Know your neighbor: Web Spam detection using web topology

Deep Prakash Singh, #B00792279

Faculty of Computer Science, Dalhousie University

April 11, 2018

Abstract

This paper is an abstract of the method implemented in [1], which is mainly about web spam. Web spam can significantly deteriorate the quality of the search engine results. With increasing number of websites (more than 1.8B + websites as of today [2]), handling the spammy and non spammy content is quite a problem for web searchers. Majority of website searchers like Google, Yahoo, Bing implementing various algorithms to keep the spam quality in check. This paper is an abstract of spam detection system. This paper finds out that linked hosts tend to belong to the same class, either all the hosts are spam or non spam. There are 3 methods demonstrated while incorporating web topology [1]

1. Clustering the host graph and assigning the labels of the hosts based on the majority voting
2. Propagating the predicted labels to the neighboring hosts
3. Using the predicted labels in the hosts and retraining the classifier

General Terms: Algorithm, classifier, spam
Keywords: Web spam, content spam, link spam (keywords and general terms taken from [1])

1. Introduction

Web is one of the various mediums of sharing any information with excellent platform for sharing any products or services. Once of the majority component of the web is the search engines. Search engines are the heart of the web. With huge searches performing over the web (Google alone crossing more than 40, 000 searches per second!), there is a need to spam control which is being generated over the web. With various spam algorithms in place, this paper proposes one more host graph searching algorithm, where a host graph is created for all the .uk crawled hosts. Spamdexing (also known as search engine spamming) as defined by is defined in [Gyongyi and Garcia-Molina, 2005] as “any action which is meant to trigger an unjustifiably favourable relevance or important for a web page, considering the page true value. Another definition of spam as given in [Perkins, 2001] is defined as “any attempt to deceive the search engine relevancy algorithm” or simply saying, anything which couldn’t have been done if the search engine, didn’t exists.

Some of the characteristics which are taken in consideration which considers a page as spam [5]

1. Any keywords which are not intended for a given web page (i.e. the web page is deceiving the users from the original content which they originally searched)
2. The use of keywords in the web page url, which is another black-hat SEO (search engine optimization) technique where many keywords are added to the url so that the search engine is deceived or tricked in getting the traffic.
3. Redirection or adding a web frame within the page, one of the common technique where a redirection URL is added to the page or a web frame is added to the page which deceives the users from the original content thereby reducing the quality of the web spam.
4. Creation of many duplicate content web pages – Creation of duplicate web pages where same content is diversified across many hosts (or web pages).
5. Adding content or keyword which is of same background as of the page thereby deceiving the original content for which the visitors had visited the page.

From a point of view of a search engine, the web is mainly divided into 2 parts [6]. The closed web (which more closely resembles the smaller, trusted collection information retrieval system which were originally designed for and second the open web, which includes the majority of the web pages. The more openness of the web has seen a one of the important factors for the majority of growth of the web pages. With more amount of success also included the more amount of increasing challenges to the information retrieval system.

Information retrieval tasks such as indexing, gathering, filtering and ranking information from collections can be manipulated easily. That is also known as **spamdexing** as discussed above, this technique is used for manipulating the search engine ranking.

From the perspective of the search engine. Even if they reduce the amount of web spam by delisting any spammy URL but still there is cost of crawling that web page (a crawler is a bot where it gets the content of the web page and pings back to the host). So thereby setting the host as spam will reduce the crawl for that page and thereby reducing the cost of the search engine crawling.

The following image is taken from [2] which depicts the whole web for our current database. As you can see that the hosts (will be discussed in section 2 of the paper) are interconnected to each other (each interconnection means that there is at least 100 pages between them). Black nodes are represented by the spam nodes and white nodes are represented by the non spam nodes. You can see that the similar web pages are linked to each other (i.e. spam nodes are linked back to the spam nodes and non spam are linked back to the non spam nodes. Most of the other connected components represents a single class.

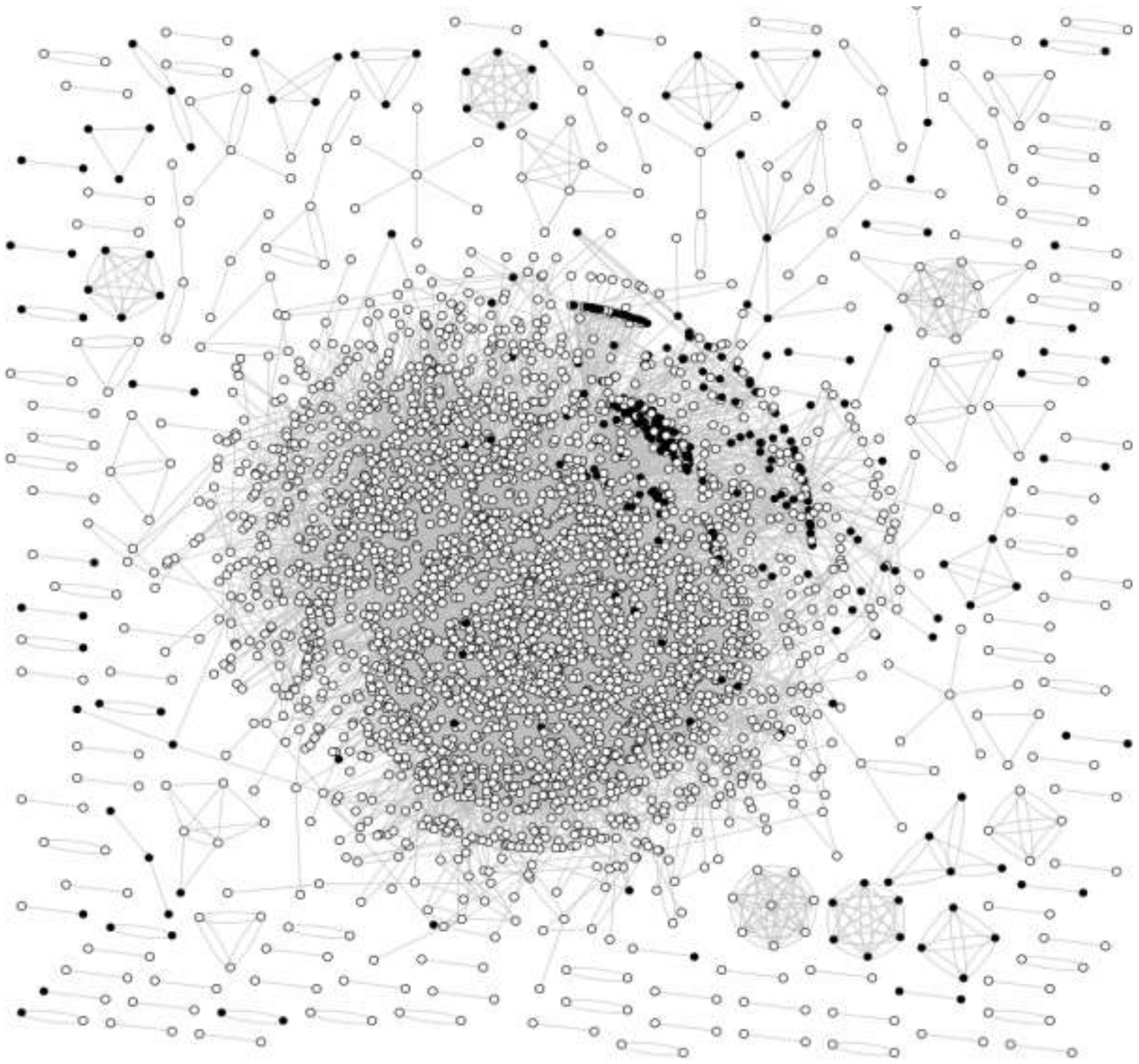

Figure 1: Graphical depiction of the host graph (undirected), pruned to only include the labelled nodes of 100 links between them (Source: Castillo *et al,*" Know your Neighbors: Web Spam Detection using the Web Topology", SIGR'07, July 23-27, 2007, Amsterdam, The Netherlands, pp 423-430)

This paper is organized as follows

1. Section 2 depicts the important keywords which will be used in this paper
2. Section 3 depicts the database which has been used for crawling
3. Section 4 depicts the features and the labels
4. Section 5 depicts the algorithm and code walkthrough which has been used for detecting the spam
5. Section 6 Demo and the code walkthrough of the code (with input and output)
6. Section 7 depicts the results and conclusion for the same

7. Section 8 are the references which have been used for the same

2. Important Keywords which will be used in this paper

This paper will contain some important keywords which will be mentioned again and again. Here is the list of important keywords.

1. **Hosts:** Host refers to the domain here (Please note that the subdomain comes under another entity as compared to the hosts, but the final calculation of the spam score will depend on the subdomain + domain value)
2. **Host Graph:** The undirected links between the hosts, as you can see in the figure 1, it represents the host graph. White nodes are the non spammy nodes and the black nodes are the spammy nodes. The links between the nodes depicts that there are minimum 100 links between the two hosts.
3. **Features:** These are the inputs which will be taken while calculating the spam values
4. **Labels:** These are the output values for a particular host (Here since we can get the output as spam or non spam, so it's a binary classifier)

3. Dataset

This paper is using the publically available WEBSPAM-UK2006 dataset [4]. It is based on the set of pages obtained from crawling the .uk domains. The data set was collected in May 2006 by the research group of the Laboratory of Web Algorithmics1 at the Universit`a degli Studi di Milano. [1]

The dataset was obtained using the UbiCrawler [7] using bfs(breadth first search). The crawling started from the large groups of seed pages as listed in Open Directory project. This seed of web pages contained more than 190.000 URLs in about 150.000 hosts. As the result 77.900.000 pages were collected as the part of process corresponding to around 11.400 hosts.

For each page which was crawled, both the links and the content was obtained. The graph which was formed after collection contained more than 3.000.000.000 edges and is stored in the compressed format as described in Boldi and Vigni [8] using 2.9bits per edge for a total size of around 1.2GB. The content data is distributed in 8 compressed volumes of 55GB each.

There were a group of volunteers coordinated by Universita di Roma, was asked to label each hosts as **normal**, **borderline** or **spam**. The complete description of classification is presented in [4].

The distribution of the host labels as judged by the human volunteers are given as below

Label	**Frequency**	**Percentage**
Normal	4.406	61.75%
Spam	1.447	22.08%
Borderline	709	10.82%
Could not be classified	350	5.34%

Table 1: Distribution of the host labels as judged by the human volunteers

4. Features and Labels

This paper deals with 2 major features [1]

1. Link based features – Link based features are computed for the home pag and the page in each host with maximum page rank. Other features such as the trust rank are computed directly over the graph. The summary of the features is described below
    a. **Page rank:** It computes a particular score for each page. There are total 11-page rank based attributes which are measured
    b. **Trust rank:** The idea for trust rank is that, if a particular page has a high page rank but it is not linked with other high page rank then that page is most probably a spam page. Using trust rank we can estimate the spam mass of a page i.e. the amount of page rank received from a spammer. Figure 2 represents the relation between the page rank and the trust rank.

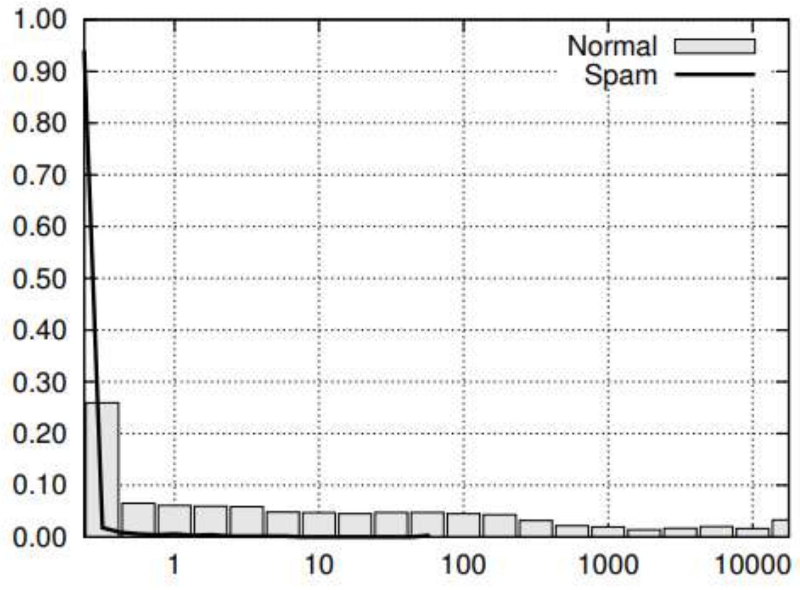

Figure 2: Histogram of the ratio between the page rank and the trust rank for most of the pages (Source: Castillo *et al,*" Know your Neighbors: Web Spam Detection using the Web Topology", SIGR'07, July 23-27, 2007, Amsterdam, The Netherlands, pp 423-430)

c. **Estimation of supporters:** Given 2 nodes between x and y, we say that x is a d-supporter of y, if the shortest path between 2 nodes is d. It is based on the classical probabilistic counting algorithm proposed by Flajolet and Martin [9].

2. Content based features – For each web page, the paper proposes various content based parameters.
    a. **Number of words in the page, title and the average word length:** For the page features we calculate the number of words in the page and the title of the page and calculate the average word length of the page + title and store that value.

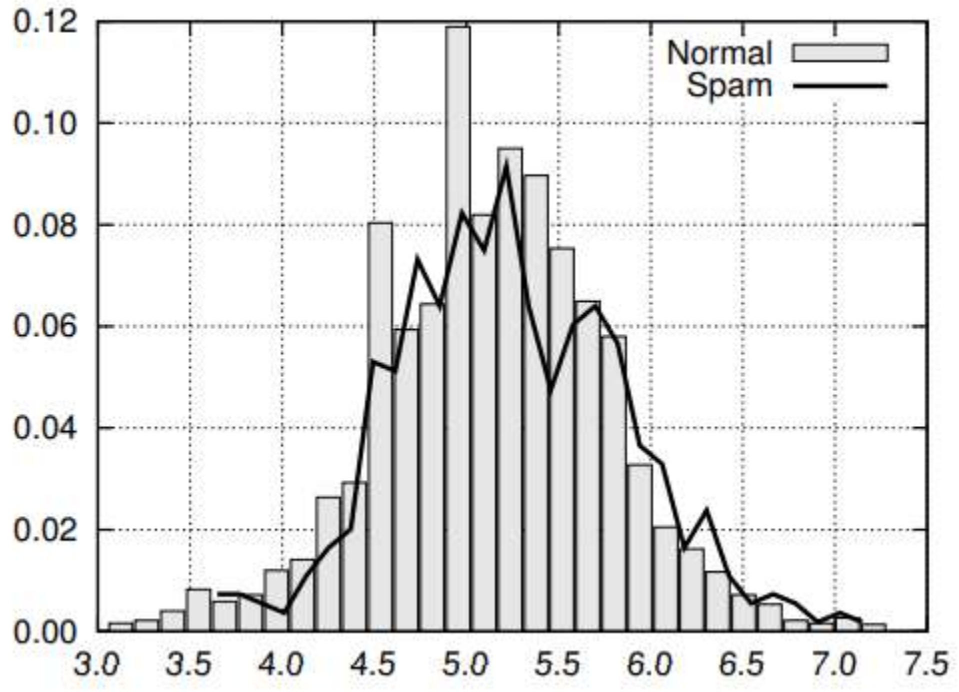

Figure 3: Histogram of the average word length between the spam and non spam pages (Source: Castillo *et al,*" Know your Neighbors: Web Spam Detection using the Web Topology", SIGR'07, July 23-27, 2007, Amsterdam, The Netherlands, pp 423-430)

b. **Fraction of the anchor and the visible text:** Fraction of the number of page in the anchor and the visible text**.**

c. **Compression rate:** Compression ratio is the size of the compressed text and the size of the uncompressed text

d. **Query precision and query recall:** Consider a set q for the most popular terms in a query then calculate the query precision and recall, these values are analogues to the precision and recall.

5. Algorithm

We are provided with the hostnames and the graph of outward linking nodes. With the hostnames and the graph, we will create a host graph (Please refer to the section 2 for the host graph definition). Once the host graph is created then we will inject the spam and non-spam values in the host graph. Once the spam and non spam values are injected we will calculate the spam score based on the spam values and once the spam score is calculated then we will display the predictions of the labels from the given training data. The complete flow is given as below: -

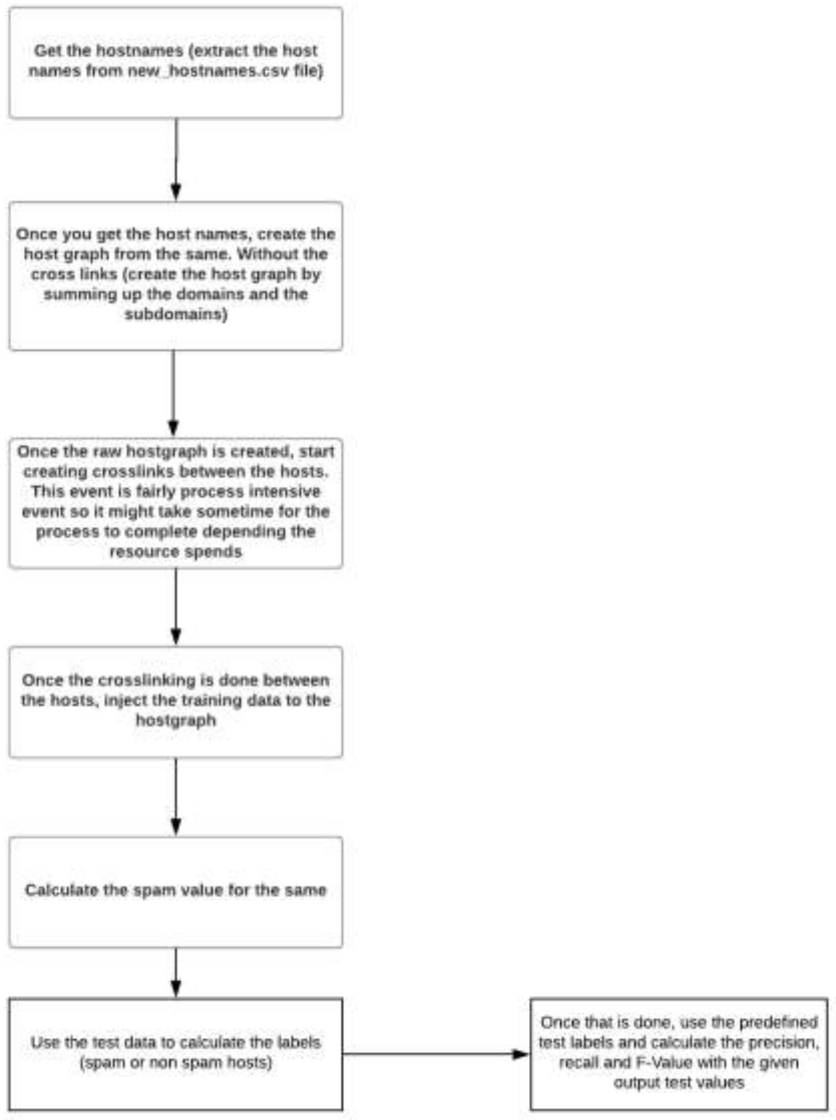
Get the hostnames (extract the host names from new_hostnames.csv file)
Once you get the host names, create the host graph from the same. Without the cross links (create the host graph by summing up the domains and the subdomains)
Once the raw hostgraph is created, start creating crosslinks between the hosts. This event is fairly process intensive event so it might take sometime for the process to complete depending the resource spends
Once the crosslinking is done between the hosts, inject the training data to the hostgraph
Calculate the spam value for the same
Use the test data to calculate the labels (spam or non spam hosts)
Once that is done, use the predefined test labels and calculate the precision, recall and F-Value with the given output test values

Figure 4: Algorithm to calculate the spam values from the training features and then get the output label from the given training features.

The algorithm works according to the following steps

Step 1: Get the host names form the csv file (please note it's mandatory to have that csv file else the program will fail). This will create a dataset of the hostnames.

Step 2: Once you get the csv files, create a host graph for the same. The host-graph should have the domains and the subdomain linked to each other and form the dependencies between various hosts

Step 3: Once your host-graph is created, then start injecting the infected and non infected hosts in the graph and calculate the spam values for the same.

Step 4: Once the spam values have been calculated then use the testing labels to test the output for the same.

Step 5: Calculate the precision, recall and F-Values for the same

6. Demo and code walkthrough

The file structure is as below

1. consoleColors.py: This is a class file displaying the colors on the output screen
2. customExceptions.py: Class containing all the custom exceptions which are created
3. fielLoader.py: The core file which loads all the csv files and create the hostgraph
4. start.py: The beginning point of the code, the code will run via this python file
5. tree.py: This file contains the structure of the nodes which will be created
6. new_hostnames.csv: File containing all the hostnames
7. uk-2006-05.hostgraph_weighted.txt: File containing the ID of all hosts along with destination links
8. webspam-uk2006-set1-labels-DomainOrTwoHumans.txt: This file contains the input labels. This is the train data file.
9. webspam-uk2006-set2-labels-DomainOrTwoHumans.txt: This file contains the labels for the test file. This is the test file.

The code structures is given as below

1. consoleColors.py
    a. class bcolors – Class containing all the console colors labels
2. customExceptions.py
    a. coreFileNotFoundException – This will arise when one of the core file is not found
    b. NoDataAvailableException – This exception will arise when there is no data available
    c. invalidRowException –This exception will arise when there is an invalid row (row not in correct provided format)

    d. invalidDataException – If there is an invalid data (although the row is in correct format but data is not correct)

3. fileLoader.py
    a. Function fnLoadHostNames()- This function will load the host names
    b. Function fnLoadHostGraph(paramListHosts) – This function will load the hostgraph provided with the input parameter as the list of hosts and will create the crosslinks (graph) for the same.
    c. Function fnInjectTrainLabelsTohostGraph(paramhostGraph, paramHostNames) – This function will load the hostnames and hostgraph and will update the graph with the training labels.
    d. Function fnPrecdictTestLabels(paramTestURL, paramHostNames) – This function will predict the output from the hostnames and then it will calculate recall, precision and F-Measure.
4. Tree.py
    a. Class node – This is a class containing the structure body of the node created.

The output screen is as below

```
------------- start of output ---------------

Initializing constructor for data load class

Checking core files

Checking file at path /home/ubuntu/workspace/new_hostnames.csv File exists

Checking file at path /home/ubuntu/workspace/webspam-uk2006-set1-labels-
DomainOrTwoHumans.txt File exists

Checking file at path /home/ubuntu/workspace/uk-2006-05.hostgraph_weighted.txt File exists

Checking file at path /home/ubuntu/workspace/webspam-uk2006-set2-labels-
DomainOrTwoHumans.txt File exists

Reading total Hosts available

Reading host ID : 11401

All host read

Loading host graph

Creating Raw HostGraph (without crosslinks)  ( 100.0% )

Raw Host graph created successfully
```

```
Cross linking hosts in graph

This might take some time (depending on the processor/server)

Cross links Hosts for Host ID : 11401 ( 100.0% )

Host Graph Created Successfully

Injecting the training data labels to the host graph

Updating host spam values based on the training data

Training the model with the input label data  ( 100.0% )

Model Trained

Optimizing the model

Optimizing the model with the input label data  ( 100.0% )

Successfully trained the host graph

Predicting labels from the test file

Predicting the host [host ID : 1850]  ( 100.0% )

Results successfully written to output.txt file

Time taken : 158.41 seconds

------------- end of output ---------------
```

The output.txt file is as below

```
----------------- start of output -----------------

Spam prediction using host graph

 Name : Deep Prakash Singh

 B00 Number : B00792279

 -----------------------------

007cleaningagent.co.uk normal normal 0.0  Correct

0800.loan-line.co.uk normal normal 0  Correct

1-hydroponics.co.uk normal normal 0.0  Correct

102belfast.boys-brigade.org.uk normal spam 0  Incorrect

10bristol.boys-brigade.org.uk spam spam 0.9375  Correct
```

```
……………………. Output hosts truncated ………………………..

dolphin.wmin.ac.uk normal spam 0  Incorrect

dolphinhotelcambs.co.uk normal normal 0.0  Correct


 --------------------

Recall : 0.2

False positive rate : 0.0

Precision : 1.0

F-Measure : 0.33

Accuracy : 0.46

----------------- end of output -----------------
```

7. Results and conclusion

The recall was 0.2, false positive rate : 0 and the precision is 1 and the accuracy is 0.46 and the f-measure is 0.33. The f-measure could have been improved with various techniques provided in the paper, but due to time constraint those smoothing techniques could not be implemented. The false positive rate of 0 is a good indicator that there was no spam nodes labelled as non spam. All non spam nodes were labelled as non spam, but some of the spam nodes were labelled as non-spam.

Further research can be conducted by implementing some tree optimization techniques and how some of the tree optimizations can be improved by implementing some smoothing techniques for the same.

8. References

1. Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, Fabrizio Silvestri" Know your Neighbors: Web Spam Detection using the Web Topology", SIGR'07, July 23-27, 2007, Amsterdam, The Netherlands, pp 423-430

2. http://www.internetlivestats.com/total-number-of-websites/

3. http://www.internetlivestats.com/google-search-statistics/

4. C. Castillo, D. Donato, L. Becchetti, P. Boldi, M. Santini, and S. Vigna. A reference collection for web spam detection. Technical report, DELIS, September 2006.

5. http://chato.cl/papers/becchetti_2008_link_spam_techniques.pdf

6. T. A. Brooks. Web search: how the Web has changed information retrieval. Information Research, 8(3), April 2003.

7. P. Boldi, B. Codenotti, M. Santini, and S. Vigna. Ubicrawler: a scalable fully distributed web crawler. Software, Practice and Experience, 34(8):711–726, 2004.

8. P. Boldi and S. Vigna. The webgraph framework I: compression techniques. In WWW ’04: Proceedings of the 13th international conference on World Wide Web, pages 595–602, New York, NY, USA, 2004. ACM Press

9. P. Flajolet and N. G. Martin. Probabilistic counting algorithms for data base applications. Journal of Computer and System Sciences, 31(2):182–209, 1985.