Learning to Detect Web Spam by Genetic Programming

Xiaofei Niu[1,3], Jun Ma[1,*], Qiang He[1], Shuaiqiang Wang[2], and Dongmei Zhang[1,3]

[1] School of Computer Science and Technology, Shandong University, Jinan 250101, China
[2] Department of Computer Science, Texas State University, San Marcos, US
[3] School of Computer Science & Technology of Shandong Jianzhu University, Shandong 250101, China
xiaofein@sdjzu.edu.cn, majun@sdu.edu.cn

Abstract. Web spam techniques enable some web pages or sites to achieve undeserved relevance and importance. They can seriously deteriorate search engine ranking results. Combating web spam has become one of the top challenges for web search. This paper proposes to learn a discriminating function to detect web spam by genetic programming. The evolution computation uses multi-populations composed of some small-scale individuals and combines the selected best individuals in every population to gain a possible best discriminating function. The experiments on WEBSPAM-UK2006 show that the approach can improve spam classification recall performance by 26%, F-measure performance by 11%, and accuracy performance by 4% compared with SVM.

Keywords: Web Spam; Information Retrieval; Genetic Programming; Machine Learning.

1 Introduction

With the explosive growth of information on the web, search engine has become an important tool to help people find their desired information in daily lives. The page ranking is highly important in the search engine design. So some techniques are employed to enable some web pages or sites to achieve undeserved relevance and importance. All the deceptive actions that try to increase the ranking of a page illogically are generally referred to as Web spam [1]. People who make spam are called spammers. A spam page is a page that is either made by spammer or receives a substantial amount score for its ranking from other spam pages. Web spam seriously deteriorates search engine ranking results. Detecting web spam is considered as one of the top challenges in the research of web search engines.

Web spam can be broadly classified into term (content) spam and link spam [2]. Term spam refers to deliberate changes in the content of special HTML text fields in the pages in order to make spam pages relevant to some queries. Link spam refers to unfairly gaining a high ranking on search engines for a web page by means of trickily manipulating the link graph to confuse the hyper-link structure analysis algorithms. Previous work on web spam identification mostly focused on these two categories.

[*] Corresponding author.

In the previous research, Ntoulas et al. [3] proposed to detect spam pages by building up a classification model which combines multiple heuristics based on page content analysis. Gyongyi et al. [4] proposed the TrustRank algorithm to separate normal pages from spam. Their work was followed by much effort in spam page link analysis such as Anti-Trust Rank [5] and Truncated PageRank [1]. C. Castillo et al. [6], which is the first paper that integrates link and content attributes to build a system to detect Web spam, extracted transformed link-based features with PageRank, TrustRank , and Truncated PageRank etc. G. G. Geng et al.[7] proposed a predicted spamicity-based ensemble under-sampling strategy for spamdexing detection. Within this strategy, many existing learning algorithms, such as C4.5, bagging and adaboost, can be applied; distinguishing information involved in the massive reputable websites were fully explored and solved the class-imbalance problem well. Yiqun Liu et al. [8] proposed three user behavior features to separate web spam from ordinary ones. Na Dai et al. [9] used content features from historical versions of web pages to improve spam classification.

However, once a type of spam is detected and banned, usually new Web spam techniques will be created instantly. Therefore to study how to detect Web spam automatically based on machine learning is very meaningful. In this paper we discuss how to detect web spam by Genetic Programming (GP) [10]. Since GP has been used in binary classification problem [11], it inspires us to use GP to detect Web spam because detecting Web spam is a special binary classification, where Web pages are labeled spam or normal.

We define an individual as a discriminating function for detecting Web spam. The individuals are evolved based on the feature values of training set of the Web pages. Further the individuals are combined based on GP to generate an optimized discriminant for Web spam detection. We study the key techniques in using GP to detect Web Spam, which include the representation of individual, e.g. the architecture and the GP operation in an individual, the features of Web pages that can be used in detecting Web spam and using multi-populations and combination to generate the discriminating function, where we study the effect of the depth of the binary trees representing the individual in the GP evolution process and the efficiency of the combination. We carried out the experiments on WEBSPAM-UK2006 to evaluate the validity of the approach. The experimental results show that the new method can improve spam classification recall performance by 26%, F-measure performance by 11%, and accuracy performance by 4% compared with SVM.

2 GP Adapted to Web Spam Detection

2.1 Individual Representation and Population

Usually the tree-based structure is used to represent genetic programs [10]. In this paper we let an individual be represented as a binary tree. We use two kinds of terminals: feature terminals and numeric/constant terminals. Feature terminals are the transformed link-based features of a web spam, such as log of in-degree, log of out-degree/pagerank and so on. Constants are 11 predefined floating numbers which are 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0. The internal nodes denote the simple arithmetic operations, which include $\{+,-,\times,\div\}$. The $+$, $-$, and $\times$ operators have their usual meanings, while ÷ represents ‘‘protected’’ division, which is the usual division operator except that a divide by zero gives a result of zero. Although we can use more operators, e.g. *log, sin, cos* etc al, it is shown that the classification accuracy of using only simple arithmetic operators is sufficient to achieve high accuracy and using simple operators is capable of reducing computation cost [12]. In fact we did the experiment to verify the assumption in our experiment, where results show that the linear function is better than non-linear function for web spam dataset. Therefore we only use the four arithmetic operators.

Unlike decision trees that are used in data classification, e.g., ID3, C4.5 [3], GP algorithms do not directly construct a solution to a problem, but rather search for a solution in a space of possible solutions. That means that some of the individuals in a population may not be able to detect spam pages accurately, however it is well known since GP can search the space to find an optimal solution in its evolution process, the final output will be a discriminative function based on the features of example data with high probability.

The output of an individual in the standard tree-based GP system is a floating point number. In our approach for web spam detection, a population *P* is a set of individuals and denoted as $p = \{I_1, I_2, \hbar \ , I_{|p|}\}$, and for a given instance *x* and an individual *I*, we consider that *I* recognizes *x* as spam if $I(x) \geq 0$, otherwise normal.

2.2 The Features of Web Pages Considered in Web Spam Detection

The features used in our experiments are the 138 transformed link-based features [7], which are the simple combination or logarithm operation of the link-based features. The 138 features can be divided into five categories: *Degree-related features*, *PageRank-related features, TrustRank-related features, Truncated PageRank-related features* and *Supporter-related features.*

2.3 Mutation Rate and Crossover Rate

The genetic operators used in our experiments are reproduction, crossover, and mutation. They are performed according to predefined rates R_r, R_c and R_m. The reproduction operator keeps several selected individuals alive to the next generation. It mimics the natural principle of survival of the fittest. The crossover operator consists in taking two individuals from *p*, selecting a subtree from each individual randomly and exchanging the two selected subtrees to generate two new individuals. The mutation operator was implemented in such a way that a randomly selected subtree is replaced by a new tree also created randomly.

Because the mutation operator can generate individuals with new structures that have never occurred on existing individuals, it is mainly used to escape local optimum. If R_m is too high, the population tends to generate diverse individuals instead of discovering solutions from present individuals. If R_m is too low, individuals may not have sufficient opportunity to mutate and the diversity is limited. In our experiments, (R_m,R_c) is initialized to (0.2,0.8). In the *g*th generation, if the ratio of the fitness value of the best individual f_{max} to the average fitness value of all individuals $f_{average}$ is less than 2, R_m and R_c are tuned by formula 1, otherwise, R_m and R_c are not changed[13].

$$(R_m, R_c) = \begin{cases} (\frac{\alpha}{R_c+\alpha}, 1-\frac{\alpha}{R_c+\alpha}) & \frac{f_{\max}}{f_{average}} < 2 \\ (R_m, R_c) & \text{otherwise} \end{cases} \qquad (1)$$

where $\alpha = R_m \times (\frac{R_c}{R_m})^{g/G}$ and G is the maximum generation.

3 Multi-population GP and Combination

Traditionally, GP works with a single population. Recently it is shown that Multi-population GP (MGP) is better than single GP in terms of finding the optimal solution [14]. Therefore we adopt MGP in finding the optimal discriminating function for detecting spam pages. It is pointed out that it is difficult to know the proper length of an individual before finding an optimal solution [13]. Different from the normal definition of the depth of trees, the length of an individual is defined as the maximum number of available nodes of the individual. It is because the definition is more suitable according to the heuristic or empirical assumptions based on previous research. Since a long function can be viewed as a composition of a number of small GP solutions, we generate an optimal solution by the combination of sub-solutions, i.e. we make use of multi-populations composed of some small-scale individuals and combine the selected best individual in every population to gain better result.

Table 1 shows an example of the combination where F stands for features and H stands for Hosts. At first, we employ GP with two populations in which the depth of the individual is 3 and get two best discriminative functions F_1-F_4 (The first best)and F_2-(F_3+F_5)(The second Best). From Table 1, the accuracy of the first best function is 60% and the accuracy of the second best function is 80%, but the accuracy of the combination of the two best discriminating functions (F_1-F_4)+ (F_2-(F_3+F_5)) is 100%. As a result, it is likely that the combination might prove effective.

Table 1. An example of Combination

T	F1	F2	F3	F4	F5	class	First Best	Second Best	Combination
H1	0.8	0.67	0.1	0.1	0.2	Spam	0.7	0.37	1.07
H2	0.1	0.5	0.8	0.1	0.5	Normal	0	-0.8	-0.8
H3	0.6	1.14	0.7	0.9	0.3	Normal	-0.3	0.14	-0.16
H4	0.9	1	0.1	0.1	0.2	Spam	0.8	0.7	1.5
H5	1.4	0.86	0.6	0.8	0.9	Normal	0.6	-0.64	-0.04

4 The Fitness Function

The fitness function F(I_j) of an individual I_j is denoted by *Accu* (I_j, *T*).

$$Accu(I_j, T) = \frac{\text{the number of examples of } T \text{ that are correctly classified by } I_j}{|T|} \qquad (2)$$

where T is a data set. The best individual produced by a population P is denoted as *BI*.

5 The Evolution Computation Algorithm

The evolution algorithm is a revised basic GP with multi-populations. The Vector Space Model is used to represent every page. We let T denote the training data set, N_p be the total number of populations; the number of individuals and the maximum generation of every population are stored in array $N[N_p]$ and $G[N_p]$ respectively; K is the maximum depth of the binary trees which represent the individuals; C is the times of the combination for the found possible best individuals; N_{pc}, $Nc[N_{pc}]$, $Gc[N_{pc}]$ are the corresponding parameters of the cth combination ($1 \le c \le C$). Then the process of GP evolution can be described as follows.

```
Algorithm 1 (The evolution for finding the possible discriminating function)
Input: $T$, $C$, $N_p$, $K$, $N[N_p]$, $G[N_p]$
       $N_{pc}$, $Nc[N_{pc}]$, $Gc[N_{pc}]$ of the $c$th combination($1 \le c \le C$)
Output: The best individual $BI$ as the discriminating function.
  (1) $i=0$.
  (2) Initialize population $P_i$ with randomly generated individuals according to
      $N[i]$ and $K$; let $g=0$, $R_m=0.2$ , $R_c=0.8$; generate an empty population $P'$ and
      an empty set $S$.
  (3) Calculate the fitness value $F(I_t)$ of every individual $I_t \in P_i$ with $T$.
  (4) Perform genetic operators:
      (4.1) Perform reproduction operator: Compare the fitness values of the indi-
            viduals in $P_i$ and put two best individuals into $P'$.
      (4.2) Compute the average fitness value of all individuals in $P_i$ and find
            $\max\{F(I_b) | I_b \in P_i\}$, then tune $R_m$ and $R_c$ according to formula 1.
      (4.3) Select the crossover or mutation operator according to $R_m$ and $R_c$.
      (4.4) If $|P'|=N[i]-1$ or the mutation is selected, then perform mutation op-
            erator on a randomly selected individual, calculate the fitness value
            for the mutant and compare its fitness value with that of the selected
            individual, and put the better individual into $P'$.
      (4.5) If the crossover is selected and $|P'| < N[i]-1$, then perform crossover
            operator on selected two individuals and evaluate the fitness values of the
            two newly generated individuals. Compare their fitness values with those
            of the two selected individuals and put the best two individuals to $P'$.
      (4.6) Continue to perform step (4.3)-(4.5) if $|P'| < N[i]$.
  (5) Store the best individual of this generation $BI_g$. Let $P_i = P'$, $P'=\Phi$, $g=g+1$.
  (6) If $g<G[i]$, then repeat (3)-(5).
  (7) Output $BI_j (0 \le j < G[i])$ to be the result of the population $P_i$ if $F(BI_j)$ is equal to
      $\max\{F(BI_j) | 0 \le j < G[i]\}$. Let $S = S \cup BI_j$ , $i=i+1$.
  (8) if $i < N_p$, then repeat (2)-(7).
  (9) Combine the best individual in every population as following if $C>0$
      (9.1) Let $c=1$.
      (9.2) With individuals in $S$ as new features, compute a new instance $e_m'$ for
            every instance $e_m$ in $T$; $T'=\{ e_m' | 1 \le m \le |T| \}$
      (9.3) Let $T = T'$, $N_p=N_{pc}$, $N[N_p]=Nc[N_{pc}]$, $G[N_p]=Gc[N_{pc}]$, then repeat (1)-(8).
      (9.4) Let $c=c+1$
      (9.5) If $c \le C$, repeat (9.2)-(9.4).
```

Algorithm 1 is an iterative process. Starting with a labeled training data set *T*, the algorithm generates N_p populations. All initial individuals in the *i*th population are randomly generated by the ramped half-and-half method [11].

After all individuals in the *i*th population are evaluated by the fitness function, the algorithm evolves the *i*th population by applying genetic operators in order to generate the best individual of the *i*th population. As a result, there are N_p individuals. Then we can combine N_p individuals using GP to gain better discriminating functions. In the *c*th combination process, the first step is to figure out a new instance for each instance in *T* with the above N_p individuals as new features, thus all the new instances can make up of a new data set T'; the next step is to repeat step (1)-(8) according to T' and the corresponding parameters N_{pc}, $Nc[N_{pc}]$, $Gc[N_{pc}]$. We can continue to perform the next combination on the basis of the above combined result. Please note that N_p should be assigned to 1 in the last combination because we need a possible best discriminating function finally. When the single population GP is executed, *C* and N_{pc} should be assigned to 0 and N_p should be assigned to 1.

6 Experiments

6.1 Data Set

In our experiments, we use the publicly available WEBSPAM-UK2006 dataset [15]. It is based on a set of pages obtained from a crawler of the .uk domain. The set includes 77.9 million pages, corresponding to 11402 hosts, among which over 8000 hosts have been labeled as “spam”, “non-spam(normal)” or “borderline”. We use the labeled hosts in webspam-uk2006-set1-labels –DomainOrTwoHumans.txt and webspam-uk2006-set2-labels-DomainOrTwoHumans.txt, which can be downloaded from http://barcelona.research.yahoo.net/webspam/datasets/uk2006/, where 4948 hosts are marked normal and 674 hosts are marked spam. We select 470 spam hosts and 500 normal hosts as train set, 204 spam hosts and 300 normal hosts as test set. Each host is presented as a 139-dimensional vector including 138 features and an associated class label.

6.2 Evaluations

The evaluations used in our experiments are precision, recall, F-measure and accuracy. For the binary classification, all instances are divided into positive instances if they belong to the target class and negative instances if they do not. Our target class is web spam. The confusion matrix is as follows:

Table 2. The confusion matrix

	actual positive	actual negative
predicted positive	TP	FP
predicted negative	FN	TN

where TP represents the number of positive examples that are correctly classified as positive, FP represents the number of negative examples that are falsely classified as positive, FN represents the number of positive examples that are falsely classified as negative, TN represents the number of negative examples that are correctly classified as negative. The evaluations are defined as following:

Precision: the percentage of truly positive examples in those classified as positive by the classifier:

$$\mathrm{Pr}\,ecision = \frac{TP}{TP + FP} \tag{3}$$

Recall: the percentage of correctly classified positive examples out of all positive examples:

$$\mathrm{Re}\,call = \frac{TP}{TP + FN} \tag{4}$$

F-measure: a balance between Precision and Recall:

$$F\_measure = \frac{2 \times \mathrm{Pr}\,ecision \times \mathrm{Re}\,call}{\mathrm{Pr}\,ecision + \mathrm{Re}\,call} \tag{5}$$

Accuracy: the percentage of correctly classified examples out of all examples:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{6}$$

6.3 Experimental Results

Because of the randomicity of GP process, we perform every experiment 10 times and get the average value of every measure to compare. The experiments shown in Table 3-4 make use of 3 populations at first. They have 10, 20 and 30 individuals respectively. Their maximum generations are 50, 30 and 20 respectively. Then combinations are employed. The first combination also uses 3 populations. They have 20, 30 and 40 individuals respectively. Their maximum generations are 100, 50 and 50 respectively. The last combination uses 1 population. It has 50 individuals and its maximum generation is 50.

Table 3. Comparison between the linear function and the non-linear function

	precision_Avg	recall_Avg	F_measure_Avg	accuracy_Avg
Linear function	0.77828174	0.836764706	0.805602567	0.836706349
Non-Linear function	0.630408109	0.759313725	0.685843185	0.718253968

Table 3 shows the comparison between the linear function and the non-linear function in condition that the depth of the binary tree is eight. As we can see in Table 3, the linear function is better than the non-linear function.

Table 4. Comparison among the depth of the tree that ranges between 1 and 9

depth	precision_Avg	recall_Avg	F_measure_Avg	accuracy_Avg
1	0.505585508	0.90294118	0.646087527	0.597222222
2	0.655315651	0.76372549	0.700587373	0.733928571
3	0.716419012	0.82303922	0.764246563	0.794642857
4	0.795592113	0.81911765	0.80486974	0.839880952
5	0.712093992	0.825	0.763331324	0.793452381
6	0.739557878	0.85833333	0.794075354	0.819642857
7	0.761283597	0.84705882	0.800857247	0.829365079
8	0.77828174	0.83676471	0.805602567	0.836706349
9	0.755325083	0.82794118	0.788169917	0.819444444

In order to present the effect of the depth of the binary tree, we perform experiments with the depth of the tree that ranges between 1 and 9. Table 4 shows the results. From Table 4, when the depth of the tree is 4, the average precision and the average accuracy are the best. When the depth of the tree is 8, the average F_measure is the best, the average accuracy is the second. However from Fig. 1, the running time when the depth of the tree is 8 is six times longer than that of when the depth of the tree is 4. From Fig.1, the running time increases exponentially with the increasing of the depth of the binary tree. So the depth of the binary tree should not be too large when we use GP to learn a possible best discriminating function.

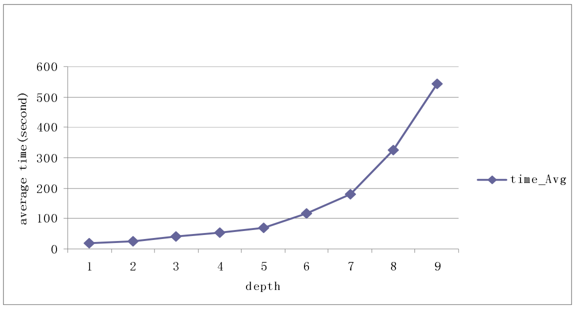

Fig. 1. Time comparison among the depth of the binary tree that ranges between 1 and 9

In order to show the effect of the combination, we perform experiments when the depth of the tree is 4. When the single population GP is carried out, both the total number of individuals and the maximum generation are 50. In the experiments of one combination, the parameters are same to those of the experiments shown in Table 3-4 except there is no the first combination. In the experiments of two combinations, all parameters are same to those of the experiments shown in Table 3-4. From Table 5, we can see that compared with the single population GP, the multi-population GP with two combinations can improve spam classification recall performance by 5.6%, F-measure performance by 2.25% and accuracy performance by 2.83%. The experimental results (the depth of the tree is 8) in Table 6 further confirm that the combination can improve the classification performance.

Table 5. Comparison between the single population and the combination(4)

	precision_Avg	recall_Avg	F_measure_Avg	accuracy_Avg	time
Single population	0.73915245	0.83480392	0.78235325	0.811507937	32.0746
One Combination	0.751195618	0.82892157	0.787326785	0.818650794	39.1965
Two Combinations	0.795592113	0.81911765	0.80486974	0.839880952	53.5418

Table 6. Comparison between the single population and the combination(8)

	precision_Avg	recall_Avg	F_measure_Avg	accuracy_Avg	time
Single population	0.763119123	0.82401961	0.791372801	0.823809524	163.0511
One Combination	0.77828174	0.83676471	0.805602567	0.836706349	324.5528

Table 7. Comparison between GP_4 and SVM

	precision_Avg	recall_Avg	F_measure_Avg	accuracy_Avg
2CGP_4	0.795592113	0.81911765	0.80486974	0.839880952
SVM	0.904761905	0.55882353	0.690909091	0.797619048

Table 7 shows the comparison between SVM and the multi-population GP with two combinations when the depth of the binary tree is 4(2CGP_4). From Table 7, we can find that 2CGP_4 improves spam classification recall performance by 26%, F-measure performance by 11% and accuracy performance by 4% compared with SVM. The precision of SVM is higher than GP_4, but the recall is too lower.

7 Conclusions and Future Work

In this paper, we propose the method using GP to learn a discriminating function to detect web spam. We study the representation of an individual, the features of Web pages that are used to detect Web spam, the effect of the depth of the binary trees in the GP evolution process and the efficiency of the combination. We performed the experiments on WEBSPAM-UK2006 collection. Experimental results show that the method can improve the web spam detection performance effectively compared with SVM. However the method does not detect content spam because only the link based features are used. We also do not consider the problem of class imbalance.

Future work involves using other proposed features in GP, e.g. the content features, finding better features by GP, considering the problem of class imbalance and proposing better classification method to detect web spam.

Acknowledgments

This work is supported by the Natural Science Foundation of China (60970047), the Natural Science Foundation of Shandong Province (Y2008G19) and the Key Science-Technology Project of Shandong Province (2007GG10001002, 2008GG10001026).

References

1. Becchetti, L., Castillo I, C., Donato I, D., Leonardi, S., Baeza-Yates, R.: Using Rank Propagation and Probabilistic Counting for Link Based Spam Detection. In: Nasraoui, O., Spiliopoulou, M., Srivastava, J., Mobasher, B., Masand, B. (eds.) WebKDD 2006. LNCS (LNAI), vol. 4811, pp. 127–146. Springer, Heidelberg (2007)
2. Gyongyi, Z., Garcia-Molina, H.: Web Spam Taxonomy. In: Proc. of First Workshop on Adversarial Information Retrieval on the Web (2005)
3. Ntoulas, A., Najork, M., Manasse, M., Fetterly, D.: Detecting spam web pages through content analysis. In: Proceedings of the 15th International Conference on World Wide Web, WWW 2006, Edinburgh, Scotland, May 23 - 26, pp. 83–92. ACM Press, New York (2006)
4. Gyöngyi, Z., Garcia-Molina, H., Pedersen, J.: Combating web spam with trustrank. In: Proceedings of the Thirtieth international Conference on Very Large Data Bases, vol. 30, pp. 576–587 (2004)
5. Krishnan, V., Raj, R.: Web Spam Detection with Anti-Trust-Rank. In: The 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb) (August 2006)
6. Castillo, C., Donato, D., Gionis, A., Murdock, V., Silvestri, F.: Know your Neighbors: Web Spam Detection using the Web Topology. Technologies Project (November 2006)
7. Geng, G.G., Wang, C.H., Li, Q.D., Xu, L., Jin, X.B.: Boosting the Performace of Web Spam Detection with Ensemble Under-Sampling Classification. In: Proc. of the 4th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007 (August 2007)
8. Liu, Y., Cen, R., Zhang, M., Ma, S., Ru, L.: Identifying Web Spam with User Behavior Analysis. In: AIRWeb 2008, Beijing, China, April 22 (2008)
9. Dai, N., Davison, B.D., Qi, X.: Looking into the Past to Better Classify Web Spam. In: AIRWeb '09, Madrid, Spain (April 21, 2009)
10. Koza, J.R.: Genetic Programming: on the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
11. Zhang, M., Wong, P.: Genetic programming for medical classification: a program simplification approach. Genet. Program. Evolvable Mach. 9, 229–255 (2008)
12. Kishore, J.K., Patnaik, L.M., Mani, V., Agrawal, V.K.: Application of genetic programming for multi-category pattern classification. IEEE Trans. Evol. Comput. 4(3), 242–258 (2000)
13. Lin, J.-Y., Ke, H.-R., Chien, B.-C., Yang, W.-P.: Designing a classifier by a layered multi-population genetic programming approach. Pattern Recognition 40, 2211–2225 (2007)
14. Fernández, F., Tomassini, M., Vanneschi, L.: An Empirical Study of Multi-population Genetic Programming. Genet. Programming Evolvable Mach. 4, 21–51 (2003)
15. Castillo, C., Donato, D., Becchetti, L., Boldi, P., Leonardi, S., Santini, M., Vigna, S.: A reference collection for web spam detection. ACM SIGIR Forum 40(2), 11–24 (2006)