Improving Link Spam Detection using Spamizer

Ela Kumar, Shruti Kohli

Abstract— **Spammers intend to increase the PageRank of certain spam pages by creating a large number of links pointing to them. We have designed and develop a system, Spamizer that detects spam hosts or pages on the Web. The UK Web Spam UK 2007 data set has been used for experimentation. It is a public web spam dataset annotated at the level of hosts, for all results reported here. System uses the key features of popular link based algorithms to detect spam in improved manner. It simultaneously exploits the structure of the web graph as well as page contents and features. The method is found to be efficient, scalable, and provides state-of-the-art accuracy on a standard Web spam benchmark.**

Index Terms— **Content Feature, Page Rank, link spam, Search Engine, Spam, Spam Detection.**

I. Introduction

Spamming is one of the most emerging challenges for search engines [1]. Numerous researchers have put-in their sweat to explore new vistas, as this is arena of continual improvement. Spamming not only hampers the quality of search results but also impacts the "user–search engine" and "search engine–web publisher" dependency. Users may get de-motivated on receiving spammed, low-quality results. Similarly, web publishers may lose faith in search engines and explore other options to attract customers. Keeping in mind the impact of web spam, lot of work is being done at various corners of the globe. Our goal has been to add value on this journey, a.k.a find unique and meaningful solution for various stakeholders. We are eyeing a solution to the problem of spamming on the World Wide Web, mainly caused by misuse of the *link structure of the net* and through unethical use of techniques like cloaking. Besides term-based relevance metrics, search engines also rely on link information to determine the importance of web pages. Since search engines now use the link structure of the web, link analysis algorithms were incorporated into search engines.

According to these algorithms, more links point to a web page, which makes the web page look more valuable from search engine parse. Taking advantage of these link-based ranking techniques like Page Rank [2] spammers started to construct spam farms. The plummeting cost of web publishing has rendered a boom in link spamming. The size of various spam farms has increased dramatically and many farms span even thousands of different domain names, rendering naive counter measures ineffective. Skilled spammers, whose activity remains largely undetected by search engines, often manage to obtain very high rankings for their spam pages.

The paper is organized as follows. In Section II we have outlined overview of link spam algorithms and discuss related work done in this arena. Section III talks, about design and development of SPAMIZER in detail. Moving down further, Section IV explains how the system can be implemented and used to detect link spam efficiently. Lastly Section V, where we conclude the paper and discuss possible directions for future work.

Manuscript received January 18, 2011; revised April 4, 2011.

Ela Kumar is with the Gautam Budha University, Gautam Budh Nagar, Uttar Pradesh (India) - 201308,India (corresponding author to provide phone: 0120-2344200; e-mail: ela_kumar@rediffmail.com).
Shruti Kohli is with Birla Institute of Technology, Mesra, Extension centre Noida,A-7,Sector 1,Noida, UP-201301, India. (e-mail: kohli.shruti@gmail.com).

II. Overview of Link Spam Algorithms

Earlier search engines used to provide results only on the basis of the relevance of the web page, and term "spamming" significantly reduced the quality of these search engines. With the invention of link-based search engines like Google, that combat well with spamming, better ranking for the results was being seen. Link-based algorithms like Page Rank and HITS proved to be good tools to calculate the importance of the web page [3]. Page Rank [2] uses incoming link information to assign global importance scores to all pages on the Web. It assumes the number of incoming links to a page is related to that page's popularity among average web users (people would point to the page that they found important). The intuition behind the algorithm is that a web page is important if several other important web pages point to it.

Link spamming techniques add numerous outgoing links to popular pages or they gather many incoming links to a single target page or group of pages. Link spamming mainly attacks link based ranking algorithms such as PageRank [16] and HITS [10] that consider a link to a page as an endorsement of that page. To increase these ranking scores, spammers often add outgoing links to popular sites and gather many links to their target sites in various ways, for example, by connecting their sites each other [8]. Influences of link spamming techniques on PageRank are examined in [4, 7]. Various techniques have been proposed for combating link spam. Some of them use machine learning for classifying spam and non-spam pages based on features related to link structure, such as in-degree, out-degree, and link based ranking scores of pages [10, 11, 12]. Another approach is to improve link-based ranking algorithms to be robust against link spamming. Various ranking algorithms are proposed and tested on real web data in [13, 17]. There have also been some link-based approaches [15, 17, 13] that extract link spam using some graph theoretical notions such as bi-connected components and commonality of neighbors. Still combating link spam with at most accuracy is future.

III. DESIGN AND DEVELOPMENT OF SPAMIZER

A. *Design*

A system has been developed to analyze and improve currently popular link-based algorithms available for detection for spam. The system has been coined as *'Spamizer'* as it detects and analyzes the intensity of spam in the given data. The goal is to merge the results obtained from the implementation of the link-based algorithms with the content analysis for the web hosts and to determine the accuracy of the current link spam algorithms in detecting spammed pages. *'Spamizer'* is based on following five link spam algorithms:

1. Link Spam Detection by Mass Estimation [2].
2. Web Spam Detection using Trust Rank [8].
3. Web Spam Detection using Anti-Trust Rank [5].
4. Propagating Trust and Distrust for detecting spam Web pages[6].
5. R-SpamRank: A Spam Detection Algorithm Based on Link Analysis [18].

Spamizer takes input from data set containing content-feature and host graph. Each of these algorithms has been implemented in C and their spamicity score is generated. The spamicity scores for each algorithm are combined to generate ***'Spamizer spamicity score'*** for the input web hosts. This score depicts the intensity of the spam determined in the experimented web pages. In the output, we assign labels to web hosts based on the calculated spamicity measure. Working of Spamizer has been depicted in figure 1 below.

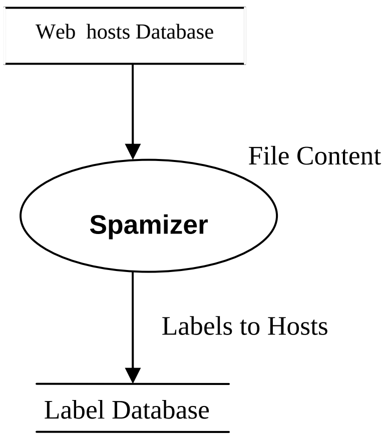


Fig 1. Working of Spamizer

B. *Architectural Design*

Spamizer includes three basic components:

a. Web Graph Generator: Generates web graph for all the five link based algorithms.
b. Spamicity score generator: Generates spamicity score for each five linked based algorithm
c. See 5 Classifier: Used to integrate the spamicity score for each link based algorithm with the content features to label webhosts.

Architectural design has been depicted in the figure 2.

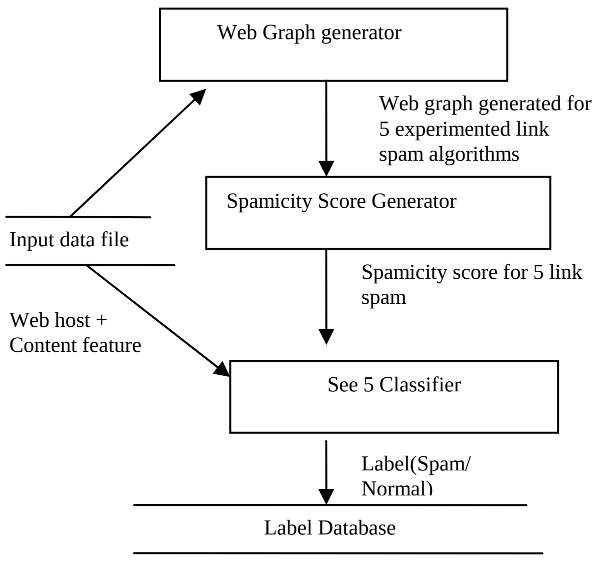

Web Graph generator
Web graph generated for 5 experimented link spam algorithms
Spamicity Score Generator
Input data file
Spamicity score for 5 link spam
Web host + Content feature
See 5 Classifier
Label(Spam/ Normal)
Label Database

Fig 2. Architectural Design of Spamizer

C. *Input Data Files*

The UK Web Spam UK 2007[9] data set has been used for experimentation. It is a public web spam dataset annotated at the level of hosts, for all results reported here. It contains files for content feature, hostnames, host graph and judge's files. This collection is the same used in the Web Spam Challenge Tracks I and II during 2007 [4], and represents a graph of 11,402 hosts in the .uk domain. Out of these, 7,473 were labeled. The hyperlink graph is represented as a list of 7,30,774 triples (nodei;nodej;#links) which specify the number of links from host i to host j. The data set includes information about each of the website like number of words in the website, amount of anchor text present in the website, etc. This is in CSV format (comma separated values) which is required format for the See5 classifier to classify the class of the website whether that is spam or normal. Input data set contains three files, namely:

a. Host Graph
b. Content Features
c. Judges Label

The Host Graph file is a file containing <**source, destination**> pairs. Each pair represents a hyperlink from source URL to destination URL. Here source and destination are numbers representing URLs. The mapping from URL to numbers is also kept in a separate file named as **Hostname file**. It is assumed that the value pairs read from the files are numbers such that the set containing the numbers from the source, and destination pairs are numbers ranging from 0 to n-1, where "n" is the total number of unique numbers (size of the set) [4]. File format is as follows:

```
Host_id -> host_id:no_of_links
Sample web hosts are:
2 -> 3794:2506 4704:3
3 ->4765:1
4 -> 24:1 52:1 530:6 4520:1 4765:5 5302:1 8303:1 9326:11
10037:1
5 -> 2079:1 4765:1
6 -> 362:1 2460:1 2794:1 2958:1 3805:14426 4300:1
4358:2 4520:4 4772:1 5948:1 6113:1 6351:2 6907:4 827
:1 8340:1 8487:1 8494:1 8500:1 8804:1 8825:1 10174:1
10397:1 11233:1
```

Content feature file consists of 96 content-based attributes of unique host-ids. The format of Content feature file and a sample record is:

```
hostid,host,HST_1,HST_2,HST_3,……HMG_48,AVG_49,
AVG_50,….,AVG_72,STD_73,STD_74,……. STD_96

0,xyz.com,521,2,4.10940,…..1.3953,0.0833,2.42220,…..0.2
2840,0.26103,…….0.56784
1,rty.co.uk,  786,8,9,678,…..45.677,9.5666,….89.7866
2, rty.co.uk
```

Here HST_1---HST_48, AVG_49---AVG_72, STD_73--STD_96 are the 96 content feature attributes of a host which have been represented by host_id and hostname.

Judges Label File contains label for a web host depicting whether it is spam or not spam. The format is URL, spam/non spam label. Sample records available in the file are:

```
007cleaningagent.co.uk normal
1-hydroponics.co.uk normal
102belfast.boys-brigade.org.uk normal
10enfield.boys-brigade.org.uk normal
15nr.co.uk normal
1portrush.boys-brigade.org.uk normal
```

D. *Working of Web Graph Generator*

Initially, content feature File and Host Graph File are taken as input files, and recorded the Common host ids, and their in-links, out-links information. This information in the file is represented in the form of multi-linked list data structure. The representation of multi-link list is in the form of row header and column header which is generated through the in-links, out-links information. Web graphs are generated for each of the above discussed link spam algorithms [2,8,5,6,18] and corresponding spamicity scores are also generated using the process depicted in figure 3.

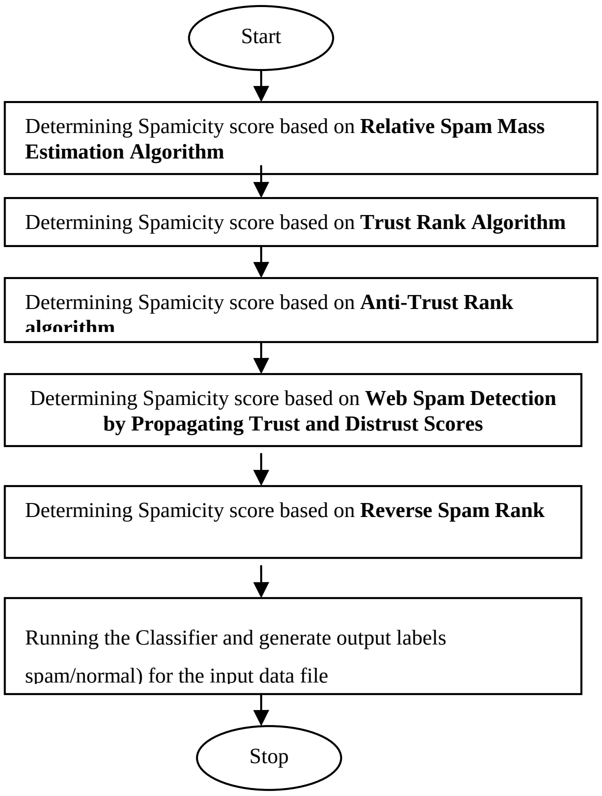

Start
Determining Spamicity score based on Relative Spam Mass Estimation Algorithm
Determining Spamicity score based on Trust Rank Algorithm
Determining Spamicity score based on Anti-Trust Rank algorithm
Determining Spamicity score based on Web Spam Detection by Propagating Trust and Distrust Scores
Determining Spamicity score based on Reverse Spam Rank
Running the Classifier and generate output labels spam/normal) for the input data file
Stop

Fig 3. Proposed methodology to detect link spam

E. *Functional Design*

Spamizer takes input from data set containing content feature and host graph. These algorithms, individually, will allot a spamicity score for all the hosts that are common to the host graph and content feature file present in the dataset. The result from the five link-based algorithms is appended with the content feature file to create two new files as Training and Testing Files, which will be fed into the See5 classifier[7] to generate a decision tree and thus allot a label (spam or non-spam) to the hosts in the testing file. The validation of results will be done according to the following criteria:

1. If a certain web page is good and Spamizer is detecting it as spam, this is called the case of false positive.

2. If a web page is spam and is being detected as normal, that is called as false negative.

So our purpose is to count the number of false positives and false negatives produced by our scheme. During the experimental analysis a tool was designed to provide labels (spam or normal) to the web hosts taken from the Dataset of the UK Web Spam 2007[9]. This tool will evaluate the total number of false positives and false negatives. This data will provide the detail of hosts which are actually spammed or not. The system takes data set as input which contains three files, namely host graph, judges label and content features. Final label is to be assigned to each of the sample hosts, and a number of false positives and false negatives are calculated in combined classifier. The output of the system is a file containing the labels for the entire web host in the data set.

IV. IMPLEMENTATION

A. *Assumption and Dependencies*

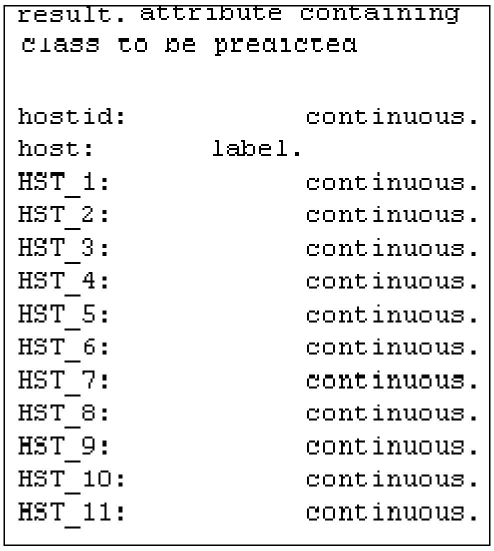
There are certain assumptions regarding the implementation of Spamizer and its use. Spamizer is based on the five link-based algorithms discussed in the previous section. Implementation has been done on Windows operating system. C programming language has been used as it provides convenience for reading from and writing to a text file which is used as database, thus assisting in performance optimization. The use of a Multi-Linked Structure has also been encouraged with the view to optimize the performance since it allows faster and efficient access of memory. See5 Classifier [7], a third-party tool, is being used for determining false positive and negatives. The use of this third-party software, See5 Classifier, requires us to strategize the interface of the system with the See5 Classifier. For this purpose our basic requirement is the conversion to .data file which is required as input to the classifier.

File Formats: The spamicity scores for the 5 link spam algorithms (discussed above) are appended in the host file and this file is used as case file in the See5 Classifier. Format of the file is:

hostid,host,HST_1,HST_2,HST_3,……HMG_48,AVG_49,AVG_50 ,….,AVG_72,STD_73,STD_74,……. STD_96,T_R

E.g:
0,xyz.com,521,2,4.10940,…..1.3953,0.0833,2.42220,…..0.22840,0.261 03,…….0.56784,0.8879

Fig 4 depicts the format of .name file used by see 5 classifier.



```
result. attribute containing
class to be predicted

hostid:              continuous.
host:          label.
HST_1:               continuous.
HST_2:               continuous.
HST_3:               continuous.
HST_4:               continuous.
HST_5:               continuous.
HST_6:               continuous.
HST_7:               continuous.
HST_8:               continuous.
HST_9:               continuous.
HST_10:              continuous.
HST_11:              continuous.
```

Fig 4. Sample .name file used by the See 5 classifier.

B. *Coding*

The code for the project has been developed in C language. Major files of the project are:

Definitions.h, Commonfunc.c , Rel_spam_mass.c , Trust_rank.c, Anti_trust_rank.c, Prop_trust_distrust.c, Rspam_rank.c

Major prototypes are:

void File_Reading(Hnode *sparse, Hnode *T_M, Hnode *U_TransOf_M, Hnode *Trans_T, Start_List *start, Start_List *temp, Label_Header *normal, Label_Header *spam)

This function is responsible to read each of the input file and to transfer the contents of each file in the FDB to the creation of webgraph sparse matrices.

void Create_T_Trans_T_U (Hnode *sparse, Hnode *T_M, Hnode U_trans_M, Hnode *Trans_T)

This function is responsible for creating different types of matrices from the existing webgraph sparse matrix.

Void Create_Label_Node(unsigned int host_id, char hostname[], Label_Header *spam, Label_Header *normal)

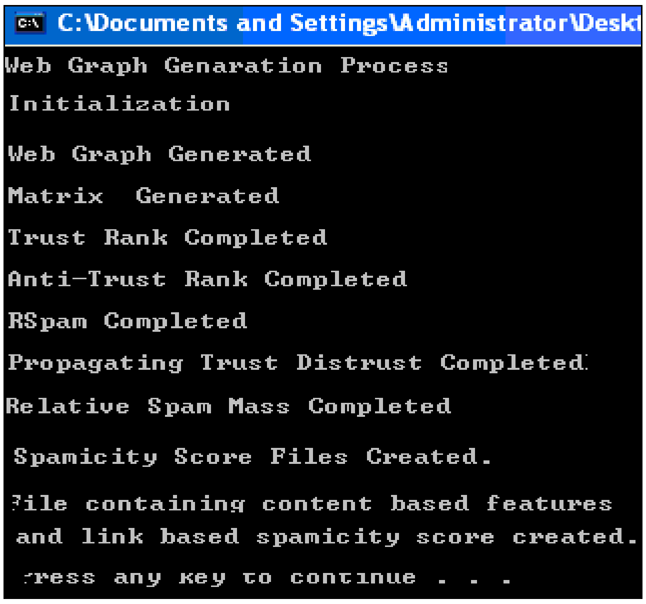
This function is responsible for storing all spam and normal nodes in separate linked lists of spam list and normal list.

unsigned int Find_Inlinks_Outlinks(Hnode *sparse,unsigned int i,int flag)

This function is responsible for finding in-links/out-links saved in the row header node for given row id.

C *Running of Spamizer*

Step 1: Generation of web graph for all the 5 link spam algorithms.

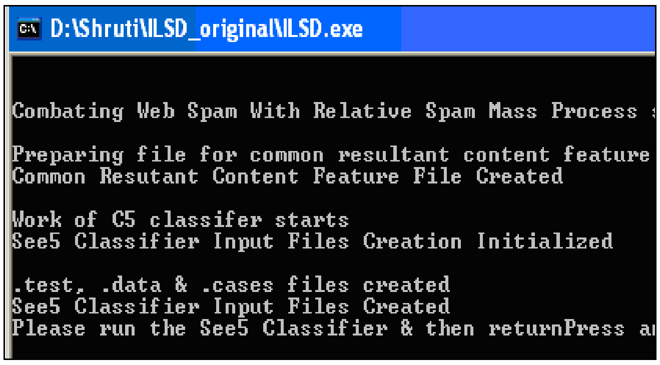


```
C:\Documents and Settings\Administrator\Deskt
Web Graph Genaration Process
Initialization

Web Graph Generated
Matrix  Generated
Trust Rank Completed
Anti-Trust Rank Completed
RSpam Completed
Propagating Trust Distrust Completed
Relative Spam Mass Completed

 Spamicity Score Files Created.

File containing content based features
 and link based spamicity score created.

 ress any key to continue . . .
```

Fig 5. Webgraph Generation

Step 2: Calculation of spamicity score for all the 5 link spam algorithms

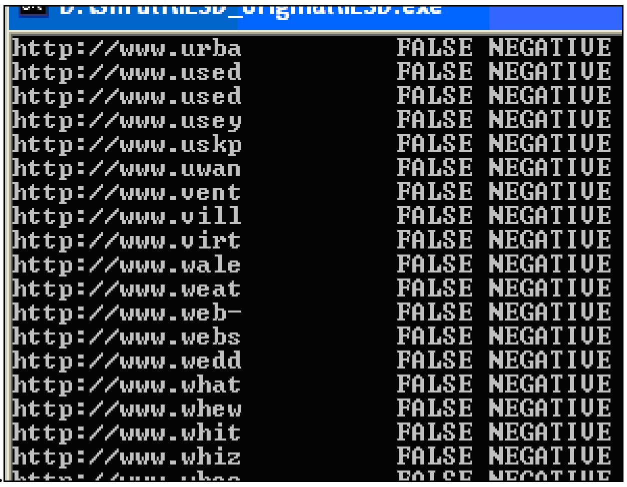
Step 3: Appending spamicity scores to the host file and inputting the resultant file in the See5 Classifier to prepare input for the see 5 classifier [depicted in figure 6].



```
D:\Shruti\ILSD_original\ILSD.exe

Combating Web Spam With Relative Spam Mass Process

Preparing file for common resultant content feature
Common Resutant Content Feature File Created

Work of C5 classifer starts
See5 Classifier Input Files Creation Initialized

.test, .data & .cases files created
See5 Classifier Input Files Created
Please run the See5 Classifier & then returnPress a
```

Fig 6. Preparing input files for the See5 Classifier

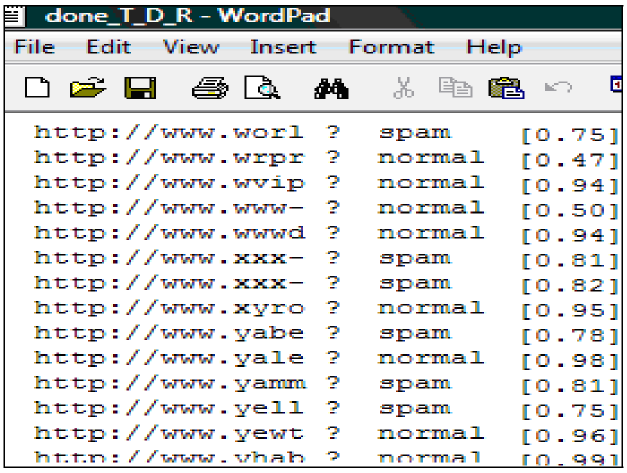
Step 4: Calculation of False Positive/False Negative for each web hosts as depicted in figure 7.



```
http://www.urba            FALSE NEGATIVE
http://www.used            FALSE NEGATIVE
http://www.used            FALSE NEGATIVE
http://www.usey            FALSE NEGATIVE
http://www.uskp            FALSE NEGATIVE
http://www.uwan            FALSE NEGATIVE
http://www.vent            FALSE NEGATIVE
http://www.vill            FALSE NEGATIVE
http://www.virt            FALSE NEGATIVE
http://www.wale            FALSE NEGATIVE
http://www.weat            FALSE NEGATIVE
http://www.web-            FALSE NEGATIVE
http://www.webs            FALSE NEGATIVE
http://www.wedd            FALSE NEGATIVE
http://www.what            FALSE NEGATIVE
http://www.whew            FALSE NEGATIVE
http://www.whit            FALSE NEGATIVE
http://www.whiz            FALSE NEGATIVE
```

Fig 7. Identifying False Positive/False Negative

Step 5: See5 Classifier labels each web host as spam/normal as depicted in figure 8.



```
http://www.worl  ?  spam     [0.75]
http://www.wrpr  ?  normal   [0.47]
http://www.wvip  ?  normal   [0.94]
http://www.www-  ?  normal   [0.50]
http://www.wwwd  ?  normal   [0.94]
http://www.xxx-  ?  spam     [0.81]
http://www.xxx-  ?  spam     [0.82]
http://www.xyro  ?  normal   [0.95]
http://www.yabe  ?  spam     [0.78]
http://www.yale  ?  normal   [0.98]
http://www.yamm  ?  spam     [0.81]
http://www.yell  ?  spam     [0.75]
http://www.yewt  ?  normal   [0.96]
http://www.yhab  ?  normal   [0.99]
```

Fig 6. Spamicity Scores for web hosts

D. *Findings*

Spam detection output (normal/spam) from all five algorithms was combined as a input to the SPAMIZER. Table I list the sample output for "Reverse Spam Algorithm" [18].

Similar output were generated for other link based algorithms and was input in the see 5 classifier to generate Spamizer output as depicted in Table II.

Table I
SAMPLE OUTPUT FOR "REVERSE SPAM ALGORITHM"

HOST ID	**HOSTNAME**	**Label in .test file**	**Label in output file**	**FP/FN**
252	http://all-car-	spam	normal	FALSE NEGATIVE
261	http://allcar-i	spam	normal	FALSE NEGATIVE
279	http://alpha.dc	spam	normal	FALSE NEGATIVE
327	http://andbarne	spam	normal	FALSE NEGATIVE
792	http://blackdog	spam	normal	FALSE NEGATIVE
866	http://books.my	spam	normal	FALSE NEGATIVE
102	http://business	spam	normal	FALSE NEGATIVE
1063	http://californ	spam	normal	FALSE NEGATIVE

Table II
SPAMIZER OUTPUT

Case ID	**Given Class**	**Predicted Class**
http://167glasg ?	Normal	[0.98]
http://24shoppi ?	Normal	[0.49]
http://365sport ?	Normal	[0.94]
http://a2zcheat ?	Normal	[0.47]
http://accelera ?	Normal	[0.65]
http://accentua ?	Normal	[0.97]
http://accessib ?	Normal	[0.97]
http://adamwalt ?	Normal	[0.99]
http://adhesive ?	Normal	[0.48]
http://ads.necg ?	Normal	[0.98]
http://adsys.ne ?	Normal	[0.97]
http://agenda-i ?	Normal	[0.98]
http://airporth ?	spam	[0.63]
http://alexwy.2 ?	spam	[0.72]
http://all-car- ?	spam	[0.48]
http://all-mort ?	spam	[0.69]
http://allcar-i ?	spam	[0.48]
http://allmoney ?	spam	[0.46]
http://alpha.dc ?	Normal	[0.98]
http://amail.co ?	Normal	[0.97]
http://amegrito ?	Normal	[0.99]
http://andbarne ?	Normal	[0.98]
http://angielsk ?	Normal	[0.97]
http://annesudw ?	Normal	[0.97]

E. *Summary of Results*

Table III provides the detail of the experiment conducted with the Spamizer.

Table III
EXPERIMENTAL RESULT

Items	Statistics
Total Number of Items	400
Classified as “Spammed”	60
Classified as “Not Spammed”	40
Correctly Classified Instances	70
Incorrectly classified Instances	30
Mean Absolute Error	0.3022
Precision of “Of Satisfied”	0.72
Recall of “Of Satisfied”	0.87
Precision of “Of Not Satisfied”	0.80
Recall of “Of Not Satisfied”	0.60

Table IV compares the result of Spamizer with the other five link based algorithms discussed in section III. Spamizer works with minimum mean absolute error as compared to other algorithms.

Table IV
COMPARATIVE ANALYSIS

Link Based Algorithm	Mean Absolute Error
Relative Spam Mass Estimation Algorithm	0.501
Trust Rank Algorithm	0.433
Anti-Trust Rank Algorithm	0.420
Web Spam Detection by Propagating Trust and Distrust Scores	0.384
Reverse Spam Rank	0.370
Spamizer (system under study)	**0.3022**

IV. CONCLUSION

Spamizer has been developed to compare and analyze various available link spam algorithms. Five major link spam algorithms, namely Relative Spam Mass Estimation, Trust Rank, Anti-Trust Rank, Propagating trust, and Distrust Scores and Reverse Spam Rank have been implemented in C and spamicity scores have been calculated for each of them. Data set used in Web spam challenge has been used to analyze these algorithms. Spamacity scores generated for each algorithm have been combined to test the spamming intensity in the data set. It was found that by combining the spamicity scores it was possible to predict the label (spam/normal) for the input web hosts with improved accuracy. Spamizer can be further improved to predict more accurate results by improving on content features and combining other anti-spam algorithms. Predicting the label for web host with improved accuracy is the future scope for this research work.

REFERENCES

[1] Monika Henzinger, Rajeev Motwani, and Craig Silverstein, "Challenges in web Search Engines.",SIGIR Forum, 36(2), 2002.
[2] Zoltan Gyongyi, Hector Garcia-Molina," Link Spam Detection Based on Mass Estimation”, http://ilpubs.stanford.edu:8090/697/1/2005-33.pdf.
[3] Bianchini M.,Gori M.,Scarselli F.,"Inside PageRank”, ACM Transactions on Internet, Volume 5 , Issue 1 (February 2005).
[4] Voorhees E. M.,"Evaluation by highly relevant documents.", In SIGIR ’01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, pages 74–82, New York, NY, USA, 2001. ACM Press. http://www2007.org/workshops/paper_51.pdf.
[5] Krishnan V. and Raj R.,"Web spam detection with anti-trust rank",AIRWEB’06, August 10, 2006, Seattle, Washington, USA.
[6] Wu Baoning, Goel Vinay , Davison Brian D. “Propagating Trust and Distrust to Demote Web Spam”, http://www.cse.lehigh.edu/~brian/pubs/2006/MTW/propagating-trust.pdf.
[7] RuleQuest, http://www.rulequest.com/see5-pubs.html.
[8] Zolt´an Gy¨ongyi et al,“Combating Web Spam with TrustRank”, Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004
[9] Web Spam –UK 2007, http://213.27.241.151/webspam/datasets/uk2007/
[10] L.Becchetti, C. Castillo, and D. Donato. Link-based characterization and detection of web spam. In Proc. Of AIRWEB 2006, Seattle, 2006.
[11] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Using rank propagation and probabilistic counting for link-based spam detetection. In Proc. of KDD 2006, Philadelphia, Pennsylvania, 2006.
[12] A. Benczur, K. Csalogany, and T. Sarlos. Link-based similarity search to ¯ght web spam. In Proc. of AIRWEB 2006, Seattle, 2006.
[13] A. Benczur, K. Csalogany, T. Sarlos, and M. Uher. Spamrank { fully automatic link spam detection. In Proc.of AIRWEB 2005, Chiba, 2005.
[14] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. Computer Networks, 33:309{320, 2000.
[15] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In Proc. of KDD 2000, Boston, 2000.
[16] T. Ono, M. Toyoda, and M. Kitsuregawa. An examination of techniques for identifying web spam by link analysis. In Proc. of DEWS 2006, Tokyo, 2006.
[17] B. Wu and B. Davidson. Identifying link farm spam pages.In Proc. of WWW 2005, Tokyo, 2005.
[18] Chenmin Liang1, Liyun Ru2, Xiaoyan Zhu1,” R-SpamRank: A Spam Detection Algorithm Based on Link Analysis”, http://www.sogou.com/labs/paper/R-SpamRank_A_Spam_Detection_Algorithm_Based_on_Link_Analysis.pdf.