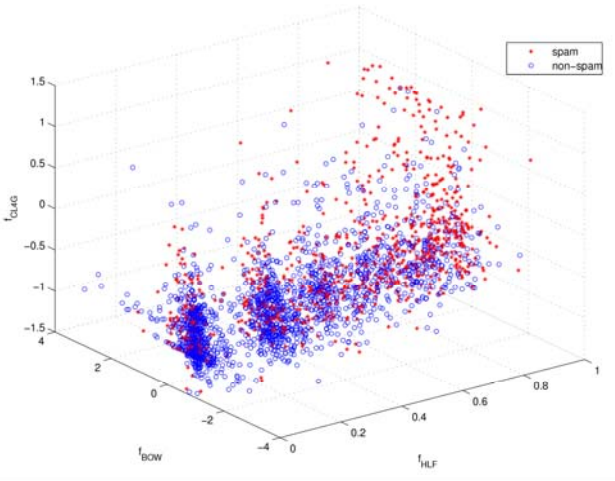

Figure 3: Feature space of meta-classifier

Several classification methods are tested for combining the classifier outputs. Again, we used 10-fold cross validation for validating the performance of meta-classifier. The results are given in Table 2. Except for recall, random forest classifier outperforms other methods. Comparing to the best results of base classifiers, the meta-classifier has increased the performance, both in terms of AUC and F-measure, by 22% and 15%, respectively. The result is not surprising, as it was previously shown that boosting-based classification methods, such as random forest, are robust and effective for adversarial environments [24].

The great advantage of multi-view learning is its capability to use the best possible classification algorithm independently for each of the discriminant feature sets. For instance, we benefited from SVM's ability to learn in high-dimensional feature sets, while using different models, i.e. KNN, for smaller-sized feature sets. However, if any of the base models is updated, the meta-model should be retrained.

C. Parallelization

Total classification time per page is 31 milliseconds in average. Despite the host-level classification models in

TABLE I.
PERFORMACE OF BASE CLASSIFIERS

Feature set	Classifier	Precision	Recall	F-measure	AUC
HTML features	KNN (N=7)	0.4944	**0.5636**	**0.5268**	**0.6789**
Bag-of-Words	SVM	0.4769	0.2097	0.2913	0.6467
Character-level n-grams	SVM	**0.6087**	0.1245	0.2068	0.6114

TABLE II.
PERFORMANCE OF CLASSIFICATION ALGORITHMS FOR ENSEMBLING

Classifier	Precision	Recall	F-measure	AUC
Random Forest	**0.892**	0.552	**0.682**	**0.901**
Decision Tree (C4.5)	0.71	0.484	0.575	0.793
KNN (Best K=5)	0.704	**0.652**	0.677	0.804
Weighted Naive Bayes	0.565	0.473	0.515	0.708
Weighted Logistic Regression	0.511	0.624	0.562	0.716
Bayes Net	0.523	0.612	0.564	0.737

which concurrent access to the web graph is a speedup bottleneck, the proposed method is linearly scalable. Multiple spam detection systems can be run in parallel, each for processing a batch of web pages, with no interrelation. In our industrial experiment, using three 12-core machines which classified the crawled pages in a pipeline, we reached our desired throughput, that is roughly a thousand pages per second.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an ensemble learning method for classifying spam web pages. Taking a page-level approach allows efficient and scalable classification of web pages, because the system only requires the page content to predict its spamness. Therefore, multiple instances of the spam detection module can be used in parallel without synchronization, which brings linear speedup. We used three state-of-the-art solutions for spam classification, and trained a meta-classifier to enhance their performance. Results show that this ensembling method successfully outperforms its base classifiers by a large margin. We are currently extending the algorithm with effective feature sets and more robust classification models, and preliminary results are encouraging.

ACKNOWLEDGMENT

The authors benefited from discussions with Azadeh Shakery, Ali Mohammad Zareh-Bidoki, Carlos Castillo, Gordon Cormack, Brian Davison, and Saeed Shahrivari. They also wish to thank Hadi Sharifi, Amin Nikookaran, and Ali Shirvani for their collaboration.

REFERENCES

[1] Z. Gyongyi and H. Garcia-Molina, "Web spam taxonomy," in *Proceedings of the first international workshop on adversarial information retrieval on the web*, 2005.

[2] C. Castillo and B. D. Davison, "Adversarial web search," *Information Retrieval*, vol. 4, no. 5, pp. 377–486, 2010.

[3] G. V. Cormack, M. D. Smucker, and C. L. A. Clarke, "Efficient and effective spam filtering and re-ranking for large web datasets," *Information Retrieval*, pp. 1–25, 2010.

[4] F. Raiber, "Adversarial content manipulation effects," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2012, p. 993.

[5] M. Erdélyi and A. A. Benczúr, "Temporal Analysis for Web Spam Detection: An Overview," in *Temporal Web Analytics Workshop TWAW 2011*, 2011, p. 17.

[6] Z. Gyongyi and H. Garcia-Molina, "Spam: It's not just for inboxes anymore," *IEEE Computer Magazine*, vol. 38, no. 10, pp. 28–34, 2005.

[7] Z. Gyongyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with trustrank," in *Proceedings of the 30th International Conference on Very Large Data Bases*, Morgan Kaufmann, 2004, pp. 576–587.

[8] V. Krishnan and R. Raj, "Web spam detection with Anti-Trust Rank," in *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*.pp. 37–40, 2006.

[9] H. Yang, I. King, and M. R. Lyu, "Diffusionrank: a possible penicillin for web spamming," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 431–438.

[10] R. Baeza-Yates, P. Boldi, and C. Castillo, "Generalizing pagerank: Damping functions for link-based ranking algorithms," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, pp. 308–315.

[11] L. Becchetti, C. Castillo, D. Donato, R. Baeza-Yates, and S. Leonardi, "Link analysis for web spam detection," *ACM Transactions on the Web (TWEB)*, vol. 2, no. 1, pp. 1–42, 2008.

[12] Y.-j. Chung, M. Toyoda, and M. Kitsuregawa, "Identifying spam link generators for monitoring emerging web spam," in *Proceedings of the 4th workshop on Information credibility*. ACM, 2010, pp. 51–58.

[13] M. Bendersky, W. B. Croft, and Y. Diao, "Quality-biased ranking of web documents," in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 95–104.

[14] M. Erdélyi, A. Garzó, and A. A. Benczúr, "Web spam classification: a few features worth more," in *Proceedings of the 2011 Joint WICOW/AIRWeb Workshop on Web Quality*, ACM, 2011, pp. 27–34.

[15] X. Geng, X. B. Jin, and D. Zhang, "Evaluating web content quality via multi-scale features," in *Proceedings of the ECML/PKDD 2010 discovery challenge*, 2010.

[16] K. M. Svore, Q. Wu, C. J. C. Burges, and A. Raman, "Improving web spam classification using rank-time features," in *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*. ACM, 2007, pp. 9–16.

[17] Y. Liu, F. Chen, W. Kong, H. Yu, M. Zhang, S. Ma, and L. Ru, "Identifying Web Spam with the Wisdom of the Crowds," *ACM Transactions on the Web (TWEB)*, vol. 6, no. 1, pp. 1–30, 2012.

[18] A. A. Benczúr, K. Csalogány, T. Sarlós, and M. Uher, "SpamRank: Fully Automatic Link Spam Detection Work in progress," *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[19] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, "Detecting spam web pages through content analysis," in *Proceedings of the 15th International Conference on the World Wide Web*. Edinburgh, Scotland: ACM, 2006, pp. 83–92.

[20] D. Fetterly, M. Manasse, and M. Najork, "Spam, damn spam, and statistics," in *Proceedings of the 7th International Workshop on the Web and Databases*, 2004.

[21] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977.

[22] T. Joachims, "Making large scale SVM learning practical," *Advances in Kernel Methods - Support Vector Learning*, 1999.

[23] G. V. Cormack, "University of waterloo participation in the TREC 2007 spam track," in *Sixteenth Text REtrieval Conference (TREC-2007)*, vol. 100, 2007.

[24] B. Biggio, G. Fumera, and F. Roli, "Multiple classifier systems for robust classifier design in adversarial environments," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 27–41, 2010.