Department of Computing

Methods for Demoting and Detecting Web Spam

Goh Kwang Leng

This thesis is presented for the Degree of
Master of Philosophy (Computer Science)
of
Curtin University

April 2013

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature : …………………………

(GOH KWANG LENG)

Date : 21st April 2013

Abstract

Web spamming has tremendously subverted the ranking mechanism of information retrieval in Web search engines. It manipulates data source maliciously either by contents or links with the intention of contributing negative impacts to Web search results. The altering order of the search results by spammers has increased the difficulty level of searching and time consumption for Web users to retrieve relevant information. In order to improve the quality of Web search engines results, the design of anti-Web spam techniques are developed in this thesis to detect and demote Web spam via trust and distrust and Web spam classification.

A comprehensive literature on existing anti-Web spam techniques emphasizing on trust and distrust model and machine learning model is presented. Furthermore, several experiments are conducted to show the vulnerability of ranking algorithm towards Web spam. Two public available Web spam datasets are used for the experiments throughout the thesis - WEBSPAM-UK2006 and WEBSPAM-UK2007.

Two link-based trust and distrust model algorithms are presented subsequently: Trust Propagation Rank and Trust Propagation Spam Mass. Both algorithms semi automatically detect and demote Web spam based on limited human experts' evaluation of non-spam and spam pages. In the experiments, the results for Trust Propagation Rank and Trust Propagation Spam Mass have achieved up to 10.88% and 43.94% improvement over the benchmark algorithms.

Thereafter, the weight properties which associated as the linkage between two Web hosts are introduced into the task of Web spam detection. In most studies, the weight properties are involved in ranking mechanism; in this research work, the weight properties are incorporated into distrust based algorithms to detect more spam. The experiments have shown that the weight properties enhanced existing distrust based Web spam detection algorithms for up to 30.26% and 31.30% on both aforementioned datasets.

Even though the integration of weight properties has shown significant results in detecting Web spam, the discussion on distrust seed set propagation algorithm is presented to further enhance the Web spam detection experience. Distrust seed set propagation algorithm propagates the distrust score in a wider range to estimate the probability of other unevaluated Web pages for being spam. The experimental results have shown that the algorithm improved the distrust based Web spam detection algorithms up to 19.47% and 25.17% on both datasets.

An alternative machine learning classifier - multilayered perceptron neural network is proposed in the thesis to further improve the detection rate of Web spam. In the experiments, the detection rate of Web spam using multilayered perceptron neural network has increased up to 14.02% and 3.53% over the conventional classifier – support vector machines. At the same time, a mechanism to determine the number of hidden neurons for multilayered perceptron neural network is presented in this thesis to simplify the designing process of network structure.

Acknowledgement

I would like to express my warm and sincere gratitude to my supervisor, Associate Professor Dr. Ashutosh Kumar Singh for his endless support and guidance throughout my Master's studies - Master of Philosophy in Computer Science. His availability for discussion on both technical and non-technical issues has been very beneficial. His vision and foresight have inspired and motivated me in my research projects. Deeply Appreciated.

I would also like to gratefully thank my research partners, Ravi Kumar Patchmuthu and Garenth Lim King Hann for their valuable assistance that has provided me numerous ways in my research. Their supports have made my work stronger and also better.

I would also like to express my appreciation to my fellow mate, Lue Ik Hong and colleague, Dr. Wee Siaw Khur who have been there for my good times and difficult times. Also, special thanks to Bridgid Chin Lai Fui, Dr. Khor Ee Huey and Lai Zhen Yue who have dedicated their time to assist in my thesis writing. At the same time, I would like to thank postgraduate students and academic staffs in Curtin University.

Last but not least, I would like to thank Curtin University, Sarawak Campus for the scholarship entitled "Curtin University, Sarawak Malaysia Higher Degree by Research Scholarship 2011" and award entitled "Curtin Sarawak Postgraduate Stipend Scholarship Award". This scholarship and stipend award have been a great support for me throughout my research projects.

Publications

JOURNAL PAPERS

1. **Goh, Kwang Leng**, Ravi Kumar Patchmuthu, Ashutosh Kumar Singh, and Anand Mohan. "TPRank Contend Web Spam with Trust Propagation", Taylor and Francis Cybernetics and Systems An International Journal. (Under Review)

2. **Goh, Kwang Leng**, Ravi Kumar Patchmuthu, and Ashutosh Kumar Singh. "Link-based Web Spam Detection using Weight Properties", Springer Journal of Intelligent Information Systems. (Under Review)

3. **Goh, Kwang Leng**, Ravi Kumar Patchmuthu, and Ashutosh Kumar Singh. "Distrust Seed Set Propagation Algorithm To Detect Web Spam", Elsevier Information Processing & Management. (Under Review)

4. **Goh, Kwang Leng**, Ravi Kumar Patchmuthu, Ashutosh Kumar Singh, and Anand Mohan. 2012. "Link Based Spam Algorithms in Adversarial Information Retrieval" Cybernetics and Systems: An International Journal 43 (6): 459-475. doi: 10.1080/01969722.2012.707491.

REFERRED CONFERENCE PAPERS

1. **Goh, Kwang Leng**, Ashutosh Kumar Singh, and King Hann Lim. "Multilayer Neural Network based Web Spam Detection Application", 1st IEEE China Summit & International Conference on Signal and Information Processing (ChinaSIP 2013), Beijing, China. (Accepted)

2. **Goh, Kwang Leng**, Ravi Kumar Patchmuthu, and Ashutosh Kumar Singh. "Incorporating Weight Properties in Detection of Web Spam", In Proceedings of 2012 International Conference on Uncertainty Reasoning and Knowledge Engineering, Aug 14-15, 2012, Jakarta, Indonesia, IEEE CPMG, NJ (ISBN: 978-1-4673-1458-9) 18-21.

3. Singh, Ashutosh Kumar, Ravi Kumar Patchmuthu, **Kwang Leng Goh**, "An Experimental Study on Spam Detection Algorithms" International Technical Conference of IEEE Region 10 Asia Pacific TENCON 2011, Nov 21-24, 2011, Bali, Indonesia, (ISBN: 978-1-4577-0256-3) 1382-1385.

Table of Contents

List of Tables

List of Figures

Abbreviations

ATR	Anti-TrustRank
AUC	Area Under an Receiver Operating Characteristic Curve
BFGS	Broyden-Fletcher-Goldfrab-Shanno
BP	Back Propagation
BR	BadRank
CGL	Conjugate Gradient Algorithm with Line Search
DISTR	Wu et al. (Wu, Goel, and Davison 2006a) Distrust Algorithm
Distrust	Nie et al. (Nie, Wu, and Davison 2007) Distrust Algorithm
DSP	Distrust Seed Set Propagation
DT	Decision Tree
GNN	Graph Neural Network
HITS	Hyperlink-Induced Topic Search
HR	HostRank
IGR	In-Link Growth Rate
IPR	Inverse PageRank
IR	Information Retrieval
LDA	Latent Dirichlet Allocation
LV	Link Variable
MLP	Multilayer Perceptrons
PM	Probability Mapping
PR	PageRank
SALSA	Stochastic Approach for Link-Structure Analysis
SCG	Scaled Conjugate Gradient
SM	Spam Mass
SOM	Self-organizing Maps
SVM	Support Vector Machine
TP	Trust Propagation
TR	TrustRank
WATR	Weight Anti-TrustRank
WDISTR	Weighted Wu et al. (Wu, Goel, and Davison 2006a) Distrust Algorithm
WDistrust	Weighted Nie et al. (Nie, Wu, and Davison 2007) Distrust Algorithm
WITCH	Web Identification Through Content and Hyperlinks

Glossary

c	Constant
c_1	Positive constant 1
c_2	Positive constant 2
d	Distrust score
$\overline{d_i}$	Normalized distrust score
e	Exponential
f	Function 1
g	Function 1
h	Diffusion score vector
i_G	Number of pure good vertices
i_X	Number of unevaluated vertices
n	Positive constant 3
n_0	Positive constant 4
o	Step size
r	Row vector
s	A vector which denote $1/\deg^{+}(\upsilon)$
t	Trust score
$\bar{t}$	Normalized Trust score
u	The iterations
$\mathbf{a}$	Input for neurons in hidden layer
$\mathbf{b}$	Output for neurons in hidden layer
$\mathbf{c}$	Input for neurons in output layer
$\mathbf{d}$	Desired output
$\mathbf{i}$	Iterative variable for input neurons
$\mathbf{j}$	Iterative variable for hidden neurons
$\mathbf{k}$	Iterative variable for output neurons
$\mathbf{p}$	Number of neurons in input layer
$\mathbf{q}$	Number of neurons in hidden layer
$\mathbf{r}$	Number of neurons in output layer
$\mathbf{w}$	Weights between neurons
$\mathbf{x}$	A set of inputs
$\mathbf{y}$	A set of outputs
B	Spam vector
$\overline{B}$	Normalized spam vector

E	Global error function
$E^{'}$	Gradient to global error function
$E^{''}$	Hessian Matrix to global error function
G	Graph
G_H	Host graph
G_W	Weighted host graph
I	Inverse transition matrix
K	Number of features
M	Number of iterations
N	Number of vertices
O	The new weight function
P	Bucket position
R	Average promotion
S_{in}	The set of in links
Sn	Propagation Coverage
T	Transition matrix
$\mathbf{F}$	Activation functions in hidden layer
$\mathbf{H}$	Activation functions in output layer
$\deg^{-}$	In-degrees
$\deg^{+}$	Out-degrees
υ	Vertices
υ_E	Evaluated vertices
υ_G	Pure good vertices
υ_H	Host vertices
υ_N	Non-spam vertices
υ_S	Spam vertices
υ_U	Ugly vertices
υ_X	Unknown vertices
$\overline{\upsilon}_E$	Unevaluated vertices
ε	Edges
ε_H	Host edges
α	Decay factor
β	Bias term
ω	Weight function
ϖ	Total weight
∞	Infinite
$\Re$	Real numbers
$\Re_+$	Positive real numbers
ξ	Error

σ	Scaling value 1
λ	Scaling value 2
$\overline{\lambda}$	Scaling value 3
ψ	Conjugate gradient direction
ϑ	Steepest descent direction
η	Percentage of trust propagated
τ	Distribution vector
γ	Thermal conductivity coefficient
μ	Timeline
κ	Special Evaluation
δ	Second order information
Δ_u	Comparison parameter
ℓ	Total number of weights linkage
^{T}	Transpose

Chapter 1 Introduction

According to a survey conducted by an Internet service company - NetCraft, an estimation of 629,939,191 Web sites are scattered around in the World Wide Web (Netcraft 2013). Nowadays, the Web search engine has become default information retrieval tool to ease Web users' needs to extract relevant information; however searching for relevant data in this information warehouse can be a challenging task since the World Wide Web is known to be the largest knowledge repository mankind ever created.

Traditionally, Web search engines did not take the ranking order of Web documents into serious consideration. The search engines employed a computer program known as Web crawlers or Web spiders to find and download Web pages, and incorporate another program to arrange the documents based on some wordings such as domain name, headings of Web page, page title, anchor text and meta data (Kobayashi and Takeda 2000; Baeza-Yates and Ribeiro-Neto 1999). In recent years, Web search engines have incorporated hyperlinks into the ranking mechanism. Authors of Web pages created hyperlinks as references to link up with another Web page. These referrals provide valuable information between documents and records of user behaviour. The idea of studying these referrals in information retrieval is commonly known as *link analysis* (Henzinger 2000).

Link analysis is an emerging technology that tries to comprehend the relationships between Web documents, thus providing an order of search results according to its importance and relevance based on users' queries. This technology developed algorithms: The first link analysis algorithm was developed by Li YanHong, is the RankDex technology (Li 1998); It was incorporated in the search engine to measure the quality of Websites ("About Rankdex" 1997). PageRank (Brin and Page 1998), developed by Sergey Brin and Larry Page, which was used in the famous Google search engine, modelled its algorithm based on probability of a random surfer for their search engine. Jon Kleinberg (Kleinberg 1999) proposed of hyperlink-induced topic search (HITS), which introduced the authorities and hubs of a Web page to rate Web pages. And lastly, stochastic approach for link-structure analysis (Lempel and Moran 2001) also known as SALSA, proposed by Lempel and Moran, examined random walks on graphs derived from the link-structure to rank Web pages. Borodin et al. (Borodin et al. 2005) had already provided a detailed study on link analysis algorithms, including its background theory and experimental results.

With exponential growth of the World Wide Web, retrieving the right information in a short time remains a challenging task. Web users only look at the top few pages of the search results (Jansen, Spink, and Saracevic 2000). This is one of the reasons the commercial industries are striving to have their Web sites appear at the top of search results. As more viewers visit, the more financial gain one would be generated.

In recent times, there are a lot of indecent tricks used by the content providers to have their pages rank higher than they deserved. This is because the order of the results is highly correlated to the profit gain of one business model. The most efficient way is to manipulate the link analysis algorithms. This unethical way of affecting the ranking order of search engines has evolved into *Web spamming*, also known as *spamdexing* (Gyöngyi and Garcia-Molina 2005).

In 2006, it was estimated that approximately one seventh of English webpages were spam, which became obstacles in users' information-acquisition process (Wang, Ma, et al. 2007). In 2007, the cost of Web spam was estimated at US$ 100 billion globally and United States alone suffered an estimated cost of US$ 35 billion (Bauer, Eeten, and Wu 2008). The intention of Web spam is to mislead search engines by boosting one page to undeserved rank. Consequently, it leads Web user to the irrelevant information. This kind of exploitation degrades the Web search engines by providing inappropriate or bias query results. Henzinger et al. (Henzinger, Motwani, and Silverstein 2002) have identified Web spam as one of the most important challenges in Web search engine industries. Many people become frustrated by constantly finding spam sites when they look for legitimate content. In addition, Web spam has an economic impact since a high ranking provides large free advertising and so an increase in the Web traffic volume (Araujo and Martinez-Romo 2010). Even worse, at least 1.3% of all search queries directed to the Google search engine contain results that link to malicious pages (Egele, Kolbitsch, and Platzer 2011). In addition, one consultancy estimated that Russian spammers earned roughly US$2–3M per year and one IBM representative claimed that a single spamming botnet was earning close to $2M per day (Kanich et al. 2011). Search engine companies generally employ human experts who specialized in detecting Web spam, constantly scanning the web looking for spamming activities. However, the spam detection process is time-consuming, expensive and difficult to automate.

Gyongyi et al. (Gyöngyi and Garcia-Molina 2005) raised the interest of the anti-Web spam community by writing a comprehensive taxonomy of all spamming techniques including boosting and hiding techniques. *Boosting techniques* refer to methods that achieve high relevance or importance for one page; *hiding techniques* refer to methods that do not influences the ranking of search engine but assist boosting techniques from the view of the user, one example is to manipulate the color scheme of the anchor text. Boosting techniques can be further expanded into *term spamming* (which also refers as content spamming) and *link spamming* while hiding techniques can be expanded into content hiding, cloaking and redirection as shown in Figure 1.1.

In addition, Wu and Davison (Wu and Davison 2005a) did a detailed research on cloaking and redirection. *Cloaking* can be explained by giving the Web user different content from what a search engine sees. *Redirection* on the other hand can be explained by sending the Web user to another URL (Uniform Resource Locator) while

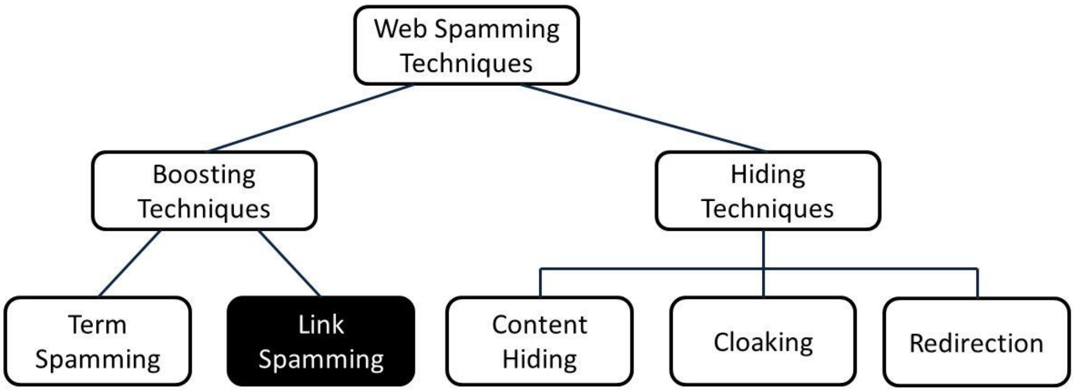


Figure 1.1: Categorization of Web spamming techniques

loading current URL. *Content hiding* refers to spam terms or links in a Web page that are invisible to the user.

Understanding spamming techniques is important in order to propose the appropriate counter-measures. In Wu's dissertation (Wu 2009), he mentioned different approaches to combat Web Spam (shown in Figure 1.2).

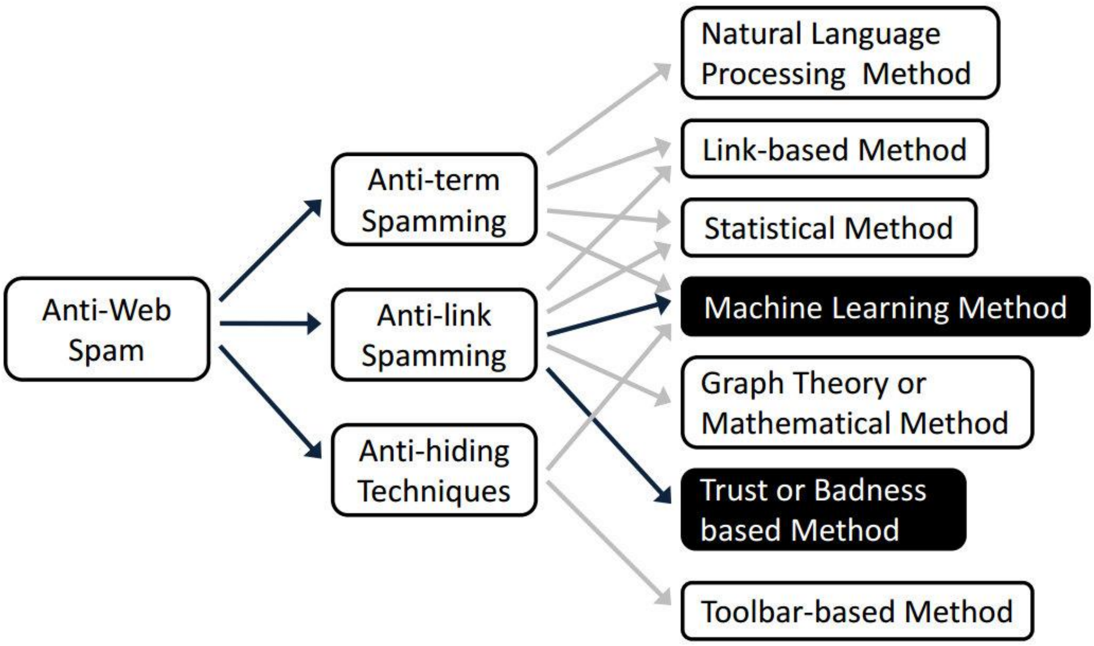


Figure 1.2: Different approaches to combat Web spam

Among the anti-Web spam techniques, *trust or badness based method* (or *trust and distrust model*) algorithms have shown significant results in eliminating Web spam (Zhang, Wang, et al. 2011). Initially the algorithms run a *seed selection process*, which a portion of a large Web is selected and evaluated as spam or non-spam to form *seed sets*. Based on the evaluated seed sets, spam and non-spam are used to propagate distrust for detection and trust for demotion of Web spam.

Trust and distrust model can be categorized into two types of algorithms: Web spam detection and Web spam demotion. Both detection and demotion of Web spam are equally important in combating Web spam. Demotion of Web spam can act as a counter-bias in reducing possible rank boosts from spam whereas detection of Web spam can help out in removing them at the earliest stage.

Besides trust or badness based method, machine learning methods have been actively used for detection of Web spam in recent years. Machine learning approach in anti-Web spam community can be divided into two sections: feature and structure. A *feature* is an individual specification of an attribute whereas *structure* is a machine learning model that takes features for classification purpose. Some of the aforementioned trust or badness based algorithms are used as features to assist machines to learn the underlying patterns of Web spam.

The research objectives of this thesis are to develop anti-Web spam algorithms based on trust and badness model for detection and demotion of Web spam and to propose an alternative machine learning model to assist human experts in the task of Web spam classification.

The notion of content trust was first introduced by Gil et al. to solve the problem of reliability of the Web resource (Gil and Artz 2007). Trust is an integral component in many kinds of human interaction, allowing people to act under uncertainty and with the risk of negative consequences (Wang and Zeng 2007; Wang, Zeng, et al. 2007). Thus, trust is used to model the reliability of the information and solve the problem of Web spam detection. On the other hand, since spammers employ propagandistic techniques, it makes sense to design anti-propagandistic methods for defending them (Metaxas 2009b). These methods need to be user-initiated, that is the user decides which Web site not to trust and then seeks to distrust those supporting the untrustworthy Web site (Metaxas 2009a). Furthermore, among the anti-Web spam techniques, link-based trust and distrust algorithms that propagate human experts' judgments over a set of seed pages are the most promising, considering the effectiveness, efficiency and simplicity (Zhang, Wang, et al. 2011; Liu et al. 2013).

The development of an automatic Web spam detection system is an interesting problem as it concerns massive amounts of data to be analysed, the involvement of multi-dimensional attribute space with potentially hundreds or thousands of dimensions, and the extremely dynamic nature for novel spamming techniques that emerge continuously (Sydow et al. 2007). Often, large amount of Web spam pages are generated using machines by stitching together grammatically from a large collection of sentences (Fetterly, Manasse, and Najork 2005). Thus, machine learning method provides an ideal solution due to its adaptive ability to learn the underlying patterns for classifying spam and non-spam (Erdélyi, Garzó, and Benczúr 2011).

In this thesis, a proposed trust propagation algorithm is developed to assist in detection and demotion of Web spam. Subsequently, existing anti-Web spam algorithms combine with proposed extracted-host weight feature are developed to enhance the Web spam detection experience. Thereafter, a distrust seed set propagation algorithm also combining with anti-Web spam algorithms is proposed to increase the detection rate of Web spam. Lastly, the application of machine learning technique namely multilayered perceptrons neural network is proposed to classify Web pages into spam and non-spam.

The thesis organization is stated in the following:

In Chapter 2, the mathematical model for Web graph is presented to formulate the algorithms effectively. After that, two large public available datasets – WEBSPAM-UK2006 and WEBSPAM-UK2007 and their provided features vectors which are used in machine learning are thoroughly described. The parameters setting and performance evaluation for all the algorithms end with this chapter.

In Chapter 3, a comprehensive study on two anti-Web spam techniques is presented – trust and badness based method, and machine learning method. Firstly, a trust model link-based anti-Web spam algorithm – TrustRank (Gyöngyi, Garcia-Molina, and Pedersen 2004) is presented to show the effectiveness of the trust model. The weaknesses of TrustRank came up with the proposed of its derivatives. Thus, the derivatives of TrustRank which include Anti-TrustRank (Krishnan and Raj 2006), Topical TrustRank (Wu, Goel, and Davison 2006b), DiffusionRank (Yang, King, and Lyu 2007) and Link-Variable TrustRank (Qi, Song-Nian, and Sisi 2008) are presented. The experiments between TrustRank and HostRank (Eiron, McCurley, and Tomlin 2004) shows the vulnerability of link analysis algorithms towards spam. After that, other trust and distrust model based algorithms are briefly explained. Lastly, the machine learning techniques that are used in combating Web spam are further discussed.

Chapter 4 covers the trust model algorithms – Trust Propagation Rank (TPRank) and Trust Propagation Spam Mass (TP Spam Mass). TrustRank (Gyöngyi, Garcia-Molina, and Pedersen 2004) and Spam Mass (Gyöngyi et al. 2006) offer the advantage of the trust evaluations and propagate trust to demote and detect Web spam. The proposed trust propagation algorithms further improve the aforementioned algorithms and the experiments have shown that the proposed trust propagation algorithms outperform both TrustRank and Spam Mass based on the same small amount of evaluated sites.

Chapter 5 introduces weight properties feature extracted from Host graph to enhance the existing Web spam detection algorithms. Weight properties can be defined as the influences of one Web node towards another Web node. Weight properties had been investigated by other researchers (Xing and Ghorbani 2004; Nemirovsky and Avrachenkov 2008; Li, Shang, and Zhang 2002) to achieve better results for ranking algorithms based on PageRank and their derivatives. However, there are no studies focusing on the incorporation of weight properties in detecting Web spam hence this method is implemented in this research. It is found that the experimental results shows that the weight properties have improved the existing Web spam detection algorithms like Anti-TrustRank (Krishnan and Raj 2006), Wu et al. Distrust (Wu, Goel, and Davison 2006a) and Nie et al. Distrust (Nie, Wu, and Davison 2007).

Chapter 6 presents a distrust seed set propagation (DSP) algorithm to enhance existing Web spam detection algorithms. The distrust seed set propagation algorithm calculates the likelihood of other Web pages of becoming spam based on some untrustworthy seeds. Three Web spam detection algorithms that are experimented in Chapter 5 are attached with DSP to compare with the original. The results show that DSP enhanced the baseline algorithms and detected 17.73% more spam hosts in WEBSPAM-UK2006 and detected 8.59% more spam hosts in WEBSPAM-UK2007.

Chapter 7 proposes the application of machine learning technique to do Web spam detection. In this chapter, the structure for machine learning model is focused. C4.5 decision tree (Quinlan 1993) and support vector machines (Chang and Lin 2011) are two well-known machine learning models used in Web spam detection. Some researchers (Yuchun et al. 2008; Abernethy, Chapelle, and Castillo 2010; Zhiyang et al. 2012) have shown support vector machines outperforms decision trees. However, support vector machines have its own demerits (Biggio, Nelson, and Laskov 2012). Therefore, multilayered perceptrons neural network (Haykin 1998) is proposed for Web spam classification due to its flexible structure and non-linearity transformation to accommodate latest Web spam patterns. The experimental results have shown that multilayered perceptrons neural network has better web spam detection rate than support vector machines despite having the same features.

Finally in Chapter 8, the results of all chapters are summarized and concluded with a couple of future directions.

Chapter 2 Preliminaries

In this chapter, a foundation on the Web graph mathematical model is presented as it is used for all solutions in the rest of the chapters. Nevertheless, two standard Web spam datasets and their provided features are also presented. Finally, the parameters settings for all algorithms and performance evaluation are introduced at the end of this chapter.

2.1 WEB MODEL

Let a graph $G = (\upsilon, \varepsilon)$ where υ is a set of vertices and ε is a set of edges. If two vertices p and q form an edge, denoted as (p,q), thus ε consist an ordered pairs (p,q) of vertices such that $(p,q) \in \varepsilon$. The in-degrees of υ is the number of edges towards υ and out-degrees of υ is the number of edges leaving υ. Therefore, the sum of in-degrees or out-degrees is equal to the number of edges as shown in Equation 2.1.

$$\sum_{(p,q)\in\varepsilon} \deg^{-}(\upsilon) = \sum_{(p,q)\in\varepsilon} \deg^{+}(\upsilon) = |\varepsilon| \tag{2.1}$$

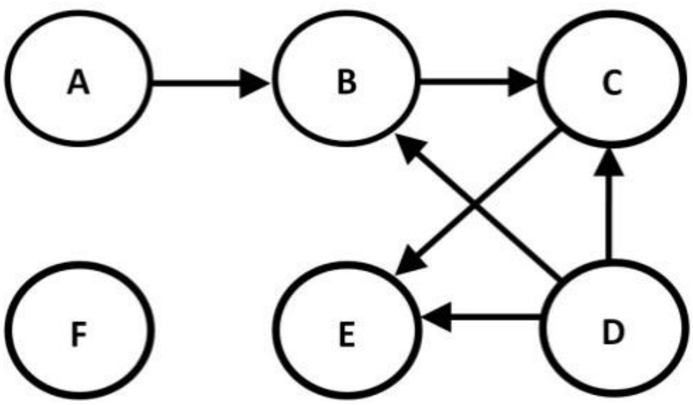


Figure 2.1: Simple Web graph

Consider a direct Web graph where $\upsilon = \{A, B, C, D, E, F\}$ showing in Figure 2.1, the in-degrees of υ is equal to the out-degrees of υ where:

$$\sum \deg^{-}(\upsilon) = \sum \deg^{+}(\upsilon) = 6$$

In a Web model, the vertices υ and the edges ε is denoted as Web pages and hyperlinks respectively. However, a Web graph can be decomposed into a host graph $G_H = (\upsilon_H, \varepsilon_H)$ where υ_H denote as a set of host vertices and ε_H denote as a set of ordered pair of hosts. A host consists of a set of Web pages under the same domain name. Assume that there are two host vertices υ_A and υ_B, υ_A and υ_B are connected such that $(\upsilon_A, \upsilon_B) \in \varepsilon_H$ if some pages under υ_A are pointing to some pages under υ_B where $\upsilon_A = \{\upsilon_{A1}, \upsilon_{A2}, \upsilon_{A3}, \ldots, \upsilon_{An}\}$, $n \in \Re$, $\forall n \geq 1$ and $\upsilon_B = \{\upsilon_{B1}, \upsilon_{B2}, \upsilon_{B3}, \ldots, \upsilon_{Bm}\}, m \in \Re, \forall m \geq 1$. Consider Figure 2.2 where host vertices $\upsilon_A = \{\upsilon_1, \upsilon_2, \upsilon_3\}$ and $\upsilon_B = \{\upsilon_4, \upsilon_5, \upsilon_6\}$, there exist direct edges $(\upsilon_1, \upsilon_4) \in \varepsilon$, $(\upsilon_1, \upsilon_6) \in \varepsilon$, $(\upsilon_2, \upsilon_4) \in \varepsilon$ and $(\upsilon_3, \upsilon_5) \in \varepsilon$ such that $(\upsilon_A, \upsilon_B) \in \varepsilon_H$, thus υ_A and υ_B are connected.

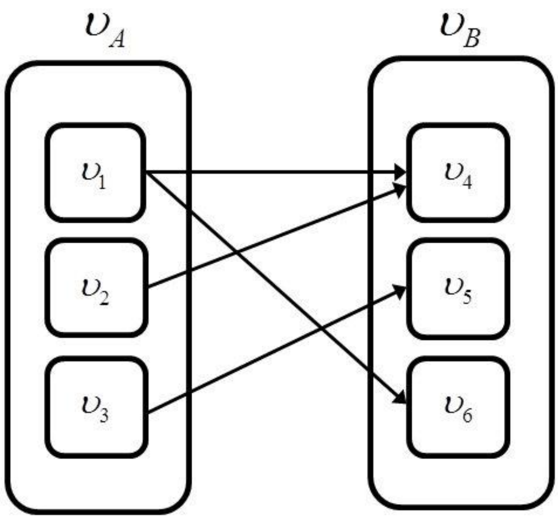


Figure 2.2: Sample host and page graph

The vertices υ can be partitioned into two categories where $\upsilon = \upsilon_E \cup \overline{\upsilon}_E$. υ_E denotes as a set of evaluated vertices while $\overline{\upsilon}_E$ denotes as an unevaluated vertex, such that $\overline{\upsilon}_E = \{\upsilon_X\}$ where υ_X stands for unknown vertices. Evaluated vertices υ_E can be assessed as non-spam vertices υ_N and spam vertices υ_S where $\upsilon_N, \upsilon_S \in \upsilon_E$ in which $\upsilon_N \cap \upsilon_S = \phi$ thus $\upsilon_E = |\upsilon_N \cup \upsilon_S| = |\upsilon_N| + |\upsilon_S|$.

A *weighted directed host graph* is a graph which each edge (a,b) has a weight function $\omega(a,b) : \upsilon_H \times \upsilon_H \to \Re_+$ with each weight is a real number. A weighted directed host graph can be represented as $G_H = (\upsilon_H, \varepsilon_H, \omega)$ where ω is the weight function of G_H .

Assume that there exists two element subset of υ_H such that $\{a,b\} \in \upsilon_H$, the weight function ω of host *a* to host *b* written as $\omega(a,b)$, is denoted as the sum of number of pages in host *a* direct to the pages in host *b* where $\upsilon_a = \{\upsilon_{a1}, \upsilon_{a2}, \upsilon_{a3}, ..., \upsilon_{an}\}$, $n \in \Re$, $\forall n \geq 1$ and $\upsilon_b = \{\upsilon_{b1}, \upsilon_{b2}, \upsilon_{b3}, ..., \upsilon_{bm}\}$, $m \in \Re$, $\forall m \geq 1$. In other words, $\omega(a,b)$ can be denoted as the sum of out-degree of host *a* direct to host *b* such that

$$\omega(a,b) = \sum_{(a,b)\in\varepsilon} \deg^+(a) \tag{2.2}$$

Figure 2.3 illustrates two host vertices υ_1 and υ_2 where υ_1 consists of page vertices $\{\upsilon_{11}, \upsilon_{12}, \upsilon_{13}\}$ while υ_2 consists of page vertices $\{\upsilon_{21}, \upsilon_{22}, \upsilon_{23}\}$.

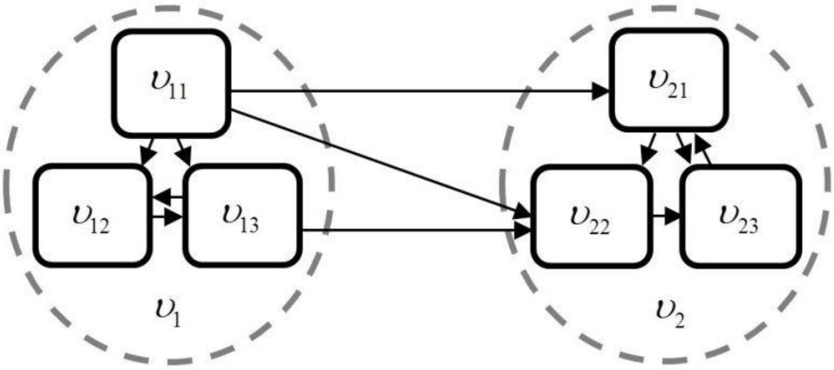


Figure 2.3: Sample of weighted graph

Assume $(\upsilon_1, \upsilon_2) \in \varepsilon$ where some pages in host vertex υ_1 are pointing to some pages in host vertex υ_2 and the weight function of υ_1 to υ_2 can be written as:

$$\omega(\upsilon_1, \upsilon_2) = \sum_{(\upsilon_1, \upsilon_2) \in \varepsilon} \deg^+(\upsilon_1) = 3$$

The graph which is representing the Web model can be transformed into matrix form as follows:

- Transition Matrix, T

$$T = \begin{cases} 1 & if\,(b,a) \in \varepsilon \\ 0 & otherwise \end{cases}$$

- Inverse Transition Matrix, I

$$I = \begin{cases} 1 & if\,(a,b) \in \varepsilon \\ 0 & otherwise \end{cases}$$

Details on adjacency-matrix representations can refer to APPENDIX B – Adjacency -Matrix Representation.

Consider Figure 2.2, the *Transition Matrix M* is written as

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The Inverse Transition Matrix N is written as

$$N = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2.2 DATASETS AND FEATURES

Two public available datasets are used throughout the whole thesis – WEBSPAM-UK2006 (Castillo et al. 2006) and WEBSPAM-UK2007 (Yahoo! 2007). Both datasets are downloaded from the Laboratory of Web Algorithmics, Università degli Studi di Milano, with the support of the DELIS EU - FET research project. The former dataset is also used in part of a Web Spam Challenge in 2007 (Castillo, Chellapilla, and Davison 2007; Castillo, Davison, et al. 2007) while the later dataset is used in Web Spam Challenge 2008 (Castillo, Chellapilla, and Denoyer 2008).

WEBSPAM-UK2006 consists of 77,741,046 Web pages while WEBSPAM-UK2007 consists of 105, 896,555 Web pages. Due to the large collection, host level is considered instead of page level. The former consists of 11,402 hosts whereas the later one consists of 114,529 hosts.

Both datasets provide evaluated sets, SET 1 for training and SET 2 for testing as the motivation behind the Web Spam Challenge Series is to provide solution to combat Web spam from machine learning perspective. For the link-based propagation algorithms, since no training and testing are required, both evaluated sets are sum to operate the experiments, as shown in Figure 2.4 and Table 2.1.

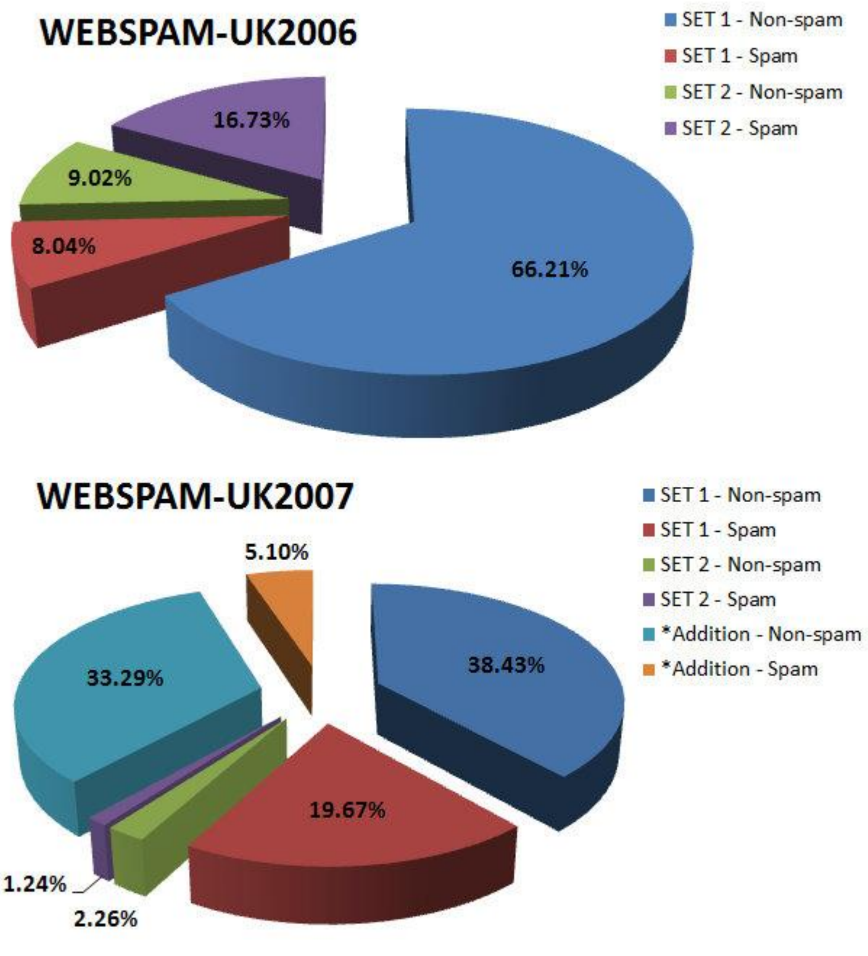


Figure 2.4: The distribution of WEBSPAM-UK2006 and WEBSPAM-UK2007 datasets

Table 2-1: Distributions of spam and non-spam in WEBSPAM-UK2006 and WEBSPAM-UK2007

	WEBSPAM-UK2006			***WEBSPAM-UK2007***			
	SET 1	***SET 2***	***TOTAL***	***SET 1***	***SET 2***	*****	***TOTAL***
Spam	674	1250	1924	222	122	157	501
Non-spam	4948	601	5549	3776	1933	3271	8980

*Additional Set from WEBSPAM-UK2006

Furthermore, the standard feature vectors as given in the Web Spam Challenge Series are used in the experiments. The features can be categorized into link-based and content-based features. Table 2-2 shows the types of features vector:

Table 2-2: Distributions of the feature vectors

Notation	***Feature Set***	***No. of Features***
A	Content-based Features	24
B	Full Content-based Features	96
C	Link-based Features	41
D	Transformed Link-based Features	138

Feature A denotes the content-based features. Most of these features are extracted from Ntoulas et al. (Ntoulas et al. 2006) and they comprise of the number of words in the page, number of words in the title, average word length, fraction of anchor text and visible text, compression rate, corpus precision and corpus recall, query precision and query recall, independent trigram likelihood, and entropy of trigrams. In total, there are 24 content-based features.

Feature B denotes the full content-based features. Since feature A are based on page feature, the authors (Castillo, Donato, et al. 2007) aggregate the content-based features for pages in order to obtain content-based features for hosts. Therefore, in total there are 96 content-based features (4 x feature A).

Feature C denotes the link-based features. Most are computed on the home page and also the page with the maximum PageRank in each host. The link-based features include degree-related measures like in-degree, out-degree, edge-reciprocity and assortativity coefficient. Besides this degree related features, PageRank, TrustRank, truncated PageRank and estimation of supporters are also included in this link-based features. In total there are 41 link-based features.

Feature D denotes the transformed link-based features. They are just simple numeric transformations and combinations of the link-based features. After transformation, there are 138 transformed link-based features.

Details on the standard feature vectors can be found in (Castillo, Donato, et al. 2007). More details on the link-based features can be found in (Becchetti et al. 2006b) while the content-based features can be found in (Ntoulas et al. 2006).

2.3 Parameters Settings and Performance Evaluation

In this section, the parameters settings that are used throughout the thesis are discussed and so as the performance evaluation so that all algorithms are standardized.

For all propagation algorithms, the decay factor α is set as 0.85 for the reason that it has become a standard since the first paper is published (Brin and Page 1998). For the seed selection regardless of spam seeds or non-spam seeds, 50 seeds are used for WEBSPAM-UK2006 while 100 seeds are used for WEBSPAM-UK2007 since there are less spam hosts in later datasets. Lastly, all the algorithms throughout the thesis execute in 50 iterations as this iteration is more than enough for the algorithms to reach convergence.

For the performance evaluation, there are three sections – trust propagation, distrust propagation and machine learning approach:

Trust propagation

- Number of non-spam hosts in each bucket
- Incremental summation of reputable hosts for all buckets
- Average promotion level for non-spam hosts (compare to benchmark)
- Number of non-spam hosts being promoted (compare to benchmark)
- Evaluated hosts represented in pages level
- Propagation coverage

Distrust propagation

- Number of spam hosts in each bucket
- Incremental summation of spam hosts for all buckets
- Average promotion level for spam hosts (compare to benchmark)
- Number of spam hosts being promoted (compare to benchmark)
- Evaluated hosts represented in pages level

For Machine Learning

- AUC (Area Under an Receiver Operating Characteristic Curve)

For trust propagation and distrust propagation, the acquired results from the derived algorithms will be sorted in descending order and divided into 10 or 20 buckets for performance evaluation. The number of non-spam hosts or spam hosts in each bucket indicates how much the algorithms have detected non-spam hosts or spam hosts. It is important to see more non-spam hosts in Web spam demotion algorithms and more spam hosts in Web spam detection algorithms as it shows the effectiveness of the algorithms. The second evaluation is the incremental summation of non-spam or spam hosts from the first to the last bucket. This evaluation shows how much the proposed algorithms have improved over all buckets. Next is the average promotion level for non-spam or spam hosts. It is used to track the movement of the particular non-spam or spam host from one bucket to the other. Let $P_O(S_i)$ be the bucket position for the non-spam or spam hosts of the benchmark algorithm and $P_m(S_i)$ be the bucket position for the non-spam or spam hosts of the proposed algorithm. For each bucket, let S_i be the labelled non-spam or spam hosts of the benchmark algorithms at the i^{th} bucket, the average promotion at i^{th} bucket, R_i can be defined as:

$$R_i = \frac{P_O(S_i) - P_\omega(S_i)}{\|S_i\|} \tag{2.3}$$

This evaluation metric tracks the improvements for each bucket over the baseline algorithms. The derived unit from the metric is called bucket per level. Moreover, the number of non-spam or spam hosts being promoted is shown, this evaluation is correlated with the previous measurement.

Throughout all experiments in this thesis, the datasets are conducted at the host level for the reason that assumed that if one host is a spam host, most likely the pages under this host are all spams. For the next experiment, the number of pages represented from the evaluated hosts is also presented. By achieving this, the number of spam and non-spam has been promoted or demoted at the page level are presented while preserving the computation on a host level. The last measurements for the trust propagation algorithms is the propagation coverage of the algorithms, this evaluation illustrates how much trust have reached other hosts, denoted as Sn and the percentage of trust propagated to evaluated hosts, denoted as η, this measurement has been introduced and used by some researchers (Zhang et al. 2009).

For machine learning approach, the area under the receiver operating characteristic curve, also known as AUC is emphasize and is used to evaluate the Web spam detection performance because it does not depend on any threshold (Erdélyi, Garzó, and Benczúr 2011) like precision and recall, and it aims at measuring the performance of the prediction of spamicity (Castillo, Chellapilla, and Denoyer 2008).

Chapter 3 Anti-Web Spam Techniques

3.1 Introduction

Various anti-Web spam techniques are constantly proposed to fight against Web spam. Among the techniques, trust and distrust model and machine learning model have shown significant results against Web spam. A comprehensive literature survey is provided on these models in this chapter.

Firstly, a well-known trust based anti-Web spam algorithm – TrustRank (Gyöngyi, Garcia-Molina, and Pedersen 2004) is presented. However, there are few weaknesses on TrustRank, thus researchers (Krishnan and Raj 2006; Wu, Goel, and Davison 2006b; Yang, King, and Lyu 2007; Qi, Song-Nian, and Sisi 2008) come up with the derivatives for TrustRank which will be thoroughly explained. An experimental study is done on TrustRank and HostRank (Eiron, McCurley, and Tomlin 2004), and shows how vulnerable it is for spam to attack. Besides TrustRank and its derivatives, there are other trust and distrust model algorithms and these algorithms are briefly explained in this section. A table on trust and distrust model is provided for comparison. Subsequently, machine learning techniques in terms of features and structure are discussed for Web spam detection.

3.2 TrustRank and its Derivatives

In this section, TrustRank and its derivatives which include Anti-TrustRank, Topical TrustRank, DiffusionRank and Link Variable TrustRank are presented.

3.2.1 TrustRank

Yang et al. (Yang, King, and Lyu 2007) mentioned that TrustRank has a strong theoretical relation with PageRank (Brinkmeier 2006). The algorithm semi-automatically separate reputable good pages from spam, and trust flows from the link structure of the good pages to identify additional good pages. The intuition behind TrustRank is that good pages seldom point to bad pages.

TrustRank starts by selecting seeds. Seed selection is done by applying inverse PageRank to the dataset in order to get pages that would be most useful to identify additional pages. The results are then ranked in descending order and choose the good pages from top L pages as good seed set because trust flows only from good seed set. TrustRank then normalizes the distribution vector and applies measurement using Equation 3.1, similar to PageRank with some minor changes:

$$TR = \alpha \cdot T \cdot TR + (1-\alpha) \cdot \tau \tag{3.1}$$

For the Equation 3.1, α is the decay factor, usually sets 0.85, T is the transition matrix, while τ is the distribution vector after normalization. As similar to PageRank, this is an iterative algorithm and calculated in M iterations.

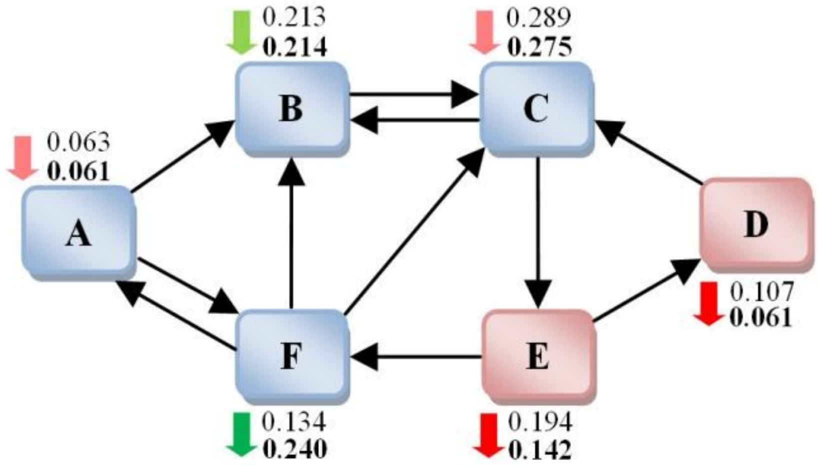


Figure 3.1: Simple Web graph with PageRank and TrustRank results

Assuming α decay factor is 0.85 running in $M=50$ iterations and set L = 3 with $s^+ = \{F\}$ and $s^- = \{D,E\}$; Figure 3.1 illustrates the results from both PageRank (upper with non-bold) and TrustRank (lower with bold). Good page *F* propagates trust to page *A*, *B* and *C* and therefore the pages are having high PageRank values while page *D* and *E* having low PageRank values. Page *F* is promoted since it is a good page while page *D* and *E* are punished for being a bad page.

3.2.2 Derivatives of TrustRank

In this section, the derivatives of TrustRank such as Anti-TrustRank (Krishnan and Raj 2006), Topical TrustRank (Wu, Goel, and Davison 2006b), DiffusionRank(Yang, King, and Lyu 2007) and Link Variable TrustRank(Qi, Song-Nian, and Sisi 2008) are presented.

3.2.2.1 Anti-TrustRank

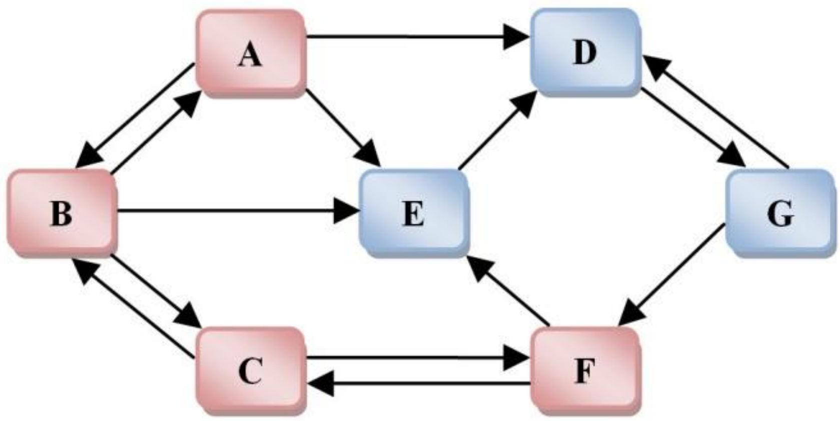


Figure 3.2: Simple Web graph with good pages (blue) and bad pages (red)

Anti-TrustRank algorithm (Krishnan and Raj 2006) uses the same approximate isolation principle used by the TrustRank algorithm but Anti-Trust is propagated in the reverse direction along incoming links from a seed set of spam pages. A page is categorized as spam page if the Anti-TrustRank score of the page is more than a given threshold value. For example in Figure 3.2, assuming page *A* is a spam seed set, Anti-TrustRank would propagate to page *B*, and page *B* would propagate to page *C* and to page *F* if these pages are more than the threshold value.

Firstly, Anti-TrustRank evaluates the dataset with PageRank algorithm and selects spam pages seed set with high PageRank; Spam pages with high PageRank are most likely to be pointed by another spam pages with high PageRank. By achieving this, Anti-TrustRank able to detect another spam pages with high PageRank. After that, Anti-TrustRank runs the biased PageRank algorithm on the transpose matrix which represents the Web graph with the spam seed set. Finally, pages are ranked in descending order by their PageRank score to estimate the spam content. Pages with score greater than the threshold value given are marked as spam.

Anti-TrustRank is able to report that the pages from which its seed set can be reached in short paths are untrustworthy. Also, the authors found that the average spam pages rank calculated by Anti-TrustRank is higher than the average spam pages rank calculated by TrustRank. In summary, Anti-TrustRank has the added benefit of returning spam pages with high precision. The intuition behind is that by starting with seed spam pages of high PageRank, it would expected that walking backward would lead to a good number of spam pages of high PageRank.

Anti-TrustRank algorithm is written as:

$$ATR(p) = \alpha \cdot \sum_{(p,q)\in\varepsilon} \left(\frac{ATR(q)}{\deg^{-}(q)} \right) + (1-\alpha)\cdot B(p) \qquad (3.2)$$

Where *ATR* represent Anti-TrustRank, α is a decay factor, $\deg^{-}(q)$ is the number of incoming links of host *q* and *B(p)* is the spam vector.

3.2.2.2 Topical TrustRank

Selecting seed function in TrustRank algorithm has a bias towards communities. The Web consists of large repositories from different kinds of topic. In addition to this, the seed set coverage used by TrustRank does not cover every topic exist on the Web. To address these issues, inspired by Topic Sensitive PageRank (Haveliwala 2002), Wu et al. (Wu, Goel, and Davison 2006b) proposed Topical TrustRank which uses topical information to partition the seed set and calculate the trust score for each topic separately.

Given a seed set, Topical TrustRank divides the seed set into different partitions corresponding to the topics as given in Equation 3.3:

$$\left(\sum_{i=1}^{n} m_i\right) \times TR = \sum_{i=1}^{n} (m_i \times TR_i) \qquad (3.3)$$

The equation is a version of the Linearity theorem proved by Jeh and Widom (Jeh and Widom 2003). Assume seed set T is given, it can be partitioned into n subsets, $T_1, T_2, \ldots, T_3$ where each containing $m_i \cdot (1 \leq i \leq n)$ seeds. TR represents the TrustRank scores calculated by using T as the seed set and $t_i \cdot (1 \leq i \leq n)$ represents the TrustRank scores calculated by using T_i as the seed set. It shows that product of TrustRank score and the total number of seeds equals the sum of products of the individual partition-specific scores and the number of seeds in that partition. The transformation of the equation is:

$$TR = \sum_{j=1}^{n} \frac{m_n}{\sum_{i=1}^{n} m_i} \cdot TR_n \qquad (3.4)$$

The authors introduced two techniques, which are called simple summation and quality bias to combine the generated topical trust score so as to present a single measure of trust for a page. Simple summation is calculated by adding up all trust scores by topic and applies on TrustRank, and then the Topical TrustRank score is generated. In the other hand, quality bias takes the average PageRank value of the seed pages of particular community into consideration.

The authors also proposed three seed selection improvements for Topical TrustRank algorithm. The improvements are seed weighting, seed filtering and finer topic hierarchy. In seed weighting, each node is assigned a constant value proportional to its quality; another way of saying is that some seed pages' trust is generally higher than some other seed pages. In seed filtering, the quality of a page can be measured using PageRank or Topical TrustRank scores, low quality seed pages can be filtered out to improve the performance of the Topical TrustRank as low quality pages might include spam pages. For finer topic hierarchy, topic directories usually provide a tree structure for each topic and calculation is expensive to involve finer topics. However, finer topic hierarchy would be ideal to categories the Web.

There is a trade-off for using simple summation. For that reason, the authors experiment using quality bias and the combination of seed weighting, seed filtering and finer topic hierarchy. The topical TrustRank results provided a reduction of 19% – 43.1% in spam sites compare to TrustRank.

3.2.2.3 DiffusionRank

Motivated by the viewpoint of the Web structure and heat diffusion phenomena, Yang et al. (Yang, King, and Lyu 2007) proposed DiffusionRank, a generalization of PageRank which additionally has the ability to reduce the effect of link manipulations. Heat diffusion is a physical phenomenon in which heat always flow from high temperature position to low temperature position.

The authors explained two points where PageRank is susceptible to Web spam. The two points are over-democratic and input-independent. The belief behind PageRank is that all pages are born equal; all pages have the right to vote in a summation of one for each page. Over-democratic can be explained when a large number of new pages are pointing to a page, since all new pages have the right to vote. For input-independent, PageRank is an iterative algorithm which it calculates until a point where it converged. Input-independent property makes it impossible to set an input to avoid Web spam, like large values for trusted pages and less or even negative value for spam sites. The heat diffusion model has an advantage to avoid over-democratic and input-independent of PageRank. Therefore, the authors proposed DiffusionRank to view the Web from another perspective and calculate the ranking values.

The DiffusionRank equation is defined in Equation 3.5:

$$h = \left(1 - \frac{\gamma}{M}\right)h + \frac{\gamma}{M}(\alpha \cdot T \cdot h + (1-\alpha) \cdot \frac{1}{N} \cdot 1) \tag{3.5}$$

Where h is a diffusion score vector, M is the number of iteration, γ is thermal conductivity coefficient, N is the number of elements where the elements refer to Web vertices and T is the transition matrix.

There are four advantages for DiffusionRank: two closed forms, group-group relations, graph cut and anti-manipulation. The two closed forms include discrete form and continuous form, the primary one has the advantage of fast computing while the secondary one has the advantage of being analysed easily from theoretical aspects. DiffusionRank is able to detect group-to-group relations easily because of the easy interpretation of the heat amount from one group to another. Another advantage is that it can partition the Web graph corresponding to the community by assigning positive and negative values among the communities. Lastly, DiffusionRank has the ability to reduce the effect of link manipulation as trusted Web pages are assigned with unit heat while all others are assigned with zero heat. The authors claimed manipulated Web pages will get lower rank until it is pointed by several good pages.

3.2.2.4 Link Variable TrustRank

Chen et al. (Qi, Song-Nian, and Sisi 2008) proposed Link Variable TrustRank algorithm (also known as LVTrustRank) based on the idea of using “bursts” of linking activity as a suspicious signal (Shen et al. 2006) with the combination of the original

TrustRank algorithm mentioned earlier. When there is a drastic change in the link structure of a spam site in a short period of time, LV TrustRank uses this opportunity to measure trust from the variance of the link structure and detect spam sites.

Spammers intend to add links to pages which the intention of promoting particular page, Shen et al. (Shen et al. 2006) introduced in-link growth rate (IGR) to measure the ratio of the increased number of incoming links of a site to the number of original incoming links. The metrics is defined in Equation 3.6:

$$IGR = \frac{|S_{in}(\mu_1)| - |S_{in}(\mu_0) \cap S_{in}(\mu_1)|}{|S_{in}(\mu_0)|} \tag{3.6}$$

μ_0 and μ_1 are two different timeline where $S_{in}(\mu_0)$ is the set of in links of a site at time μ_0 and $S_{in}(\mu_1)$ is the set of in links of a site at time μ_1. IGR is a good indicator to represent the variance of spam sites in link structure.

LVTrustRank computes the TrustRank score $TR(\mu_1)$ and $TR(\mu_2)$ at different timeline and uses IGR to get the ratio of the variance of link structure. A joint formula to compute the final trust score for the timelines is defined in Equation 3.7:

$$TR(\mu_f) = \left(\frac{TR(\mu_1) + TR(\mu_2)}{2} \right)^{1+IGR} \tag{3.7}$$

LVTrustRank performs well on detecting Web spam based on the variance of the link structure. However, there exist some spam sites that do not change their link structure and it is not possible for LVTrustRank to detect. Nevertheless, Shen et al. (Shen et al. 2006) introduced the idea of using variance of link structure to detect spam can be explored further.

3.2.3 Experiments

The experiments are conducted on two datasets – WEBSPAM-UK2006 and WEBSPAM-UK2007. Firstly, HostRank and TrustRank are compared and experimented to see the vulnerability of HostRank towards Web spam. After that, different TrustRank algorithms with various seeds (see Chapter 2 Preliminaries for parameters settings) are discussed. In WEBSPAM-UK2006, the TrustRank algorithms with 50, 75 and 100 seeds are experimented while 100, 150 and 200 seeds are experimented in WEBSPAM-UK2007.

Figure 3.3 and 3.5 show the comparison of HostRank and TrustRank on the ratio of non-spam sites and spam sites for each bucket in WEBSPAM-UK2006 and WEBSPAM-UK2007. The dark blue bar denotes the non-spam sites of HostRank while the light blue bar denotes the non-spam sites of TrustRank. The empty spaces above the bars represent the spam sites in each individual buckets.

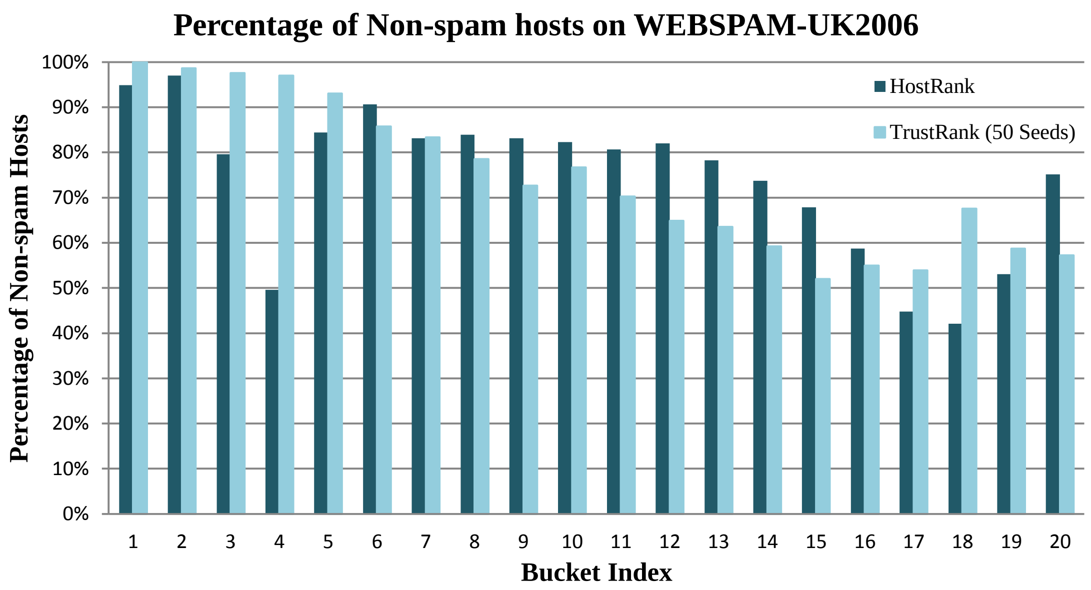


Figure 3.3: Percentage of non-spam hosts in HostRank and TrustRank (50 Seeds) buckets on WEBSPAM-UK2006.

Figure 3.4 and Figure 3.6 show the percentages of non-spam hosts in TrustRank buckets on WEBSPAM-UK2006 and WEBSPAM-UK2007.

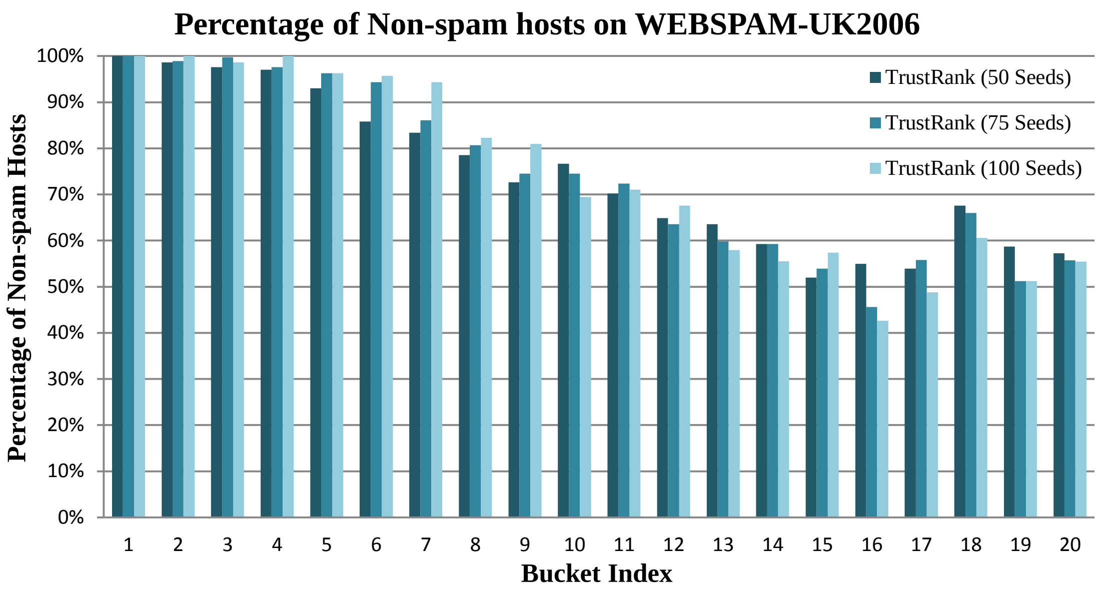


Figure 3.4: Percentage of non-spam hosts in TrustRank (50, 75 and 100 Seeds) buckets on WEBSPAM-UK2006.

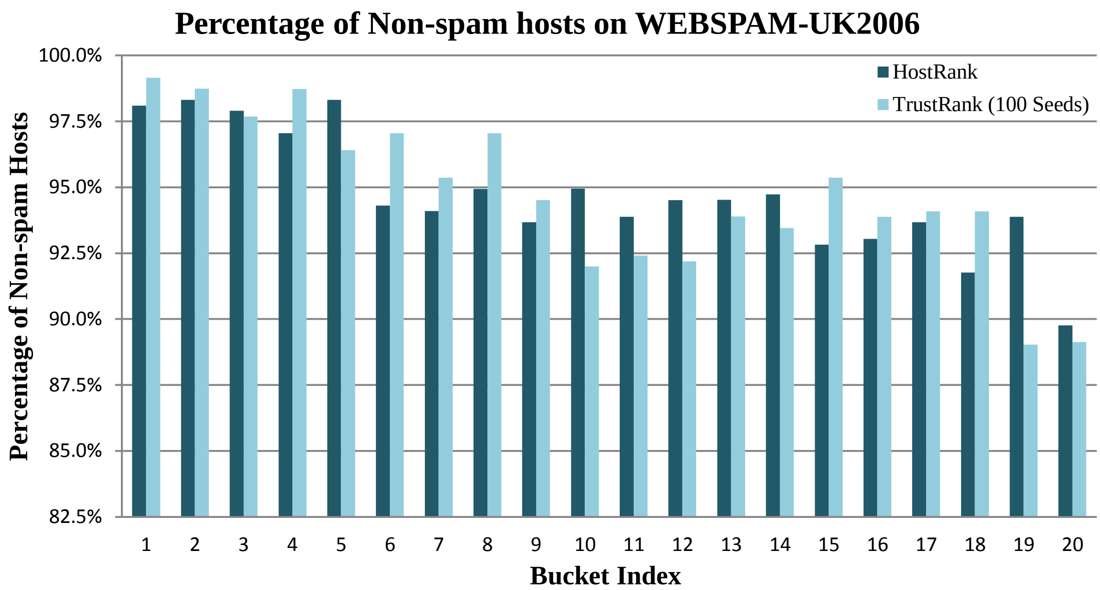


Figure 3.5: Percentage of non-spam hosts in HostRank and TrustRank (100 Seeds) buckets on WEBSPAM-UK2007.

In Figure 3.3, TrustRank (50 seeds) able to achieve more than 90% of non-spam hosts in top 5 buckets whereas in HostRank, the 4th bucket alone already consists more than 50% of spam hosts. In Figure 3.5, TrustRank outperforms HostRank by having less spam hosts as much as 7 buckets in top 9 buckets. It is important to return results in early buckets because it shows the most trustworthy and relevant results.

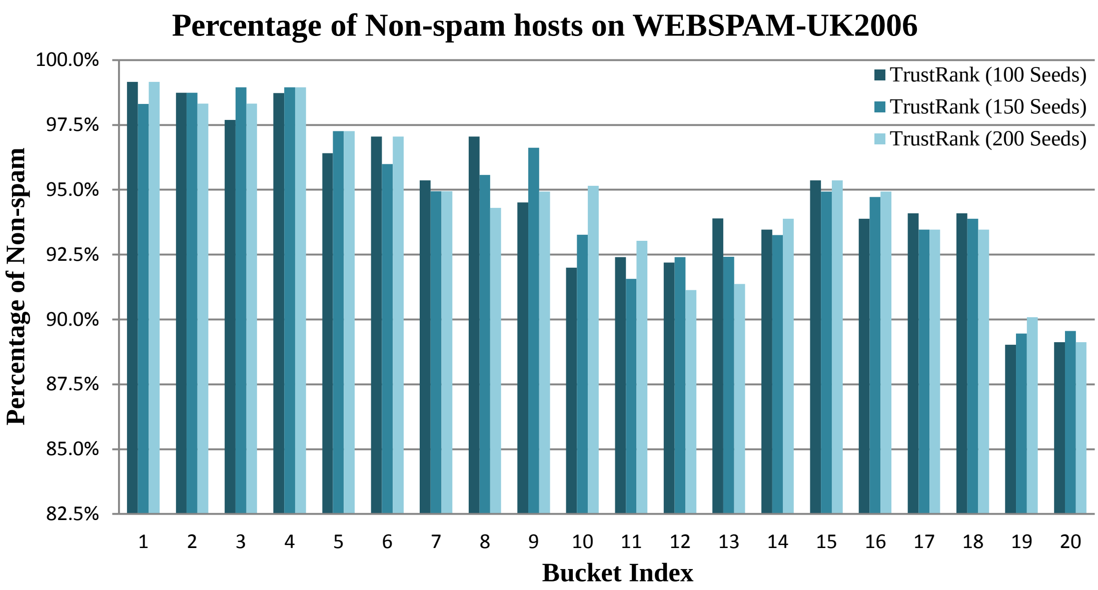


Figure 3.6: Percentage of non-spam hosts in TrustRank (100, 150 and 200 Seeds) buckets on WEBSPAM-UK2007.

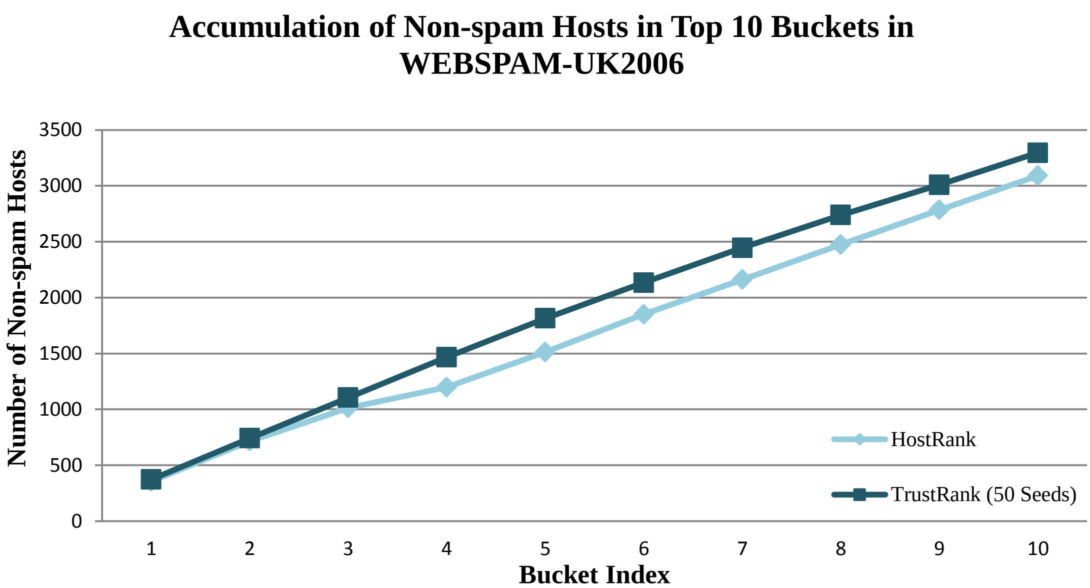


Figure 3.7: Accumulation of non-spam hosts on top 10 buckets for HostRank and TrustRank (50 Seeds) in WEBSPAM-UK2006

Figure 3.7 illustrates the accumulation of non-spam hosts on top 10 buckets for HostRank and TrustRank (50 Seeds) in WEBSPAM-UK2006. At the end of the 10th bucket, HostRank able to reach 3091 non-spam hosts whereas TrustRank with only 50 seeds able to reach 3295 non-spam hosts. Furthermore, the figure also shows that

TrustRank able to detect more trustworthy hosts as early as possible compare to HostRank.

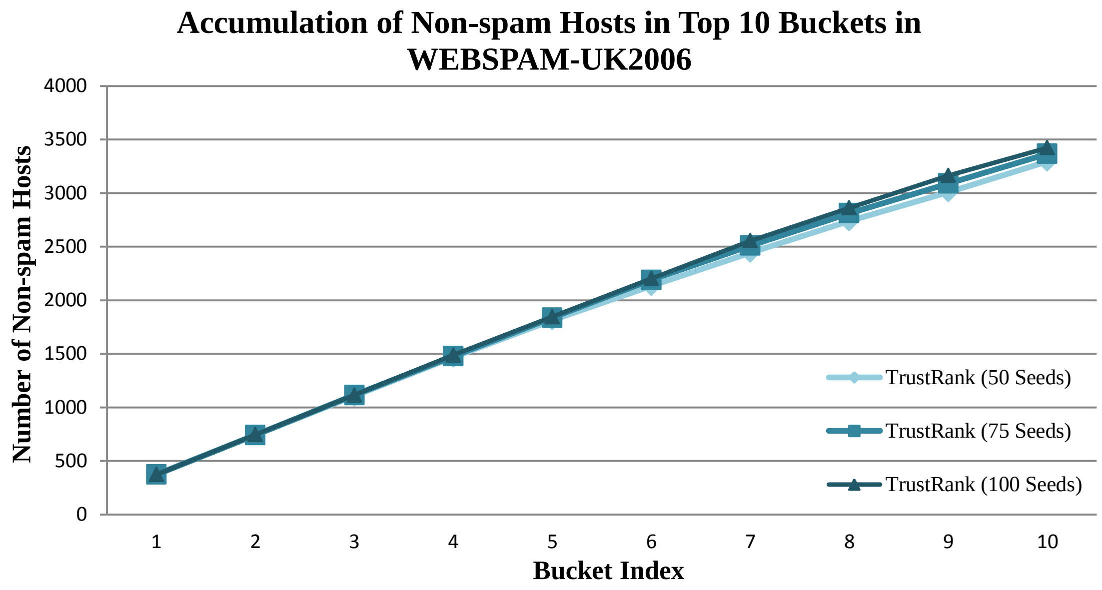


Figure 3.8: Accumulation of non-spam hosts on Top 10 buckets for TrustRank (50, 75 and 100 Seeds) in WEBSPAM-UK2006

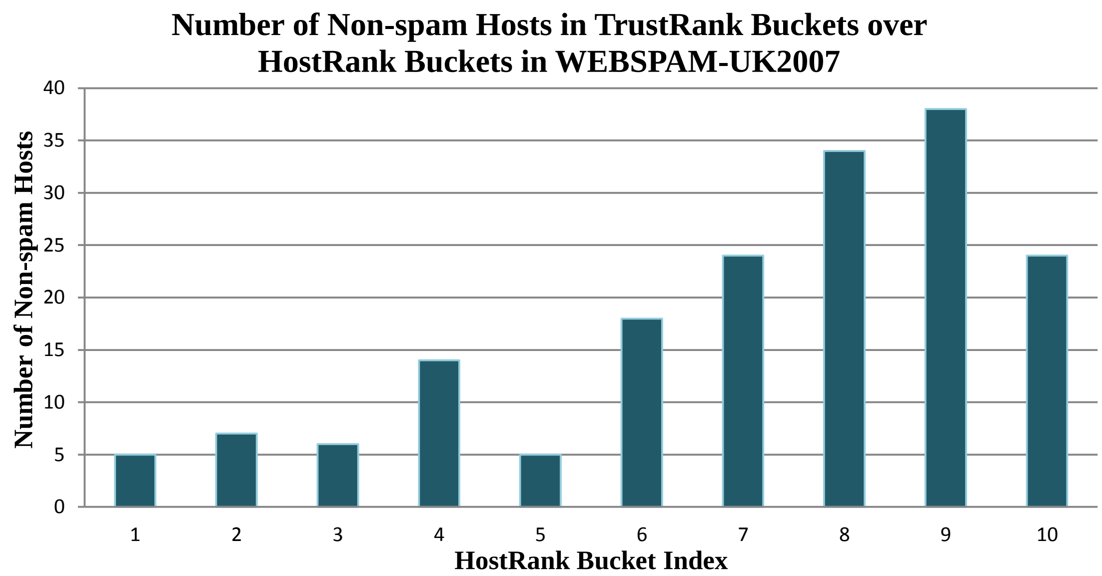


Figure 3.9: Number of non-spam hosts in TrustRank (100 Seeds) buckets over HostRank buckets in WEBSPAM-UK2007

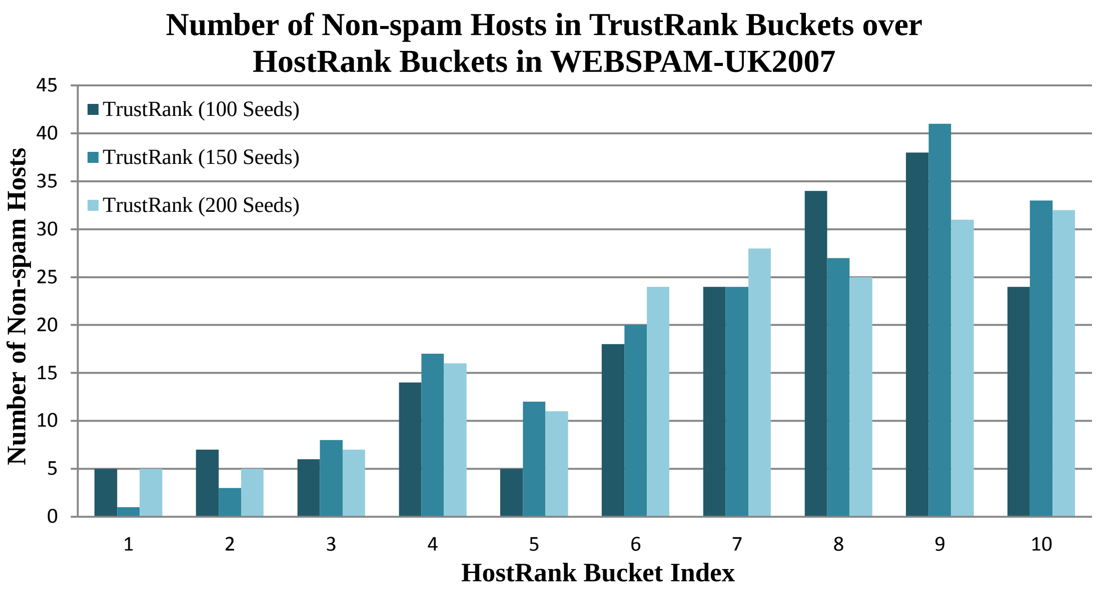


Figure 3.10: Number of non-spam hosts in TrustRank (100, 150 and 200 Seeds) buckets over HostRank buckets in WEBSPAM-UK2007

Figure 3.8 illustrates the accumulation of non-spam hosts on top 10 buckets on three TrustRank algorithms with different number of seeds (50, 75 and 100 Seeds) in WEBSPAM-UK2006 dataset. At the end of the 10^{th} bucket, TrustRank with 100 seeds has the highest accumulative sum of non-spam hosts with an amount of 3423 non-spam hosts. Second is TrustRank with 75 seeds with an amount of 3367 non-spam hosts and finally, TrustRank with 50 seeds with an amount of 3295 non-spam hosts.

Figure 3.9 depicts the number of non-spam hosts in TrustRank (100 seeds) buckets over HostRank buckets in WEBSPAM-UK2007. As shown in the figure, all TrustRank (100 seeds) buckets have more trustworthy hosts than the HostRank algorithm. In addition, the 9^{th} bucket has the highest amount as much as 38 of non-spam hosts.

Figure 3.10 further illustrates the number of non-spam hosts in three TrustRank algorithms (100, 150 and 200 seeds) buckets over HostRank buckets in WEBSPAM-UK2007. TrustRank with 150 seeds has the highest score with the sum of 186 non-spam hosts more than HostRank algorithm compare to TrustRank 200 with 184 non-spam hosts. Figure 3.11 and 3.13 illustrate average promotion level for non-spam hosts in WEBSPAM-UK2006 and WEBSPAM-UK2007. The two figures indicate the improvement over HostRank buckets. On the other hand, figure 3.12 and 3.14 show the numbers of non-spam hosts are being promoted in HostRank buckets in WEBSPAM-UK2006 and WEBSPAM-UK2007.

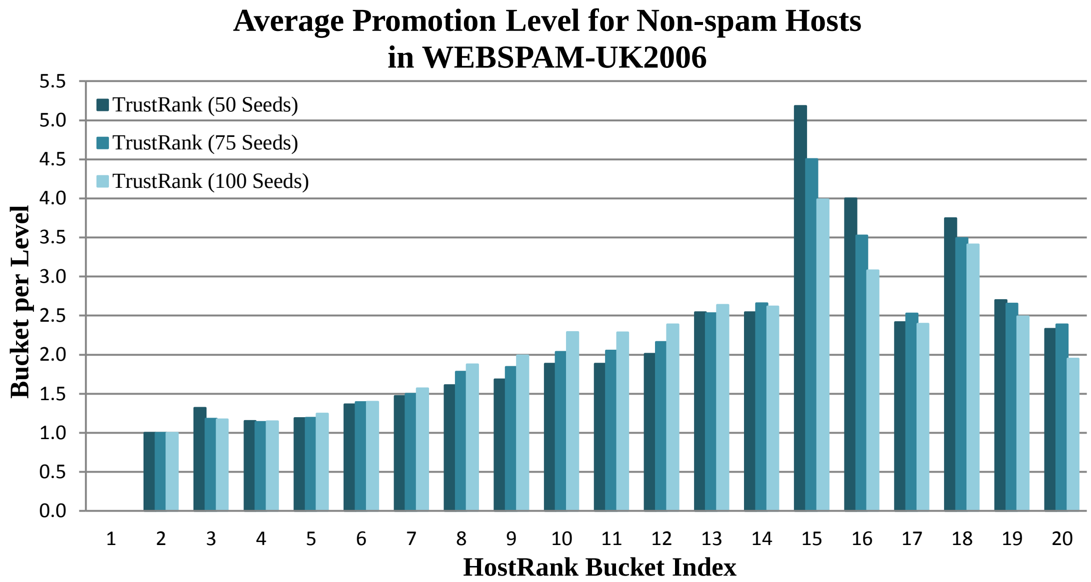


Figure 3.11: Average promotion level for non-spam hosts in WEBSPAM-UK2006 for TrustRank (50, 75 and 100 Seeds) buckets over HostRank buckets

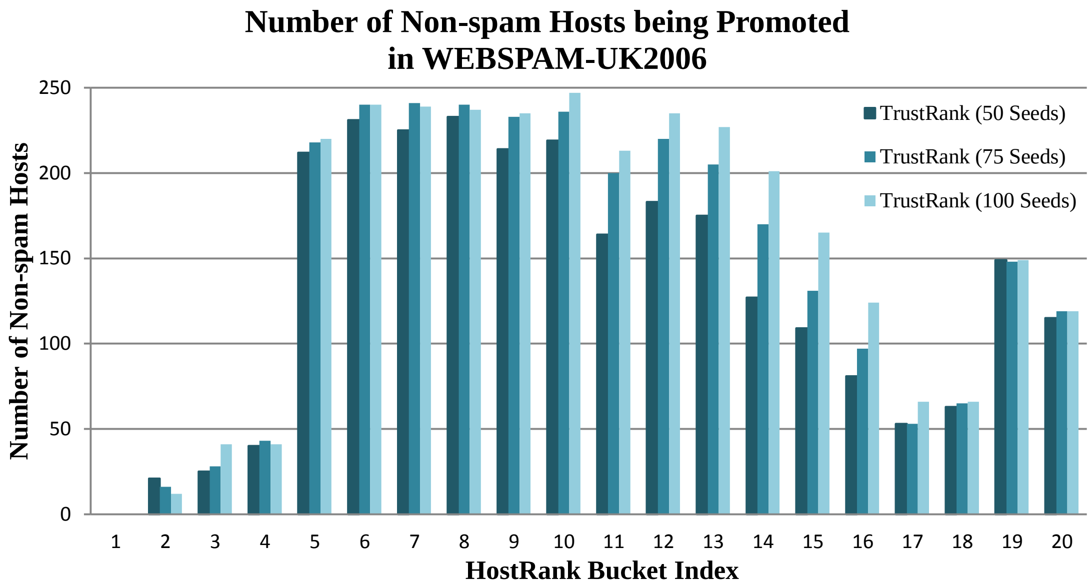


Figure 3.12: Number of non-spam hosts being promoted in HostRank buckets from TrustRank (50, 75 and 100 Seeds).

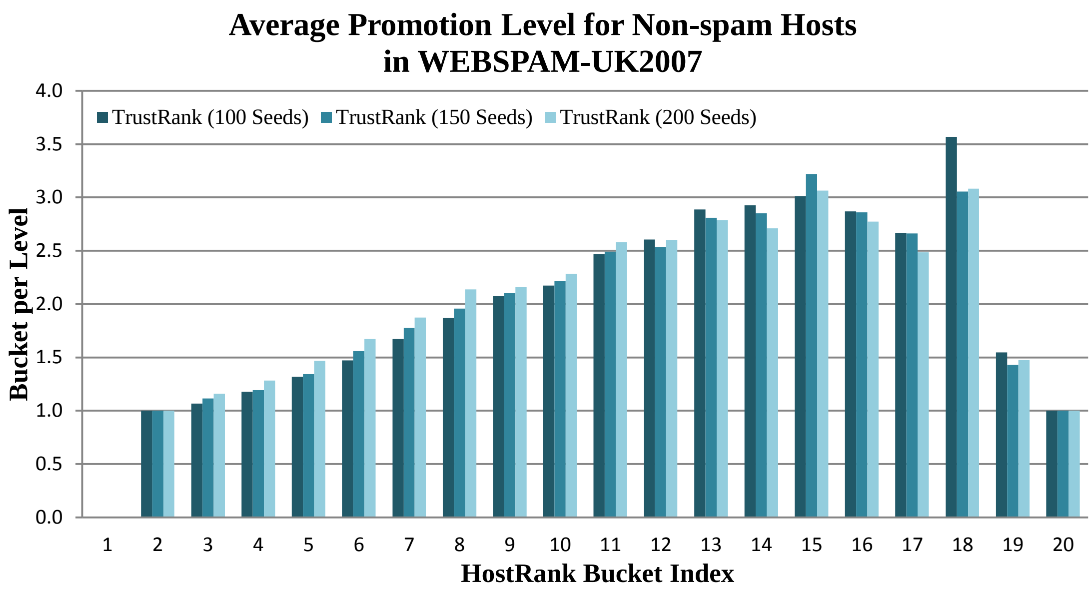


Figure 3.13: Average promotion level for non-spam hosts in WEBSPAM-UK2007 for TrustRank (100, 150 and 200 Seeds) buckets over HostRank buckets

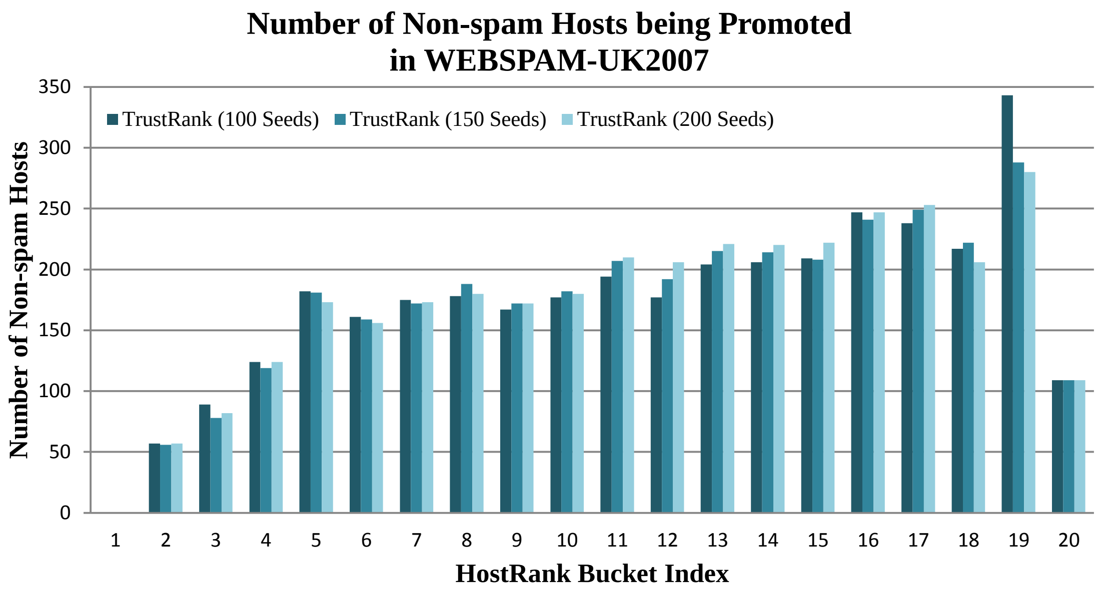


Figure 3.14: Number of non-spam hosts being promoted in HostRank buckets from TrustRank (100, 150 and 200 Seeds).

From the observation on figure 3.11 and 3.12, even though TrustRank (50 Seeds) has the highest average promotion level – 5.18 at 15th bucket, TrustRank (100 Seeds) has promoted the most number of non-spam hosts, a total of 3077, the second highest goes to TrustRank (75 Seeds) with 2903 non-spam hosts and third is TrustRank (50 Seeds) with 2639 non-spam hosts.

In Figure 3.13, TrustRank (100 seeds) has the highest average promotion of 3.57 non-spam host per level. However, TrustRank (200 seeds) has the highest number of promoted non-spam host, a total of 3471 non-spam hosts being promoted even though the highest individual bucket being promoted goes to TrustRank (100 seeds) on 19th bucket with a number of 343 non-spam hosts being promoted.

Table 3-1 and 3-2 show the number of non-spam Web pages represented from the non-spam hosts in WEBSPAM-UK2006 and WEBSPAM-UK2007. The non-spam hosts are based on the accumulation for top 5 buckets.

Table 3-1: Web pages promoted for all algorithms in WEBSPAM-UK2006

WEBSPAM-UK2006				
Bucket Index	***HostRank***	***TrustRank (50 Seeds)***	***TrustRank (75 Seeds)***	***TrustRank (100 Seeds)***
1	4633746	5772732	5911439	6045772
2	7878928	9405003	9538061	9339090
3	10707594	12453809	12656345	12648555
4	11860496	15018597	15006317	14940319
5	14333206	17326257	17267232	17165337

Table 3-2: Web pages promoted for all algorithms in WEBSPAM-UK2007

WEBSPAM-UK2007				
Bucket Index	***HostRank***	***TrustRank (100 Seeds)***	***TrustRank (150 Seeds)***	***TrustRank (200 Seeds)***
1	5433512	5762322	6048337	6221612
2	9514864	9798875	10029708	9981872

(Table 3-2 continued)

3	12113172	12601227	12790826	12721565
4	14570132	15149195	15107725	15151087
5	16374587	16952324	17088503	17095291

From both the table, it has clearly shown that all the TrustRank algorithms have detected more non-spam pages compare to HostRank algorithm. In Table 3-1, some TrustRank algorithm with more seeds might detect less non-spam pages than TrustRank with lesser seeds. This is due to more seeds might promote more spam pages too. Regardless of this, the TrustRank algorithms still outperforms the HostRank algorithm.

Table 3-3: Propagation coverage in WEBSPAM-UK2006 and WEBSPAM-UK2007

Datasets	***Algorithms***	$Sn(\upsilon_E)$	$Sn(\upsilon_N)$	$Sn(\upsilon_S)$	η_N	η_S
WEBSPAM-UK2006	TrustRank (50 Seeds)	8564	4223	1374	86.20	13.80
	TrustRank (75 Seeds)	8766	4388	1381	90.01	9.99
	TrustRank (100 Seeds)	8922	4519	1384	92.84	7.16
WEBSPAM-UK2007	TrustRank (100 Seeds)	73790	6603	242	98.98	1.02
	TrustRank (150 Seeds)	75629	6780	249	99.15	0.85
	TrustRank (200 Seeds)	76759	6858	260	99.26	0.74

Table 3-3 shows the propagation coverage from the TrustRank algorithms in WEBSPAM-UK2006 and WEBSPAM-UK2007. Observed from the table, the more seeds TrustRank has, the more non-spam and spam hosts able to reach. Even though more spam hosts are reached if more seeds are used, the trust propagation propagates to non-spam hosts is still more than spam hosts – in WEBSPAM-UK2006, TrustRank (100 Seeds) propagates 92.84% trust to non-spam hosts and 7.16% to spam hosts whereas in WEBSPAM-UK2007, TrustRank (200 Seeds) propagates 99.26% trust to non-spam hosts and 0.74% to spam hosts. The full results of all experiments in this chapter can refer to APPENDIX C - Chapter 3 Results.

3.3 OTHER TRUST AND DISTRUST MODEL ALGORITHMS

The very first algorithm to detect Web spam is the BadRank (Sobek 2002) algorithm. Based on the given spam seed set, distrust is propagated to measure the negative characteristics of one's page and its principle is to link to bad neighbours. The formula of the algorithm is written as:

$$BR(p) = \alpha \cdot \sum_{p:(p:q)\in\varepsilon} \frac{BR(q)}{\deg^{-}(q)} + (1-\alpha)\cdot\kappa(p) \tag{3.8}$$

Where *BR(p)* stands for BadRank of page *p*, $(p:q)\in\varepsilon$ denotes there is a direct link from page *p* to page *q*. *j* is the jump probability and $\deg^{-}(q)$ is the number of incoming links of page *q*. According to (Sobek 2002), $\kappa(p)$ is a special evaluation on page *p* which reflected whether this page is detected by a spam filter.

Wu et al. (Wu and Davison 2005b) introduced a technique to identify link farm spam pages, this technique consists of three steps: Generating step, expansion step and ranking step. At first the algorithm generates a spam seed set by its common incoming and outgoing links. Then the authors use ParentPenalty to expand the seed set, the assumption is that if one page points to a bunch of bad pages, it is likely that the page is a bad page. Lastly, the authors rank the Web graph by down weighting the elements in the adjacency matrix.

Gyongyi et al. (Gyöngyi et al. 2006) introduce the concept of spam mass to measure the impact of link spamming on PageRank (Brinkmeier 2006). Spam mass can identify pages that benefit from link spamming. Those pages which benefit from link spamming is bias towards search engines, identifying them can help search engines remove them as early as possible.

Wu et al. (Wu, Goel, and Davison 2006a) proposed two algorithms – one based one trust model and another based on distrust model. After calculating two algorithms individually, the authors totalled up the score to detect Web spam and experimented on three splitting methods: equal splitting, constant splitting and logarithm splitting, and three accumulation steps: simple summation, maximum share and maximum parent. Maximum share and logarithm splitting for trust and distrust model is concluded to be able to achieve the best results.

Nie et al. (Nie, Wu, and Davison 2007) did similar research with Wu et al. (Wu, Goel, and Davison 2006a) which proposed one trust and one distrust model algorithms. Two splitting methods and two accumulation steps are experimented: equal splitting and constant splitting, and simple summation and maximum share. The difference between their researches is that Wu et al. algorithms have a constant value which can be adjust to detect the most spam or demote the most spam whereas Nie et al. algorithms have a weighting value at the summation of trust and distrust algorithms. Nie et al. concluded that simple summation with constant splitting for trust and maximum share with constant splitting for distrust have the best performance.

Liang et al. (Liang, Ru, and Zhu 2007) proposed R-SpamRank, which stands for reverse spam rank, which initially uses blacklist as spam Web pages as seeds, then expand it by applying a formula similar to BadRank. The authors claimed that the algorithm ideally detect spam pages in a link farm.

Zhao et al. (Li, Qiancheng, and Yan 2008) proposed QoC-QoL algorithm to select bad seeds based on good seeds to combat Web spam. The authors concluded that using large good seeds with bad seeds is the best choice.

Zhang et al. (Zhang et al. 2009) explore the bidirectional links and proposed two page value metrics, AVRank and HVRank to detect spam easier. AVRank and HVRank are inspired by TrustRank and HITS algorithm also to expand the seed set trust propagation. The authors also proved that automatically identified large seed set works better than human manual identified seed set.

Lastly, Trust-Distrust Rank (Zhang, Wang, et al. 2011) proposed by Zhang et al. make good use of good seeds and bad seeds and also overcomes the disadvantages of both existing trust and distrust propagation algorithms which is either trust or distrust is propagating in a non-differential way.

A comparative study on all link-based trust and distrust model algorithms is listed as:

Table 3-4: List of link-based trust and distrust algorithms.

Algorithms	***Year***	***Good Seed Set***	***Bad Seed Set***	***Results***	
				Datasets	***Achieve***
BadRank (Sobek 2002)	2002		✓	-	-
TrustRank (Gyöngyi, Garcia-Molina, and Pedersen 2004)	2004	✓		AltaVista Aug 03 (31,003,946 sites)	1000 sample sites; with 178 good seeds, precision is 0.86 and recall is 0.55 for top 10 buckets
ParentPenalty (Wu and Davison 2005b)	2005		✓	search.ch (≈350,000 sites)	27,568 sites were expanded to additional 42,833 spam sites
Topical TrustRank (Wu, Goel, and Davison 2006b)	2006	✓		Web Base Jan 01 (65,000,000 pages) search.ch (≈350,000 sites)	Decrease spam by 19% - 43.1% from the top ranked sites when compared with TrustRank
Anti-TrustRank (Krishnan and Raj 2006)	2006		✓	Web Graph 2002 (18,500,000 pages)	1.721% of spam pages found using Anti-TrustRank while 0.28% spam pages found using PageRank for top 100,000 pages

(Table 3-4 continued)

Spam Mass (Gyöngyi et al. 2006)	2006	✓		Yahoo! 2004 (73,300,000 sites)	Detected ≈10,000 link spam hosts
Wu et al. (Wu, Goel, and Davison 2006a)	2006	✓	✓	search.ch (≈350,000 sites)	3,589 labelled spam sites; remove 80% of spam sites out of top 10 buckets
DiffusionRank (Yang, King, and Lyu 2007)	2007	✓		Middle Size Graph (18,542 pages) Large Size Graph (607,170 pages)	The anti-manipulation feature enables DiffusionRank to be a candidate as a penicillin for Web Spamming
Nie et al. (Nie, Wu, and Davison 2007)	2007	✓	✓	WEBSPAM-UK2006 (11,402 hosts)	Moving 23.4 more normal host to top 10 buckets while moving out 5.7 spam hosts
R-SpamRank (Liang, Ru, and Zhu 2007)	2007		✓	Sogou.com (5,000,000 pages)	Precision of 99.1% for top 10,000 pages being spam
Link Variable TrustRank (Qi, Song-Nian, and Sisi 2008)	2008		✓	WEBSPAM-UK2007 (114,529 hosts)	By combining Inlink Growth Rate with TrustRank, the experiment shows the method is effective in detecting spam sites
QoC-QoL algorithm (Li, Qiancheng, and Yan 2008)	2008	✓	✓	13.3 million Web pages and 232 million links	Mixed seed set is effective in identifying Web spam sites regardless of their way of combination.
AVRank and HVRank (Zhang et al. 2009)	2009	✓	✓	Tianwang (358,245 hosts)	By exploiting bidirectional links and large seed set, the algorithms able to achieve better performance.
Trust-Distrust Rank (Zhang, Wang, et al. 2011)	2011	✓	✓	WEBSPAM-UK2007 (105,896,555 pages) TREC ClueWeb09 (428,136,613 pages)	Overcome the disadvantages of TrustRank and Anti-TrustRank and outperform them

3.4 MACHINE LEARNING TECHNIQUES

In recent year, researchers in the adversarial information retrieval community have moved towards machine learning approach to detect Web spam. Actually the Web spam problem can be viewed as a classification problem. Machine learning constructed Web spam classifiers have shown positive results due to their adaptive ability to learn the underlying patterns for classifying spam and non-spam. Machine learning approach can be divided into two categories – features and structures. The former depicts as the input used for classification while the latter define the machine learning algorithm that is used for learning. Some aforementioned link-based trust and distrust model algorithms are used as features to assist the machine learning model.

The WEBSPAM-UK datasets have made a leap in Web spam community for using various machine learning models. In fact, previously there are few Web spam challenge series – Web spam challenge track I (Castillo, Chellapilla, and Davison 2007), II (Castillo, Davison, et al. 2007) and III (Castillo, Chellapilla, and Denoyer 2008) which aim is to bring both machine learning and information retrieval community to solve the Web spam labelling problem.

In this sub-section, a comprehensive literature review on machine learning models that have been proposed is given throughout the years. The features for machine learning model are reviewed first follow by the structures of the machine learning model.

Becchetti et al. (Becchetti et al. 2006b) study several link-based metrics which include rank propagation for links and probabilistic counting to improve the Web spam detection techniques. Moreover, the authors conducted another similar research (Becchetti et al. 2006a) which include more link-based metrics such as degree correlation and number of neighbours, and as a result the metrics achieve 80.4% detection rate with 1.1% false positive on WEBSPAM-UK2002 dataset.

Besides link-based features, some researchers (Ntoulas et al. 2006) propose several content-based features for Web spam detection. The content of Web pages can be modified in order to attract Web users, a technique known as keyword-stuffing. The authors experiment on 105 million Web pages and 86.2% spam pages detected.

Stacked graphical learning (Kou 2007), a meta-learning scheme, has shown positive results in Web spam detection (Castillo, Donato, et al. 2007). Some researchers (László and Siklósi 2007) take advantage of stacked graphical learning by generating features by averaging known and predicted labels for similar nodes of the graph. The authors achieve improvement of 0.01% F-measure for small graph and 0.111% F-measure for large graph.

Gan and Suel (Gan and Suel 2007) propose 8 content features, 14 link-based features and 3 additional features which include number of hosts in the domain, ratio of pages in this host to pages in this domain and number of hosts on the same IP address. The overall features achieved more than 90% F-measure for spam and non-spam detection in Swiss dataset.

Castillo et al. (Castillo, Donato, et al. 2007) use the combination of link-based features from (Becchetti et al. 2006a) and content-based features from (Ntoulas et al. 2006) and experiment on WEBSPAM-UK2006 dataset and result in 88.4% of spam hosts detected with 6.3% false positive.

A preliminary study on using linguistic features for Web spam detection is conducted by Piskorski et al. (Piskorski, Sydow, and Weiss 2008) and concluded by providing several discriminating Corleone and General Inquirer attributes that are promising enough to discriminate spam and non-spam.

Becchetti et al. (Becchetti, Castillo, Donato, Baeza-YATES, et al. 2008) perform a detailed statistical analysis that only consider link structure of the Web for Web spam detection. Their experiments show that the performance of all combined features is comparable with that state-of-the-art spam classifier that use content attributes.

Becchetti et al. (Becchetti, Castillo, Donato, Leonardi, et al. 2008) later use both link and content features to classify spam and non-spam. In addition, the authors use graph clustering algorithms, propagation of predicted labels and stacked graphical learning to improve the classification accuracy. As a result, their proposed methodology manages to detect up to 88% of spam pages.

Linked latent Dirichlet allocation (LDA), an extension of LDA proposed by B ŕó et al. is used for Web spam classification. The linked LDA technique consider linkage such as topics are propagated along links in such a way that the linked document directly influences the words in the linking Document. The authors concluded that linked LDA outperforms LDA and other baseline classifier about 3% to 8% in AUC performance.

Historical Web page information is important for Web spam classification. Dai et al. (Dai, Davison, and Qi 2009) propose 1270 temporal features to improve the performance of Web spam classifiers. The features are experimented on WEBSPAM-UK2007 and have shown that their approach improves the F-measure by 30% compared to the baseline classifier which only considers current page content.

Martinez-Romo and Araujo (Martinez-Romo and Araujo 2009) presented 42 language model features to represent a Web document that calculate disagreement between two Web pages. The authors experiment on WEBSPAM-UK2006 and WEBSPAM-UK2007 and show that the language model features improve the F-measure of the former dataset by 6% and latter dataset by 2%.

Later on, the authors combined their language model features with 12 qualified link analysis features (Araujo and Martinez-Romo 2010) along with both content and link-based features, the overall features achieve 0.86 F-measure and 0.88 AUC performance in WEBSPAM-UK2006, and 0.40 F-measure and 0.76 AUC performance in WEBSPAM-UK2007.

Abernethy et al. (Abernethy, Chapelle, and Castillo 2010) present WITCH (which stands for Web Identification Through Content and Hyperlinks) algorithm; not only the authors use content and link-based features, the authors also include slack features and graph regularization features. The authors achieve 0.928 for AUC 10% and 0.963 for AUC 100% in WEBSPAM-Uk2006 using support vector machine.

Li et al. (Li et al. 2011) generate 10 new features from link features based on genetic programming and show that the new features are well performed than 41 standardized link-based features and also 138 transformed link-based features.

A table which shows a list of features from various scientific publications for classification are given as:

Table 3-5: List of features for classification

Features & Authors	***Structure***	***Datasets***	***Achieve***
163 Link-based Features (Becchetti et al. 2006a)	Decision Tree with Boosting	WEBSPAM-UK2002	80.4%% of detection rate with 1.1% false positive
82 Link-based Features (Becchetti et al. 2006b)	Decision Tree (Pruning with M = 5 and M = 10)	WEBSPAM-UK2002	80% of spam pages detected with 2% false positive
Content Features (Ntoulas et al. 2006)	C4.5 Decision Tree	MSN Search 105,484,446 Web pages	86.2% spam pages detected
Stack Graphical Learning (László and Siklósi 2007)	C4.5 Decision Tree	WEBSPAM-UK2006	0.01% F-measure improvement on small graph and 0.111% F-measure improvement on large graph
8 Content Features, 14 Link-based Features and 3 Additional Features(Gan and Suel 2007)	C4.5 Decision Tree and Support Vector Machine	Swiss ch 2005 (239272 hosts)	More than 90% F-measure on Spam and Non-Spam detection

(Continued Next Page)

(Table 3-5 continued)

140 Link-based Features, 96 Content-based Features (Castillo, Donato, et al. 2007)	C4.5 Decision Tree	WEBSPAM-UK2006	88.4% of spam hosts detected with 6.3% false positive
208 Linguistic Features (Piskorski, Sydow, and Weiss 2008)	-	WEBSPAM-UK2006/2007	Certain linguistic features are useful for Web spam detection when combined with features studied elsewhere
163 Link-based Features (Becchetti, Castillo, Donato, Baeza-YATES, et al. 2008)	C4.5 Decision Tree with Bagging	WEBSPAM-UK2002/2006	87% for WEBSPAM-UK2002, 63% for WEBSPAM-UK2006
45 Link-based Features, 18 Content based Features (Becchetti, Castillo, Donato, Leonardi, et al. 2008)	C4.5 Decision Tree	WEBSPAM-UK2006	Detected up to 88% of spam pages
linked LDA Features (B ŕ ó et al. 2009)	Bayes Net, Support Vector Machine, C4.5 Decision Tree	WEBSPAM-UK2007	85.4% AUC (win 84.8% Winner of Web Spam Challenge 2008)
1270 Temporal Features (Dai, Davison, and Qi 2009)	Support Vector Machine	WEBSPAM-UK2007	F-measure outperform by 30%
42 Language Model Features (Martinez-Romo and Araujo 2009)	Metacost (cost sensitive Decision Tree with bagging)	WEBSPAM-UK2006/UK2007	Improve 6% F-measure in WEBSPAM-UK2006, Improve 2% F-measure in WEBSPAM-UK2007
42 Language Model Features and 12 Qualified Links Features (Araujo and Martinez-Romo 2010)	C4.5 Decision Tree	WEBSPAM-UK2006/UK2007	Combined with Link and Content Features achieve AUC performance of 0.88 and 0.76 for WEBSPAM-UK2006 and UK2007

(Continued Next Page)

(Table 3-5 continued)

Slack Features, Graph Regularization Features (Abernethy, Chapelle, and Castillo 2010)	Support Vector Machine	WEBSPAM-UK2006	0.928 of AUC 10% and 0.963 of AUC 100%
10 Genetic Programming Features (Li et al. 2011)	Support Vector Machine and Genetic Programming	WEBSPAM-UK2006	10 newly generated features are better than 41 link features and 138 transformed link features

Besides features, the structure that is used to determine the machine to learn is also important.

Noi et al. (Noi et al. 2010) present a spam detection approach based on probability mapping graph self-organizing maps (PM-GraphSOMs) for clustering Web pages and graph neural networks (GNNs) for classification. Their approaches achieved better results than those who participate in the Web spam challenge 2007 with F-measure of 0.9169 and AUC of 0.9301. However, using both unsupervised and supervised techniques are computationally expensive.

A harmonic function based semi-supervised learning for Web spam detection is proposed by Zhang et al. (Zhang, Zhu, et al. 2011) and conducted the experiments by comparing with other semi-supervised learning methods and achieve the highest precision, recall and F-measure.

Leon-Suematsu et al. (Leon-Suematsu et al. 2011) presented a Web spam detection algorithm that predicts the spamicity of subgraphs based on the bow-tie structure of Web graphs by a support vector machine. 0.83 precision, 0.94 recall and 0.88 F-measure are achieved in WEBSPAM-UK2006.

Zhiyang et al. (Zhiyang et al. 2012) compare three machine learning models for Web spam detection: rule-based classifier, decision tree based and support vector machine. The results have shown that support vector machine outperform both rule-based and decision tree based by precision, recall and f1-value.

Fake medical websites are increasing widespread in recent years. Abbasi et al. (Abbasi et al. 2012) propose recursive trust relabeling, an adaptive learning algorithm which uses underlying content and graph-based classifiers, coupled with a recursive labeling mechanism, for enhanced detection of fake medical websites.
There are researchers (Al-Kabi et al. 2012; Wahsheh, Al-kabi, and Alsmadi 2012) focuses on combating Arabic Web spam. The authors conducted experiments on various machine learning models such as naïve bayes, decision tree, support vector machine, k-nearest neighbor and logitboost, and achieve spam detection with more than 90% accuracy.

Below shows a list of structure from various scientific publications for Web spam detection:

Table 3-6: List of structures for classification

Authors	***Structure***	***Datasets***	***Achieve***
(Noi et al. 2010)	PM-GraphSOMs and Graph Neural Network	WEBSPAM-UK2006	F-measure 0.9169 and AUC 0.9301
(Zhang, Zhu, et al. 2011)	Harmonic Functions Based Semi-supervised Learning	WEBSPAM-UK2006	83.8% Precision, 93.1% Recall, 88.2% F-measure
(Leon-Suematsu et al. 2011)	Support Vector Machine	WEBSPAM-UK2006	83% Precision, 94% Recall, F-measure 88%
(Zhiyang et al. 2012)	Soft margin classifier - Support Vector Machine, Rule Based, Decision Trees	137640 Web pages - 9634 (7%) and 128006 (93%)	Support Vector Machine outperform Rule-based and Decision Tree-based
(Abbasi et al. 2012)	Recursive Trust Labelling	930,000 Websites	over 90% accuracy on three test bed

(Continue next page)

(Table 3-6 continued)

(Al-Kabi et al. 2012)	Naïve Bayes, Decision Tree, Support Vector Machine, K-Nearest Neighbour, LogitBoost	Arabic 15,000 Web Pages	Decision Tree is the best with 99.521% accuracy
(Wahsheh, Al-kabi, and Alsmadi 2012)	Decision Tree, Naïve Bayes	Arabic Link Spam Corpus (3,000 Web Spam pages)	91.4706% Accuracy for Decision Tree, 81.17655% Accuracy for Naïve Bayes

Some researchers proposed their own features and structures in assist of Web spam detection.

Tian et al. (Tian, Weiss, and Ma 2007) employ a combinatorial feature-fusion method for compressing enormous amount of word-based features and produce 200 combinatorial feature-fusion features. The researchers experiment on three learning models - alternating decision tree, sequential minimal optimization based support vector machine and naïve bayes, and their alternating decision tree achieve the best result with 0.931 AUC, 0.716 F-measure, 0.797 precision and 0.649 recall.

Tang et al. (Tang et al. 2007) extract features from link-based data and combine with text-based data and produce 4,924,007 features for Web spam detection. However, the authors only select 28,051 features out of 4,924,007 features due to the limit of computation. Random forest and support vector machine with radial basis function are used for host classification while linear support vector machine is used for page classification. The authors experiment on WEBSPAM-UK2006 and achieve F-measure of 75.46% and 95.11% AUC for small dataset, 90.20% F-measure and 98.92% AUC for large dataset.

Except for content features and page-level link analysis feature, Geng et al. (Geng, Zhu, and Wang 2009) extract host-level link analysis feature for Web spam classification. The researchers have shown that by incorporating the three feature set, the best performance can be achieved.

Erdélyi et al. (Erdélyi, Garzó, and Benczúr 2011) propose a new feature set - a bag of words derived from BM25 term weighting scheme to improve classification tasks. The researchers experimented using various machine learning models such as ensemble selection, logitboost and random forest, and show that these three machine learning models improve the accuracy results.

Even though there are plenty of algorithms and machine learning techniques used to combat Web spam, content providers still try to think another way to manipulate the search engines, this field is known as "Adversarial Information Retrieval (Adversarial IR)", a war between search engines and those who tries to manipulate them (Castillo and Davison 2011).

A list of scientific publications that use features and structures are shown in a table in the next page:

Table 3-7: List of features and structures for classification

Features & Authors	***Structure***	***Datasets***	***Achieve***
200 Combinatorial Feature-Fusion Features (Tian, Weiss, and Ma 2007)	Alternating Decision Tree, Sequential Minimal Optimization (Support Vector Machine) and Naïve Bayes	WEBSPAM-UK2006	Alternating Decision Tree is the best classifier among with 0.931 AUC, 0.716 F-measure, 0.797 Precision and 0.649 Recall after Semi-supervised learning and Fusion.

(Continue next page)

(Table 3-7 continued)

28,051 features (Tang et al. 2007)	Support Vector Machine, Random Forest	WEBSPAM-UK2006	(Small) 75.46% F-measure, 95.11% AUC (Large) 90.20% F-measure, 98.92% AUC
40 Host Graph based features (Geng, Zhu, and Wang 2009)	C4.5 Decision Tree with Bagging, Adaboost with Decision Stump	WEBSPAM-UK2006	85.5% Precision, 88.7% Recall, 87.1% F1-measure, 97.1 AUC
10,000 BM25 Features (Erdélyi, Garzó, and Benczúr 2011)	Bagged and Boost Decision Tree, Logistic Regression, naïve Bayes, random forest, Support Vector Machine	WEBSPAM-UK2007 & DC-2010	All 10273 including link, content and BM25 Features achieve 0.902 AUC

3.5 SUMMARY

In this chapter, two anti-Web spam techniques are covered – trust and distrust based model and machine learning model. TrustRank, a trust based anti-Web spam algorithm is presented based on some initial trustworthy seeds, and propagate trust to detect other trustworthy pages. However, there are some flaws in TrustRank algorithm. Thus, other researchers propose the derivatives of TrustRank such as Anti-TrustRank , Topical TrustRank , DiffusionRank and Link Variable TrustRank . The derivatives of TrustRank algorithms are thoroughly described.

Experiments are conducted for HostRank and TrustRank on two large public available datasets – WEBSPAM-UK2006 and WEBSPAM-UK2007 to show how vulnerable it is for link analysis algorithms. As a result, TrustRank detect more trustworthy hosts and demote spam hosts. Furthermore, TrustRank has shown that more seeds eventually will lead to better performance. Other trust and distrust model algorithms are then briefly explained in this chapter. A comparison table for trust and distrust model algorithms is also presented. After that, machine learning approaches in Web spam detection are discussed which include features that assist machine to learn and structures that define the machine learning model.

Chapter 4 Trust Propagation Algorithms

4.1 INTRODUCTION

The quantity and quality of the seed sets are the key factors for the success of trust and distrust based anti-Web spam algorithms. This kind of approach is simple and yet effective, but the manual evaluation of seed sets is very time-consuming. For this reason, the manual evaluation process becomes vital and valuable.

In this chapter, Trust Propagation Rank (TPRank) is proposed with the idea of calculating trust scores for all pages based on limited evaluation of non-spam and spam seeds to demote Web spam. To enhance the proposed algorithm, “ugly” pages are underlined for the reason that the categorization of “ugly” pages and pure good pages can avoid promoting spam pages. Furthermore, spam pages are punished by giving them zero rank so that it would not affect the ranks of other pages.

In addition to this proposed algorithm, Spam Mass (Gyöngyi et al. 2006) algorithm is modified with trust propagation into Trust Propagation Spam Mass (TP Spam Mass) to detect Web spam. Experiments are done on two available datasets WEBSPAM-UK2006 (Castillo et al. 2006) and WEBSPAM-UK2007 (Yahoo! 2007), and the results have shown that TPRank outperforms TrustRank in demotion of Web spam and TP Spam Mass outperforms Spam Mass in detection of Web spam.

4.2 TRUST PROPAGATION

In this section, the ugly vertices, a new trust score calculation and a way to handle spam vertices are introduced. Furthermore, two anti-Web spam algorithms are proposed – Trust Propagation Rank (TPRank) and Trust Propagation Spam Mass (TP Spam Mass).

4.2.1 Definition

The definitions are provided for the ease of understanding the algorithms in the rest of the chapter.

Unevaluated vertices, denoted as υ_X, refer to unknown pages which are not evaluated. *Non-spam vertices*, denoted as υ_N, refer to reputable pages that provide reliable content to the users. The opposite is *spam pages*, denoted as υ_S, which refer to pages that deliberately provide unreliable content to the user. TrustRank follows the intuition that non-spam pages seldom point to spam pages and trust flows. However, it does not work in the real Web. Spammers can get lots of incoming links from non-spam pages using indecent ways (Qi, Song-Nian, and Sisi 2008). One way of doing this is by leaving comments on *accessible pages*, i.e. pages that can be edited by external like blog and Wikipedia. This kind of pages are distinguished as ugly pages υ_U which apart from the pure good pages υ_G. *Ugly pages* refer to the aforementioned accessible pages or reputable pages that unintentionally link to spam pages. The ugly pages are one of the reasons that spam pages got promoted easily. On the other hand, pure good pages are reputable pages that there is no way that one would link to spam pages.

For the assessment of ugly vertices υ_U, this can be done after the evaluation of non-spam vertices υ_N and spam vertices υ_S. For all non-spam vertices υ_N, if any of the outgoing vertices of the non-spam vertex is a spam vertex, the non-spam vertex then categorize into set of ugly vertices υ_U, otherwise set of pure good vertices υ_G.

4.2.2 Trust Score Calculation

A new trust score calculator is introduced to calculate the trust score of the unknown vertices υ_X. The ugly vertices that introduced earlier are used to enhance the new trust score calculation. The equation of the trust score calculator is written as

$$t_p = \frac{\sum_{(q:p)\in E} t_q}{(i_G + i_X)} \tag{4.1}$$

t_p is the trust score for unknown vertex *p*. i_G is the number of pure good vertices while i_X is the number of unevaluated vertices. The vertices in i_G, i_X and t_q vertices refer to the incoming vertices. The new trust score of page *p* is calculated by the trust score of the incoming links. For all incoming links, spam vertices and ugly vertices are simply ignored for the reason that their trust is not trustworthy as the vertices might be pointing to spam pages.

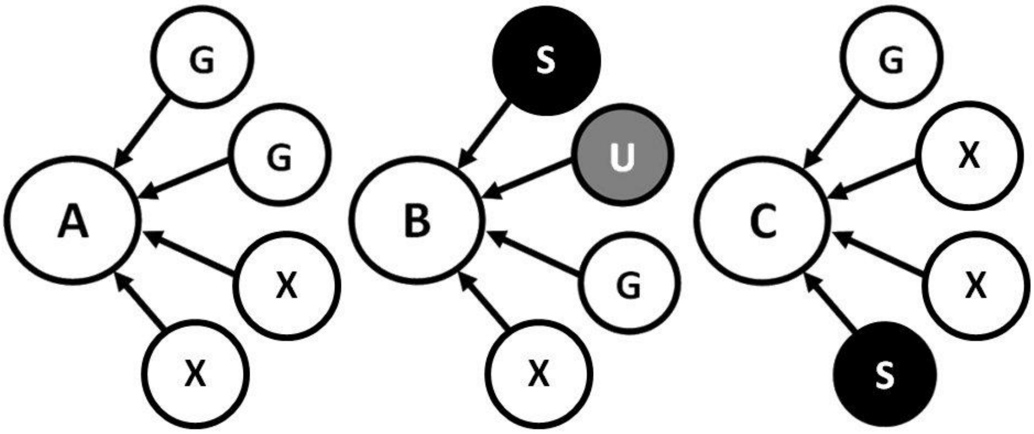


Figure 4.1: Three examples of trust score calculation

Assume page *A* where it is pointed by two υ_G and two υ_X (showing in the left in Figure 4.1), the new trust score is calculated as:

$$t_A = \frac{\sum_{(A:p)\in\varepsilon} t_q}{\left(i_G + i_X\right)} = \frac{2}{\left(2+2\right)} = \frac{2}{4} = 0.500$$

Assume page *B* where it is pointed by one υ_S, one υ_U, one υ_G and one υ_X (showing in the middle in Figure 4.1), the new trust score is calculated as:

$$t_B = \frac{\sum_{(B:p)\in\varepsilon} t_q}{(i_G + i_X)} = \frac{1}{(1+1)} = \frac{1}{2} = 0.500$$

Assume page *C* where it is pointed by one υ_G, two υ_X and one υ_S (showing in the right in Figure 4.1), the new trust score is calculated as:

$$t_B = \frac{\sum_{(B:p)\in\varepsilon} t_q}{(i_G + i_X)} = \frac{1}{(1+2)} = \frac{1}{3} = 0.333$$

4.3.3 Handling Spam Vertices

During the assessment of the seed set, both non-spam seeds and spam seeds are evaluated. Often either one of the seed sets is used to propagate trust or distrust, for example TrustRank only uses non-spam seed set to propagate trust with the spam seed set remain unused. Seed sets are expensive to be evaluated and should therefore make good use of both non-spam and spam seed set. TrustRank has shown that non-spam vertices will receive high trust score while spam vertices receive low trust score. Even though it is low, spam vertices can work together and boost one target page. In other words, spam vertices can still affect other vertices. ParentPenalty (Wu and Davison 2005b; Wu, Goel, and Davison 2006a) penalize non-spam vertices that point to spam vertices. However, non-spam vertices might unintentionally point to spam vertices; spammers might leave comments to make non-spam vertices point to them. In this research, the spam vertices are punished by giving them zero rank. By achieving this, the spam vertices have no chance of affecting non-spam vertices with low trust score and will not get ranked even though pointed by other vertices.

4.3.4 Trust Propagation Rank (TP Rank)

Trust Propagation Rank (TPRank), a Web spam demotion algorithm that works similar to TrustRank is proposed but propagates trust further based on the same limited set of evaluation seeds. Unlike TrustRank, TPRank use both non-spam seed set and spam seed set to demote spam. The seeds are selected based on inverse PageRank for the reason that to choose the seeds that propagate the widest coverage (Gyöngyi, Garcia-Molina, and Pedersen 2004). The equation for inverse PageRank can be written as:

$$IPR = \alpha \cdot I \cdot IPR + (1-\alpha) \cdot \frac{1}{N} \cdot 1_N \tag{4.2}$$

Where IPR is the inverse PageRank score, α is a decay factor usually set as 0.85, I is the inverse transition matrix of the Web graph and N is the number of the vertices. During the process of seed selection, spam seeds are collected too. After the collection, both ugly vertices and pure good vertices can be extracted out of the non-spam vertices.

Algorithm Trust Propagation Rank (*TPRank*)

Input

T	*Transition matrix*
N	*number of pages*
α	*decay factor*
M	*number of iterations*
$\sigma(i)$	*i^{th} vertex*
t	*trust score*
$\bar{t}$	*trust score*
i_G	*Number of incoming pure good vertices of page i*
i_X	*Number of incoming unknown vertices of page i*

Output

TPR	*TPRank scores*

Begin

Assume that the seeds are evaluated as ugly vertices υ_U, *pure good vertices* υ_G *and spam vertices* υ_S.

(See next page)

1) //*calculate trust score for unknown pages*
for i = 1 to N do
if $\sigma(i) \notin \upsilon_G$ and $\sigma(i) \notin \upsilon_S$ then

$$t(\sigma(i)) = \frac{\sum_{(q:\sigma(i))\in\varepsilon} t_q}{(i_G + i_X)}$$

end if
end for

2) *// normalize trust score vector*

$\bar{t} = t / |t|$

3) *// compute TPRank scores*

$TPR = \bar{t}$

for i = 1 to M do

$$TPR = \alpha \cdot T \cdot TPR + (1-\alpha) \cdot \bar{t}$$

end for

End

Figure 4.2: Trust Propagation Rank (TPRank) Algorithm

4.2.5 Trust Propagation Spam Mass (TP Spam Mass)

In (Gyöngyi et al. 2006), the authors proposed the concept of Spam Mass, a measure for the impact of link spamming on PageRank. By estimating Spam Mass, it can help by identifying pages that significantly benefit from link-spamming. Spam Mass is built on top of PageRank and TrustRank, so the equation for Spam Mass is:

$$SM = \frac{PR - TR}{PR} \tag{4.3}$$

where SM stands for Spam Mass, PR stands for PageRank and TR stands for TrustRank. A vertex's Spam Mass is calculated based on its PageRank score minus TrustRank score and divided by its PageRank score. Note that both PageRank and TrustRank should be normalized first before proceed.

In this research, Trust Propagation Rank (TPRank) can be extended to Trust Propagation Spam Mass (TP Spam Mass) where the equation is written as:

$$TP\_SM = \frac{PR - TP}{PR} \tag{4.4}$$

where TP_SM stands for Trust Propagation Spam Mass and TP stands for Trust Propagation Rank. It has shown that Spam Mass works more effective than Anti-TrustRank (Qureshi 2011). Equation 4.3 and Equation 4.4 are important to show the detection of Web spam.

4.2.6 Example

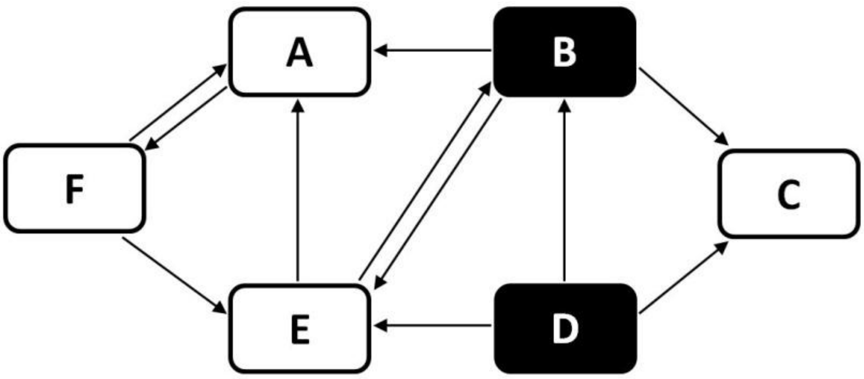


Figure 4.3: Sample Web graph

Figure 4.3 illustrates a sample Web graph which contains 4 non-spam vertices (highlighted in white box) and 2 spam vertices (highlighted in black box).

An example on the above figure is provided in this sub-section by executing TrustRank and TPRank. In addition, the results on Spam Mass and TP Spam Mass are also provided.

Firstly, non-spam seeds are selected based on inverse PageRank (see Equation 4.2), this is similar to TrustRank (Gyöngyi, Garcia-Molina, and Pedersen 2004) seed selection method as the inverse PageRank selected seeds have the most widest propagation. In this example, top three seeds are selected and evaluate:

$$\upsilon_E = \{A, B, E\},\ \ \upsilon_N = \{A, E\}, \upsilon_S = \{B\}$$

Since $\upsilon_N = \{A, E\}$ and vertex *E* is pointing to vertex *B*, a spam vertex, then υ_N can be further categorize into υ_U, set of ugly vertices and υ_G, set of good vertices such as $\upsilon_U = \{E\}$ and $\upsilon_E = \{A\}$. υ_N is used in TrustRank while υ_S, υ_U and υ_G are used in TPRank to find the trust score for the vertices. The trust scores are:

In TrustRank,

$$t = [0.5 \quad 0 \quad 0 \quad 0 \quad 0.5 \quad 0];$$

In TPRank,

$$\bar{t} = [0.33 \quad 0 \quad 0 \quad 0 \quad 0.33 \quad 0.33].$$

After that, the results from both algorithms are used for Spam Mass (see Equation 4.3) and TP Spam Mass (see Equation 4.4).

The results are shown below:

Table 4-1: Results from various algorithms on sample Web graph

Algorithms	***Vertices***					
	A	***B***	***C***	***D***	***E***	***F***
TrustRank	0.147	0.066	0.027	0.019	0.155	0.126
TPRank	0.139	0	0.025	0.018	0.147	0.168
Spam Mass	-0.072	0.279	0.652	0.645	-0.109	0.082
TP Spam Mass	-0.201	0.322	0.675	0.667	-0.046	-0.229

As shown in Table 4-1, TPRank actually punishes vertex *B* for being a spam vertex. In TrustRank, spam vertex *B* is actually higher than vertex *C* and *D* for the reason that the unevaluated vertices are treated the same status even though spam vertices have various way to get rank higher than some innocent unknown vertices. In Spam Mass and TP Spam Mass comparison, the biggest difference is vertex *F* where it shows negative value in TP Spam Mass while Spam Mass is showing a positive value (negative value actually shows how trustworthy it is while positive value shows its spamicity). In TP Spam Mass, trust are propagated to vertex *F* as it is pointed to vertex *A* and *E*, thus it is most likely that this vertex is a non-spam vertex.

4.3 Experimental Results

In the experiment, 50 non-spam seeds are used for WEBSPAM-UK2006 and 100 non-spam seeds are used for WEBSPAM-UK2007. During the selection for the good seeds, 179 spam seeds are detect in WEBSPAM-UK2006 while in WEBSPAM-UK2007, 214 spam seeds are detected. For TPRank purpose, the ugly seeds are evaluated based on non-spam seeds and spam seeds. As a result, there are 20 ugly vertices and 30 pure good vertices in non-spam seeds for WEBSPAM-UK2006 and 24 ugly vertices and 76 pure good vertices in non-spam seeds for WEBSPAM-UK2007.

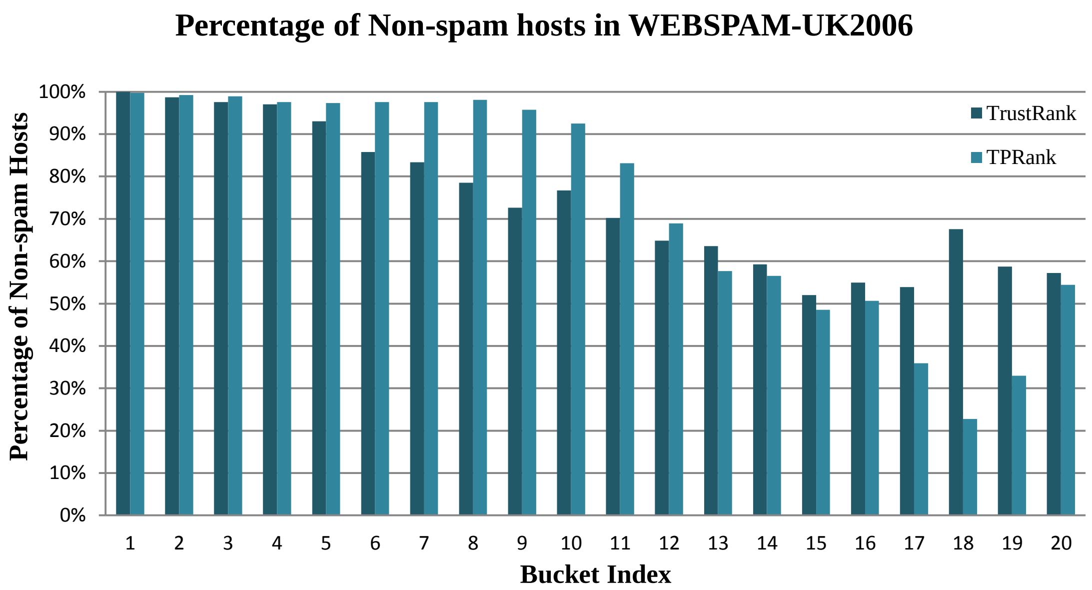


Figure 4.4: Percentage of non-spam hosts in WEBSPAM-UK2006

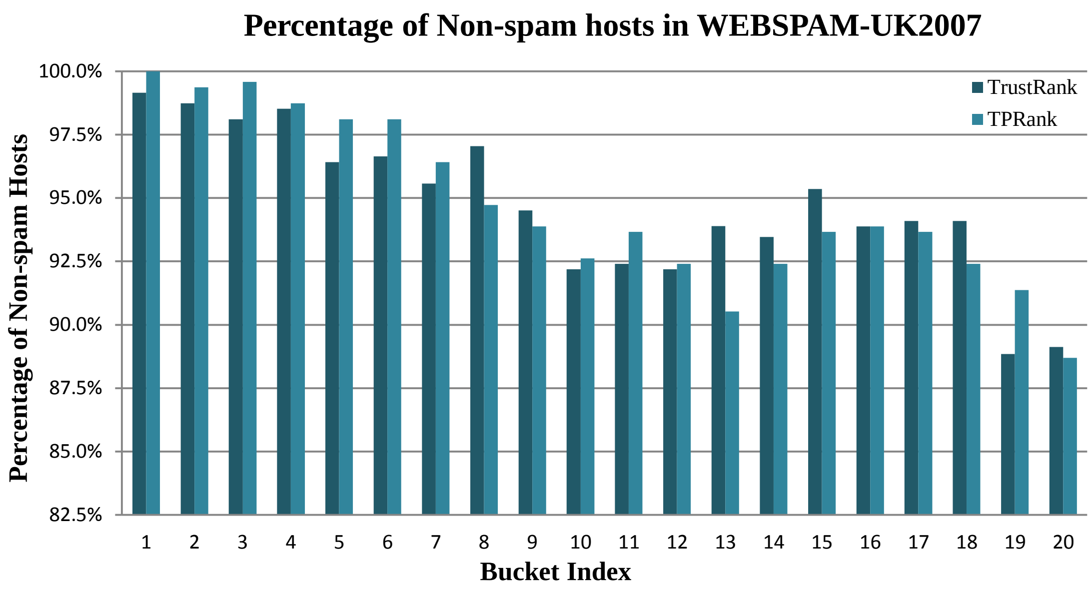


Figure 4.5: Percentage of non-spam hosts in WEBSPAM-UK2007

Figure 4.4 illustrates the percentage of non-spam hosts in WEBSPAM-UK2006 while Figure 4.5 illustrates the percentage of non-spam hosts in WEBSPAM-UK2007. TPRank able to detect more non-spam hosts than TrustRank for the first twelve buckets in WEBSPAM-UK2006 shown in Figure 4.4 and for the first seven buckets in WEBSPAM-UK2007 show in Figure 4.5. It is important to demote spam hosts as early as possible so that spam hosts do not appear much at the top results.

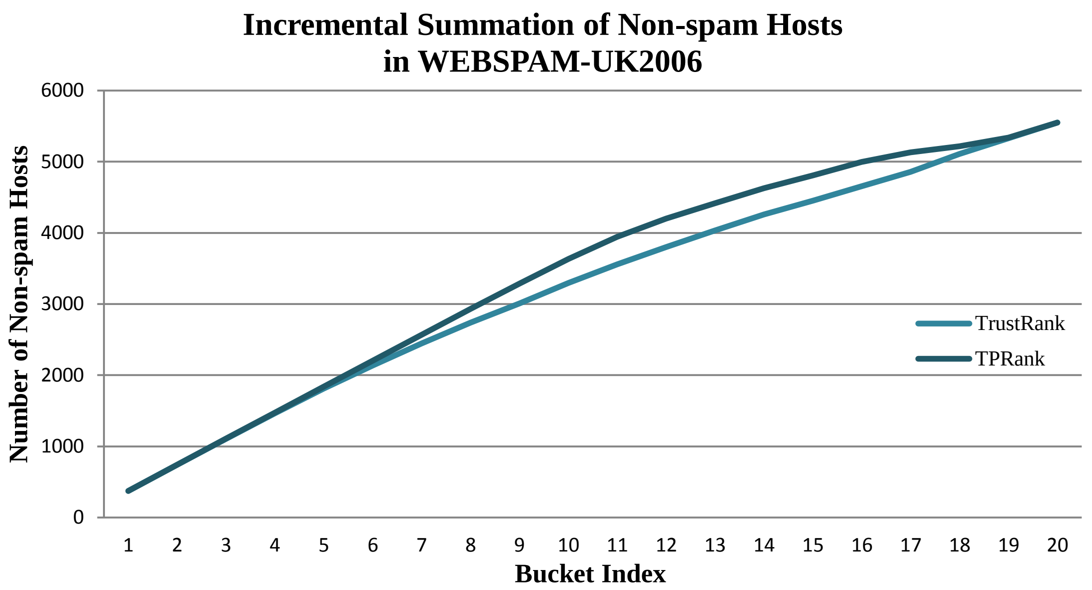


Figure 4.6: Incremental summation of non-spam hosts in WEBSPAM-UK2006

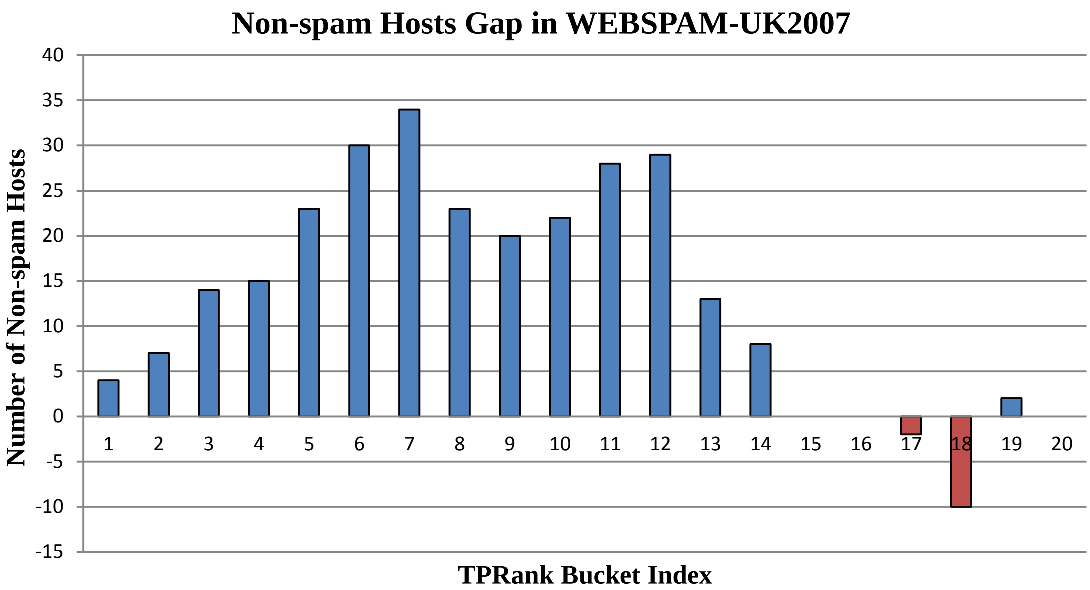


Figure 4.7: Non-spam hosts gap in WEBSPAM-UK2007

Observed from Figure 4.6, TrustRank detects 3799 non-spam hosts and TPRank detects 4201 non-spam host in the 12^{th} bucket in WEBSPAM-UK2006, it has the biggest improvement with 402 non-spam hosts detected. On the other hand for WEBSPAM-UK2007 showing in Figure 4.7, the 7^{th} bucket has the biggest improvement gap of 34 non-spam hosts detected. There is only a slight improvement for WEBSPAM-UK2007 dataset for the reason that the number of label spam hosts is small, thus it is relatively hard to see the improvement of the non-spam hosts. Nevertheless, it has shown that TPRank able to detect more non-spam hosts compare to TrustRank algorithm.

Figure 4.8 to Figure 4.11 illustrate the average promotion for non-spam hosts and the number of non-spam hosts promoted from TPRank over TrustRank buckets in WEBSPAM-UK2006 and WEBSPAM-UK2007.

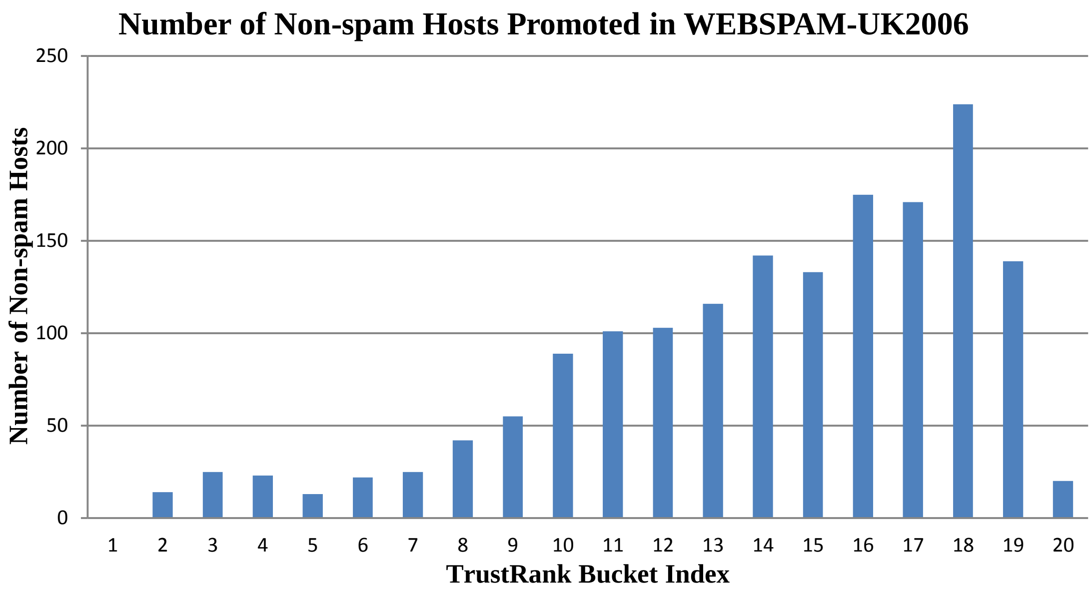


Figure 4.8: Number of non-spam hosts promoted in WEBSPAM-UK2006

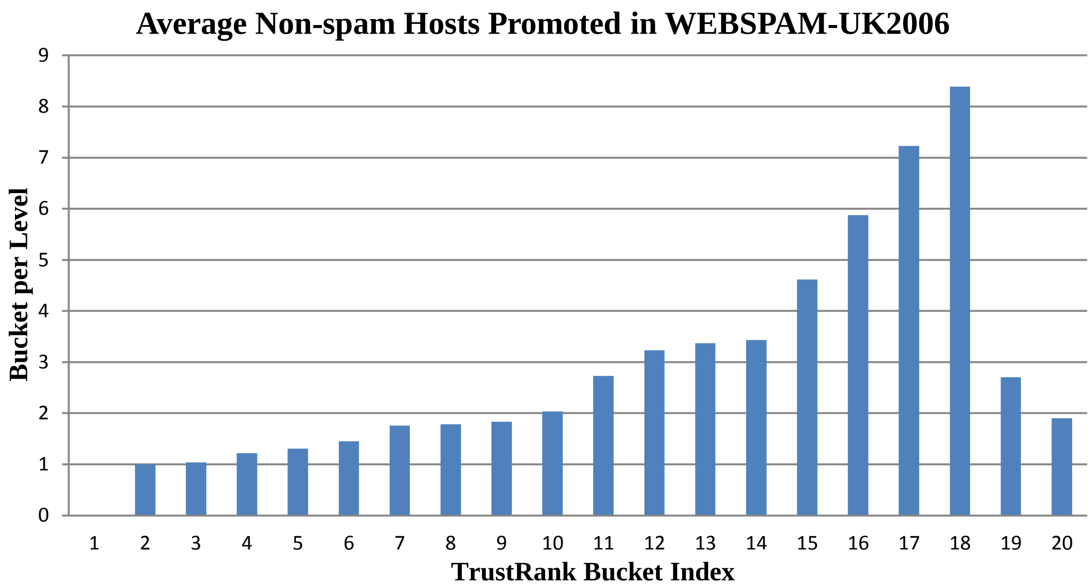


Figure 4.9: Average non-spam host promoted in WEBSPAM-UK2006

Observed from Figure 4.8 and 4.9, the 18th bucket has the highest improvement with an average non-spam hosts promotion of 8.388 bucket per level promoting 224 non-spam hosts in WEBSPAM-UK2006.

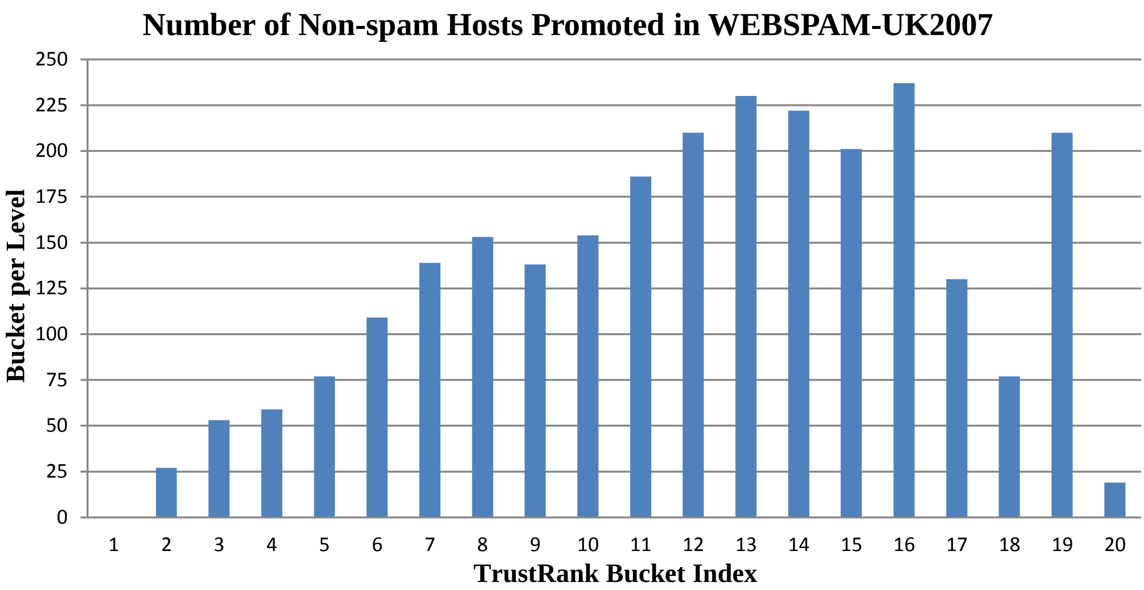


Figure 4.10: Number of non-spam hosts promoted in WEBSPAM-UK2007

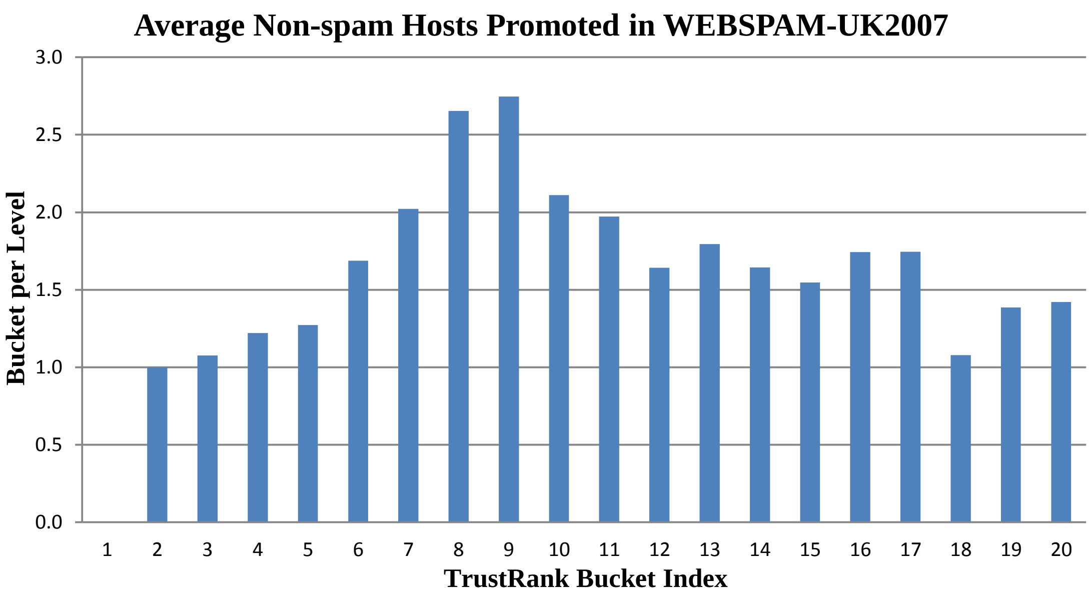


Figure 4.11: Average non-spam host promoted in WEBSPAM-UK2007

In Figure 4.10 and 4.11, the highest average non-spam host promotion is the 9th bucket with the promotion of 2.746 bucket per level and the bucket that has the highest number of promoted non-spam hosts is the 16th bucket promoting 237 non-spam hosts.

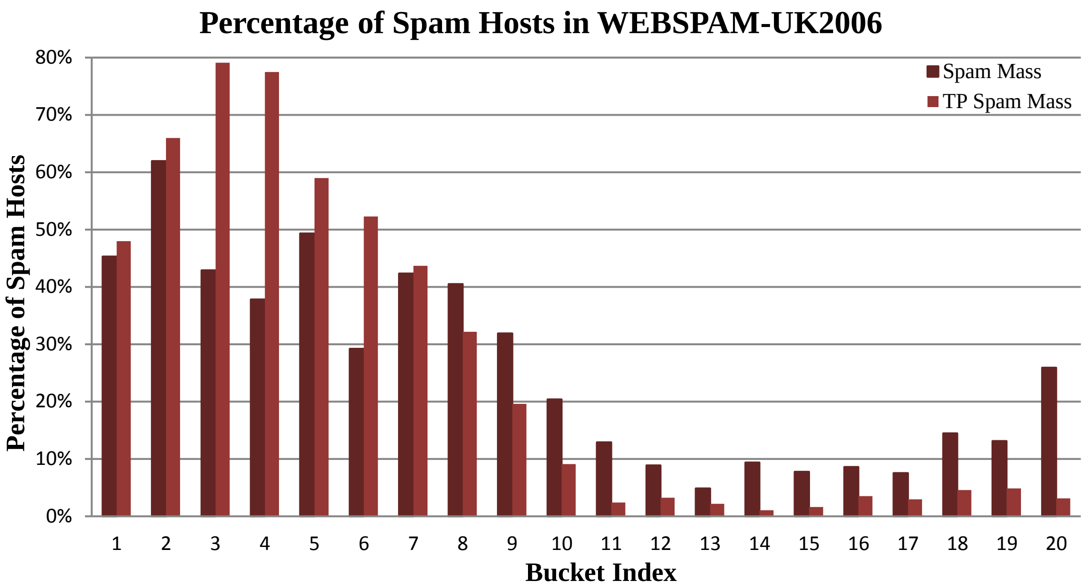


Figure 4.12: Percentage of spam hosts in WEBSPAM-UK2006

Apart from Web spam demotion algorithm, two Web spam detection algorithm – Spam Mass and TP Spam Mass are discussed. Figure 4.12 and 4.13 illustrates the percentage of spam hosts and the summation of all spam hosts on WEBSPAM-UK2006. Figure 4.14 and 4.15 illustrates the percentage of spam hosts and the summation of all spam hosts on WEBSPAM-UK2007.

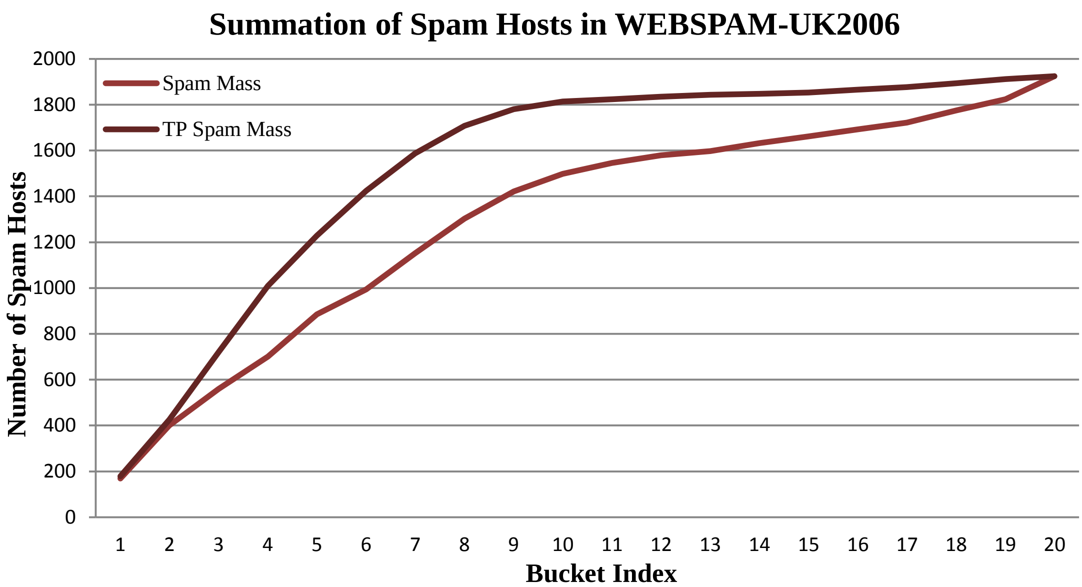


Figure 4.13: Summation of spam hosts in WEBSPAM-UK2006

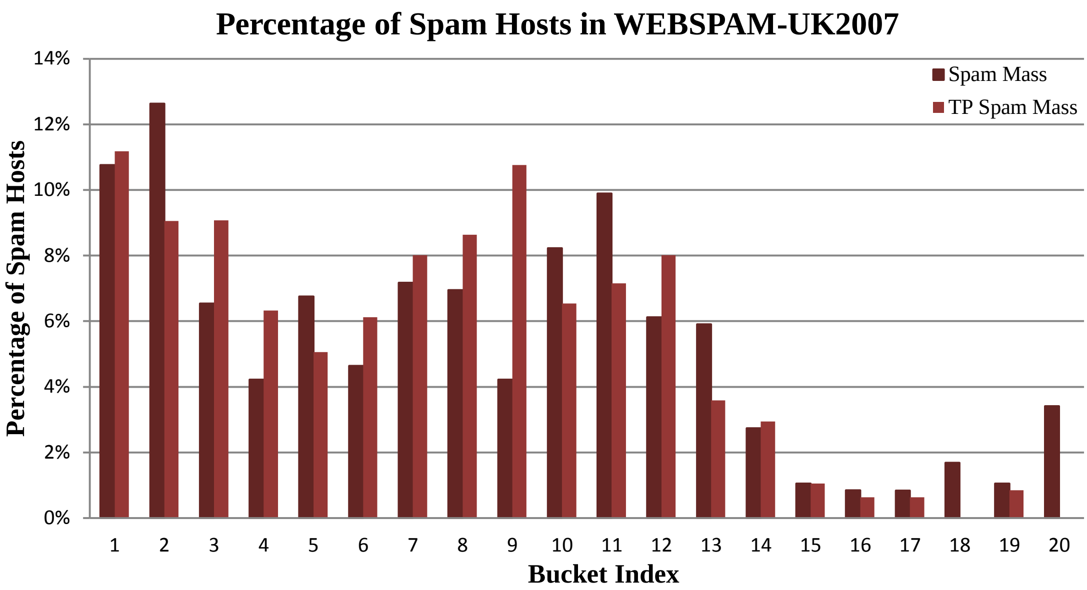


Figure 4.14: Percentage of spam hosts in WEBSPAM-UK2007

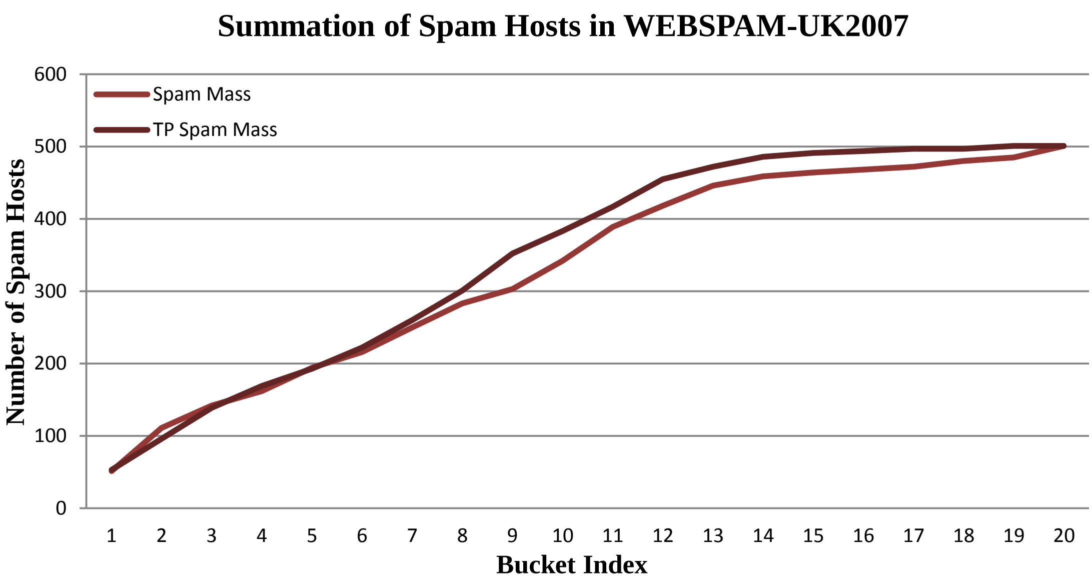


Figure 4.15: Summation of spam hosts in WEBSPAM-UK2007

In Figure 4.12, TP Spam Mass has shown that the algorithm detect more spam hosts than Spam Mass for the first seven buckets in WEBSPAM-UK2006 but for WEBSPAM-UK2007 showing in Figure 4.14, the result is not so clear for the reason that the spam seed set for the dataset is relatively small. However as shown in Figure 4.15, TP Spam Mass actually accumulate more spam hosts as the bucket moves further even though Spam Mass manages to detect more spam at the second bucket. As for WEBSPAM-UK2006 in Figure 4.13, TP Spam Mass manages to accumulate more spam hosts for all buckets compare to Spam Mass algorithm.

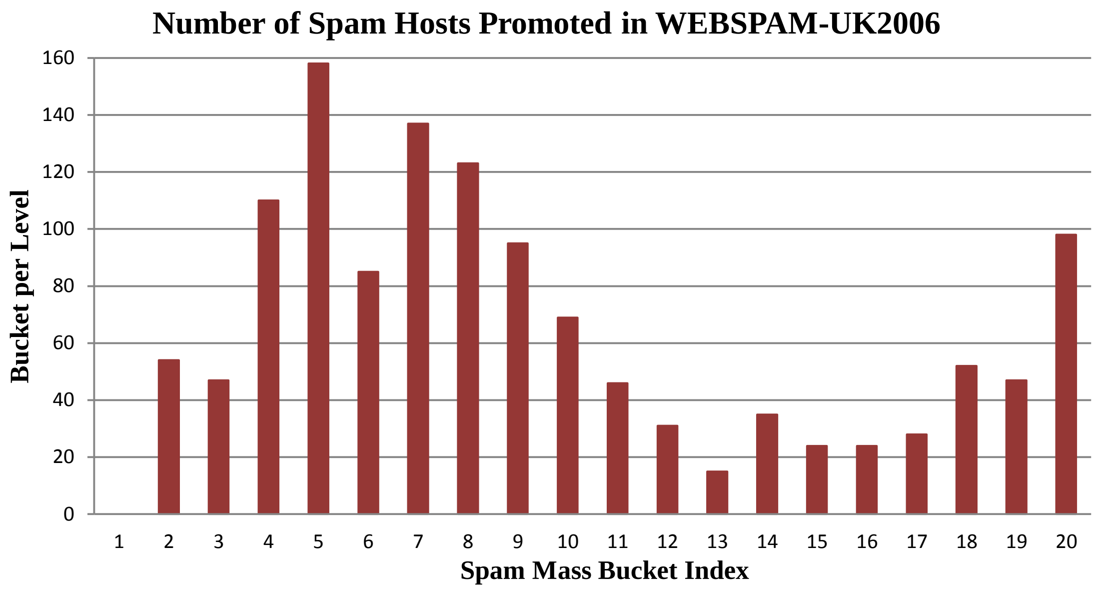


Figure 4.16: Average spam hosts promoted in WEBSPAM-UK2006

In Figure 4.16 and Figure 4.17, TP Spam Mass promotes as much as 10.38 bucket per level for spam host with the 5th bucket promoting 158 spam hosts which is an improvement of 42.36% on detection of Web Spam over Spam Mass algorithm in WEBSPAM-UK2006. For WEBSPAM-UK2007 showing in Figure 4.18 and Figure 4.19, TP Spam Mass able to promote up to 5.875 bucket per level in the last bucket and promotes 31 spam hosts at the 11th bucket.

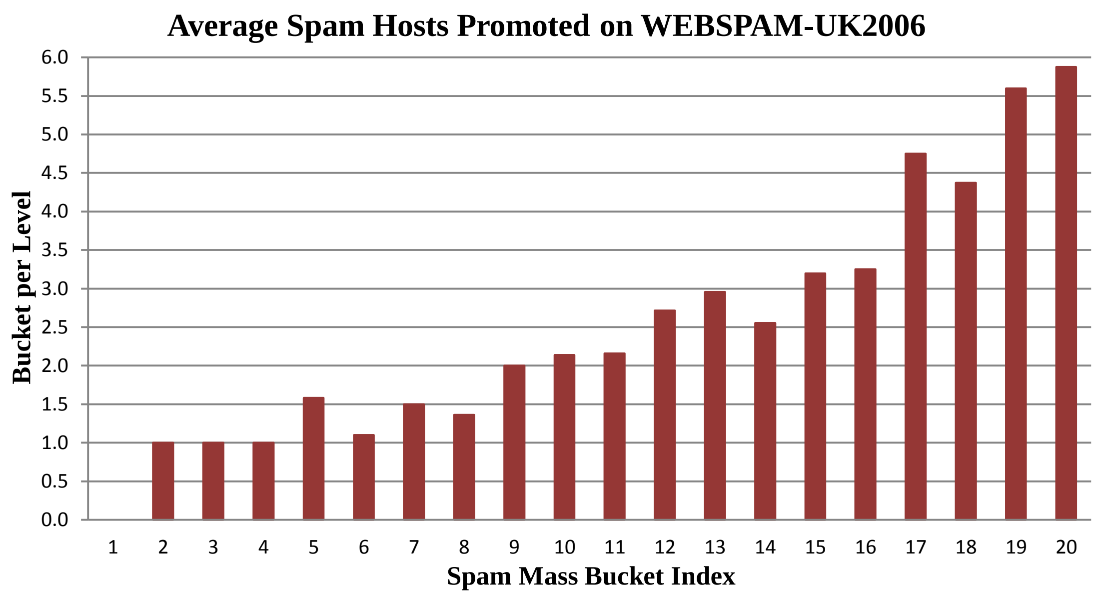


Figure 4.17: Number of spam hosts promoted in WEBSPAM-UK2006

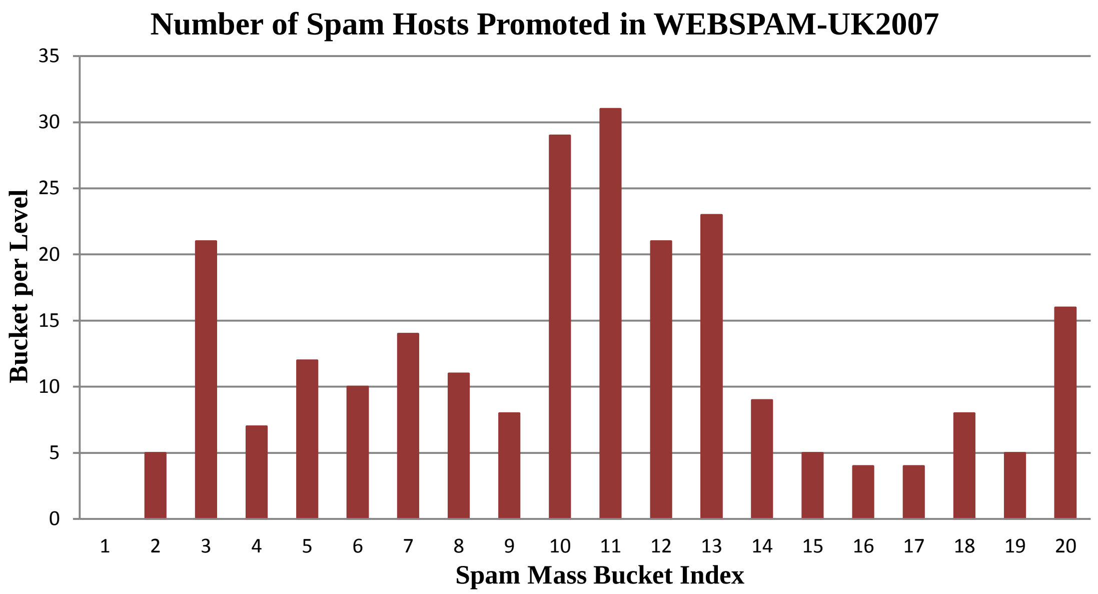


Figure 4.18: Average spam hosts promoted in WEBSPAM-UK2007

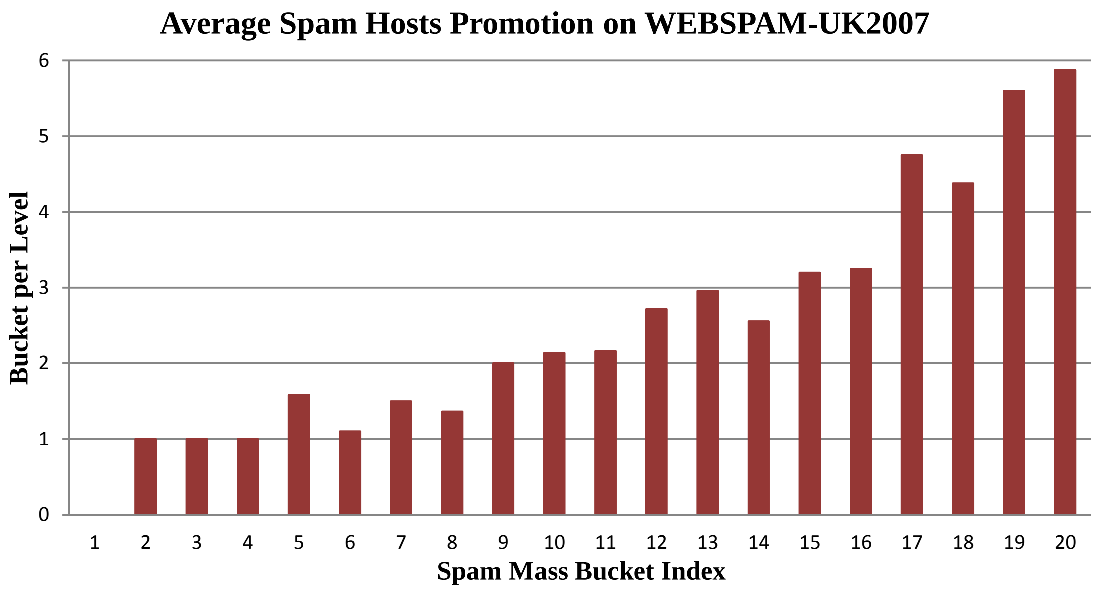


Figure 4.19: Number of spam hosts promoted in WEBSPAM-UK2007

Table 4-2: Number of Web pages represented from evaluated hosts in WEBSPAM-UK2006

WEBSPAM-UK2006				
Bucket Index	***Algorithms***			
	TrustRank	***TPRank***	***Spam Mass***	***TP Spam Mass***
1	5772732	5105405	151418	92751
2	9405003	8617492	1548226	1545822
3	12453809	11389610	2496185	3922350
4	15018597	13548654	2963797	5340260
5	17326257	15972170	3510497	6723305
6	19436425	18080304	3701269	7640583
7	21183616	19746985	4329449	8539854
8	23057903	21699476	4854187	9318171
9	24886572	23498479	5513986	9680861
10	26594747	25371805	5962484	9929897

Table 4-3: Number of Web pages represented from evaluated hosts in WEBSPAM-UK2007

WEBSPAM-UK2007				
Bucket Index	***Algorithms***			
	TrustRank	***TPRank***	***Spam Mass***	***TP Spam Mass***
1	5762322	5893117	5278	5288
2	9798875	9487775	6684	6436
3	12601227	12041690	7385	7286
4	15149195	14092532	8013	8866
5	16952324	16163390	9342	9310
6	18311165	17929892	9519	9532
7	19565437	19312567	16486	19081
8	20322920	20280246	19305	20145
9	21056879	20846941	20188	155152
10	21581252	21627362	63341	157259

Table 4-2 and 4-3 illustrate the number of Web pages represented from evaluated hosts in WEBSPAM-UK2006 and WEBSPAM-UK2007. The evaluated hosts are retrieved from Figure 4.13 and 4.15 where the summation of hosts is shown from the first bucket to the last bucket. However, only the top 10 buckets for evaluation are concerned. From the tables, it shows that TP Spam Mass actually detected more spam hosts than Spam Mass. However TPRank does not really outperform TrustRank in terms of number of Web pages represented from the evaluated hosts. It is believe that TrustRank choose the seeds with the largest propagation but TPRank propagates trust to the widest seeds, therefore TrustRank able to detect more Web pages compare to TPRank. However, TPRank still outperforms TrustRank in term of host level.

Table 4-4 illustrates the propagation coverage denote as Sn from the evaluated vertices υ_E, non-spam vertices υ_N and spam seeds υ_S; In addition, the percentage of trust that have propagated to non-spam and spam hosts are in the table.

Table 4-4: Propagation coverage in WEBSPAM-UK2006 and WEBSPAM-UK2007

Datasets	***Algorithms***	$Sn(\upsilon_E)$	$Sn(\upsilon_N)$	$Sn(\upsilon_S)$	η_N	η_S
WEBSPAM-UK2006	TrustRank	8564	4223	1374	86.20%	13.80%
	TPRank	10183	5242	1553	98.02%	1.98%
WEBSPAM-UK2007	TrustRank	73790	6603	242	98.98%	1.02%
	TPRank	95192	7900	353	99.54%	0.46%

$Sn(\upsilon_E)$ denotes the number of hosts that are covered from the seed set. $Sn(\upsilon_N)$ denotes the number of non-spam hosts while $Sn(\upsilon_S)$ denotes the number of spam hosts propagated. η_N denotes the percentage of trust propagated to non-spam hosts and η_S denotes the percentage of trust propagated to spam hosts. From Table 4-4, it has shown that TPRank has propagated trust to more non-spam hosts just so as spam hosts over TrustRank algorithm. Even though spam hosts are propagated more in

TPRank, the trust propagated to spam hosts are relatively small compare to TrustRank, 1.98% than 13.80% in WEBSPAM-UK2006 and 0.42% than 1.18% in WEBSPAM-UK2007 for the reason that TPRank actually propagate trust more towards non-spam hosts. The full results of all experiments in this chapter can refer to APPENDIX D - Chapter 4 Results.

Aside from TrustRank and Spam Mass, TPRank outperform T-Rank (Zhang, Wang, et al. 2011) and TP Spam Mass outperform LVTrustRank (Qi, Song-Nian, and Sisi 2008) in detection and demotion of Web spam on WEBSPAM-UK2007. The parameter settings are similar to this thesis; in the T-Rank experiments, T-Rank obtained around 100 spam sites but TPRank obtained 20 spam sites after demotion for the top five buckets; in the LVTrustRank experiments, the algorithm detects up to 6% for the first three buckets but TP Spam Mass detects at least 8% for the first three buckets with 2nd bucket detects 13% of spam sites.

The proposed trust propagation algorithm can be further improves existing link-based trust model algorithms such as Topical TrustRank (Wu, Goel, and Davison 2006b), Wu et al. trust algorithm (Wu, Goel, and Davison 2006a), DiffusionRank (Yang, King, and Lyu 2007), Nie et al. trust algorithm (Nie, Wu, and Davison 2007), LVTrustRank (Qi, Song-Nian, and Sisi 2008), QoC-QoL algorithm (Li, Qiancheng, and Yan 2008), AVRank & HVRank (Zhang et al. 2009), and T-Rank (Zhang, Wang, et al. 2011). By incorporating the proposed trust propagation algorithm into existing link-based trust model algorithms, little computation is needed (prove in next section) while the enhancement of Web spam demotion is achieved.

4.4 Computational Complexity

In terms of time complexity, assume a graph G where it consists of vertices υ and edges ε. The new trust score calculation checks all the connected vertices of all vertices, thus the operation costs $O(\upsilon+\varepsilon)$. In TrustRank, the algorithm just assigns the trust scores so therefore operate in of $O(\upsilon)$ time. The core operation in both algorithms is where for all vertices, all the incoming links of the vertices is checked; this operation cost $O(\upsilon+\varepsilon)$ in both algorithms. So in total time, in worst case both algorithms still run in $O(\upsilon+\varepsilon)$ time. Details on Big O notation can refer to APPENDIX A - Asymptotic Notation. For Spam Mass and TP Spam Mass, PageRank algorithm costs $O(\upsilon+\varepsilon)$, similar to the core operation in TrustRank and TPRank, while both TrustRank and TPRank cost $O(\upsilon+\varepsilon)$. So therefore for Spam Mass and TP Spam Mass, both algorithms in worst case operate in $O(\upsilon+\varepsilon)$ time.

4.5 SUMMARY

Various link-based Anti-Web spam techniques are constantly proposed in recent years. Trust Propagation Rank (TPRank) is proposed to demote Web spam and Trust Propagation Spam Mass (TP Spam Mass) to detect Web spam. The proposed algorithms are experimented on two large public available dataset WEBSPAM-UK2006 and WEBSPAM-UK2007, and have shown that the proposed algorithms outperform both TrustRank and Spam Mass in various measurements. TPRank has improved the detection rate over TrustRank up to 10.88% in WEBSPAM-UK2006 and up to 1.08% in WEBSPAM-UK2007. TP Spam Mass has improved the detection rate over Spam Mass up to 43.94% in WEBSPAM-UK2006 and up to 16.17% in WEBSPAM-UK2007. In terms of host to page level, TP Spam Mass has shown significant results compare to Spam Mass, for up to 106% improvement in WEBSPAM-UK2006 and 668% improvement in WEBSPAM-UK2007. In terms of propagation coverage, TPRank has also shown significant results as the algorithm has propagated to more trust scores to non-spam hosts compare to TrustRank. Even though it is slightly more computation in TPRank compare to TrustRank, the experiments have shown noteworthy results that both TPRank and TP Spam Mass are worthy in exchange for better performance

Chapter 5 Incorporating Weight Properties

5.1 INTRODUCTION

The weight properties in the Web model indicate the value of linkage between two unknown Web vertices. These weight properties have been exploited by other researchers to achieve better relevancy in query results for link analysis algorithms such as weighted PageRank (Xing and Ghorbani 2004; Nemirovsky and Avrachenkov 2008) and weighted HITS algorithm (Li, Shang, and Zhang 2002). Link spam, a broad class of Web spam on other hand, tries to attack link analysis algorithm by manipulating the linkages between vertices in the Web. Undoubtedly, there are some associates between weight properties in the Web model and link spamming. However, no research has been done correlating these two.

In this chapter, a novel metric is proposed based on weight properties to enhance the detection rate for distrust based Web spam detection algorithms. This metric calculates the weights based on outgoing links of the vertices which indicate the relevancy linkage between two vertices. The weights are used along with distrust based Web spam detection algorithms such as Anti-TrustRank (Krishnan and Raj 2006), Wu et al. Distrust algorithm (Wu, Goel, and Davison 2006a) and Nie et al. Distrust algorithm (Nie, Wu, and Davison 2007) to detect more spams. The experimental results have shown that by incorporating weight properties, it enhanced the detection rate by 30.25% for Anti-TrustRank, 12.14% for Wu et al. Distrust algorithm and 10.92% for Nie et al. Distrust algorithm in WEBSPAM-UK2006, and 31.30% for Anti-TrustRank, 26.38% for Wu et al. Distrust algorithm and 20.31% for Nie et al. Distrust in WEBSPAM-UK2007.

In most studies, the weight properties has been widely used to achieve better results for link analysis algorithms based on PageRank (Brinkmeier 2006) and their derivative. However in this work, the weight properties are incorporated for the purpose of detecting Web spam.

5.2 APPROACH

In this section, the seed selection, weight function and some modified algorithms along with the new weight function are discussed. In addition, some examples are provided to give an insight on the new weight function.

5.2.1 Seed Selection

The seed selection process for trust and distrust model Web spam algorithms either select spam seeds to propagate distrust or select non-spam seeds to propagate trust to filter Web spam.

In this research, Web spam detection algorithms are focused, in which spam seeds are crucial to propagate distrust to detect Web spam. According to Krishnan and Raj (Krishnan and Raj 2006), the seed selection algorithm that efficiently detects more spam with high PageRank is the PageRank algorithm (Brinkmeier 2006). High PageRank spam seeds travel in the reverse direction to detect additional high PageRank spam. Detection of high PageRank spam is important as the spam pages manipulate other Web pages easily. In this research, HostRank (Eiron, McCurley, and Tomlin 2004) is used rather than PageRank as the ranking mechanism is implemented at the host level. The HostRank algorithm is written as:

$$HR(p) = \alpha \cdot \sum_{(q,p)\in\varepsilon} \left(\frac{HR(q)}{\deg^{+}(q)} \right) + (1-\alpha) \cdot \frac{1}{N} \tag{5.1}$$

Where *HR(p)* is the HostRank result on host *p*, α is a decay factor, $O(q)$ is the number of outgoing links of host *q*. Top rank results are then evaluated as spam seed set and selected seeds are labelled as spam to form spam vector *B*, where,

$$B(p)=\begin{cases}1 & p\in \upsilon_S\\ 0 & otherwise\end{cases} \tag{5.2}$$

Spam vector *B* then normalized by,

$$\overline{B}=B/\|B\| \tag{5.3}$$

The normalized spam vector $\overline{B}$ is used later in both the original and modified version of Web spam detection algorithms to propagate distrust to detect more spam.

5.2.2 Weight Function

Assume a weighted graph is given; a weighted graph associates a label with every edge in the graph. The weights denote as the number of outgoing links between one host towards another host. The *computeOLweight* metric is introduced which compute and normalize the weight, the metric is written as:

$$\varpi_p=\sum_{(p,q)\in\varepsilon}\omega_{pq} \tag{5.4}$$

Where ϖ stands for the total weight for host *p*, $(p,q)\in\varepsilon$ denotes as there is a direct connection from host *p* to host *q*, ω_{pq} is a weight vector which denote as the number of pages from host *p* to host *q* of the weighted graph.

Let *T* represents the transition weight matrix of the graph, such that

$$T=\begin{array}{c} \\ \upsilon_1\\ \upsilon_2\\ \vdots\\ \upsilon_n\end{array}\begin{array}{c}\begin{array}{cccc}\upsilon_1 & \upsilon_2 & \cdots & \upsilon_n\end{array}\\ \begin{bmatrix}\omega_{11} & \omega_{12} & \cdots & \omega_{1n}\\ \omega_{21} & \omega_{22} & \cdots & \omega_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \omega_{n1} & \omega_{n2} & \cdots & \omega_{nn}\end{bmatrix}\end{array}$$

The row vector, r_i is then calculated where $i = 1,2,\ldots,m$ such that

$$\left.\begin{array}{l} r_1 = [\omega_{11} \quad \omega_{12} \quad \cdots \quad \omega_{1n}] \\ r_2 = [\omega_{21} \quad \omega_{22} \quad \cdots \quad \omega_{2n}] \\ \vdots \\ r_m = [\omega_{m1} \quad \omega_{m2} \quad \cdots \quad \omega_{mn}] \end{array}\right\} \tag{5.5}$$

Then, let the sum of elements for each of the row, r_m in matrix T applied into Equation 5.4 where *m* is the row index such that

$$\sum_{j=1}^{m} \omega_{mj}(p,q) = \omega_{m1} + \omega_{m2} + \cdots + \omega_{mn} = \varpi_m \tag{5.6}$$

Where ϖ is a scalar value.

$$O_{pq} = \frac{\omega_{pq}}{\varpi} \tag{5.7}$$

O_{pq} is the new weight which indicates the normalized

Note that, the transition matrix T is basically a matrix form of representation on each of the weight function in a weighted graph. Hence,

$$T = \omega(p,q). \tag{5.8}$$

Each row vectors are multiplied with the corresponding reciprocal of the summation for each row respectively in order to normalize the transition weight matrix to a transition weight matrix.

For instance,

$$\left.\begin{aligned}
O_1 &= r_1 \times \frac{1}{\varpi_1} = \begin{bmatrix}\omega_{11} & \omega_{12} & \cdots & \omega_{1n}\end{bmatrix}\frac{1}{\varpi_1} \\
O_2 &= r_2 \times \frac{1}{\varpi_2} = \begin{bmatrix}\omega_{21} & \omega_{22} & \cdots & \omega_{2n}\end{bmatrix}\frac{1}{\varpi_2} \\
&\vdots \\
O_n &= r_n \times \frac{1}{\varpi_n} = \begin{bmatrix}\omega_{n1} & \omega_{n2} & \cdots & \omega_{nn}\end{bmatrix}\frac{1}{\varpi_n}
\end{aligned}\right\} \tag{5.9}$$

The weight matrix O which is the normalized matrix such that

$$O = \begin{bmatrix} O_1 \\ O_2 \\ \vdots \\ O_n \end{bmatrix}$$

Hence,

$$O_{pq} = \frac{\omega(p,q)}{\varpi_p} \tag{5.10}$$

This weight gives us valuable information on how much one host is affecting another host. This approach is similar to the act of Web spamming, which is boosting one targeted page or host. In later section, the weight features along with the Web spam detection algorithms are experimented.

5.2.3 Algorithms

Let weighted host graph represented as $G_W = (\upsilon, \varepsilon, \omega)$, where υ is a set of vertices, ε is a set of edges and ω is a weight function that denotes the number of pages for each edge of the weighted directed graph G_W. The weight function mentioned in the previous sub-section is then applied onto the weighted host graph G_W to get the new weight O which denote as the outgoing links of one host to another. The weight is used to modify existing Web spam detection algorithm to enhance the Web spam detection. The Web spam detection algorithms that are presented here to modify and show comparisons are Anti-TrustRank (Krishnan and Raj 2006), Wu et al. distrust algorithm (Wu, Goel, and Davison 2006a) and Nie et al. distrust algorithm (Nie, Wu, and Davison 2007).

The principle of Anti-TrustRank is based on the intuition that pages that point to spam pages are likely to be spam pages themselves. Unlike TrustRank (Gyöngyi, Garcia-Molina, and Pedersen 2004), Anti-TrustRank travel in the reverse direction from a set of high PageRank spam seeds to detect more spam pages. The Anti-TrustRank algorithm can be seen in Equation 3.2. The algorithm is then modified by adding the weight and is written as:

$$WATR(p) = \alpha \cdot \sum_{p:(p:q)\in E_h} \left(\frac{WATR(q)}{\deg^{-}(q)} \cdot O_{pq} \right) + (1-\alpha) \cdot B(p) \qquad (5.11)$$

Where *WATR(p)* is the result of the weighted Anti-TrustRank algorithm of host *p*, α is a decay factor, $\deg^{-}(q)$ is the number of incoming links of host *q*, O_{pq} is the new weight function from host *p* to host *q* and *B(p)* is the spam vector.

Wu et al. (Wu, Goel, and Davison 2006a) proposed the combination of both trust and distrust to demote Web spam and experimented on three types of summation steps and two types of splitting steps for both trust and distrust, the summation steps are simple summation, maximum share and maximum parent while the splitting steps are constant splitting and logarithm splitting. The authors have shown that by combining the two propagations, it will improve the overall performance score. However, only the distrust is concerned as it is used to detect Web spam. The authors have shown that using maximum share for accumulation and logarithm splitting for splitting with constant *c* of 0.9 has the best performance for detecting Web spam. The Wu et al. distrust algorithm is written as:

$$DISTR(p) = \alpha \cdot c \cdot MaxShare\left[\sum_{\forall(p:q)\in G}\left(\frac{DISTR(q)}{\log(1+\deg^{-}(q))}\right)\right] + (1-\alpha)\cdot B(p) \tag{5.12}$$

Where *DISTR* stands for weighted Wu et al. distrust algorithm. *MaxShare* is a function that only takes the maximum distrust values from the children.

The best performance of Wu et al. distrust algorithm is modified and the resulting algorithm is:

$$WDISTR(p) = \alpha \cdot c \cdot MaxShare\left[\sum_{\forall(p:q)\in G}\left(\frac{WDISTR(q)}{\log(1+\deg^{-}(q))}\cdot O_{pq}\right)\right] + (1-\alpha)\cdot B(p) \tag{5.13}$$

Where *WDISTR* stands for weighted Wu et al. distrust algorithm.

The next Web spam detection algorithm is Nie et al. (Nie et al., 2007) algorithm. Similar with Wu et al. (Wu, Goel, and Davison 2006a), the authors use both trust and distrust propagation. The authors calculate the overall trust score by also including the subtraction of the distrust score. The authors found that using maximum share for accumulation and equal splitting for splitting actually achieves the best performance. The algorithm is written as:

$$Distrust(p) = \alpha \cdot MaxShare\left[\sum_{\forall(p:q)\in G}\left(\frac{Distrust(q)}{\deg^{-}(q)}\right)\right] + (1-\alpha)\cdot B(p) \tag{5.14}$$

Where *Distrust(p)* represent Nie et al. Distrust algorithm.

The best performance of Nie et al. distrust algorithm is modified by including the weight for the experiments. The algorithm can be written as:

$$WDistrust(p) = \alpha \cdot MaxShare\left[\sum_{\forall(p:q)\in G}\left(\frac{WDistrust(q)}{\deg^{-}(q)} \cdot O_{pq}\right)\right] + (1-\alpha)\cdot B(p) \quad (5.15)$$

Where *WDistrust* stands for Nie et al. distrust algorithm.

For the next sub-section, the algorithms are executed on a sample weighted Web graph as a simple example.

5.2.4 Example

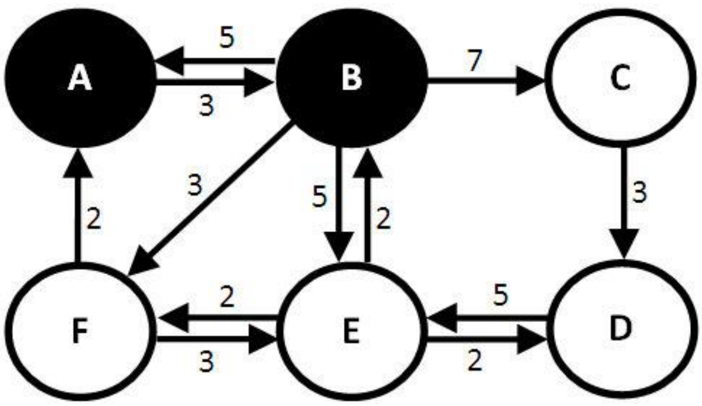


Figure 5.1: Sample weighted Web graph

Figure 5.1 illustrates a Web graph where $\upsilon_S = \{A, B\}$ and $\upsilon_N = \{C, D, E, F\}$. Initially, HostRank is applied to select the spam hosts. Assume that the jumping probability *j* is 0.85, running in 20 iterations, the HostRank results on Figure 5.1 are:

$$HR = [0.133 \quad 0.215 \quad 0.071 \quad 0.162 \quad 0.271 \quad 0.148]$$

From the result, top HostRank hosts are selected to evaluate. Assume that top three hosts are evaluated; the evaluated hosts $\upsilon_E = \{B, D, E\}$ where $\upsilon_S = \{B\}$ and

$\upsilon_N = \{D, E\}$. The spam vector B would give

$$B = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The spam vector B is now ready to apply into the Web spam detection algorithms. Before that, the new weight O is calculated for the modified algorithms. Below are the step based on the graph in Figure 5.1.

The transition weight matrix T of this figure is written as:

$$T = \begin{bmatrix} 0 & 3 & 0 & 0 & 0 & 0 \\ 5 & 0 & 7 & 0 & 5 & 3 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 2 & 0 & 2 & 0 & 2 \\ 2 & 0 & 0 & 0 & 3 & 0 \end{bmatrix}$$

The row vectors of the matrix T can be written as:

$$r_1 = \begin{bmatrix} 0 & 3 & 0 & 0 & 0 & 0 \end{bmatrix};\ r_2 = \begin{bmatrix} 5 & 0 & 7 & 0 & 5 & 3 \end{bmatrix};\ r_3 = \begin{bmatrix} 0 & 0 & 0 & 3 & 0 & 0 \end{bmatrix};$$

$$r_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 5 & 0 \end{bmatrix}; r_5 = \begin{bmatrix} 0 & 2 & 0 & 2 & 0 & 2 \end{bmatrix}; r_6 = \begin{bmatrix} 2 & 0 & 0 & 0 & 3 & 0 \end{bmatrix}.$$

Then the summation of each row can be written as:

$$\varpi_1 = \sum r_1 = 3;\ \varpi_2 = \sum r_2 = 20;\ \varpi_3 = \sum r_3 = 3;$$

$$\varpi_4 = \sum r_4 = 5;\ \varpi_5 = \sum r_5 = 6;\ \varpi_6 = \sum r_6 = 5.$$

The weight then can be computed as:

$$O = \begin{bmatrix} O_1 \\ O_2 \\ O_3 \\ O_4 \\ O_5 \\ O_6 \end{bmatrix} = \begin{bmatrix} r_1 \times \dfrac{1}{\varpi_1} \\ r_2 \times \dfrac{1}{\varpi_2} \\ r_3 \times \dfrac{1}{\varpi_3} \\ r_4 \times \dfrac{1}{\varpi_4} \\ r_5 \times \dfrac{1}{\varpi_5} \\ r_6 \times \dfrac{1}{\varpi_6} \end{bmatrix} = \begin{bmatrix} 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0.25 & 0 & 0.35 & 0 & 0.25 & 0.15 \\ 0 & 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0.33 & 0 & 0.33 & 0 & 0.33 \\ 0.4 & 0 & 0 & 0 & 0.6 & 0 \end{bmatrix}$$

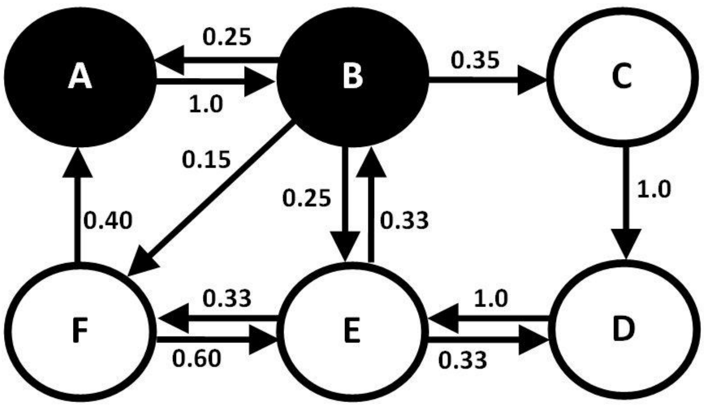


Figure 5.2: Sample weighted Web graph after computeOLweight metric

The normalized weight is also shown in Figure 5.2.

After the spam vector B and new weight O are calculated, both spam vector and the new weight are applied into both original and weighted Web spam detection algorithms individually. Assume that the jumping probability is 0.85 and run in 20 iterations, the results for the algorithms are:

Table 5-1: Host results on different Web spam detection algorithms

Web Spam Detection Algorithms	***Hosts***					
	A	***B***	***C***	***D***	***E***	***F***
Anti-TrustRank	0.156	0.368	0.029	0.069	0.243	0.135
Wu et al. Distrust	0.197	0.286	0.075	0.108	0.197	0.137
Nie et al. Distrust	0.204	0.337	0.050	0.082	0.204	0.123
Weighted Anti-TrustRank	0.268	0.363	0.049	0.066	0.135	0.119
Weighted Wu et al. Distrust	0.305	0.287	0.091	0.085	0.101	0.131
Weighted Nie et al. Distrust	0.312	0.367	0.050	0.059	0.104	0.108

Table 5-1 shows the results of both the original and weighted Web spam detection algorithms. Observed from the table, all the algorithms gave host *B* the highest rank as it is a spam host. However, not all algorithms able to detect host *A* as a spam host. The closest original algorithms are Wu et al. distrust and Nie et al. distrust as both these algorithms gave the same rank values for host *A* and host *E*. With the computed weight, the modified version of the algorithms able to give a high rank to host *A* and host *B* for the reason that host *A* is performing link exchange with spam host *B* only.

5.3 EXPERIMENTAL RESULTS

In this section, the proposed algorithms are experimented on two public available datasets – WEBSPAM-UK2006 and WEBSPAM-UK2007 (See Chapter 2 for more details on datasets). The percentages of spam hosts, summation of spam hosts, average spam host promotion, number of spam hosts promoted and number of Web pages represented from the evaluated spam hosts are shown in the experiments.

Figure 5.3 to Figure 5.8 illustrate the percentage of spam hosts for Anti-TrustRank versus Weighted Anti-TrustRank, Wu et al. Distrust versus weighted Wu et al. Distrust, Nie et al. Distrust vs. weighted Nie et al. Distrust in WEBSPAM-UK2006 and WEBSPAM-UK2007.

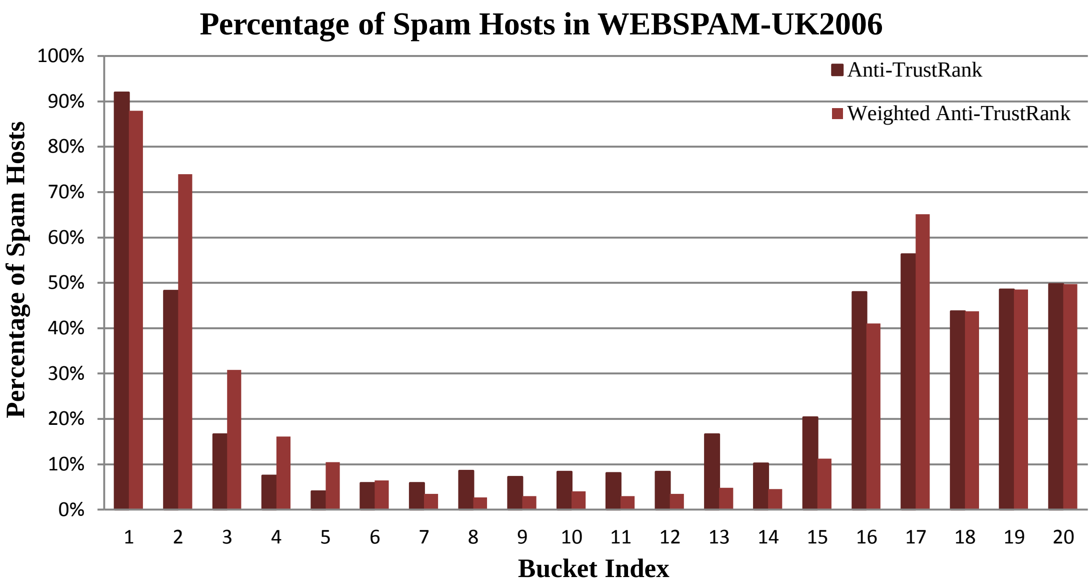


Figure 5.3: Percentage of spam hosts for Anti-TrustRank and weighted Anti-TrustRank in WEBSPAM-UK2006.

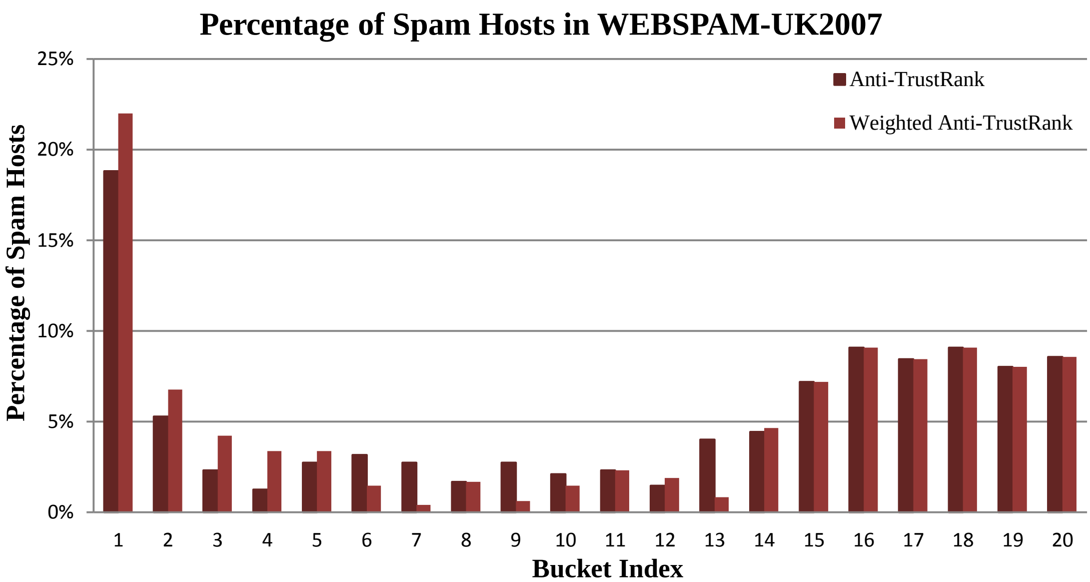


Figure 5.4: Percentage of spam hosts for Anti-TrustRank and weighted Anti-TrustRank in WEBSPAM-UK2007.

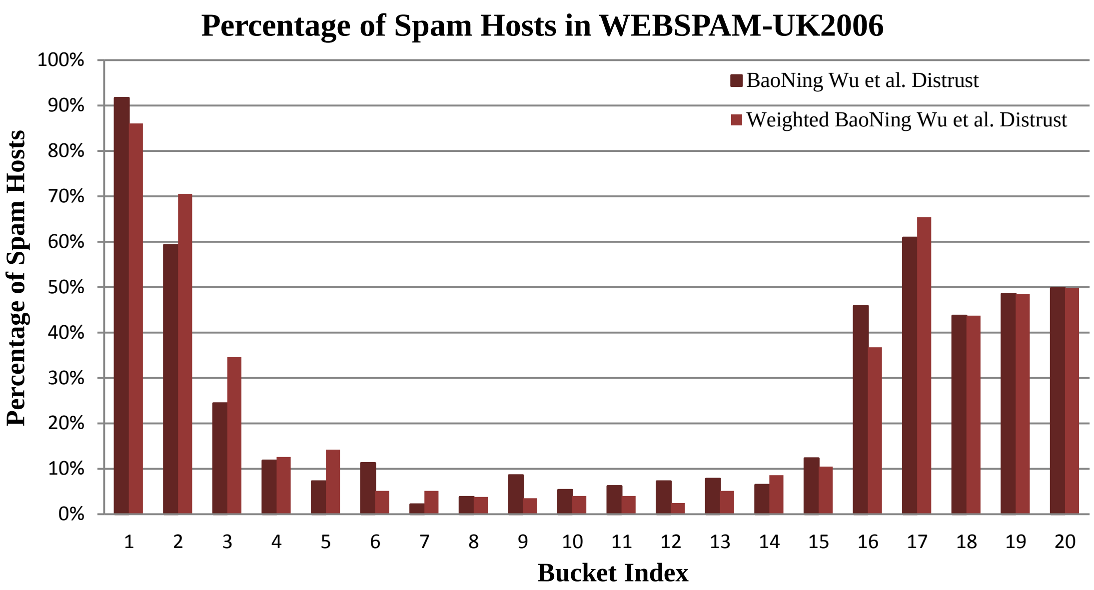


Figure 5.5: Percentage of spam hosts for Wu et al. Distrust and weighted Wu et al. Distrust in WEBSPAM-UK2006.

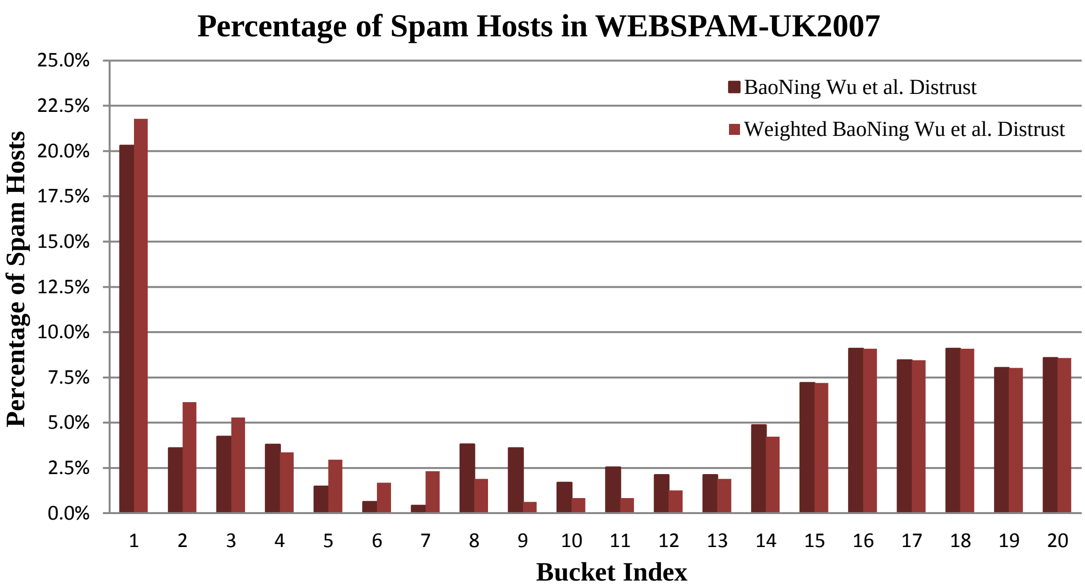


Figure 5.6: Percentage of spam hosts for Wu et al. Distrust and weighted Wu et al. Distrust in WEBSPAM-UK2007.

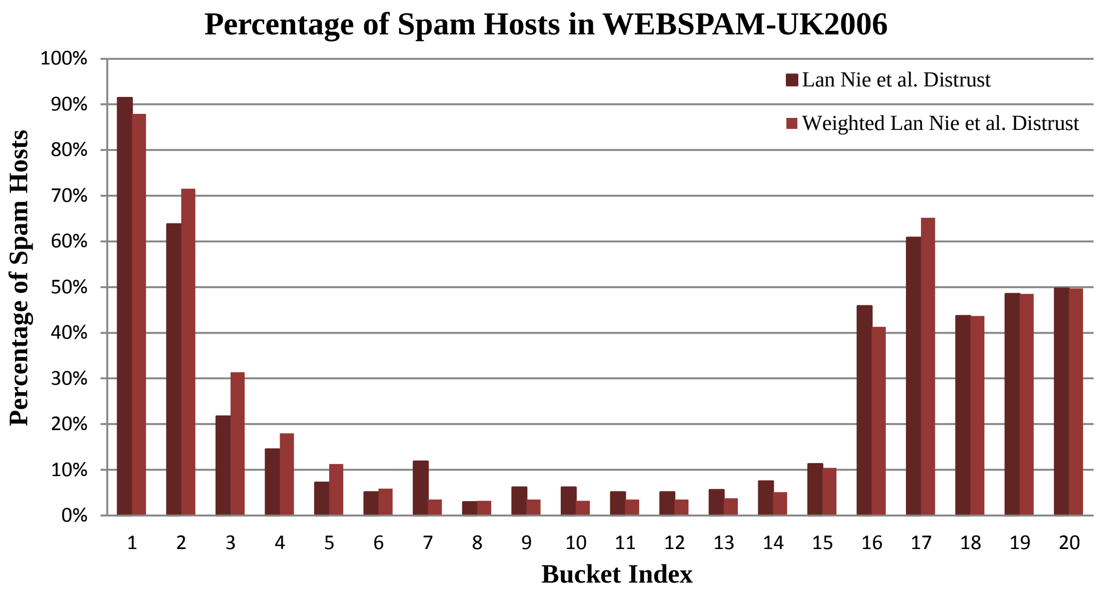


Figure 5.7: Percentage of spam hosts for Nie et al. Distrust vs weighted Nie et al. Distrust in WEBSPAM-UK2006.

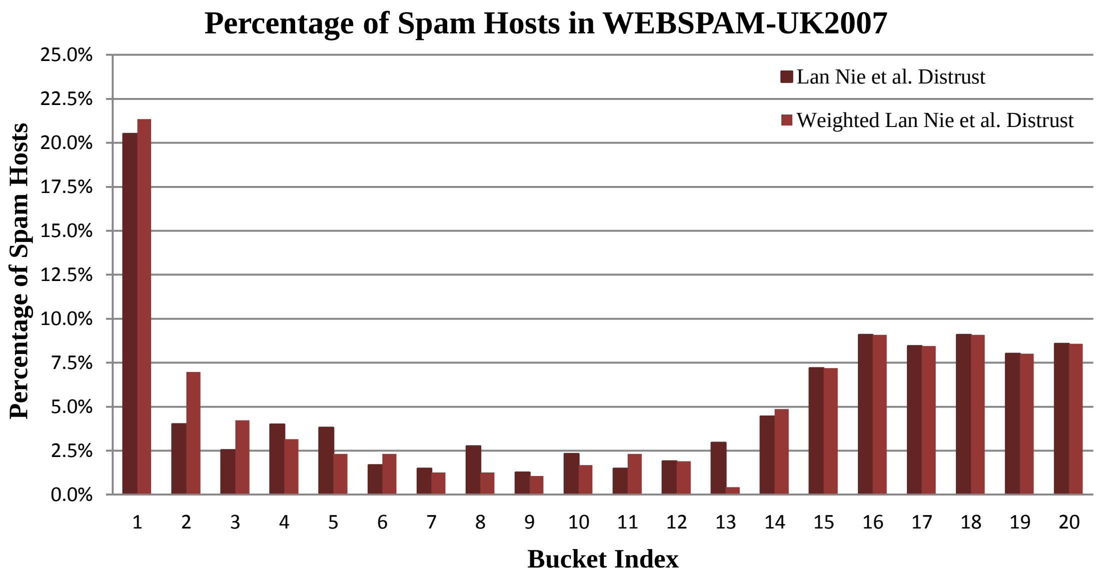


Figure 5.8: Percentage of spam hosts for Nie et al. Distrust vs weighted Nie et al. Distrust in WEBSPAM-UK2007.

The figures (Figure 5.3 to Figure 5.8) under the same dataset show similar patterns. In WEBSPAM-UK2006, the benchmark algorithm works slightly better than the weighted one for the first bucket but from the second bucket to the fifth bucket, the weighted algorithms works better than the benchmark. In WEBSPAM-UK2007, weighted Anti-TrustRank managed to detect more spam host for the first five bucket than Anti-TrustRank. However, weighted Wu et al. Distrust and weighted Lan Nie et al Distrust algorithms managed to detect more spam host for the first three bucket than the benchmark.

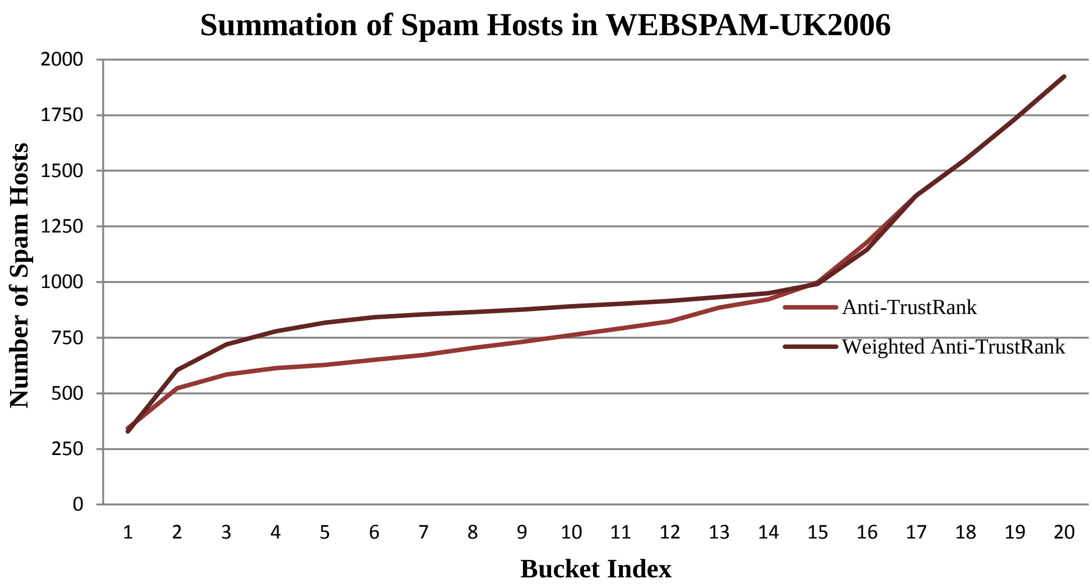


Figure 5.9: Summation of spam hosts for Anti-TrustRank and weighted Anti-TrustRank in WEBSPAM-UK2006.

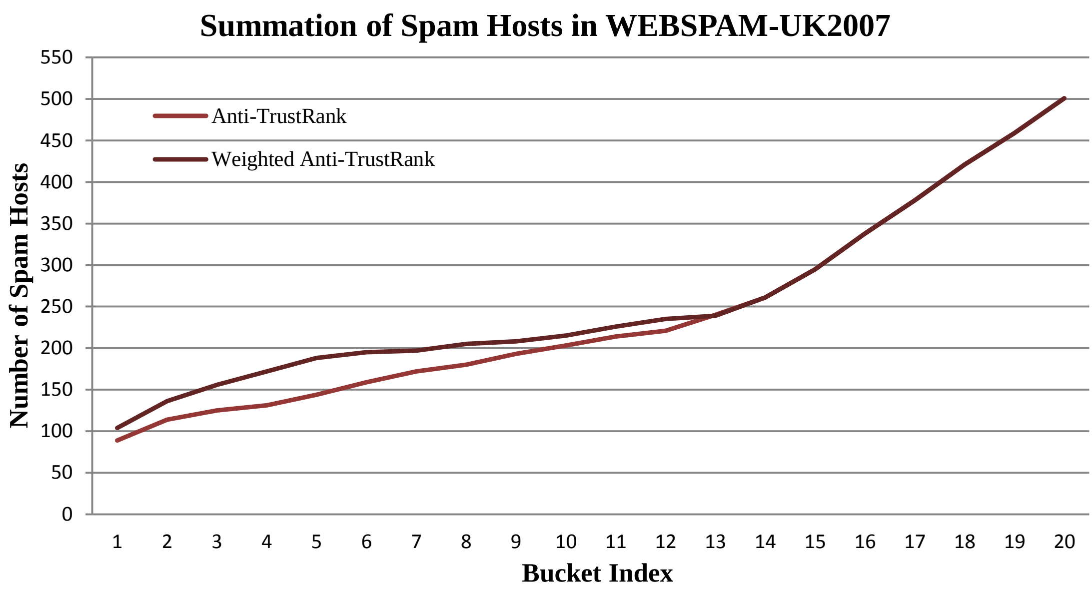


Figure 5.10: Summation of spam hosts for Anti-TrustRank and weighted Anti-TrustRank in WEBSPAM-UK2007.

Figure 5.9 to Figure 5.14 illustrate the summation of spam hosts for Anti-TrustRank versus weighted Anti-TrustRank, Wu et al. Distrust versus weighted Wu et al. Distrust, Nie et al. Distrust vs. weighted Nie et al. Distrust in WEBSPAM-UK2006 and WEBSPAM-UK2007.

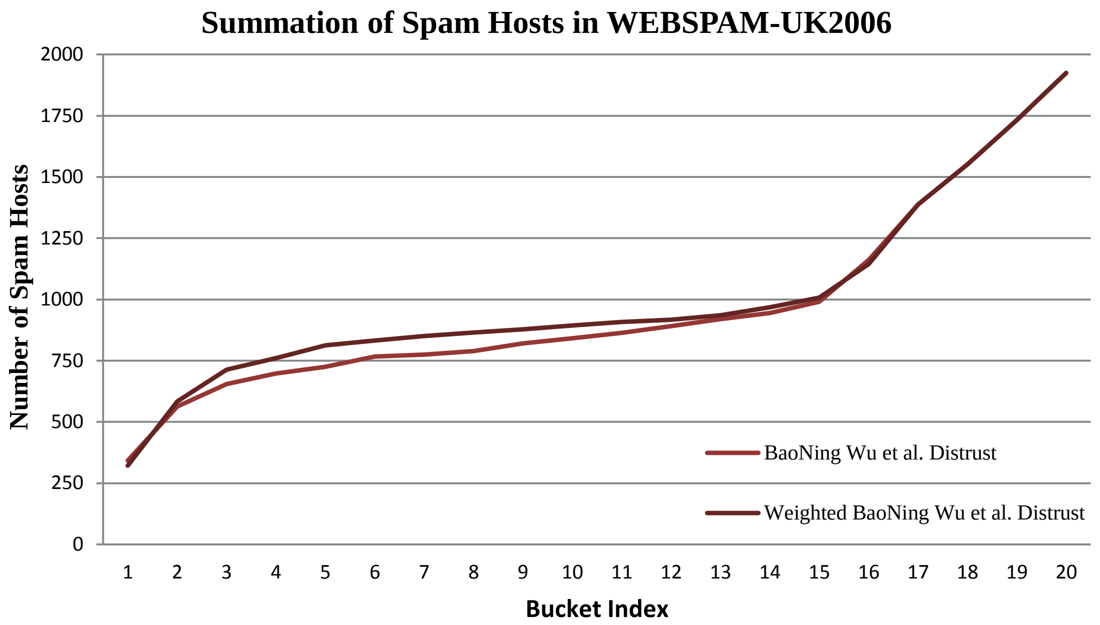


Figure 5.11: Summation of spam hosts for Wu et al. Distrust and weighted Wu et al. Distrust in WEBSPAM-UK2006.

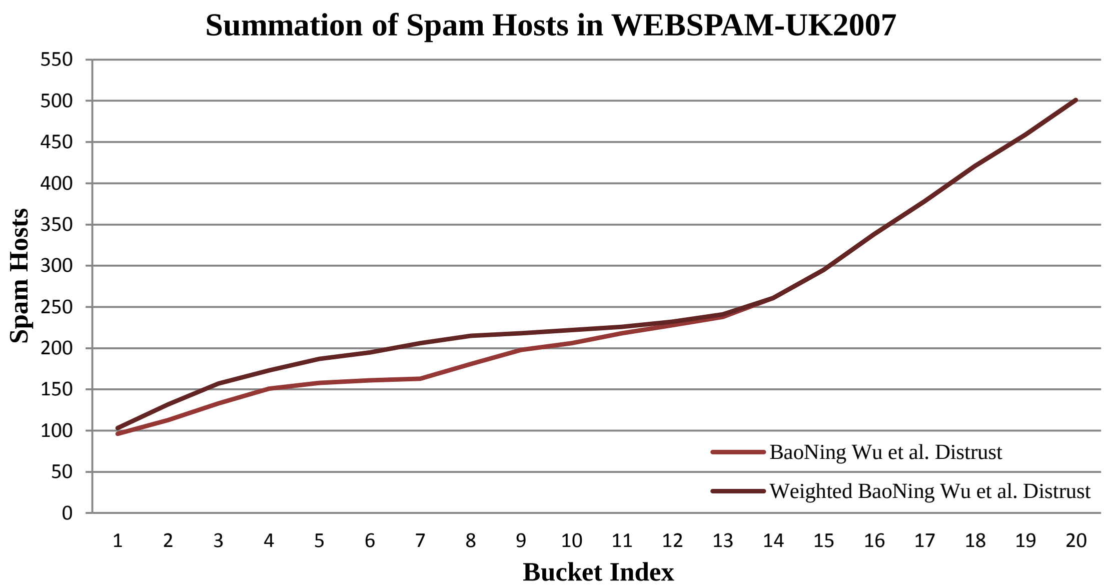


Figure 5.12: Summation of spam hosts for Wu et al. Distrust and weighted Wu et al. Distrust in WEBSPAM-UK2007.

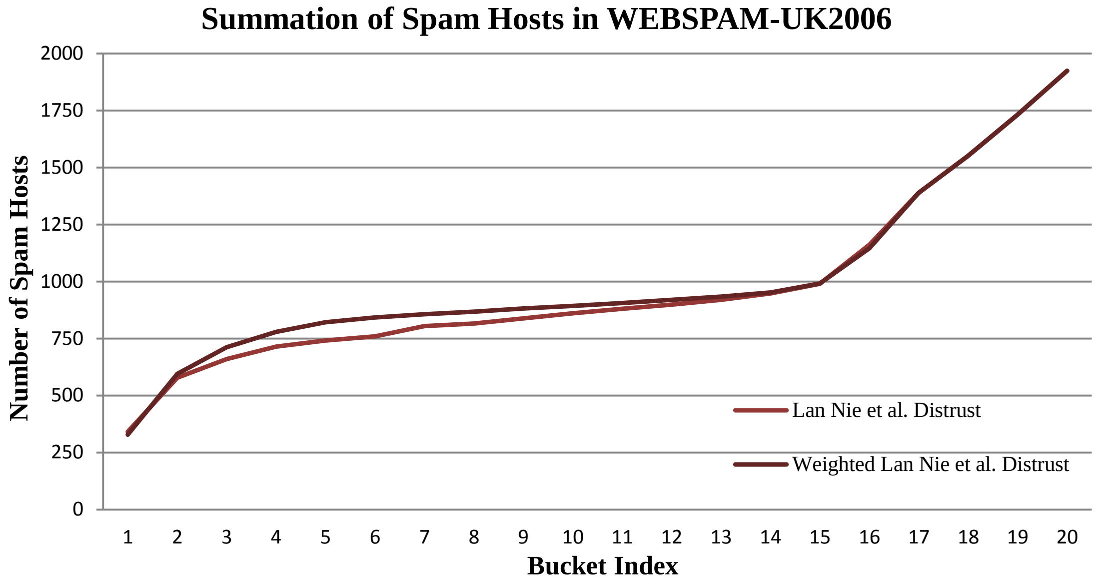


Figure 5.13: Summation of spam hosts for Nie et al. Distrust and weighted Nie et al. Distrust in WEBSPAM-UK2006.

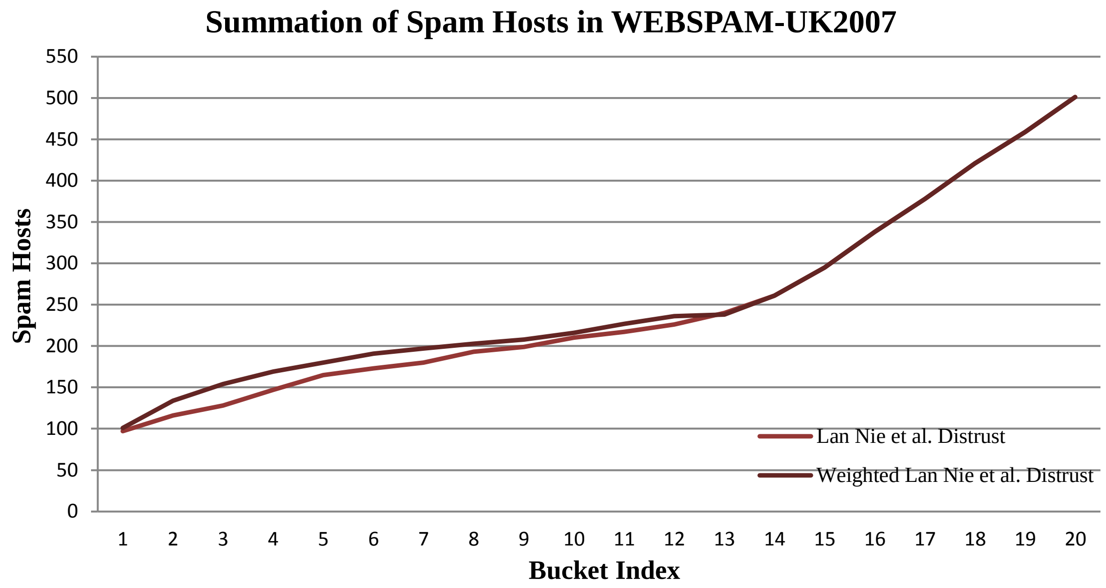


Figure 5.14: Summation of spam hosts for Nie et al. Distrust and weighted Nie et al. Distrust in WEBSPAM-UK2007.

Figure 5.9 to Figure 5.14 is highly correlated to the previous figure - Figure 5.3 to Figure 5.8 as the figures show the summation of each bucket for various algorithms in the two datasets. Observed from figure Figure 5.9 to Figure 5.14, all the weighted algorithms accumulated more spam hosts until it reached the same point at 15th bucket in WEBSPAM-UK2006 and 13th bucket in WEBSPAM-UK2007. The biggest gap for the summation of spam hosts in WEBSPAM-UK2006 is Anti-TrustRank versus weighted Anti-TrustRank (showing in Figure 5.9) with 192 spam hosts difference in the 6th bucket. On the other hand, the biggest gap for summation of spam hosts in WEBSPAM-UK2007 is also Anti-TrustRank versus weighted Anti-TrustRank (showing in Figure 5.10) with 44 spam hosts difference in the 5th bucket.

Figure 5.15 to Figure 5.18 show the average spam hosts promotion and number of spam hosts being promoted for weight Anti-TrustRank over Anti-TrustRank in the two datasets.

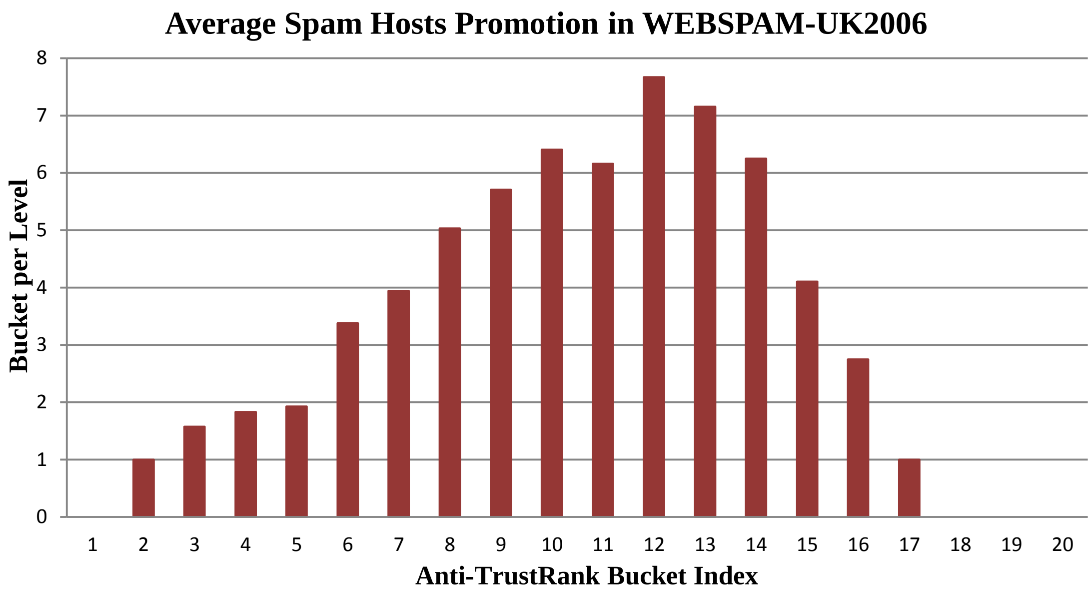


Figure 5.15: Average spam hosts promotion for weighted Anti-TrustRank over Anti-TrustRank in WEBSPAM-UK2006.

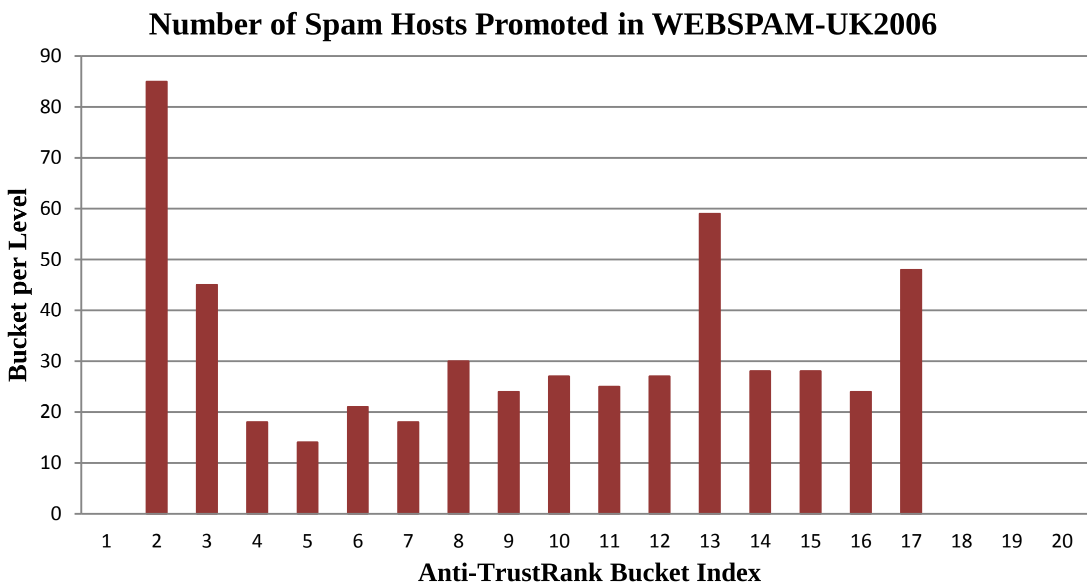


Figure 5.16: Number of spam hosts promoted for weighted Anti-TrustRank over Anti-TrustRank in WEBSPAM-UK2006.

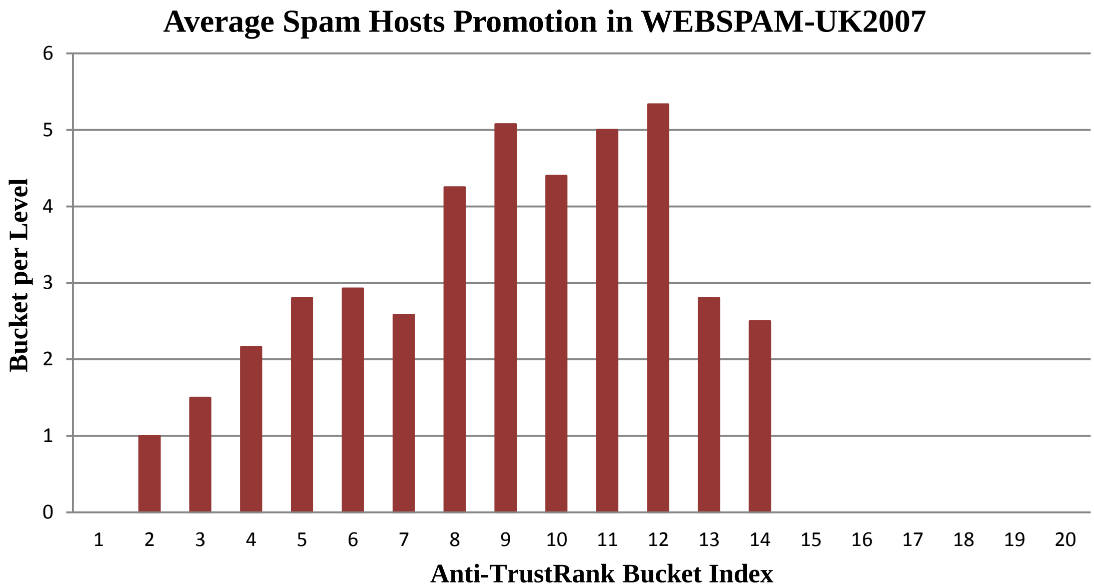


Figure 5.17: Average spam hosts promotion for weighted Anti-TrustRank over Anti-TrustRank in WEBSPAM-UK2007.

In WEBSPAM-UK2006, the highest average spam host promotion came from 12th bucket with 7.67 bucket per level. However, the most spam hosts that a bucket promoted is the 2nd bucket where it promotes 85 spam hosts. In WEBSPAM-UK2007, a sum of 113 spam hosts being promoted and the largest number of spam hosts being promoted is the 13th bucket with 15 spam hosts promoted.

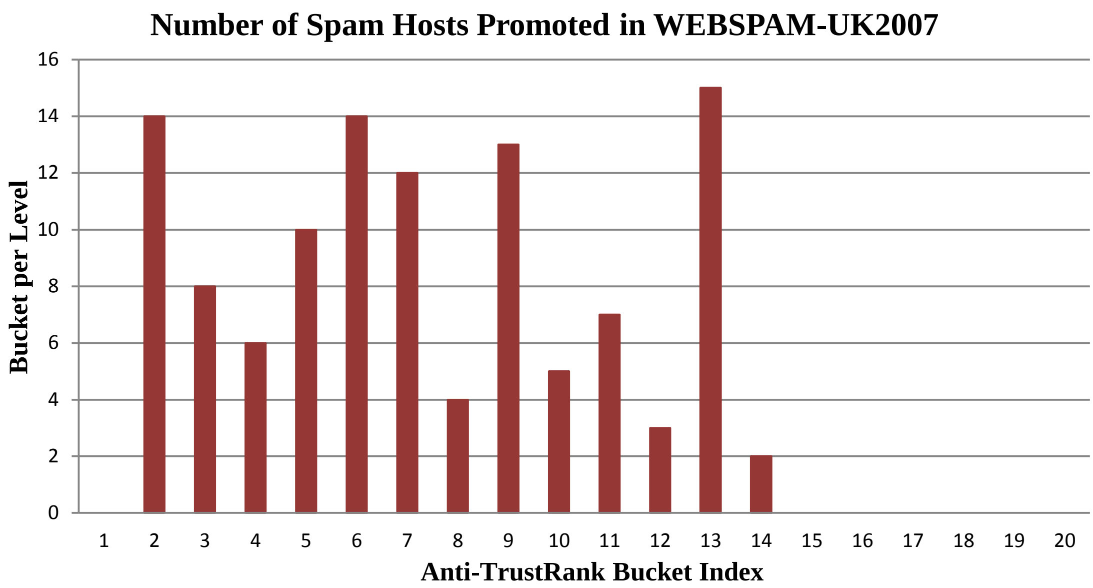


Figure 5.18: Number of spam hosts promoted for weighted Anti-TrustRank over Anti-TrustRank in WEBSPAM-UK2007.

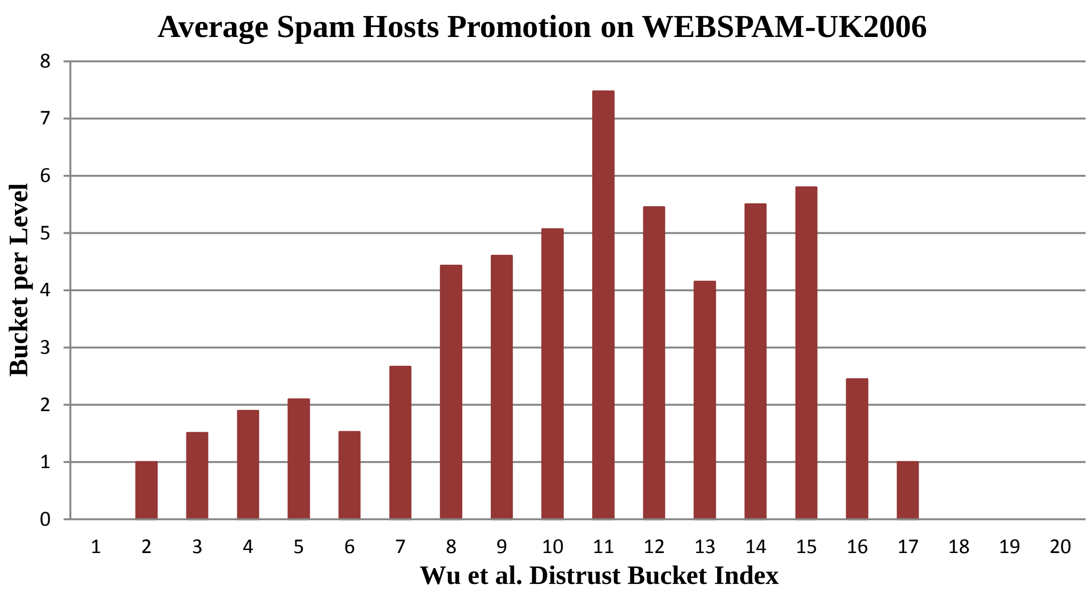


Figure 5.19: Average spam hosts promotion for weighted Wu et al. Distrust over Wu et al. Distrust in WEBSPAM-UK2006.

Figure 5.19 to Figure 5.22 illustrate the average spam hosts promotion and number of spam hosts being promoted for weighted Wu et al. Distrust over the benchmark Wu et al. Distrust in WEBSPAM-UK2006 and WEBSPAM-UK2007 datasets.

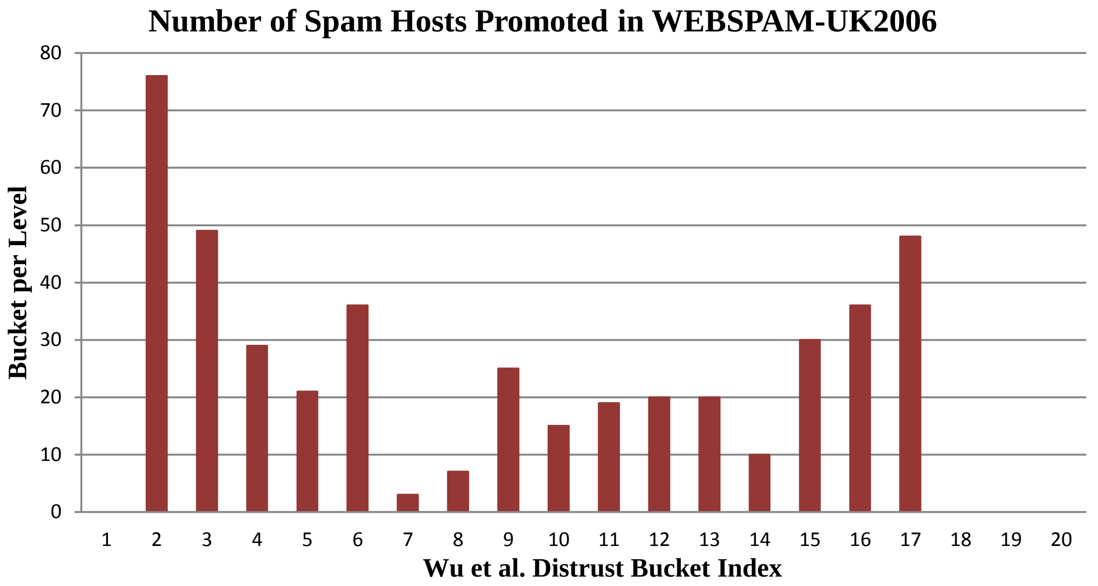


Figure 5.20: Number of spam hosts promoted for weighted Wu et al. Distrust over Wu et al. Distrust in WEBSPAM-UK2006.

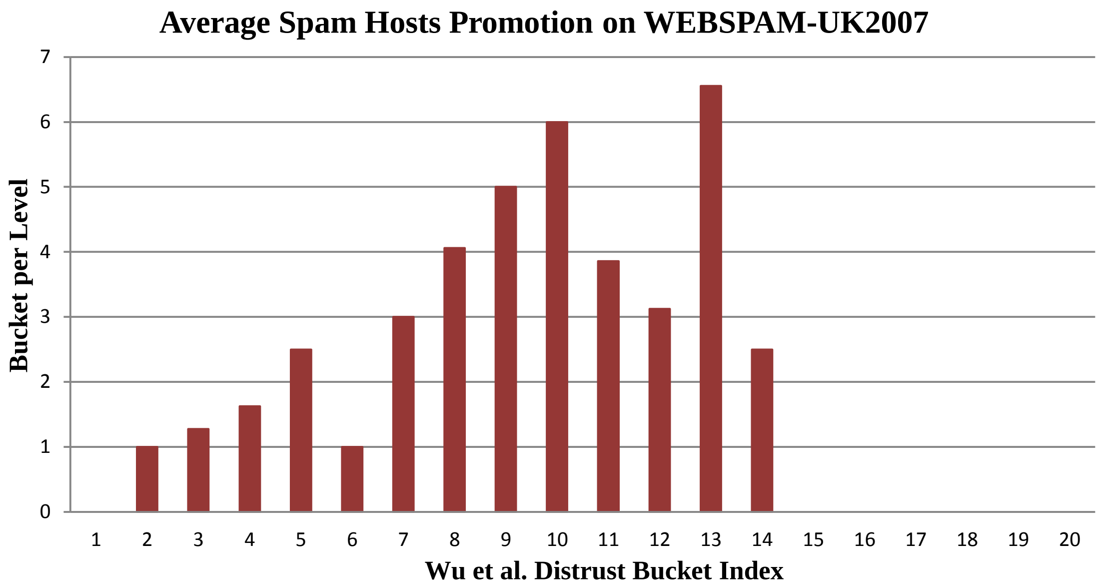


Figure 5.21: Average spam hosts promotion for weighted Wu et al. Distrust over Wu et al. Distrust in WEBSPAM-UK2007.

In WEBSPAM-UK2006, the highest average spam host promotion bucket goes to 11th bucket with bucket per level of 7.47 but the highest number of spam hosts bucket is the 2nd bucket with 76 spam hosts. On the other hand in WEBSPAM-UK2007, the bucket with the largest average spam host promotion is 13th bucket with average promotion of

6.55 bucket per level. The total sum of spam host being promoted in WEBSPAM-UK2007 by weighted Wu et al. Distrust is 101 spam hosts.

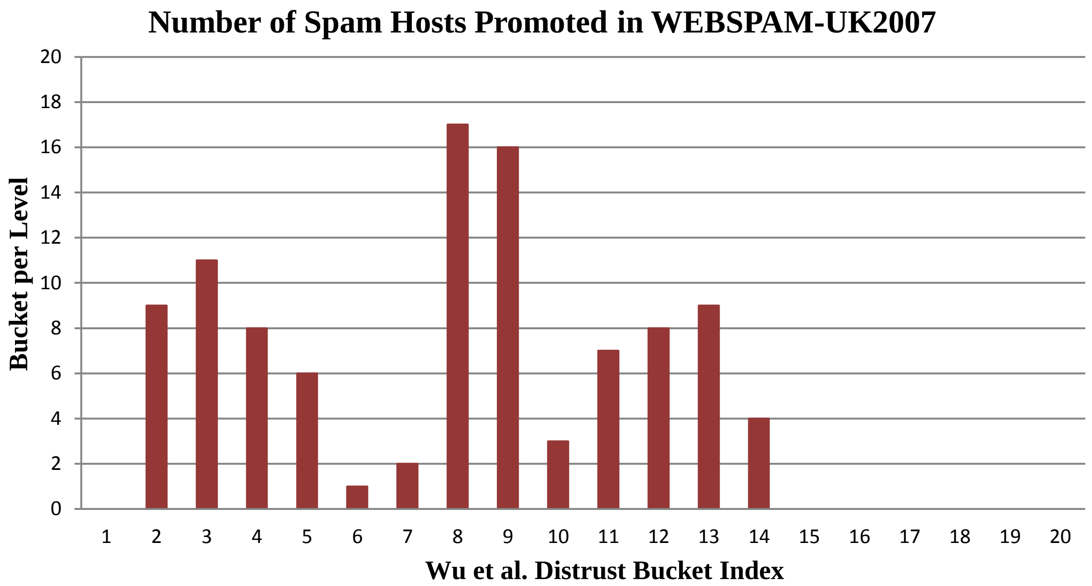


Figure 5.22: Number of spam hosts promoted for weighted Wu et al. Distrust over Wu et al. Distrust in WEBSPAM-UK2007.

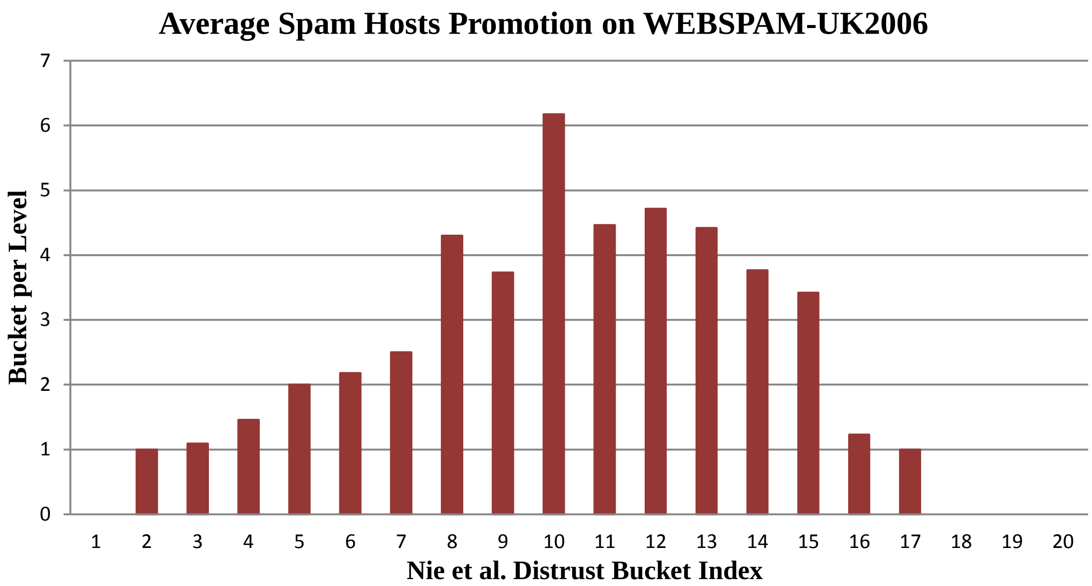


Figure 5.23: Average spam hosts promotion for weighted Nie et al. Distrust over Nie et al. Distrust in WEBSPAM-UK2006.

Figure 5.23 to Figure 5.26 illustrate weighted Nie et al. Distrust over Nie et al. Distrust in term of average spam hosts promotion and number of spam hosts promoted in WEBSPAM-UK2006 and WEBSPAM-UK2007.

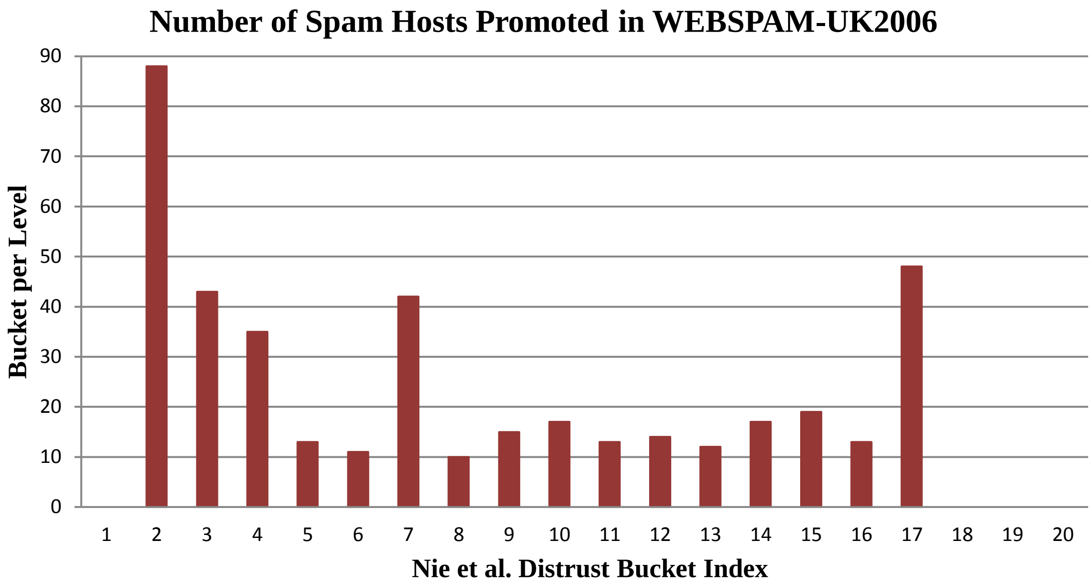


Figure 5.24: Number of spam hosts promoted for weighted Nie et al. Distrust over Nie et al. Distrust in WEBSPAM-UK2006.

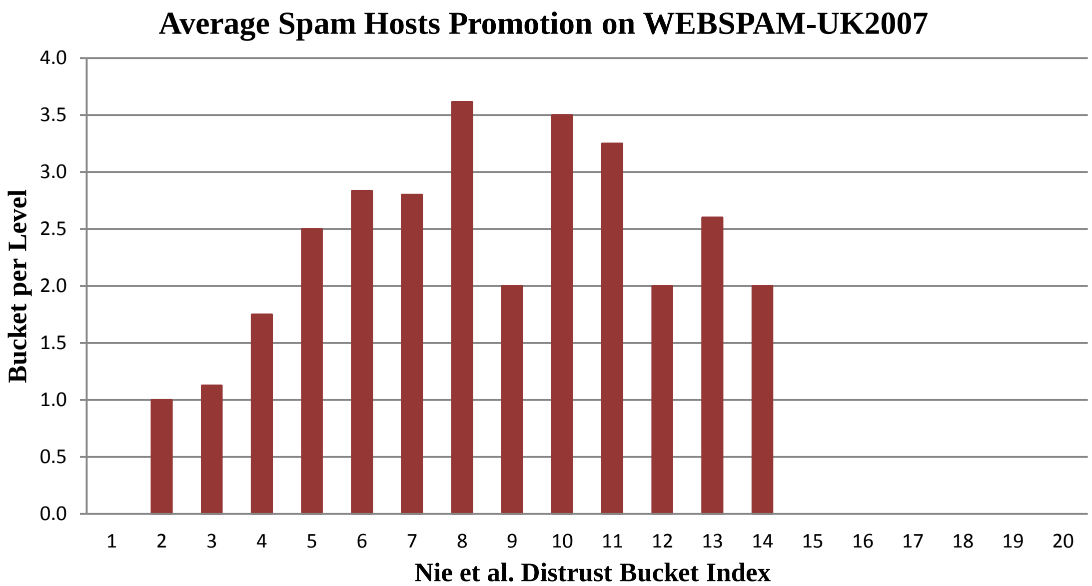


Figure 5.25: Average spam hosts promotion for weighted Nie et al. Distrust over Nie et al. Distrust in WEBSPAM-UK2007.

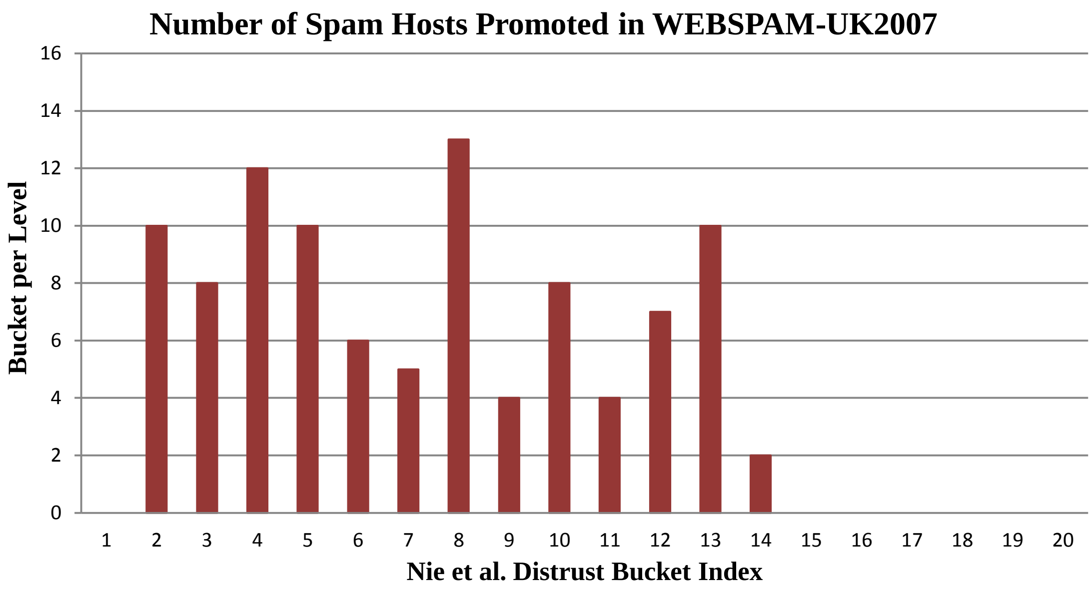


Figure 5.26: Number of spam hosts promoted for weighted Nie et al. Distrust over Nie et al. Distrust in WEBSPAM-UK2006.

For WEBSPAM-UK2006, weighted Nie et al. Distrust has the highest average spam host promotion over the benchmark algorithm at the 10^{th} bucket with bucket per level of 6.18. However the bucket that promotes the most spam hosts is the 2^{nd} bucket, just like weighted Anti-TrustRank and weighted Wu et al. Distrust over their benchmark algorithms. For WEBSPAM-UK2007, similar to previous weighted algorithms, there is almost no spam host being promoted at the last few buckets. This is because the amount of spam hosts is not large enough compare to the WEBSPAM-UK2007 dataset. Despite of this, the weighted algorithm promoted up to 3.62 bucket per level and promotes up to 13 spam hosts for each bucket.

Table 5-2: Number of Web pages represented from evaluated hosts in WEBSPAM-UK2006

WEBSPAM-UK2006						
Bucket Index	***Algorithms***					
	ATR	***WATR***	***DISTR***	***WDISTR***	***Distrust***	***WDistrust***
1	1313291	1349790	1622823	1295424	1653512	1273941
2	2456555	2901785	2728892	2613922	2822220	2872809

(Table 5-2 continued)

3	2841133	3780350	3336290	3681108	3438744	3821501
4	3063894	4282150	3739390	4124931	3929221	4296223
5	3204105	4461991	3945059	4454984	3995253	4448293
6	3370887	4570601	4049277	4546177	4166130	4566529
7	3508984	4615278	4094948	4632867	4282662	4639370
8	3754869	4727802	4282774	4749903	4369946	4694204
9	4061543	4792877	4506142	4806178	4445917	4797769
10	4214193	4857809	4631553	4921995	4713938	4919528

Table 5-3: Number of Web pages represented from evaluated hosts in WEBSPAM-UK2007

WEBSPAM-UK2007						
Bucket Index	***Algorithms***					
	ATR	***WATR***	***DISTR***	***WDISTR***	***Distrust***	***WDistrust***
1	929200	970876	947715	892901	960910	910721
2	1005378	1069127	1026238	1025691	1020807	1063244
3	1012441	1101626	1058704	1077870	1057994	1079379
4	1037990	1115851	1075044	1139643	1076417	1110244
5	1064091	1120334	1112379	1181418	1117387	1116103
6	1080294	1148771	1115044	1188373	1126445	1120537
7	1099606	1149016	1115320	1190689	1134084	1130236
8	1107402	1190866	1131098	1207496	1136708	1190857
9	1139103	1191074	1164160	1207522	1184168	1206430
10	1198849	1216639	1203615	1207579	1205119	1206584

**ATR – Anti-TrustRank, , WATR – Weighted Anti-TrustRank,*

DISTR. – Wu et al. Distrust, WDISTR – Weighted Wu et al. Distrust,

Distrust – Nie et al. Distrust, WDistrust – Weighted Nie et al. Distrust

Table 5-2 and 5-3 shows the number of Web pages represented from evaluated hosts in WEBSPAM-UK2006 and WEBSPAM-UK2007. The tables show each bucket with its accumulating Web pages. The evaluated hosts are retrieved from previous experiments in Figure 5.9 to Figure 5.14. However, only the top 10 buckets are shown for the reason that detection of Web spam in early buckets are more important. As shown in the tables, at the 10th bucket, the weighted algorithms detect more spam Web pages than the benchmark algorithms. Both weighted Wu et al. Distrust and weighted Nie et al. Distrust did not managed to detect more spam pages at the first few buckets. However, both the algorithms managed to detect more in later buckets. For example , weighted Wu et al. Distrust move close to the bench algorithm at the 2nd bucket and catch up in 3rd bucket and so on in both Web spam datasets. Weighted Nie et al. Distrust algorithm manages to detect more spam pages at the 2nd bucket onwards. The full results of all experiments in this chapter can refer to APPENDIX E - Chapter 5 Results.

Aside from the aforementioned Web spam detection algorithms, the weighted algorithms (i.e. weighted Anti-TrustRank, weighted Wu et al. Distrust and weighted Nie et al. Distrust) outperform LVTrustRank (Qi, Song-Nian, and Sisi 2008) by the average detection of spam sites for the top five buckets in WEBSPAM-UK2007. LVTrustRank has an average detection of 4% while the weighted algorithms achieve average demotion of 7.8%, 7.7% and 7.4% individually for the top five buckets.

The weight properties can further improve the Web spam detection experience of other link-based distrust model algorithms such as ParentPenalty (Wu and Davison 2005b), R-SpamRank (Liang, Ru, and Zhu 2007), QoC-QoL (Li, Qiancheng, and Yan 2008), AVRank & HVRank (Zhang et al. 2009), and also Trust-Distrust Rank (Zhang, Wang, et al. 2011).

5.4 COMPUTATIONAL COMPLEXITY

Consider a weighted directed host graph $G_W = (\upsilon, \varepsilon, \omega)$, with υ as a set of vertices, ε as a set of edges and ω is the weight of the edges. Initially, Equation 5.4 sums the out degree weights for each vertex of the graph. This operation can be divided into two steps which are one, $O(\varepsilon)$ time for summation of weights, shown in Equation 5.5 and two, $O(\upsilon)$ time for performing on all vertices, shown in Equation 5.6. Thus the total time devoted for this operation is $O(\upsilon + \varepsilon)$. The weight is calculated by multiplying every weight of the edges with the reciprocal of the summation weight for every vertex, shown in Equation 5.7. The time spent on visiting all edges is $O(\varepsilon)$ while $O(\upsilon)$ for all vertices. Therefore, this operation costs $O(\upsilon + \varepsilon)$. To summarize, the total operations for performing weight function is $O(\upsilon + \varepsilon)$ time. The weight is an important feature for detection of Web spam; this method is definitely worthy in exchange of better performances. Details on Big O notation can refer to APPENDIX A - Asymptotic Notation.

5.6 SUMMARY

Many unethical ways have been adopted by the commercial industries to make their website appear at the top of the search results and thus undermining the web users' interests. Many anti-link spam techniques have been constantly proposed. In this chapter, the incorporation of weight properties is proposed to enhance the Web spam detection algorithms. In the experiment section, three well known Web spam detection algorithms are modified and compared with the original algorithms. The results have shown that based on the same quantity of spam seeds, the weight has greatly improved the baseline algorithm up to 30.25% at the host level and 39.76% at the page level in detection of Web spam for WEBSPAM-UK2006 while up to 31.30% at the host level and 8.81% at the page level for WEBSPAM-UK2007 dataset.

Chapter 6 Distrust Seed Set Propagation Algorithm

6.1 INTRODUCTION

The process of carefully choosing pages for propagation purpose is known as seed selection process. The seed selection process is crucial in terms of quality and quantity towards the performance of these trust and distrust models (Zhang, Han, and Liang 2009). For trust propagation, seeds are selected based on their outgoing links to identify pages that give the broadest propagation, as that of how HITS is calculating the hub score. Conversely, for distrust propagation, seeds are selected based on their incoming links, as that of PageRank is calculating the authority score. Krishnan and Raj (Krishnan and Raj 2006) use high PageRank spam seeds to detect more spam sites because high PageRank spam seeds are likely to detect other spam sites with relatively high PageRank. Despite the quality of the seeds, the quantity is still a problem. Manual evaluation for the seed set is tremendously expensive in terms of both cost and also in not be enough to cover the Web.

These problems have been noticed by few researchers (Zhang, Han, and Liang 2009; Wu, Goel, and Davison 2006b; Jiang et al. 2008). Automatic seed set expansion algorithm proposed by Zhang et al. (Zhang, Han, and Liang 2009) follows the intuition that if one page is pointed by many trustworthy pages, then that page can be trusted. Wu et al. (Wu and Davison 2005b) proposed Parent Penalty which follows the intuition that if one page is pointing to many spam pages, it is likely that this page is a spam page. These algorithms are the only two that expand the seed set to combat Web spam. Both however, use threshold to separate spam and non-spam. Due to the enormity of the Web, threshold is very hard to determine.

In this chapter, the purpose is to detect more spam pages which are more concern to

Web search engines. The distrust seed set propagation algorithm (DSP) is proposed which act as an extension to the spam seed set to calculate the distrust score for unevaluated pages. Unlike the expand seed set algorithms that mentioned earlier, the intention is to assist the manual evaluation by calculating the likelihood of other pages becoming spam based on the seed set. The experiments are done on WEBSPAM-UK2006 (Castillo et al. 2006) and WEBSPAM-UK2007 (Yahoo! 2007) and have shown that DSP algorithm works well with existing Web spam detection algorithms.

6.2 Distrust Seed Set Propagation

In this section, a detailed explanation is given on distrust seed set propagation algorithm (DSP) and in addition, some examples are provided on the proposed algorithm.

6.2.1 Algorithm

Web spam detection is more effective at host level rather than page level for the reason that if one page is a spam host, it can be assumed that all pages under this host are all spams. A host is denoted as a set of Web pages under the same domain name. Consequently, algorithms are all done at the host level.

The seed selection process for trust and distrust model Web spam algorithms either select spam seeds to propagate distrust or select trustworthy seeds to propagate trust to filter Web spam. Since distrust seed set propagation algorithm is correlated to Web spam detection algorithm, the seed selection process for DSP therefore select spam seeds to propagate distrust to detect Web spam.

According to Krishnan and Raj (Krishnan and Raj 2006), the seed selection algorithm that efficiently detects more high PageRank spam vertices is the PageRank algorithm (Brinkmeier 2006). Actually it is HostRank (Eiron, McCurley, and Tomlin 2004) since the experiments are done at the host level.

The HostRank algorithm can be written as:

$$HR(p) = \alpha \cdot \sum_{(q,p)\in\varepsilon} \left(\frac{HR(q)}{\deg^{+}(q)} \right) + (1-\alpha) \cdot \frac{1}{N} \tag{6.1}$$

Where *HR(p)* is the HostRank score on host *p*, α is a decay factor, $\deg^{+}(q)$ is the number of outgoing links of host *q*.

Assume that a host graph $G_H = (\upsilon_H, \varepsilon_H)$ where υ_H is a set of host vertices and ε_H is a set of ordered pair of hosts. Initially HostRank are executed on the host graph H and top selected HostRank vertices are evaluated and assigned with an initial distrust score d_1. The initial distrust score d_1 is calculated such that

$$d_1(p) = \begin{cases} 1, & p \in \upsilon_S \\ 0, & otherwise \end{cases} \tag{6.2}$$

where the initial distrust score of host *p* $\overline{d_1}$ is 1 if host *p* is a spam host, 0 otherwise. The distrust score then normalized where the normalized distrust score $\overline{d_i}$ by,

$$\overline{d_i} = \frac{d_i}{\|d_i\|} \tag{6.3}$$

Such that

$$\sum \overline{d_i}(\upsilon_H) = 1 \tag{6.4}$$

Note that the initial distrust score is similar to the results from the evaluation process for the Web spam detection algorithms. The difference is that the distrust score calculated by the distrust seed set propagation algorithm is an iterative process. Thus, at the next iteration, while the distrust score of evaluated vertices remains, the distrust scores for the unevaluated vertices are calculated as.

$$\overline{d_i}(p) = \frac{\sum_{(p,q)\in\varepsilon} \overline{d_{i-1}}(q)}{\deg^+(q)} \tag{6.5}$$

where $\overline{d_i}(p)$ is the new distrust score of page *p* at i^{th} iteration; *(p,q)* denotes as there is a hyperlink from page *p* to page *q;* $\deg^+(p)$ denotes the number of outgoing links of page *p*. After the distrust scores for the unevaluated vertices are calculated, the distrust scores then again normalized using Equation 6.3 and 6.4. The distrust seed set propagation algorithm is an iterative process where the iteration is dependent to the size of the Web graph, the iteration will reach until one point where it start to converge. The equivalent matrix equation form of Equation 6.5 is:

$$\overline{d_i} = s \cdot I \cdot \overline{d_{i-1}} \tag{6.6}$$

where $\overline{d_i}$ is the distrust score vector, *I* is an inverse adjacent matrix represent the Web structure, in which *T*(*p*,*q*) is 1 if page *p* is pointing to page *q*, otherwise 0. *s* is a vector which represent $1/\deg^+(p)$ where $\deg^+(p)$ is the number of outgoing links of page *p*. Figure 6.1 illustrates the distrust seed set propagation algorithm.

In this experiment, three well-known Web spam detection algorithms are chosen and compared with those along with the distrust seed set propagation algorithm – Anti-TrustRank (Krishnan and Raj 2006) refer to Equation 3.2, Wu et al. (Wu, Goel, and Davison 2006a) distrust algorithm refer to Equation 5.12 and Nie et al. (Nie, Wu, and Davison 2007) distrust algorithm refer to Equation 5.14.

Algorithm : Distrust Seed Set Propagation Algorithm

Input : d Distrust score

$\overline{d}$ Normalized distrust score

I Inverse adjacency matrix

υ_S Spam seed set

υ All vertices in the graph

M Number of iterations

s A vector denote by $1/\deg^{+}(p)$ for all vertices where $\deg^{+}(p)$ is the number of outgoing links of page p

Output : $\overline{d_M}$ Final normalized distrust score

Begin

1. $d_1 = 0_N$
2. for page p in υ :
3. if p in υ_S :
4. $d_1(p) = 1/sum(\upsilon_S)$
5. for $i = 2$ to M do:
6. $d_i = s \cdot I \cdot d_{i-1}$
7. $\overline{d_i} = d_i / \|d_i\|$
8. return $\overline{d_M}$

End

Figure 6.1: The distrust seed set propagation (DSP) algorithm.

6.2.2 Example

Consider Figure 6.2 as shown below.

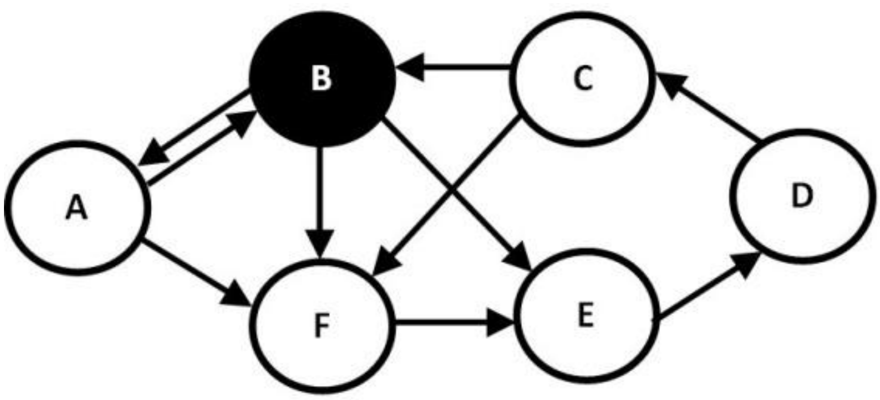


Figure 6.2: Sample Web graph.

A sample of inverse adjacent matrix on Figure 2 is shown as:

$$I = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The vectors on Figure 2 would be:

$$s = \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{2} & 1 & 1 & 1 \end{bmatrix}$$

The distrust seed set propagation algorithms would run in an iterative computation to assure the propagation spread further. Assume that Page *B* is a spam seed; at the first iteration where Equation 6.2 is applied, the distrust score vector *d* is given:

$$d_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

After that, the distrust score vector *d* is applied onto Equation 6.5 iteratively to assure that the distrust is propagating. The intention of the distrust seed set propagation algorithm is to find the probability of other unevaluated pages being spam. Table 6-1 illustrates the distrust score vector on Figure 6.2 for 10 iterations.

In Table 6-1, assume page *A* is a spam seed; the likelihood for the unevaluated pages becoming a spam is counted iteratively where each iteration is normalized to the sum of 1. A page will only become dishonest if it points to spam seeds. While observing the 2nd iteration, since page *A* and page *C* pointing to page *B*, there is a possibility that both page *A* and page *C* are spam. Distrust is propagated further as more iteration is done. The distrust distribution in Table 6-1 is applied into Anti-TrustRank algorithm with

DSP in Equation 6.6, the decay factor α is set to 0.85 and run in 20 iterations; the results are shown in Table 6-2.

Table 6-1: Distribution of distrust propagation

Distrust Score	***Pages***					
	A	***B***	***C***	***D***	***E***	***F***
$\overline{d_1}$	0	1	0	0	0	0
$\overline{d_2}$	0.25	0.50	0.25	0	0	0
$\overline{d_3}$	0.20	0.40	0.20	0.20	0	0
$\overline{d_4}$	0.167	0.333	0.167	0.167	0.167	0
$\overline{d_5}$	0.143	0.285	0.143	0.143	0.143	0.143
$\overline{d_6}$	0.187	0.25	0.187	0.125	0.125	0.125
$\overline{d_7}$	0.177	0.235	0.176	0.176	0.118	0.118
$\overline{d_8}$	0.167	0.222	0.167	0.167	0.167	0.111
$\overline{d_9}$	0.158	0.211	0.158	0.158	0.158	0.158
$\overline{d_{10}}$	0.175	0.200	0.175	0.150	0.150	0.150

Table 6-2: Results of Anti-TrustRank with Distrust Seed Set Propagation algorithm

Distrust Score	***Pages***					
	A	***B***	***C***	***D***	***E***	***F***
$\overline{d_1}$	0.164	0.359	0.164	0.141	0.125	0.053
$\overline{d_2}$	0.180	0.312	0.180	0.164	0.133	0.059
$\overline{d_3}$	0.164	0.281	0.164	0.180	0.148	0.066
$\overline{d_4}$	0.172	0.297	0.172	0.172	0.172	0.070
$\overline{d_5}$	0.164	0.281	0.164	0.164	0.164	0.094
$\overline{d_6}$	0.164	0.297	0.164	0.180	0.148	0.090
$\overline{d_7}$	0.156	0.281	0.156	0.172	0.148	0.090
$\overline{d_8}$	0.156	0.281	0.156	0.18	0.156	0.090
$\overline{d_9}$	0.156	0.281	0.156	0.172	0.156	0.098
$\overline{d_{10}}$	0.164	0.266	0.164	0.164	0.164	0.094

Observed from Table 6-2, note that $\overline{d_1}$ actually is the original algorithm. When applied the distrust seed set algorithm, the distrust values are propagating around, this enhanced the Web spam detection algorithms. The experiments are done in a large dataset namely WEBSPAM-UK2006 and WEBSPAM-UK2007 that the algorithms work well with distrust seed set algorithm and reached convergence after the 5th distrust score vector.

6.3 Experimental Results

In the experiments, distrust seed set propagation is performed for 10 iterations on well-known three Web spam detection algorithms – Anti-TrustRank (Krishnan and Raj 2006), Wu et al. Distrust (Wu, Goel, and Davison 2006a) and Nie et al. Distrust (Nie, Wu, and Davison 2007) in WEBSPAM-UK2006 and WEBSPAM-UK2007 (See Chapter 2 for more details on datasets). Even though the results are distributed in 20 buckets, only the top 10 buckets are concerned because early buckets is crucial for Web spam detection.

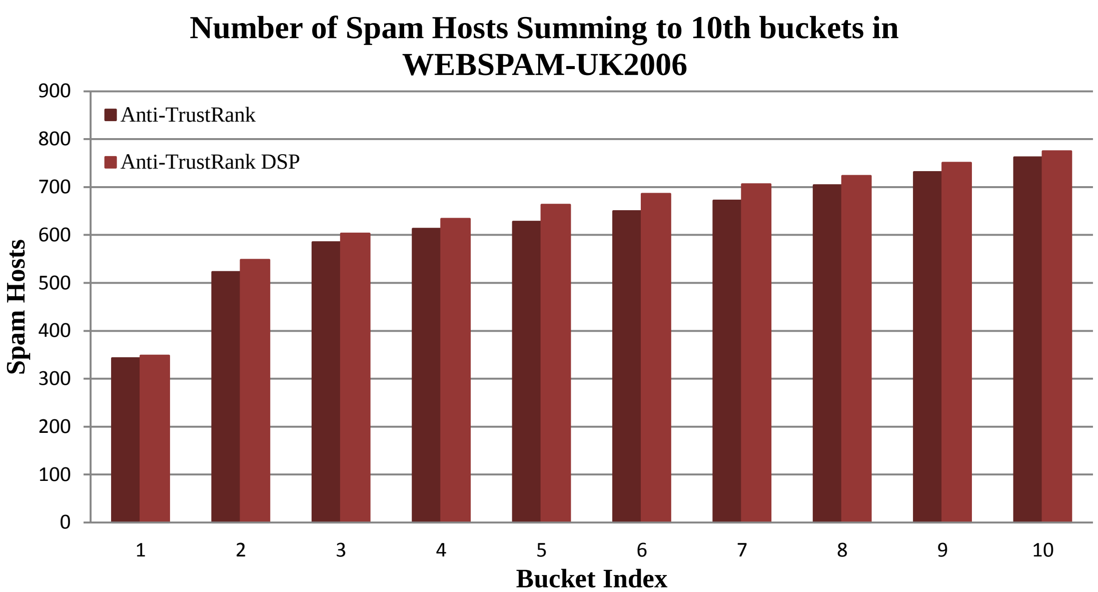


Figure 6.3: Number of spam hosts summing to the 10th bucket for Anti-TrustRank and Anti-TrustRank DSP in WEBSPAM-UK2006.

Figure 6.3 to Figure 6.8 illustrate number of spam hosts summing to the 10th bucket in WEBSPAM-UK2006 and spam hosts gap in top 10 buckets in WEBSPAM-UK2007 for Anti-TrustRank, Wu et al. Distrust and Nie et al. Distrust versus the DSP algorithms at the 2nd iteration.

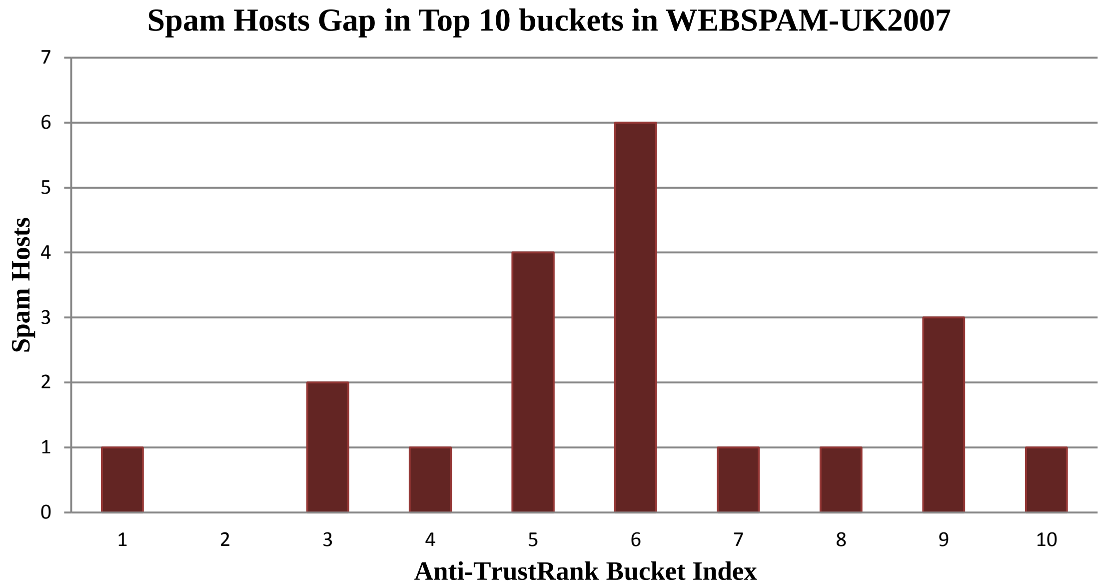


Figure 6.4: Spam hosts gap in Top 10 buckets for Anti-TrustRank DSP over Anti-TrustRank in WEBSPAM-UK2007.

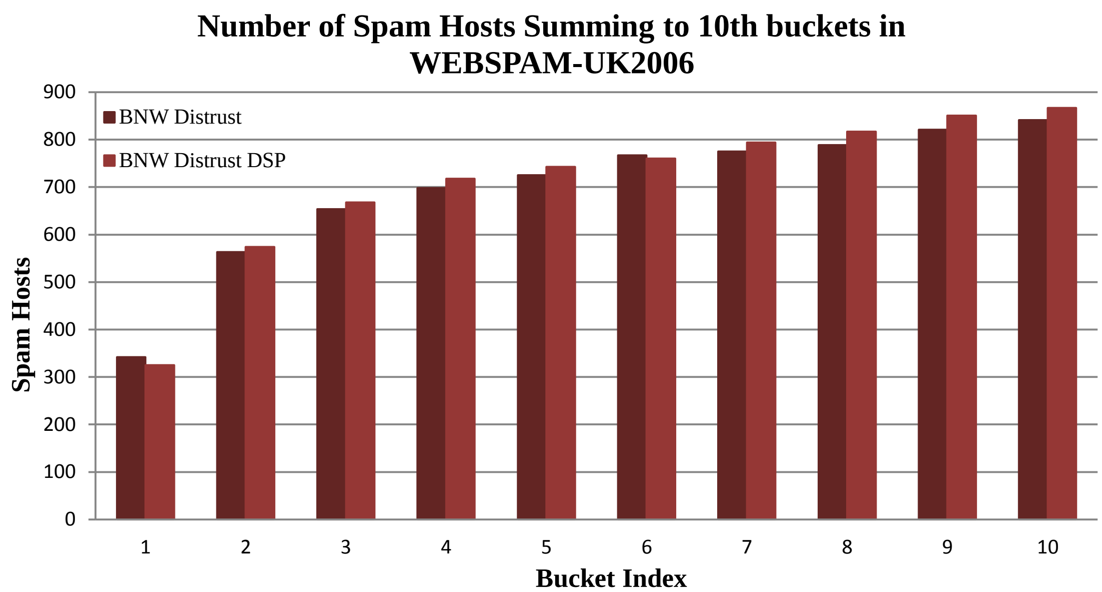


Figure 6.5: Number of spam hosts summing to the 10th bucket for Wu et al. Distrust and Wu et al. Distrust DSP in WEBSPAM-UK2006.

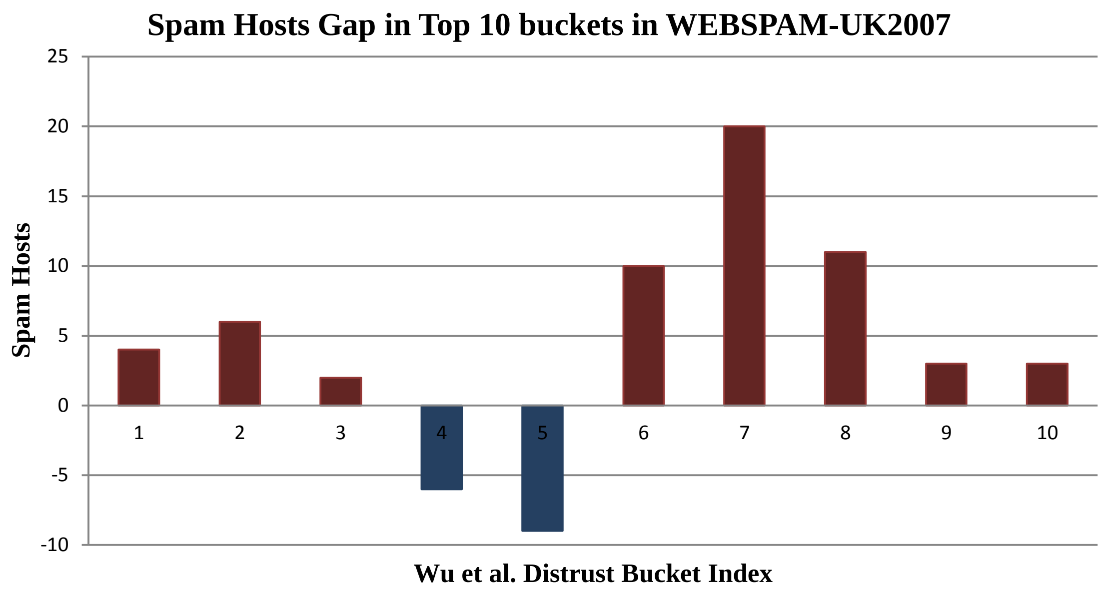


Figure 6.6: Spam hosts gap in Top 10 buckets for Wu et al. Distrust DSP over Wu et al. Distrust in WEBSPAM-UK2007.

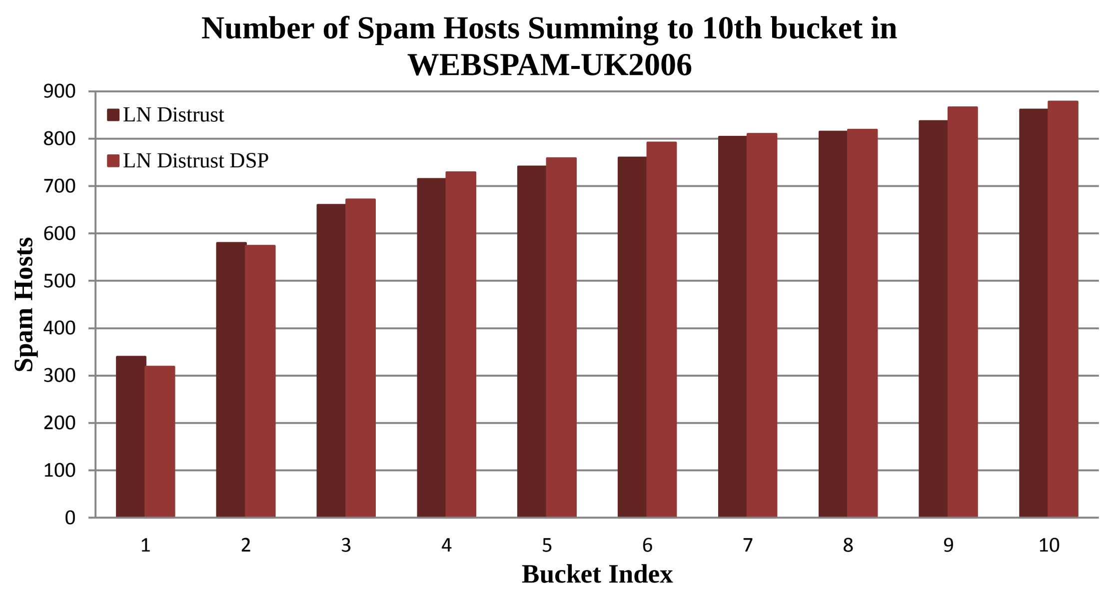


Figure 6.7: Number of spam hosts summing to the 10th bucket for Nie et al. Distrust and Nie et al. Distrust DSP in WEBSPAM-UK2006.

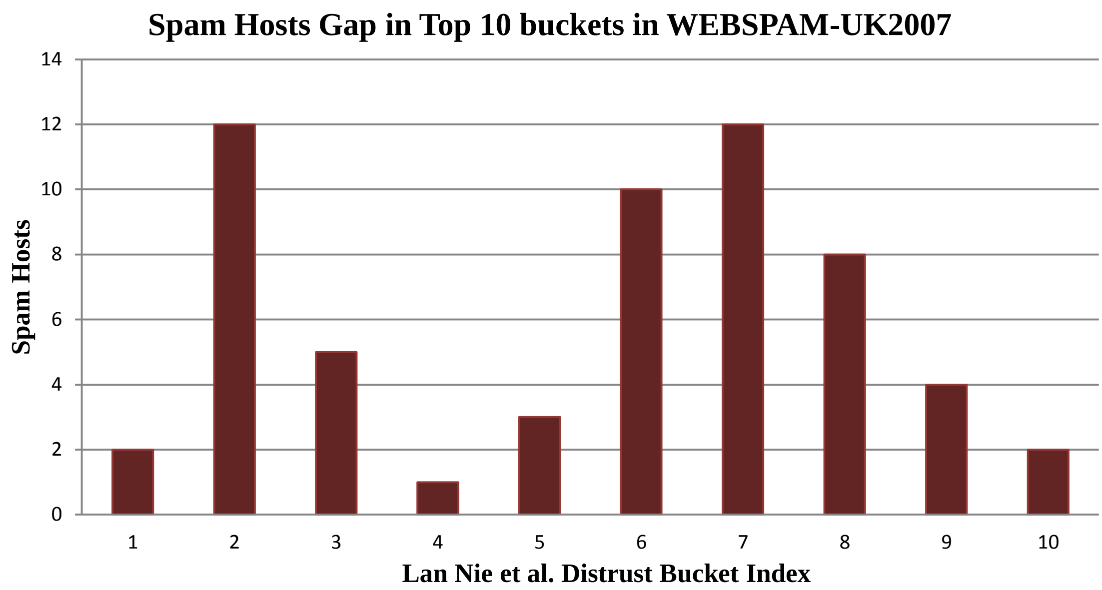


Figure 6.8: Spam hosts gap in Top 10 buckets for Nie et al. Distrust DSP over Nie et al. Distrust in WEBSPAM-UK2007.

To summarize, the DSP 2nd iteration improve Anti-TrustRank up to 5.57%, Wu et al. Distrust up to 3.54% and Nie et al. Distrust up to 4.21% in WEBSPAM-UK2006. For WEBSPAM-UK2007 on the other hand, the DSP 2nd iteration improve Anti-TrustRank up to 3.7%, Wu et al. Distrust up to 11.83% and Nie et al. Distrust up to 10.26%. Even though it is a small improvement, the distrust seed set algorithm did improve the baseline algorithms. In later experiments, the results on summation of 10 buckets of spam hosts are presented for these algorithms along with 10 iterations DSP.

Table 6-3: Summation of 10 buckets of spam hosts for DSP on 10 iterations in WEBSPAM-UK2006

WEBSPAM-UK2006										
	Iteration									
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**	**10th**
ATR	762	775	825	865	880	887	887	**888**	887	886
	-	1.71%	8.27%	13.52%	15.49%	16.40%	16.40%	**16.54%**	16.40%	16.27%
Wu	841	867	863	871	877	882	883	**884**	884	884
	-	3.09%	2.62%	3.57%	4.28%	4.88%	4.99%	**5.11%**	5.11%	5.11%
Nie	861	878	884	901	937	941	943	**944**	941	941
	-	1.97%	2.67%	4.65%	8.83%	9.29%	9.52%	**9.64%**	9.29%	9.29%

*ATR – Anti-TrustRank, Wu – Wu et al. Distrust, Nie – Nie et al. Distrust

Table 6-4: Summation of 10 buckets of spam hosts for DSP on 10 iterations in WEBSPAM-UK2007

WEBSPAM-UK2007										
	Iteration									
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**8th**	**9th**	**10th**
ATR	204 -	205 0.49%	205 0.49%	213 4.41%	216 5.88%	216 5.88%	215 5.39%	215 5.39%	**217** **6.37%**	216 5.88%
Wu	208 -	211 1.44%	213 2.40%	214 2.88%	216 3.85%	216 3.85%	**217** **4.33%**	217 4.33%	217 4.33%	217 4.33%
Nie	209 -	211 0.96%	214 2.39%	221 5.74%	226 8.13%	228 9.09%	229 9.57%	229 9.57%	**230** **10.05%**	230 10.05%

*ATR – Anti-TrustRank, Wu – Wu et al. Distrust, Nie – Nie et al. Distrust

Table 6-3 and 6.4 show the summation of spam hosts for 10 buckets for the baseline algorithms along with the distrust seed set algorithm from 2nd iteration to 10th iteration for WEBSPAM-UK2006 and WEBSPAM-UK2007. The percentages of improvement over the baseline algorithms are also shown in the tables. Note that the 1st iteration DSP is the standard propagation for all trust and distrust algorithms. In WEBSPAM-UK2006, all algorithms with DSP managed to detect the most spam hosts at the 8th iteration while reaching convergence at the 5th iteration. The biggest improvement goes to Anti-TrustRank on 8th iteration DSP with an improvement of 16.54% detecting 888 spam hosts. Despite of this, the most spam hosts detected is Nie et al. Distrust on 8th iteration DSP with 944 spam hosts detected at the 10th bucket.
For WEBSPAM-UK2007, both Anti-TrustRank and Nie et al. Distrust detect the most spam hosts with 9th iteration DSP while Wu et al. Distrust detect the most spam hosts with 7th iteration DSP. Regardless of this, the three algorithms reach convergence after the 5th iteration. The most spam hosts detected and also the biggest improvement goes to Nie et al. Distrust at 9th iteration DSP with an improvement of 10.05% detecting 230 spam hosts.

Figure 6.9 to Figure 6.12 illustrate the average spam hosts promotion and number of spam hosts promoted for Anti-Trustrank on 8th iteration DSP in WEBSPAM-UK2006 and on 9th iteration DSP WEBSPAM-UK2007 over its benchmark algorithm – Anti-TrustRank.

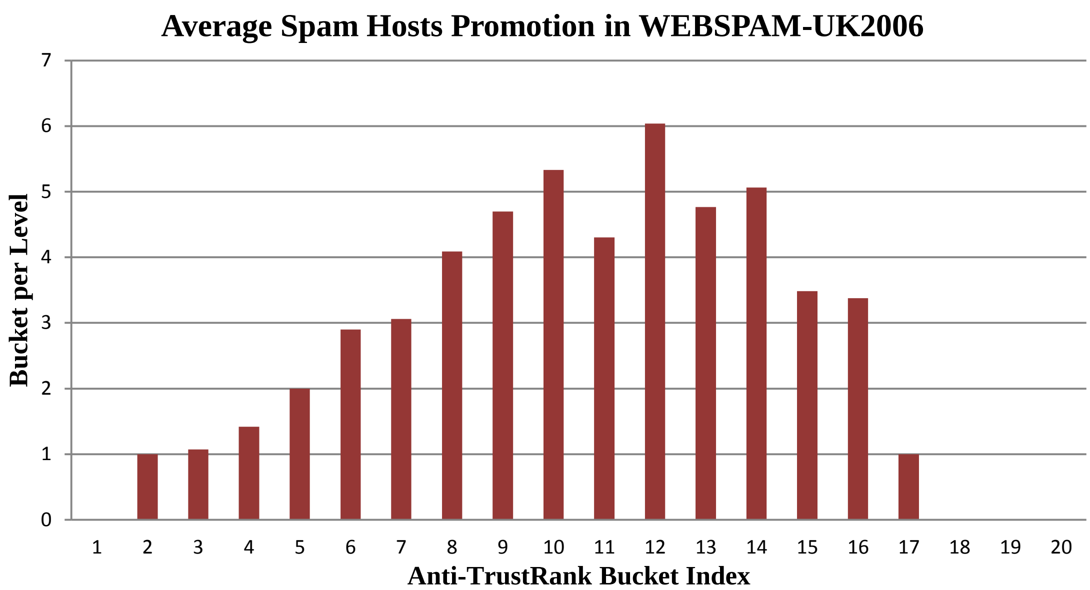


Figure 6.9: Average spam hosts promotion for Anti-TrustRank on 8th iteration DSP over Anti-TrustRank in WEBSPAM-UK2006.

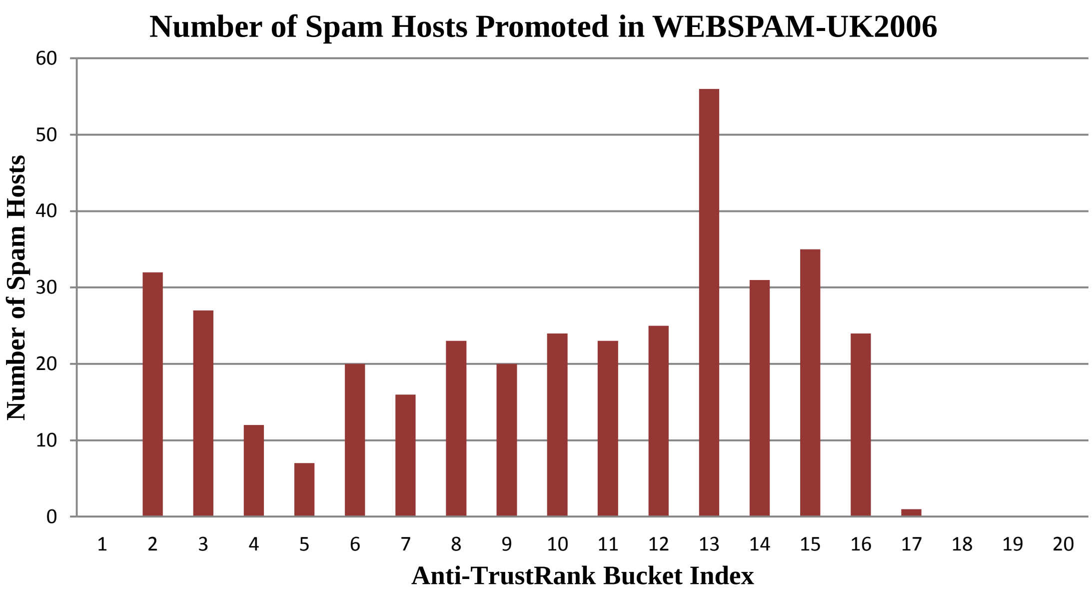


Figure 6.10: Number of spam hosts promoted for Anti-TrustRank on 8th iteration DSP over Anti-TrustRank in WEBSPAM-UK2006.

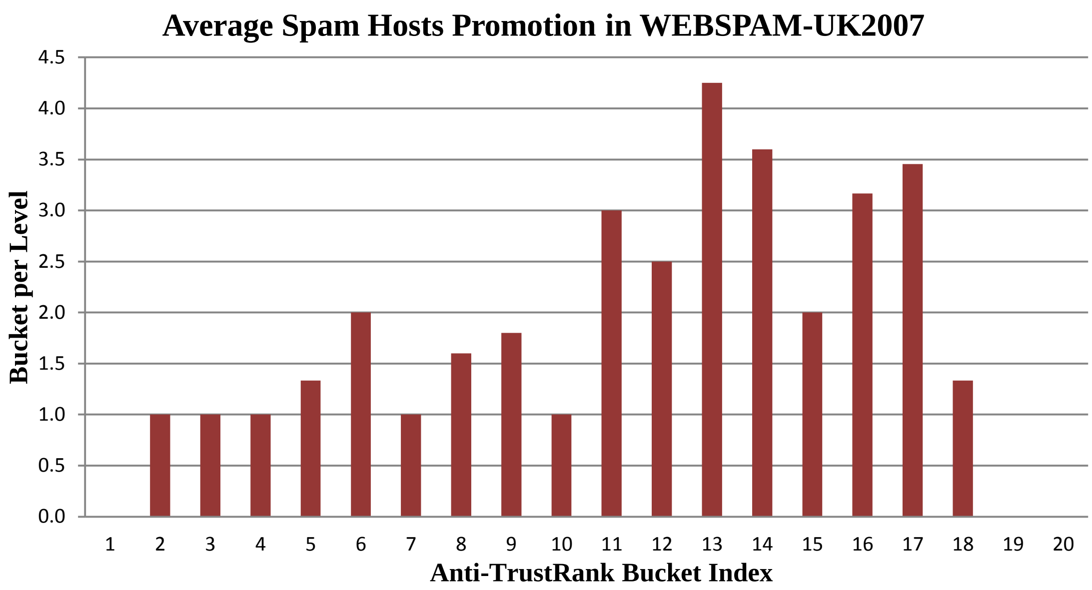


Figure 6.11: Average spam hosts promotion for Anti-TrustRank on 9th iteration DSP over Anti-TrustRank in WEBSPAM-UK2007.

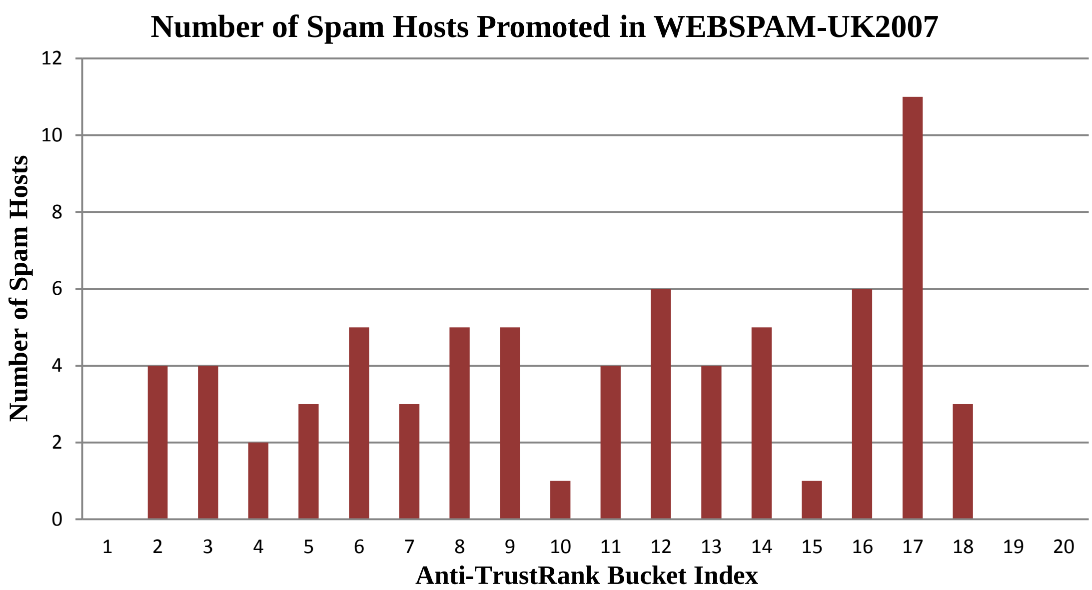


Figure 6.12: Number of spam hosts promoted for Anti-TrustRank on 9th iteration DSP over Anti-TrustRank in WEBSPAM-UK2007.

In WEBSPAM-UK2006, the highest average spam host promotion is the 12th bucket where it promotes 6.04 bucket per level for the spam hosts. However the most spam hosts being promoted is the 13th bucket with 56 spam hosts promoted. In WEBSPAM-UK2007, there are a total of 72 spam hosts promoted by DSP over the benchmark algorithm, the highest average spam hosts promotion is the 13th bucket with 4.25 bucket per level.

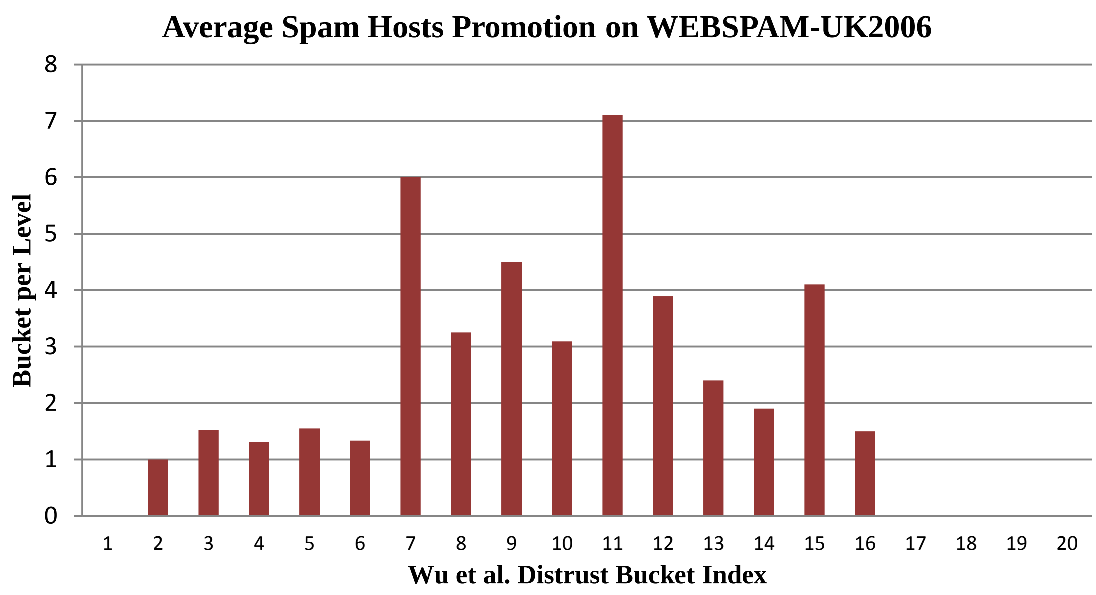


Figure 6.13: Average spam hosts promotion for Wu et al. Distrust on 8th iteration DSP over Wu et al. Distrust in WEBSPAM-UK2006.

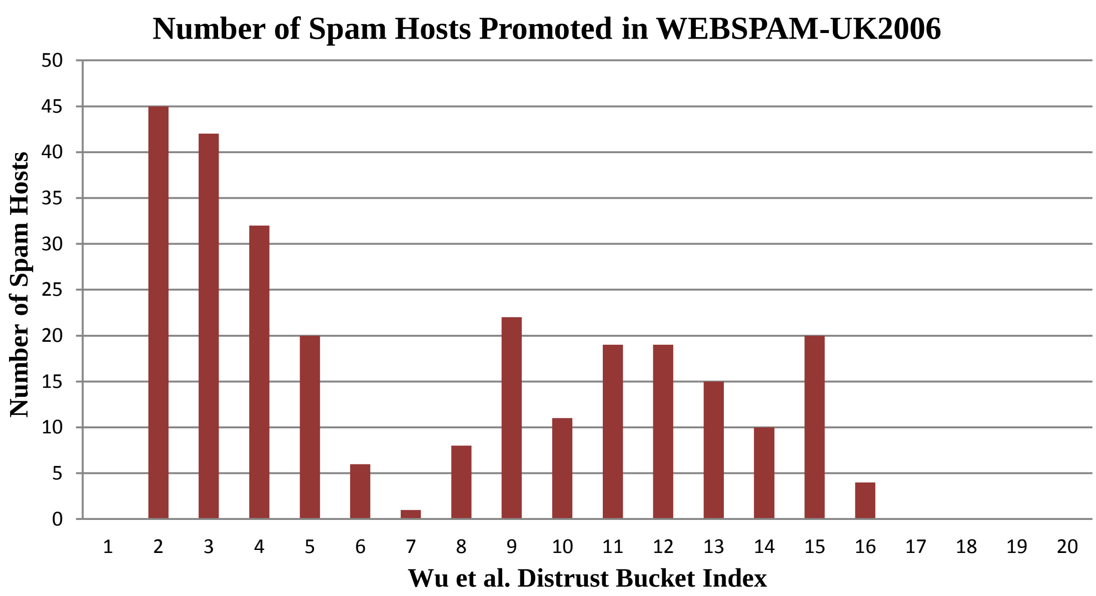


Figure 6.14: Number of spam hosts promoted for Wu et al. Distrust on 8th iteration DSP over Wu et al. Distrust in WEBSPAM-UK2006.

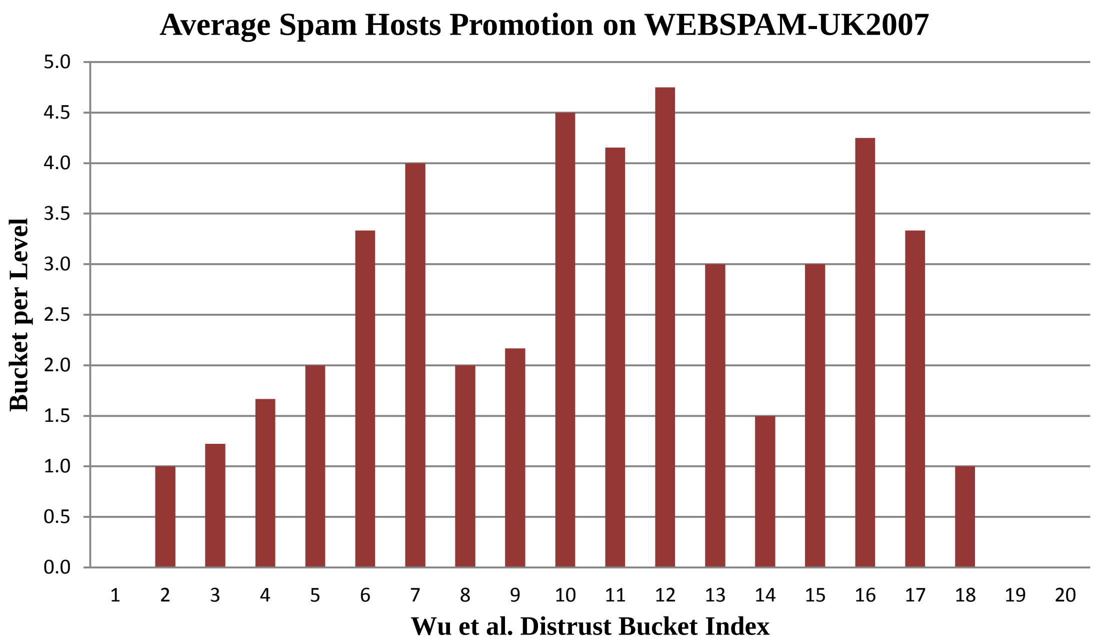


Figure 6.15: Average spam hosts promotion for Wu et al. Distrust on 7th iteration DSP over Wu et al. Distrust in WEBSPAM-UK2007.

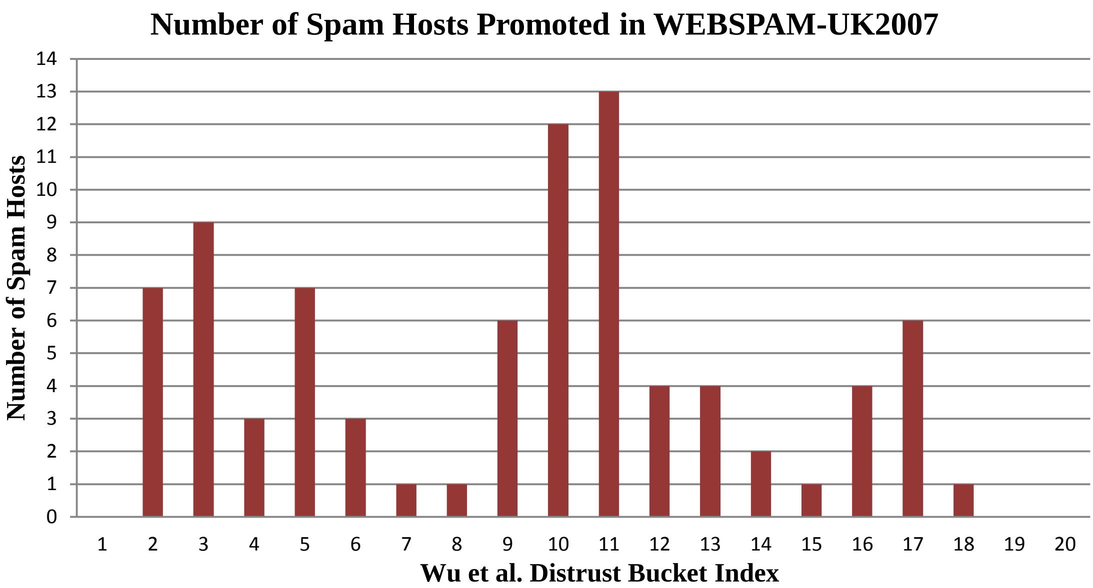


Figure 6.16: Number of spam hosts promoted for Wu et al. Distrust on 7th iteration DSP over Wu et al. Distrust in WEBSPAM-UK2007.

Figure 6.13 to Figure 6.16 illustrate the average spam hosts promotion and number of spam hosts promoted for Wu et al. Distrust on 7th iteration DSP in WEBSPAM-UK2006 and on 8th iteration DSP WEBSPAM-UK2007 over its benchmark algorithm – Wu et al. Distrust. From the figures, the highest bucket per level in WEBSPAM-UK2006 is the 11th bucket where it promotes 7.1 bucket per level for the spam hosts. Still, the bucket that promotes the most spam hosts is the 2nd bucket promoting 45 spam hosts. WEBSPAM-UK2007 on the other hand, the highest bucket per level is 4.75 bucket per level for the spam hosts while the highest number of spam hosts promoted is the 11th bucket promoting 13 spam hosts.

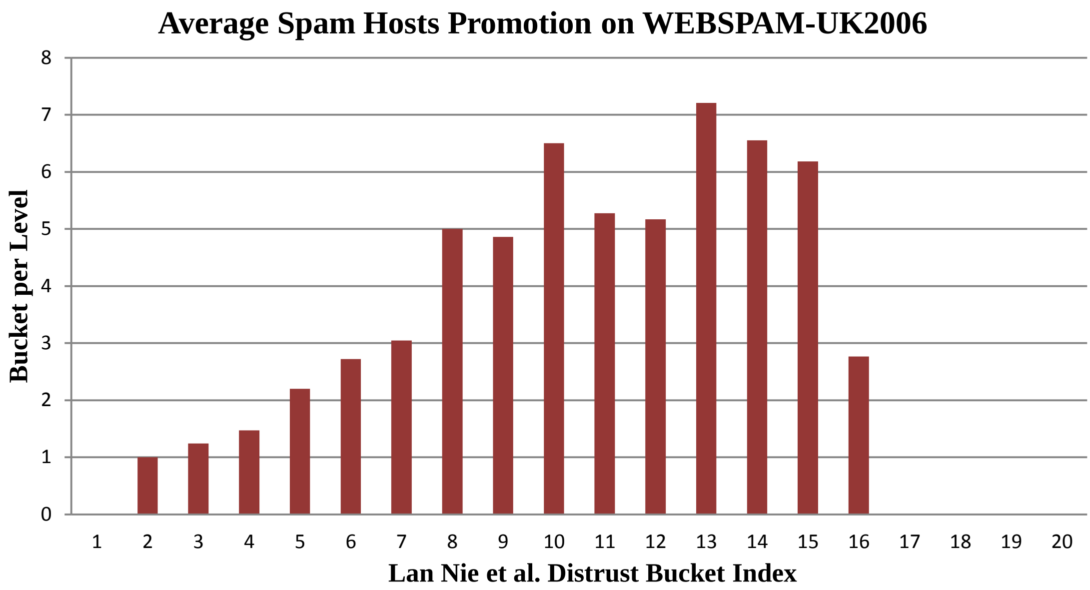


Figure 6.17: Average spam hosts promotion for Nie et al. Distrust on 8th iteration DSP over Nie et al. Distrust in WEBSPAM-UK2006.

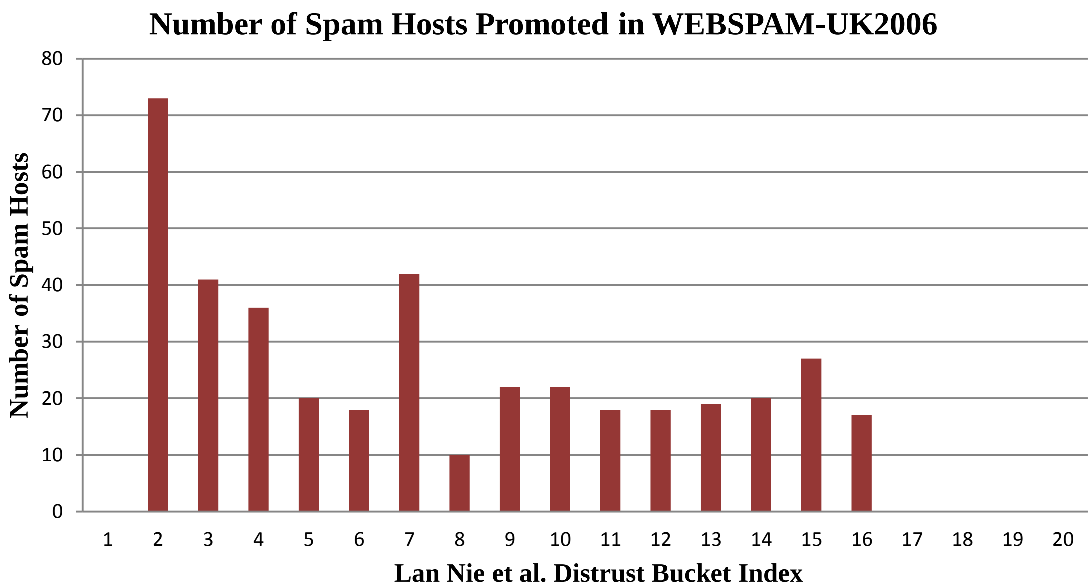


Figure 6.18: Number of spam hosts promoted for Nie et al. Distrust on 8th iteration DSP over Nie et al. Distrust in WEBSPAM-UK2006.

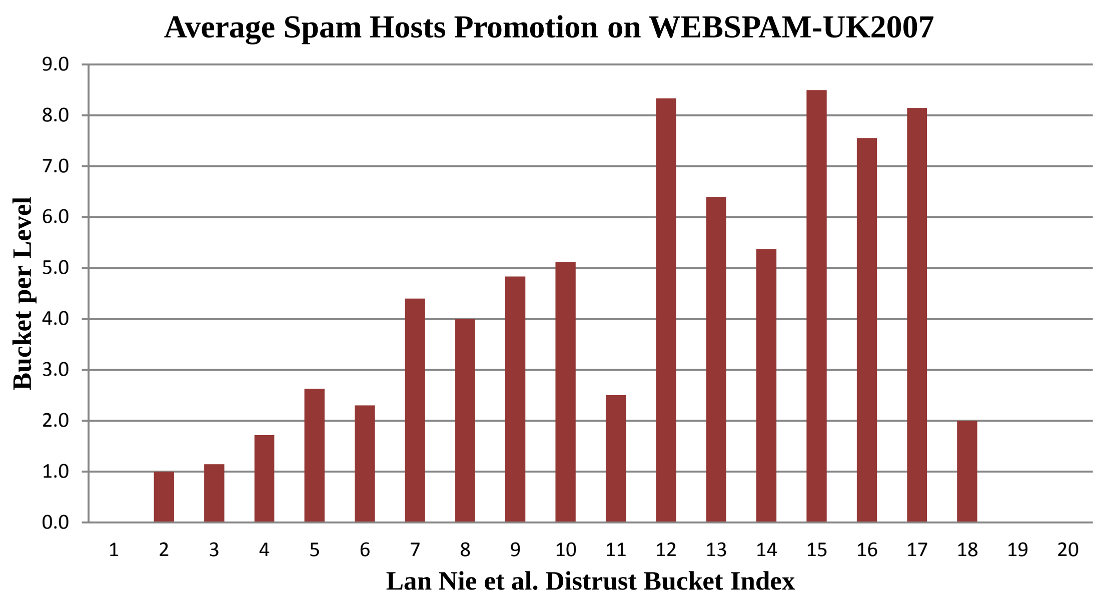


Figure 6.19: Average spam hosts promotion for Nie et al. Distrust on 9th iteration DSP over Nie et al. Distrust in WEBSPAM-UK2007.

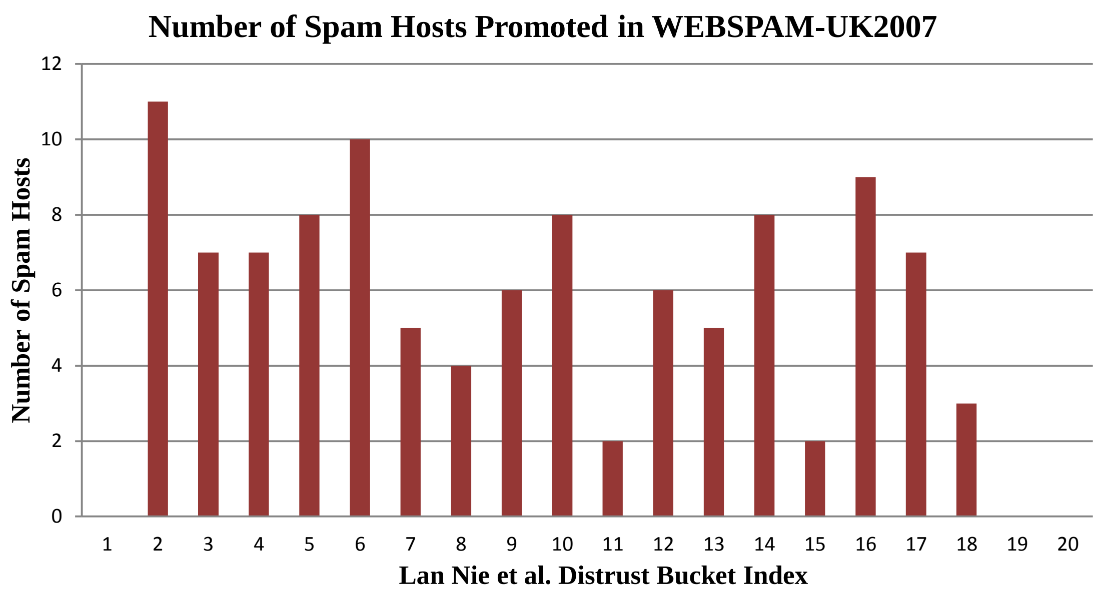


Figure 6.20: Number of spam hosts promoted for Nie et al. Distrust on 9th iteration DSP over Nie et al. Distrust in WEBSPAM-UK2007.

Figure 6.17 to Figure 6.20 illustrate the average spam hosts promotion and number of spam hosts promoted for Nie et al. Distrust on 8th iteration DSP in WEBSPAM-UK2006 and on 9th iteration DSP WEBSPAM-UK2007 over its benchmark algorithm – Nie et al. Distrust.

The highest bucket per level in WEBSPAM-UK2006 is the 13th bucket as high as 7.21 bucket per level for spam hosts while the bucket with the biggest number of spam hosts promoted is the 2nd bucket with 73 spam hosts promoted. For WEBSPAM-UK2007, the highest average spam hosts promotion is the 15th bucket with 8.5 bucket per level for spam hosts. The total sum of spam hosts being promoted by Nie et al. Distrust with 9th iteration DSP over Nie et al. Distrust in WEBSPAM-UK2007 is 108 spam hosts.

Table 6-5: Number of Web pages represented from evaluated hosts in WEBSPAM-UK2006

WEBSPAM-UK2006						
Bucket Index	***Algorithms***					
	ATR	***ATR 8th DSP***	***DISTR***	***DISTR 8th DSP***	***Distrust***	***Distrust 8th DSP***
1	1313291	1298996	1622823	1588293	1653314	1539283
2	2456555	2694632	2728892	2856597	2832581	3003471
3	2841133	3535509	3336290	3841061	3438744	4171748
4	3063894	3845866	3739390	4132319	3929393	4619347
5	3204105	4024198	3945059	4279553	3995253	4944029
6	3370887	4316557	4049277	4488813	4166130	5240632
7	3508984	4448721	4094948	4591742	4282662	5284677
8	3754869	4529346	4282774	4676556	4369946	5352723
9	4061543	4832617	4506142	4780965	4440319	5536531
10	4214193	5004972	4631553	4939534	4713938	5540404

Table 6-6: Number of Web pages represented from evaluated hosts in WEBSPAM-UK2007

WEBSPAM-UK2007						
Bucket Index	***Algorithms***					
	ATR	***ATR 9th DSP***	***DISTR***	***DISTR 7th DSP***	***Distrust***	***Distrust 9th DSP***
1	937106	932239	973094	995268	1005285	1003788
2	1009426	1032782	1027670	1064206	1022486	1068011
3	1014809	1034819	1079967	1064527	1079508	1120374
4	1039525	1060942	1096300	1078740	1112970	1193032
5	1065252	1078895	1133635	1148592	1138692	1195269
6	1084178	1085877	1136315	1158840	1150000	1220140
7	1103333	1087342	1139671	1168950	1155144	1228255
8	1147566	1169866	1153311	1185852	1157171	1258372
9	1163808	1225869	1183984	1228032	1189461	1258837
10	1200058	1226594	1194601	1242814	1205620	1309149

Table 5 and 6 depicts number of Web pages represented from evaluated hosts in WEBSPAM-UK2006 and WEBSPAM-UK2007. At the 10th bucket, both tables have shown that the algorithms with distrust seed set propagation have detected more spam pages than the benchmark algorithms. Some DSP algorithms in different datasets are not able to detect more spam pages at the first bucket, but the algorithms have shown that more spam pages are detected in later buckets. The algorithm that performs the best in both tables is Nie et al. Distrust with DSP which detect 17.73% more spam hosts in WEBSPAM-UK2006 and detected 8.59% more spam hosts in WEBSPAM-UK2007. The full results of all experiments in this chapter can refer to APPENDIX F - Chapter 6 Results.

In the experiments, DSP algorithm has improved the Web spam detection experience of Anti-TrustRank, Wu et al. Distrust and Nie et al. Distrust. Besides the aforementioned algorithms, DSP able to enhanced existing link-based distrust model algorithms such as ParentPenalty (Wu and Davison 2005b), R-SpamRank (Liang, Ru, and Zhu 2007), QoC-QoL (Li, Qiancheng, and Yan 2008), AVRank & HVRank (Zhang et al. 2009), and also Trust-Distrust Rank (Zhang, Wang, et al. 2011).

6.4 COMPUTATIONAL COMPLEXITY

In terms of computational complexity, assume that a Web graph G consists of a set of vertices υ and a set of edges ε. The most effective seed selection process for spam detection is to select seeds using HostRank algorithm which cost $O(\upsilon+\varepsilon)$ time. Then the seeds are moved to the propagation phase where in the baseline algorithm, it runs for $O(\upsilon)$ time. In DSP however, the algorithm went through all vertices and checks the edges that are connected to the particular vertices. Therefore in the worst case scenario, the algorithm run in $O(\upsilon+\varepsilon)$ time. Overall, the running time complexity is $O(\upsilon+\varepsilon)$ time. In previous experiments, distrust seed set propagation algorithm significantly improved the performance of the baseline algorithms. Details on Big O notation can refer to APPENDIX A - Asymptotic Notation.

6.5 SUMMARY

In this chapter, distrust seed set propagation algorithm (DSP) is proposed to propagate distrust further in order to detect more spam. In the experiment section, three modified Web spam detection algorithms are applied with DSP and shown that it enhanced the baseline algorithms and detected up to 17.73% more spam hosts in WEBSPAM-UK2006 and up to 8.59% more spam hosts in WEBSPAM-UK2007 at the host level, up to 5.33% more spam pages in WEBSPAM-UK2006 and up to 8.75% more spam pages in WEBSPAM-UK2007. The impact of this proposed algorithm in practical can increase the number of spam pages detected in order to clean them as soon as possible.

Chapter 7 Neural Network based Application

7.1 INTRODUCTION

The application of machine learning method in Web spam classification has shown positive results due to their adaptive ability to learn the underlying patterns for classifying spam and non-spam data. Even though the Web spam features are highly correlated with the success for Web spam detection, the structure of classifiers also play an important role. C4.5 decision tree (DT) (Quinlan 1993) and support vector machine (SVM) (Cortes and Vapnik 1995) are two commonly used machine learning approaches among the adversarial information retrieval community. However, there were some evidences showing that SVM actually outperforms DT. Abernethy et al. (Abernethy, Chapelle, and Castillo 2010) obtained the best result in Web Spam Challenge 2007 with the AUC performance of 0.963 using SVM compared to C4.5 DT with the AUC performance of 0.935. Yuchun et al. (Yuchun et al. 2008) obtained higher AUC results with less time and space using SVM than DT in spam senders behaviour analysis. Zhiyang et al. (Zhiyang et al. 2012) did some simulation research on machine learning models for Web spam detection and their results showed that SVM outperformed both rule-based classifier and decision tree classifier in terms of precision, recall and F1-value.

In spite of this, researchers have shown that the outcome of SVM is easily manipulated in adversarial classification tasks like spam filtering (Biggio, Nelson, and Laskov 2011). Furthermore, recent scientific researches (Biggio, Nelson, and Laskov 2012; Xiao, Xiao, and Eckert 2012) indicated that by injecting contaminated training data, the accuracy of the SVM will be significantly degraded.

Aside from SVM and DT, neural networks have emerged as a vital classification tool and have been demonstrated to be a competitive alternative to traditional classifiers (Zhang 2000). There are few researchers using neural networks for Web spam classification. Both Ntoulas et al. (Ntoulas et al. 2006) and Mahmoudi et al. (Mahmoudi, Yari, and Khadivi 2010) used neural networks but the authors did not mention the architecture of the neural networks. Closest to this chapter research in this chapter is Noi et al. (Noi et al. 2010) who use probability mapping graph self-organizing maps for clustering, and then graph neural network for classifying task. However, the training time for a mixture of unsupervised and supervised network is computational expensive.

In this chapter, a multilayer perceptrons (MLP) neural network is proposed for Web spam classification due to its flexible structure and non-linearity transformation to accommodate latest Web spam patterns. Using the right learning algorithm and selecting the right number of hidden neurons are crucial to obtain the optimal results. Therefore, scaled conjugate gradient (Møller 1993) algorithm is selected to supervise MLP network's weight because it could offer faster learning speed and better performances than other standard back propagation algorithms. The experiments are done on two public available Web spam datasets – WEBSPAM-UK2006 (Castillo, Chellapilla, and Davison 2007; Castillo, Davison, et al. 2007) and WEBSPAM-UK2007 (Castillo, Chellapilla, and Denoyer 2008). The experimental results have shown that MLP has improved the AUC performance up to 14.02% over SVM on former dataset and up to 3.53%% over SVM on later dataset. In addition, based on the experimental results, 3 fixed number of hidden neurons are concluded as parameters that are close to optimal results.

7.1 Multilayered Perceptrons Neural Network

For the machine learning model, multilayered perceptrons neural network is used for the role of Web spam detection. MLP neural network is a non-linear feed-forward network model which maps a set of inputs $\mathbf{x}$ onto a set of outputs $\mathbf{y}$ using multi weights connections. A basic structure of MLP is illustrated in Figure 7.1. It consists of an input layer, an output layer and one hidden layer. The input layer has $\mathbf{p}$ number of neurons relying on the input features. The output layer has $\mathbf{r}$ number of neurons depending on the number of classifying task. The hidden layer has $\mathbf{q}$ number of hidden neurons and the number of hidden neurons is varied from $[2,\infty]$ (Haykin 1998). It depends on the linearity of the mapping data.

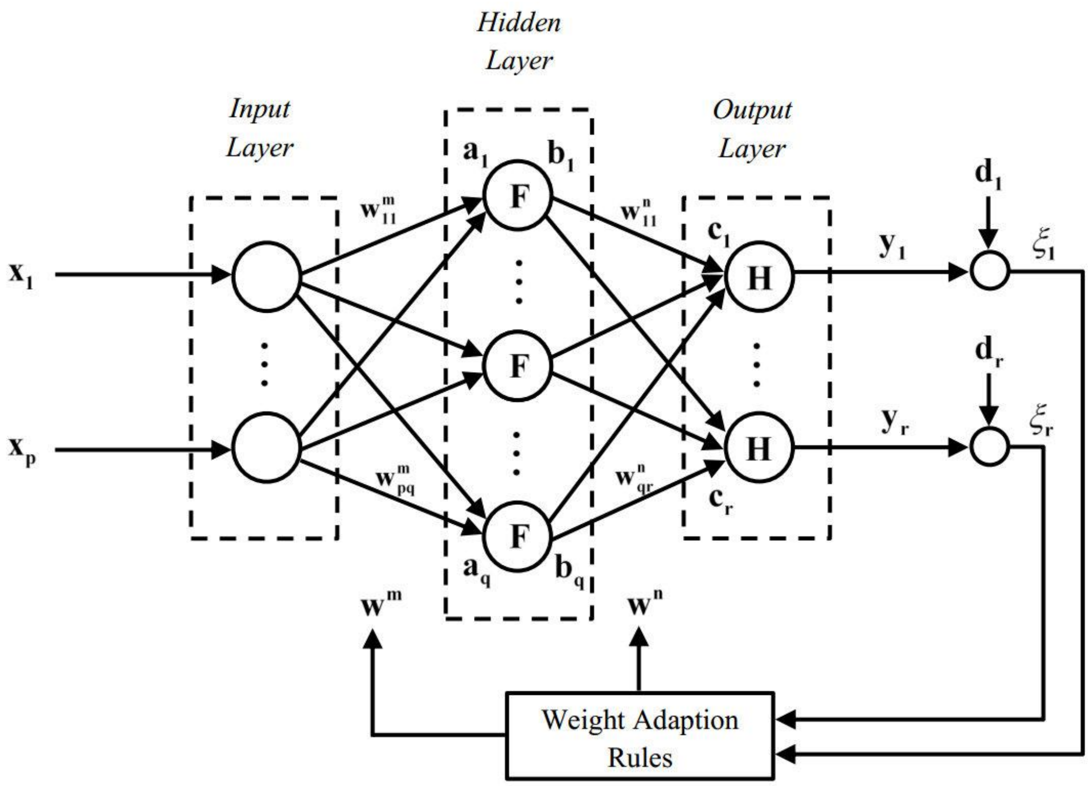


Figure 7.1: The structure of multilayer perceptrons neural network

Let $\mathbf{x}$ be the input which comprising of a column vector $(\mathbf{x_1},\mathbf{x_2},\ldots,\mathbf{x_p})^T$ while the superscript *T* denotes as matrix transpose. Let $\mathbf{w^m}$ be the weight where the superscript denotes as the layer between the input layer and hidden layer. The summation of input and weights is denoted as

$$\mathbf{a_i} = \sum_{\mathbf{i}=1}^{\mathbf{p}} \mathbf{x_i} \cdot \mathbf{w^m_{ij}} + \beta; \quad \mathbf{j} = 1,2,\cdots\mathbf{q} \tag{7.1}$$

Where $\mathbf{i}$ and $\mathbf{j}$ are the iterative variable for input and hidden neurons respectively, β is a bias term that regulates the degree of an activation to induce firing. The resultant value $\mathbf{a}$, acts as the input to $\mathbf{q}$ number of hidden neurons.

The activation functions $\mathbf{F}(.)$ and $\mathbf{H}(.)$ are implemented onto all weighted sum inputs for all neurons in the input layer and hidden layer or hidden layer and output layer. It usually refers to a sigmoid function due to the reason that it is a strictly increasing function that exhibits smoothness and has the desired asymptotic properties (Jain, Jianchang, and Mohiuddin 1996). Thus, the output of the hidden neurons denoted by $\mathbf{b}$ associated with hidden neuron $\mathbf{q}$ is written as:

$$\mathbf{b_j} = \mathbf{F}\left(\mathbf{a_j}\right) = \frac{1}{1+e^{(-\mathbf{a_j})}} \tag{7.2}$$

The input $\mathbf{c}$ for the neuron $\mathbf{r}$ in the output layers is calculated by multiplying the output of the hidden neurons $\mathbf{b}$ with the weight $\mathbf{w^n}$ (the superscript $\mathbf{n}$ denotes as the nodes between the hidden layer and the output layer) and $\mathbf{k}$ is the iterative variable for output neurons, where

$$\mathbf{c_k} = \sum_{\mathbf{j}=1}^{\mathbf{q}} \mathbf{b_j} \cdot \mathbf{w^n_{jk}} + \beta; \quad \mathbf{k} = 1,2,\cdots \mathbf{r} \tag{7.3}$$

The output $\mathbf{y}$ in the third layer transforms the input $\mathbf{c}$ using the sigmoid function, expressed by:

$$\mathbf{y_k} = \mathbf{G}(\mathbf{c_k}) = \frac{1}{1+e^{(-\mathbf{c_k})}} \tag{7.4}$$

The output $\mathbf{y}$ from the output layer is also known as the actual output. The notation

$\mathbf{d}$, on the other hand, denotes the desired output. The performance of MLP is evaluated by computing the difference between the actual output and the desire output. The difference is also known as error, which is denoted as ξ. The errors are then passed to the weight adaption rules, to adaptively update all weights in order to increase the performance of MLP by recognizing spam and non-spam. Scaled conjugate gradient algorithm is used as the weight adaption rule in this research; it will be thoroughly explained in the next section.

7.3 SCALED CONJUGATE GRADIENT

Weights updating algorithm is highly required to obtain an optimum solution for classifying a particular task. In this context, the task refers to web spam detection. One output neuron is set in the Web spam classification, where the output '0' indicates non-spam and the output '1' indicates spam. A set of training data with its relatively desired outputs is inserted to MLP neural network to iteratively adjust the weights based on the back propagated errors. Various weight updating techniques have been reviewed in (Levenberg 1944; Riedmiller 1994).

A supervised learning algorithm namely scaled conjugate gradient (SCG) (Møller 1993) is used for the reason that it has a faster learning speed and better performance than the standard back propagation algorithm (BP) (Rumelhart, Hinton, and Williams 1988), conjugate gradient algorithm with line search (CGL) (Johansson, Dowla, and Goodman 1991) and the one-step Broyden-Fletcher-Goldfarb-Shanno (BFGS) memoriless quasi-Newton algorithm (Battiti and Masulli 1990).

The notations and variables that are used throughout the algorithm descriptions are: $\mathbf{w}$ denotes the weight vector, σ denotes a scaling value which set between 0 and 10^{-4}, λ and $\bar{\lambda}$ denote another scaling values which are set between 0 and 10^{-6}, ψ denotes the conjugate gradient direction, ϑ denotes the steepest descent direction, $E(\omega)$

denotes the global error function, $E'(\omega)$ denotes the gradient to global error function, $E''(\omega)$ denotes the Hessian Matrix to global error function, u denotes the iterations, δ denotes the second order information, o denotes the step size, Δ_u denotes the comparison parameter and ℓ denotes the total number of weights linkage. Note that superscript T is denoted as transpose.

Assume the inputs $\mathbf{w}_1, \sigma$, λ are given and $\bar{\lambda}$ is set as 0, at the first iteration sets $u=1$ where,

$$\psi_u = \vartheta_u = -E'(\mathbf{w}_u) \tag{7.5}$$

Initially, the algorithm calculates the second order information δ_u where,

$$\sigma_u = \sigma / |\psi_u| \tag{7.6}$$

$$\delta_u = \psi_u^T \left(E'(\mathbf{w}_u) + \sigma_u \psi_u) - E'(\mathbf{w}_u)\right) / \sigma_u \tag{7.7}$$

After δ_u is calculated, the δ_u is then scaled where,

$$\delta_u = \delta_u + (\lambda_u - \bar{\lambda}_u) \cdot |\psi_u|^2 \tag{7.8}$$

If the δ_u is less or equal to 0, then the algorithm make the Hessian matrix positive definite such as,

$$\bar{\lambda}_u = 2(\lambda_u - \delta_u / |\psi_u|^2) \tag{7.9}$$

$$\delta_u = -\delta_u + \lambda_u |\psi_u|^2 \tag{7.10}$$

$$\lambda_u = \overline{\lambda_u} \tag{7.11}$$

After that the step size o_u and the comparison parameter Δ_u are calculated where,

$$o_u = (\psi_u^T r_u) / \delta_u \tag{7.12}$$

$$\Delta_u = \delta_u [E(\mathbf{w}_u) - E(\mathbf{w}_u + o_u \psi_u)] / (\psi_u^T \vartheta_u) \tag{7.13}$$

The comparison parameter Δ_u is used to check whether a reduction in error can be made, thus if Δ_u is greater or equal to 0, then the new weight vector is,

$$\mathbf{w}_{u+1} = \mathbf{w}_u + o_u \psi_u \tag{7.14}$$

$$\vartheta_{u+1} = -E'(\mathbf{w}_{u+1}) \tag{7.15}$$

$$\overline{\lambda_u} = 0 \tag{7.16}$$

At this point, if the number of iteration u is equal to the number of weights ℓ, then the algorithm is restarted with

$$\psi_{u+1} = \vartheta_{u+1} \tag{7.17}$$

Otherwise,

$$\psi_{u+1} = \vartheta_{u+1} + [(|\vartheta_{u+1}|^2 - \vartheta_{u+1}^T \vartheta_u) / (\psi_u^T \vartheta_u)] \cdot \psi_u \tag{7.18}$$

Even if the comparison parameter Δ_u is greater or equal to 0, if the comparison parameter Δ_u has a big value that is greater or equal to 0.75, the scale parameter λ is reduced, where

$$\lambda_u = \lambda_u / 4 \tag{7.19}$$

On the other hand, if the comparison parameter Δ_u is less than 0, then

$$\overline{\lambda}_u = \lambda_u \tag{7.20}$$

If the comparison parameter Δ_u has a small value that is lesser than 0.25, the scale parameter is increased where,

$$\lambda_u = \lambda_u + (\delta_u (1 - \Delta_u) / |\psi_u|^2) \tag{7.21}$$

After all these calculations, if the steepest descent direction ϑ is not equal to 0, the iteration u is increment with 1 and goes back to Equation 7.6. Otherwise, the desired weight $\mathbf{w}$ is given.

The SCG algorithm outperforms the BP algorithms as it does not need any user dependent parameters. Furthermore, the algorithm does not compute the expensive line search per learning iteration by using a step size scaling mechanism which makes SCG perform faster than CGL and BFGS.

7.4 EXPERIMENTAL RESULTS

In this section, an open source machine learning tool namely Weka (Hall et al. 2009), version 3.6 is used to conduct the experiments. The feature sets from two datasets - WEBSPAM-UK2006 and WEBSPAM-UK2007 are provided for this experiments. The feature sets are based on content-based such as Feature A and B, and link-based such as Feature C and D (Refer to Chapter 2 for more details on datasets and features). These features were fed into machine learning methods to evaluate the performance of web spam classification. The measurement unit in this section is AUC for the reason that it does not depend on any threshold like precision and recall (Erdélyi, Garzó, and Benczúr 2011).

Two machine learning methods were compared in the experiment, i.e. SVM and MLP. In SVM network structure, radial basis function (RBF) kernel is used for its promising performance as it non-linearly maps samples to a higher dimensional space. The sigma value of RBF is varied from 1 to 50 to obtain the optimal results. Besides RBF sigma, the scalar value are tweaked for soft margin to find a hyper plane that splits the examples as clean as possible; the range of the scalar value is set between 1 to 50. For MLP, the aforementioned scaled conjugate gradient algorithm is incorporated as a supervised learning algorithm. The weights between the neurons are randomly set between 0 and 1. Assume the datasets have K number of features; the model is executed based on 1000 epoch from 1 to K number of features. Since the weights between neurons are randomly generated, the process is executed 20 times to get the average for every epoch.

After gathering all experimental results, the main result in SVM is selected based on the best parameters given. As for MLP, the main result is calculated by averaging every single hidden neuron results of particular feature set.

Table 7-1: AUC results on WEBSPAM-UK2006

WEBSPAM-UK2006				
Feature Set	***Features***	***SVM***	***MLP***	***Improvement***
		AUC	***AUC***	
A	24	0.7511	0.7987	6.34
B	96	0.8084	0.8704	7.67
C	41	0.7280	0.8301	14.02
D	138	0.7988	0.8276	3.61
A + C	65	0.8051	0.8688	7.91
B + D	234	0.8387	0.8869	5.75

Table 7-2: AUC results on WEBSPAM-UK2007

WEBSPAM-UK2007				
Feature Set	***Features***	***SVM***	***MLP***	***Improvement***
		AUC	***AUC***	
A	24	0.6782	0.7025	3.53
B	96	0.7420	0.7470	0.67
C	41	0.6218	0.6236	0.29
D	138	0.6613	0.6691	1.18
A + C	65	0.7234	0.7352	1.63
B + D	234	0.7524	0.7685	2.13

The performance comparison of SVM and MLP network was tabulated in Table 7-1 and Table 7-2 using extracted features from WEBSPAM-UK2006 and WEBSPAM-UK2007 respectively In Table 7-1, it was obviously shown that MLP outperformed SVM for all features. Of all features classification performance, the greatest improvement comes from Feature C where MLP improve SVM for 14.02% on the AUC performance. For single set features (Feature A, B, C and D), the AUC results generated from SVM range from 0.73 to 0.81 whereas the AUC results generated from MLP range from 0.80 to 0.87. Regardless of this, the best AUC results come from the combination of Feature B and D as it gives 0.87 in SVM and 0.89 in MLP.

In Table 7-2, Feature A has the biggest improvement among all feature sets where MLP improved 3.53% over SVM on the AUC performance. Feature C in this dataset does not give much improvement unlike the one in Table 7-1. In spite of everything, the best AUC results come from the combination of Feature B and D as it gives 0.75 in SVM and 0.77 in MLP, a 2.13% Improvement.

From the observation of both Table 7-1 and 7-2, it showed that content-based features contributed higher AUC performance than link-based features. However, the best AUC performance comes from the combination of both content and link-based features.

Note that the AUC results for MLP in Table 7-1 and Table 7-2 are based on the average of all results from various hidden neurons number. In later experiments, all the AUC performances from various numbers of hidden neurons are presented based on different feature sets.

Figure 7.2 to Figure 7.7 shows the number of hidden neurons contributed to the varied of AUC performance in a certain pattern on various feature sets. Due to the variation of the results, quadratic function was used to model the AUC curve to a better representation. The highest performance point of each AUC curve was marked in a relative to its number of hidden neurons used in MLP network.

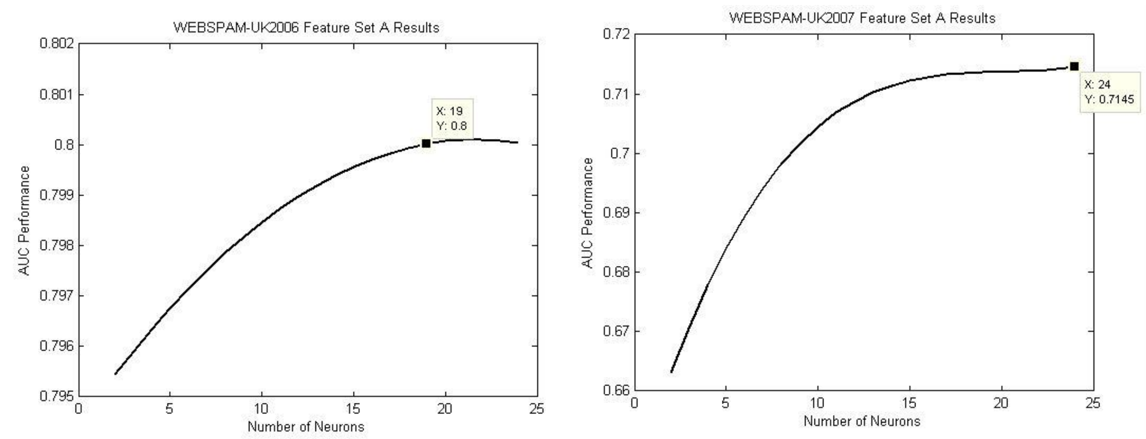


Figure 7.2: Feature set A AUC performance in WEBSPAM-UK2006 and WEBSPAM-UK2007

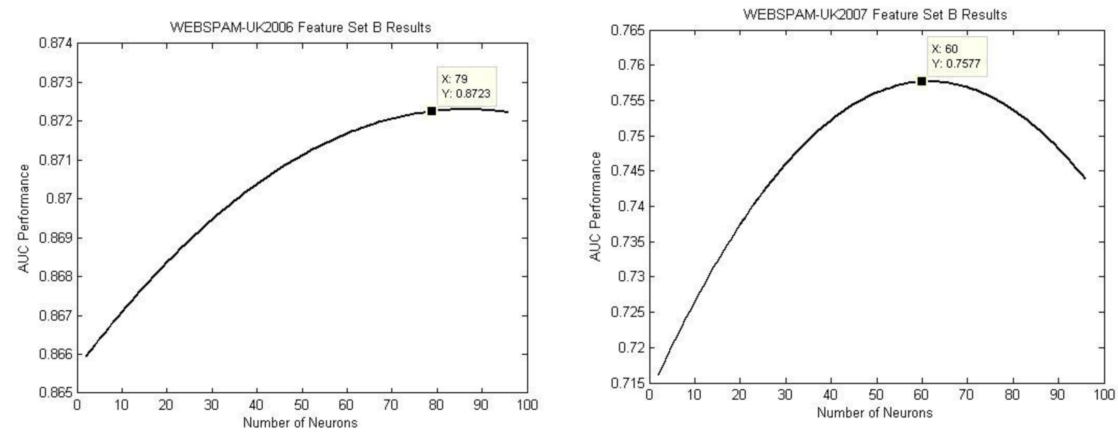


Figure 7.3: Feature set B AUC performance in WEBSPAM-UK2006 and WEBSPAM-UK2007

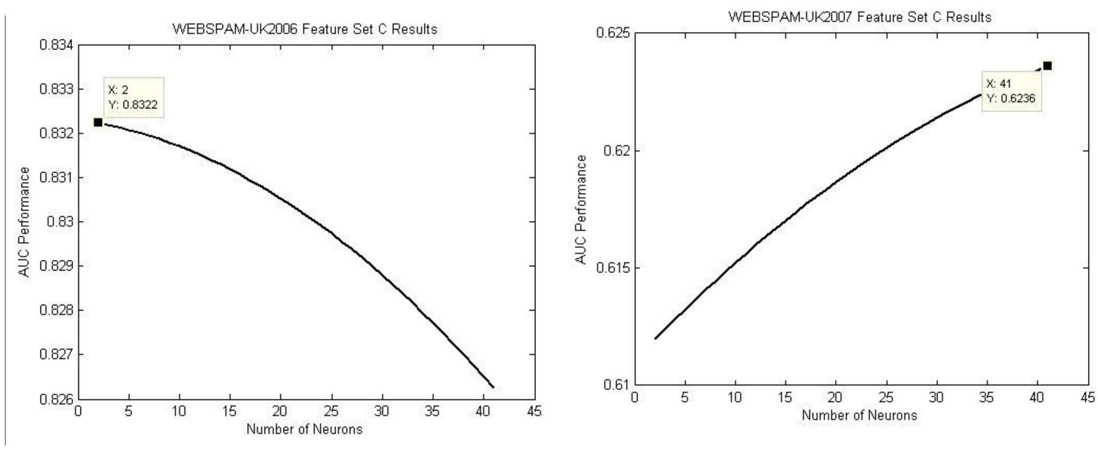


Figure 7.4: Feature set C AUC performance in WEBSPAM-UK2006 and WEBSPAM-UK2007

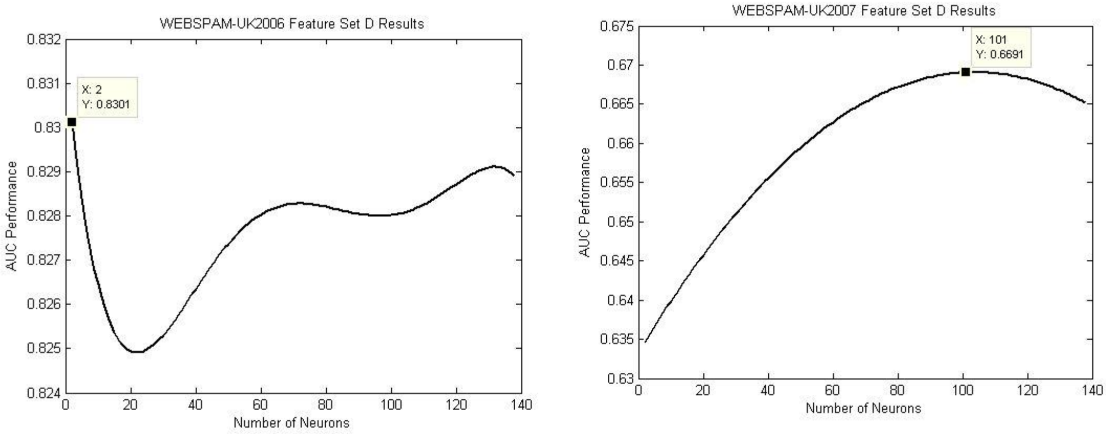


Figure 7.5: Feature set D AUC performance in WEBSPAM-UK2006

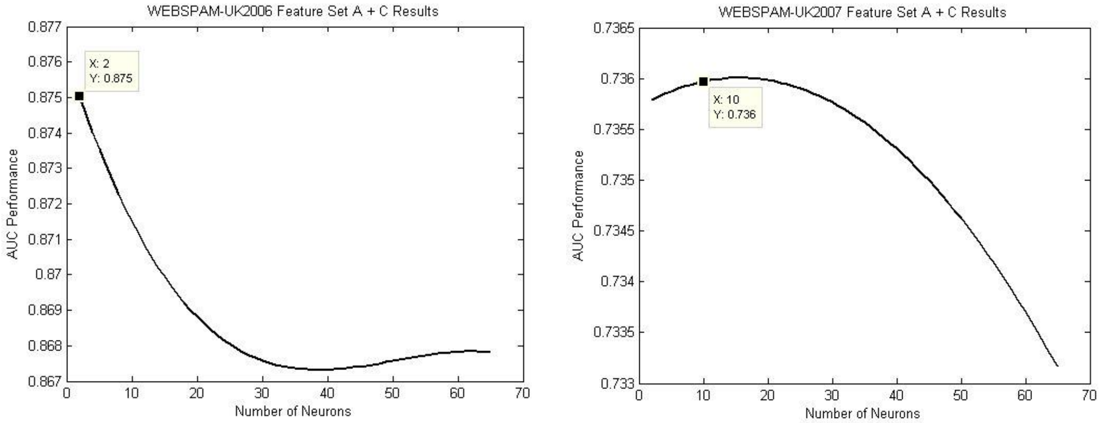


Figure 7.6: Feature set A + C AUC performance in WEBSPAM-UK2006 and WEBSPAM-UK2007

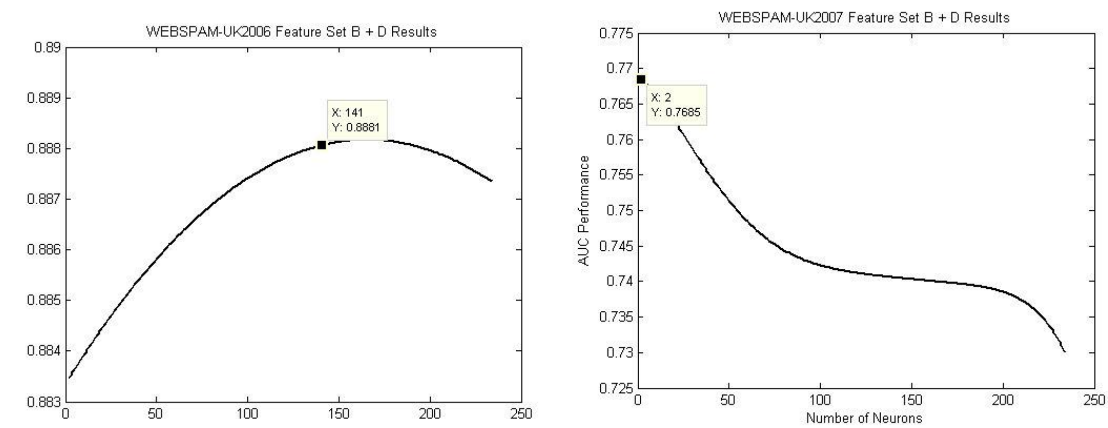


Figure 7.7: Feature set B + D AUC performance in WEBSPAM-UK2006 and WEBSPAM-UK2007

From the observation of Figure 7.2 to Figure 7.7, at most of the times, the highest plots either fall at the start or the end of the quadratic curves. In other cases, it falls slightly after the middle of the curve where the points reach convergences such as Feature B + D in WEBSPAM-UK2006, Feature B and Feature D in WEBSPAM-UK2007. Based on this observation and assume that there are K number of features, the convergences are estimated at $0.6 \cdot K$ of the curve.

Table 7-3: The AUC results on average, 2 neurons, $0.6 \cdot K$ neurons and K neurons from MLP in WEBSPAM-UK2006 and WEBSPAM-UK2007.

	WEBSPAM-UK2006				WEBSPAM-UK2007			
	Average	2	$0.6 \cdot K$	K	Average	2	$0.6 \cdot K$	K
A	0.7987	0.7953	0.7996	**0.8003**	0.7025	0.6590	0.7033	**0.7144**
B	0.8704	0.8597	0.8722	**0.8730**	0.7470	0.6695	**0.7612**	0.7532
C	0.8301	**0.8353**	0.8276	0.8272	0.6236	**0.6247**	0.6216	0.6229
D	0.8276	**0.8380**	0.8297	0.8291	0.6691	0.6583	**0.6762**	0.6628
A + C	0.8688	**0.8688**	0.8673	0.8677	0.7352	**0.7451**	0.7371	0.7423
B + D	0.8869	0.8847	**0.8881**	0.8878	0.7685	**0.7843**	0.7514	0.7194

Table 7-3 illustrates the AUC results on average AUC, 2 neurons, $0.6 \cdot K$ neurons and K neurons from MLP on WEBSPAM-UK2006 and WEBSPAM-UK2007. The highlighted bold results in Table 7-3 indicate the highest AUC performances among the results generated from the three fixed hidden neurons. As shown in Table 7-3, all highlighted bold results are actually having higher AUC than the average AUC which is shown in Table 7-1 and Table 7-2. This is a very significant finding for the reason that the computation becomes much slower when more features and more neurons are used. However, by plotting these three fixed number of hidden neurons – 2 neurons, $0.6 \cdot K$ neurons and K neurons, optimal AUC performance is achieved from MLP on Web spam detection. The full results of all experiments in this chapter can refer to APPENDIX G - Chapter 7 Results.

Abernethy et al. (Abernethy, Chapelle, and Castillo 2010) achieved 0.963 AUC using their proposed Web spam features while Li et al. (Li et al. 2011) have developed 10 new features generated by genetic programming that work better than 41 link-based features and 138 transformed link features. The authors' results are obtained on WEBSPAM-UK2006 using support vector machines. As it is indicated earlier in this chapter, the outcome of SVM is easily manipulated filtering (Biggio, Nelson, and Laskov 2011). Thus, an alternative classifier – MLP neural network is suggested for Web spam classification. Furthermore, the experiments have shown that MLP able to achieve better AUC performance than SVM in various feature sets.

Other features such as language models and qualified links achieved 0.88 and 0.76 for WEBSPAM-UK2006 and WEBSPAM-UK2007 using C4.5 Decision Tree (Araujo and Martinez-Romo 2010). In recent years, some researchers have shown that SVM works better than C4.5 Decision Tree (Yuchun et al. 2008; Abernethy, Chapelle, and Castillo 2010; Zhiyang et al. 2012); while in this chapter, it has shown that MLP works better than SVM. Using the standard feature sets, MLP achieved 0.8881 on WEBSPAM-UK2006 and 0.7842 on WEBSPAM-UK2007, it is suggested that the AUC performance using language models and qualified links features can be further improved using MLP network.

7.5 Summary

An alternative classification tool – MLP neural network is proposed in this chapter for Web spam classification. Scaled conjugate gradient algorithm is used to train MLP network for its fast learning speed and better performance than other supervised learning algorithm. Experimental results have shown that MLP network improve the AUC performance up to 14.02% on WEBSPAM-UK2006 and up to 3.53% on WEBSPAM-UK2007 over SVM based on various set of feature.

Determining the number of hidden neurons in MLP network is always a computational task as different number of input size requires a change in MLP hidden layer, which dramatically affects the performance of classification. Therefore, a mechanism of determining a MLP network structure has been proposed in this chapter. Instead of monitoring the AUC curve varied with the number of hidden neuron, the optimal performance for Web spam detection could actually be obtained by evaluating three types of hidden neuron numbers, i.e. 2 neurons, $0.6 \cdot K$ neurons and K neurons. The experiment has proved that one of these neuron numbers could achieve a promising performance with the highest point.

With the amount of Web spam features given, choosing the appropriate machine learning model is significant in order to perform the optimal detection rate for the classification task.

Chapter 8 Conclusion

Web spam has been heavily deteriorated the quality of Web search engines such as providing unrelated information to mislead Web users, and thus this disrupt the quality of search results provided by the search engines. In solving the aforementioned problem, this research involved the implementation of the link-based techniques to reduce or eliminate the problem arises by Web spam.

In Chapter 3, a comprehensive literature review on trust and distrust model algorithms and machine learning model are presented. TrustRank and its derivatives are selected as the model algorithms as TrustRank has provided more advantages in eliminating Web spam. The investigation of TrustRank and HostRank on WEBSPAM-UK2006 and WEBSPAM-UK2007 shows the vulnerability of HostRank in Web spam. The comparison of TrustRank and HostRank with 50, 75 and 100 non-spam seeds in WEBSPAM-UK2006 and with 100, 150 and 200 non-spam seeds in WEBSPAM-UK2007 are made to show the vulnerability of link-analysis algorithms when it comes to Web spam. Experimental results from TrustRank shows that it promoted up to 22.45% non-spam hosts based on 50 non-spam seeds in WEBSPAM-UK2006 and up to 1.08% based on 100 non-spam seeds in WEBSPAM-UK2007. In terms of non-spam Web pages promotion, TrustRank has promoted up to 26.63% non-spam Web pages over HostRank based on 50 non-spam seeds in WEBSPAM-UK2006 and up to 6.05% based on 100 non-spam seeds in WEBSPAM-UK2007. It is evident that TrustRank is able to achieve better performance when more seeds are used. TrustRank from WEBSPAM-UK2006 based on 100 non-spam seeds has improved HostRank up to 24.13% by promoting non-spam hosts and 30.47% by promoting Non-spam Web pages. It is found that for WEBSPAM-UK2007 based on 200 non-spam seeds, the non-spam host improvement over HostRank is the same as the one with 100 non-spam seeds. However, TrustRank with 200 non-spam seeds is able to achieve up to 14.50% in terms of non-spam Web pages promotion instead of 6.05% with 100 non-spam seeds.

In Chapter 4, two trust propagation algorithms namely Trust Propagation Rank (TPRank) and Trust Propagation Spam Mass (TP Spam Mass) are presented. The trust score for other unevaluated vertices are calculated based on the current evaluated vertices to both demote and detect Web spam. The sets of ugly vertices are introduced as a subset of non-spam vertices to assist in the algorithms. For the experiments, TPRank is compared with TrustRank while TP Spam Mass is compared with Spam Mass based on limited evaluated seeds in WEBSPAM-UK2006 and WEBSPAM-UK2007 datasets. As a result, TPRank outperforms TrustRank up to 10.88% on promoting non-spam hosts and achieves average promotion of 8.39 bucket per level for non-spam hosts in WEBSPAM-UK2006, and up to 1.08% on promoting non-spam hosts and achieves average promotion of 2.75 bucket per level for non-spam hosts in WEBSPAM-UK2007. In terms of non-spam Web pages, TPRank improves the promotion up to 2.14% in WEBSPAM-UK2006 and 2.27% in WEBSPAM-UK2007 over TrustRank. On the other hand, TP Spam Mass outperforms with Spam Mass up to 43.94% in WEBSPAM-UK2006 and up to 16.17% in WEBSPAM-UK2007 on detection of Web spam. In term of spam pages detection, TP Spam Mass has achieved up to 106.43% improvement in WEBSPAM-UK2006 and up to 668.54% improvement in WEBSPAM-UK2007.

In Chapter 5, a novel metric based on weight properties to enhance the detection rate of distrust based Web spam detection algorithms is presented. The novel metric calculates the weights based on the outgoing links of the vertices which indicate the relevancy linkage between two vertices. The weights used along with several distrust based Web spam detection algorithms such as Anti-TrustRank (Krishnan and Raj 2006), Wu et al. Distrust algorithm (Wu, Goel, and Davison 2006a) and Nie et al. Distrust algorithm (Nie, Wu, and Davison 2007) to detect additional Web spam. The results have shown the improvement on the detection of spam hosts up to 30.26% in Anti-TrustRank, 12.14% in Wu et al. Distrust and 10.92% in Nie et al. Distrust in WEBSPAM-UK2006, up to 31.30% in Anti-TrustRank, 26.38% in Wu et al. Distrust and 20.31% in Nie et al. Distrust in WEBSPAM-UK2007. In terms of Web spam pages, the weight properties have increased the detection rate for up to 39.76%, 13.14% and 11.34% in WEBSPAM-UK2006, and 8.81%, 6.76% and 4.76% in WEBSPAM-UK2007 on Anti-TrustRank, Wu et al. Distrust and Nie et al. Distrust algorithms.

In Chapter 6, distrust seed set propagation algorithm (DSP) which act as an extension to the spam seed set to calculate the distrust score for unevaluated vertices are introduced. There are several iterations derived from DSP. In the experiments, 10 iterations DSP are conducted on several Web spam detection algorithms, similarly to the experiments conducted as in Chapter 5. The results have shown that all 10 iterations of DSP have improved the detection of Web spam over the baseline algorithms. Furthermore, the results from DSP in WEBSPAM-UK2006 has improved up to 18.6%, 7.95% and 19.47%, and 6.94%, 24.78% and 25.17% in WEBSPAM-UK2007 on Anti-TrustRank, Wu et al. Distrust algorithm and Nie et al. Distrust algorithm respectively. In terms of Spam pages detection, DSP in WEBSPAM-UK2006 has made improvement up to 28.05%, 15.13% and 25.79% , and 5.33%, 4.04 and 8.75% in WEBSPAM-UK2007 on Anti-TrustRank, Wu et al. Distrust algorithm and Nie et al. Distrust algorithm respectively.

In Chapter 7, MLP neural network is proposed for Web spam classification due to its flexible structure and non-linearity transformation which can accommodate the latest Web spam patterns. From the experimental results, MLP is compared with the state-of-the-art SVM for Web spam classification. Scaled conjugate gradient training algorithm is applied to adaptively adjust the weight parameters in MLP network. As a result, MLP improve the AUC performance of SVM up to 14.02% in WEBSPAM-UK2006 and up to 3.53% in WEBSPAM-UK2007. Computing every single hidden neurons in MLP is computationally expensive. Thus assuming there are K number of features, it is found that 3 fixed numbers of hidden neurons as parameters are close to the optimal results – 2 neurons, K neurons and $0.6 \cdot K$ neurons.

For future work, it would be recommended to investigate the combination of the trust and distrust model techniques mentioned in above chapters (Chapter 3 to Chapter 6) to both detect and demote Web spam. The accumulation and splitting steps in this model are also crucial for the success of the algorithms, since limited studies are carried out in this area. Over the years, machine learning has grown rapidly and more Web spam features are constantly proposed for detection. In addition, finding alternatives for further enhancements and improvements on the multilayered perceptrons neural network can be incorporated into the link based technique which could increase the learning rate and detection rate.

References

Abbasi, Ahmed, Fatemeh, Mariam, Zahedi, and Siddharth Kaza. 2012. "Detecting Fake Medical Web Sites Using Recursive Trust Labeling." *ACM Trans. Inf. Syst.* 30 (4): 1-36. doi: 10.1145/2382438.2382441.

Abernethy, Jacob, Olivier Chapelle, and Carlos Castillo. 2010. "Graph Regularization Methods for Web Spam Detection." *Mach. Learn.* 81 (2): 207-225. doi: 10.1007/s10994-010-5171-1.

"About Rankdex." 1997. About Rankdex. Accessed 2 Jan 2013, http://www.rankdex.com/about.html.

Al-Kabi, Mohammed, Heider Wahsheh, Izzat Alsmadi, Emad Al-Shawakfa, Abdullah Wahbeh, and Ahmed Al-Hmoud. 2012. "Content-Based Analysis to Detect Arabic Web Spam." *J. Inf. Sci.* 38 (3): 284-296. doi: 10.1177/0165551512439173.

Araujo, Lourdes, and Juan Martinez-Romo. 2010. "Web Spam Detection: New Classification Features Based on Qualified Link Analysis and Language Models." *Trans. Info. For. Sec.* 5 (3): 581-590. doi: 10.1109/tifs.2010.2050767.

Baeza-Yates, Ricardo A., and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*: Addison-Wesley Longman Publishing Co., Inc.

Battiti, Roberto, and Francesco Masulli. 1990. "Bfgs Optimization for Faster and Automated Supervised Learning" *International Neural Network Conference, INNC 90 PARIS*: Dordrecht, Kluwer Academic Publishers.

Bauer, Johannes M., Michel J. G. van Eeten, and Yuehua Wu. 2008. "Itu Study on the Financial Aspects of Network Security: Malware and Spam." ICT Applications and Cybersecurity Division, International Telecommunication Union. http://www.itu.int/ITU-D/cyb/cybersecurity/docs/itu-study-financial-aspects-of-malware-and-spam.pdf.

Becchetti, Luca, Carlos Castillo, Debora Donato, Ricardo Baeza-YATES, and Stefano Leonardi. 2008. "Link Analysis for Web Spam Detection." *ACM Trans. Web* 2

(1): 1-42. doi: 10.1145/1326561.1326563.

Becchetti, Luca, Carlos Castillo, Debora Donato, Stefano Leonardi, and Ricardo Baeza-Yates. 2006a. "Link-Based Characterization and Detection of Web Spam" *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), Seattle, USA.*

———. 2006b. "Using Rank Propagation and Probabilistic Counting for Link-Based Spam Detection" *Proceedings of the Workshop on Web Mining and Web Usage Analysis (WebKDD 2006), Philadelphia, Pennsylvania, USA*: ACM Press.

———. 2008. "Web Spam Detection: Link-Based and Content-Based Techniques" *The European Integrated Project Dynamically Evolving, Large Scale Information Systems (DELIS): proceedings of the final workshop*.

Biggio, Battista, Blaine Nelson, and Pavel Laskov. 2011. "Support Vector Machines under Adversarial Label Noise" *JMLR: Workshop and Conference Proceedings 20, Taoyuan, Taiwan*: MIT Press.

———. 2012. "Poisoning Attacks against Support Vector Machines" *Proceeding of the 29th International Conference on Machine Learning (ICML 2012), Edinburgh, Scotland, Great Britain*: Omnipress.

Bíró, István, Dávid Siklósi, Jácint Szabó, and András A. Benczúr. 2009. "Linked Latent Dirichlet Allocation in Web Spam Filtering." In *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web, Madrid, Spain*, 37-40. 1531922: ACM. doi: 10.1145/1531914.1531922.

Borodin, Allan, Gareth O. Roberts, Jeffrey S. Rosenthal, and Panayiotis Tsaparas. 2005. "Link Analysis Ranking: Algorithms, Theory, and Experiments." *ACM Trans. Internet Technol.* 5 (1): 231-297. doi: 10.1145/1052934.1052942.

Brin, Sergey, and Lawrence Page. 1998. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." In *Proceedings of the seventh international conference on World Wide Web 7, Brisbane, Australia*, 107-117. 297827: Elsevier Science Publishers B. V.

Brinkmeier, Michael. 2006. "Pagerank Revisited." *ACM Trans. Internet Technol.* 6 (3): 282-301. doi: 10.1145/1151087.1151090.

Castillo, Carlos, Kumar Chellapilla, and Brian D. Davison. 2007. "Web Spam Challenge Track I." In *Proceedings of the 3rd international workshop on Adversarial Information Retrieval on the Web, Banff, Alberta, Canada*. ACM.

Castillo, Carlos, Kumar Chellapilla, and Ludovic Denoyer. 2008. "Web Spam Challenge 2008." In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb '08), Beijing, China*. ACM, New York, NY, USA.

Castillo, Carlos, and Brian D. Davison. 2011. "Adversarial Web Search." *Found. Trends Inf. Retr.* 4 (5): 377-486. doi: 10.1561/1500000021.

Castillo, Carlos, Brian D. Davison, Ludovic Denoyer, and Patrick Gallinari. 2007. "Web Spam Challenge Track Ii." In *ECML/PKDD Graph Labelling Workshop, Warsaw, Poland*.

Castillo, Carlos, Debora Donato, Luca Becchetti, Paolo Boldi, Massimo Santini, and Sebastiano Vigna. 2006. "A Reference Collection for Web Spam." *SIGIR Forum* 40 (2).

Castillo, Carlos, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. 2007. "Know Your Neighbors: Web Spam Detection Using the Web Topology." In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, Amsterdam, The Netherlands*, 423-430. 1277814: ACM. doi: 10.1145/1277741.1277814.

Chang, Chih-Chung, and Chih-Jen Lin. 2011. "Libsvm: A Library for Support Vector Machines." *ACM Trans. Intell. Syst. Technol.* 2 (3): 1-27. doi: 10.1145/1961189.1961199.

Cormen, Thomas H., Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. 2001. *Introduction to Algorithms*: McGraw-Hill Higher Education.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-Vector Networks." *Machine Learning* 20 (3): 273-297. doi: 10.1007/bf00994018.

Dai, Na, Brian D. Davison, and Xiaoguang Qi. 2009. "Looking into the Past to Better Classify Web Spam." In *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web, Madrid, Spain,* 1-8. 1531916: ACM. doi: 10.1145/1531914.1531916.

Egele, Manuel, Clemens Kolbitsch, and Christian Platzer. 2011. "Removing Web Spam Links from Search Engine Results." *Journal in Computer Virology* 7 (1): 51-62.

Eiron, Nadav, Kevin S. McCurley, and John A. Tomlin. 2004. "Ranking the Web Frontier." In *Proceedings of the 13th international conference on World Wide Web, New York, NY, USA*, 309-318. ACM, New York, NY, USA. doi:

10.1145/988672.988714.

Erdélyi, Miklós, András Garzó, and András A. Benczúr. 2011. "Web Spam Classification: A Few Features Worth More." In *Proceedings of the 2011 Joint WICOW/AIRWeb Workshop on Web Quality, Hyderabad, India*, 27-34. 1964121: ACM. doi: 10.1145/1964114.1964121.

Fetterly, Dennis, Mark Manasse, and Marc Najork. 2005. "Detecting Phrase-Level Duplication on the World Wide Web" *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*: ACM.

Gan, Qingqing, and Torsten Suel. 2007. "Improving Web Spam Classifiers Using Link Structure." In *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web, Banff, Alberta, Canada*, 17-20. 1244412: ACM. doi: 10.1145/1244408.1244412.

Geng, Guanggang, Pengfei Zhu, and Deliang Wang. 2009. "Web Spam Detection with Feature Fusion " *Journal of Computational Information Systems* 5 (3): 1511-1519.

Gil, Yolanda, and Donovan Artz. 2007. "Towards Content Trust of Web Resources." *Web Semantics: Science, Services and Agents on the World Wide Web* 5 (4): 227-239.

Gyöngyi, Zoltán, Pavel Berkhin, Hector Garcia-Molina, and Jan Pedersen. 2006. "Link Spam Detection Based on Mass Estimation" *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea*, 1164166: VLDB Endowment.

Gyöngyi, Zoltán, and Hector Garcia-Molina. 2005. "Web Spam Taxonomy" *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), Chiba, Japan*: http://ilpubs.stanford.edu:8090/646/.

Gyöngyi, Zoltán, Hector Garcia-Molina, and Jan Pedersen. 2004. "Combating Web Spam with Trustrank" *Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada*, 1316740: VLDB Endowment.

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. "The Weka Data Mining Software: An Update." *SIGKDD Explor. Newsl.* 11 (1): 10-18. doi: 10.1145/1656274.1656278.

Haveliwala, Taher H. 2002. "Topic-Sensitive Pagerank." In *Proceedings of the 11th international conference on World Wide Web, Honolulu, Hawaii, USA*, 517-526. 511513: ACM. doi: 10.1145/511446.511513.

Haykin, Simon. 1998. *Neural Networks: A Comprehensive Foundation*: Prentice Hall PTR.

Henzinger, Monika R., Rajeev Motwani, and Craig Silverstein. 2002. "Challenges in Web Search Engines." *SIGIR Forum* 36 (2): 11-22. doi: 10.1145/792550.792553.

Henzinger, Monika Rauch. 2000. "Link Analysis in Web Information Retrieval." *IEEE Data Engineering Bulletin* 23 (3): 3-8.

Jain, A. K., Mao Jianchang, and K. M. Mohiuddin. 1996. "Artificial Neural Networks: A Tutorial." *Computer* 29 (3): 31-44. doi: 10.1109/2.485891.

Jansen, Bernard J., Amanda Spink, and Tefko Saracevic. 2000. "Real Life, Real Users, and Real Needs: A Study and Analysis of User Queries on the Web." *Inf. Process. Manage.* 36 (2): 207-227. doi: 10.1016/s0306-4573(99)00056-4.

Jeh, Glen, and Jennifer Widom. 2003. "Scaling Personalized Web Search." In *Proceedings of the 12th international conference on World Wide Web, Budapest, Hungary*, 271-279. 775191: ACM. doi: 10.1145/775152.775191.

Jiang, Qiancheng, Lei Zhang, Yizhen Zhu, and Yan Zhang. 2008. "Larger Is Better: Seed Selection in Link-Based Anti-Spamming Algorithms" *Proceedings of the 17th International Conference on World Wide Web, Beijing, China*, 1367658: ACM, New York, NY, USA. doi: 10.1145/1367497.1367658.

Johansson, Erik M., Farid U. Dowla, and Dennis M. Goodman. 1991. "Backpropagation Learning for Multilayer Feed-Forward Neural Networks Using the Conjugate Gradient Method." *International Journal of Neural Systems* 2 (4): 291-301.

Kanich, Chris, Nicholas Weaver, Damon McCoy, Tristan Halvorson, Christian Kreibich, Kirill Levchenko, Vern Paxson, Geoffrey M Voelker, and Stefan Savage. 2011. "Show Me the Money: Characterizing Spam-Advertised Revenue" *USENIX Security Symposium*.

Kleinberg, Jon M. 1999. "Authoritative Sources in a Hyperlinked Environment." *J. ACM* 46 (5): 604-632. doi: 10.1145/324133.324140.

Kobayashi, Mei, and Koichi Takeda. 2000. "Information Retrieval on the Web." *ACM Comput. Surv.* 32 (2): 144-173. doi: 10.1145/358923.358934.

Kou, Zhenzhen. 2007. "Stacked Graphical Learning." University of Massachusetts Amherst.

Krishnan, Vijay, and Rashmi Raj. 2006. "Web Spam Detection with Anti-Trustrank" *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), Seattle, USA.*

László, A.A.B.K.C., and L.D. Siklósi. 2007. "Semi-Supervised Learning: A Comparative Study for Web Spam and Telephone User Churn" *Graph Labelling Workshop and Web Spam Challenge*.

Lempel, R., and S. Moran. 2001. "Salsa: The Stochastic Approach for Link-Structure Analysis." *ACM Trans. Inf. Syst.* 19 (2): 131-160. doi: 10.1145/382979.383041.

Leon-Suematsu, Yutaka I., Kentaro Inui, Sadao Kurohashi, and Yutaka Kidawara. 2011. "Web Spam Detection by Exploring Densely Connected Subgraphs." In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01, Campus Scientifique de la Doua, Lyon, France*, 124-129. 2052339: IEEE Computer Society. doi: 10.1109/wi-iat.2011.152.

Levenberg, Kenneth. 1944. "A Method for the Solution of Certain Non-Linear Problems in Least Squares." *Quarterly of Applied Mathematics* 2: 164-168.

Li, Longzhuang, Yi Shang, and Wei Zhang. 2002. "Improvement of Hits-Based Algorithms on Web Documents" *Proceedings of the 11th international conference on World Wide Web*: ACM.

Li, Shengen, Xiaofei Niu, Peiqi Li, and Lin Wang. 2011. "Generating New Features Using Genetic Programming to Detect Link Spam" *2011 International Conference on Intelligent Computation Technology and Automation (ICICTA), Shenzhen, China*, doi: 10.1109/icicta.2011.41.

Li, Yanhong. 1998. "Toward a Qualitative Search Engine." *IEEE Internet Computing* 2 (4): 24-29. doi: 10.1109/4236.707687.

Li, Zhao, Jiang Qiancheng, and Zhang Yan. 2008. "From Good to Bad Ones: Making Spam Detection Easier" *Computer and Information Technology Workshops, 2008. CIT Workshops 2008. IEEE 8th International Conference on*, doi: 10.1109/CIT.2008.Workshops.49.

Liang, Chenmin, Liyun Ru, and Xiaoyan Zhu. 2007. "R-Spamrank: A Spam Detection Algorithm Based on Link Analysis." *Journal of Computational Information Systems* 3 (4): 1705-1712.

Liu, Xinyue, You Wang, Shaoping Zhu, and Hongfei Lin. 2013. "Combating Web Spam through Trust-Distrust Propagation with Confidence." *Pattern Recognition Letters*.

Mahmoudi, M., A. Yari, and S. Khadivi. 2010. "Web Spam Detection Based on Discriminative Content and Link Features" *5th International Symposium on Telecommunications (IST), 2010*, doi: 10.1109/istel.2010.5734084.

Martinez-Romo, Juan, and Lourdes Araujo. 2009. "Web Spam Identification through Language Model Analysis." In *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web, Madrid, Spain*, 21-28. 1531920: ACM. doi: 10.1145/1531914.1531920.

Metaxas, Panagiotis. 2009a. "Using Propagation of Distrust to Find Untrustworthy Web Neighborhoods" *Internet and Web Applications and Services, 2009. ICIW'09. Fourth International Conference on*: IEEE.

Metaxas, Panagiotis Takis. 2009b. "On the Evolution of Search Engine Rankings" *WEBIST*.

Møller, Martin Fodslette. 1993. "Original Contribution: A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning." *Neural Netw.* 6 (4): 525-533. doi: 10.1016/s0893-6080(05)80056-5.

Nemirovsky, Danil, and Konstantin Avrachenkov. 2008. "Weighted Pagerank: Cluster-Related Weights." DTIC Document.

Netcraft. 2013. January 2013 Web Server Survey. Accessed 2 Jan 2013, http://news.netcraft.com/archives/2013/01/07/january-2013-web-server-survey-2.html.

Nie, Lan, Baoning Wu, and Brian D. Davison. 2007. "Winnowing Wheat from the Chaff: Propagating Trust to Sift Spam from the Web" *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands*, 1277950: ACM, New York, NY, USA. doi: 10.1145/1277741.1277950.

Noi, Lucia Di, Markus Hagenbuchner, Franco Scarselli, and Ah Chung Tsoi. 2010. "Web Spam Detection by Probability Mapping Graphsoms and Graph Neural Networks" *Proceedings of the 20th International Conference on Artificial*

Neural Networks: Part II, Thessaloniki, Greece, 1889052: Springer-Verlag, Germany.

Ntoulas, Alexandros, Marc Najork, Mark Manasse, and Dennis Fetterly. 2006. "Detecting Spam Web Pages through Content Analysis." In *Proceedings of the 15th international conference on World Wide Web, Edinburgh, Scotland*, 83-92. 1135794: ACM. doi: 10.1145/1135777.1135794.

Piskorski, Jakub, Marcin Sydow, and Dawid Weiss. 2008. "Exploring Linguistic Features for Web Spam Detection: A Preliminary Study." In *Proceedings of the 4th international workshop on Adversarial information retrieval on the web, Beijing, China*, 25-28. 1451990: ACM. doi: 10.1145/1451983.1451990.

Qi, Chen, Yu Song-Nian, and Cheng Sisi. 2008. "Link Variable Trustrank for Fighting Web Spam" *Proceedings of International Conference on Computer Science and Software Engineering, Wuhan, China,* doi: 10.1109/csse.2008.1099.

Quinlan, J. Ross. 1993. *C4.5: Programs for Machine Learning*: Morgan Kaufmann Publishers Inc.

Qureshi, Muhammad Atif. 2011. "Improving the Quality of Web Spam Filtering by Using Seed Refinement." Department of Computer Science Korea Advanced Institute of Science and Technology.

Riedmiller, Martin. 1994. "Advanced Supervised Learning in Multilayer Perceptrons - from Backpropagation to Adaptive Learning Algorithms." *Special Issue on Neural Networks, International Journal Computer Standards & Interfaces* 16 (3): 265-278.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1988. "Learning Representations by Back-Propagating Errors." In *Neurocomputing: Foundations of Research*, 696-699. MIT Press.

Shen, Guoyang, Bin Gao, Tie-Yan Liu, Guang Feng, Shiji Song, and Hang Li. 2006. "Detecting Link Spam Using Temporal Information." In *Proceedings of the Sixth International Conference on Data Mining*, 1049-1053. 1193305: IEEE Computer Society. doi: 10.1109/icdm.2006.51.

Sobek, Markus. 2002. Pr0 - Google's Pagerank 0 Penalty. Accessed 25 February, http://pr.efactory.de/e-pr0.shtml.

Sydow, Marcin, Jakub Piskorski, Dawid Weiss, and Carlos Castillo. 2007. "Application of Machine Learning in Combating Web Spam."

Tang, Y., Y. He, S. Krasser, and P. Judge. 2007. "Web Spam Challenge 2007 Track Ii Secure Computing Corporation Research" *Graph Labelling Workshop and Web Spam Challenge*.

Tian, Ye, Gary M. Weiss, and Qiang Ma. 2007. "A Semi-Supervised Approach for Web Spam Detection Using Combinatorial Feature-Fusion" *Proceeedings of The Graph Labelling Workshop and Web Spam Challenge, Warsaw, Poland.*

Wahsheh, Heider A., Mohammed N. Al-kabi, and Izzat M. Alsmadi. 2012. "Evaluating Arabic Spam Classifiers Using Link Analysis." In *Proceedings of the 3rd International Conference on Information and Communication Systems, Irbid, Jordan*, 1-5. 2222456: ACM. doi: 10.1145/2222444.2222456.

Wang, Wei, and Guosun Zeng. 2007. "Content Trust Model for Detecting Web Spam." In *Trust Management*, 139-152. Springer.

Wang, Wei, Guosun Zeng, Mingjun Sun, Huanan Gu, and Quan Zhang. 2007. "Evirank: An Evidence Based Content Trust Model for Web Spam Detection." In *Advances in Web and Network Technologies, and Information Management*, 299-307. Springer.

Wang, Yi-Min, Ming Ma, Yuan Niu, and Hao Chen. 2007. "Spam Double-Funnel: Connecting Web Spammers with Advertisers" *Proceedings of the 16th international conference on World Wide Web*: ACM.

Wu, Baoning. 2009. *Finding and Fighting Search Engine Spam: Algorithms and Evaluations*: VDM Verlag.

Wu, Baoning, and Brian D. Davison. 2005a. "Cloaking and Redirection: A Preliminary Study" *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), Chiba, Japan.*

———. 2005b. "Identifying Link Farm Spam Pages" *Proceedings of Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, Chiba, Japan*, 1062762: ACM, New York, NY, USA. doi: 10.1145/1062745.1062762.

Wu, Baoning, Vinay Goel, and Brian D. Davison. 2006a. "Propagating Trust and Distrust to Demote Web Spam." In *World Wide Web (WWW2006) Workshop on Models of Trust for the Web (MTW), Edinburgh, Scotland*.

———. 2006b. "Topical Trustrank: Using Topicality to Combat Web Spam" *Proceedings of the 15th international conference on World Wide Web, Edinburgh, Scotland,* 1135792: ACM, New York, NY, USA. doi:

10.1145/1135777.1135792.

Xiao, Han, Huang Xiao, and Claudia Eckert. 2012. "Adversarial Label Flips Attack on Support Vector Machines." In *20th European Conference on Artificial Intelligence (ECAI), Montpellier, France*.

Xing, Wenpu, and Ali Ghorbani. 2004. "Weighted Pagerank Algorithm" *Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on*: IEEE.

Yahoo! 2007. Web Spam Collections. http://barcelona.research.yahoo.net/webspam/datasets/.

Yang, Haixuan, Irwin King, and Michael R. Lyu. 2007. "Diffusionrank: A Possible Penicillin for Web Spamming" *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands*, 1277815: ACM, New York, NY, USA. doi: 10.1145/1277741.1277815.

Yuchun, Tang, S. Krasser, He Yuanchen, Yang Weilai, and D. Alperovitch. 2008. "Support Vector Machines and Random Forests Modeling for Spam Senders Behavior Analysis" *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, doi: 10.1109/glocom.2008.ecp.419.

Zhang, G. P. 2000. "Neural Networks for Classification: A Survey." *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 30 (4): 451-462. doi: 10.1109/5326.897072.

Zhang, Weifeng, Danmei Zhu, Yinzhou Zhang, Guoqiang Zhou, and Baowen Xu. 2011. "Harmonic Functions Based Semi-Supervised Learning for Web Spam Detection." In *Proceedings of the 2011 ACM Symposium on Applied Computing, TaiChung, Taiwan*, 74-75. 1982204: ACM. doi: 10.1145/1982185.1982204.

Zhang, Xianchao, Bo Han, and Wenxin Liang. 2009. "Automatic Seed Set Expansion for Trust Propagation Based Anti-Spamming Algorithms" *Proceedings of the Eleventh International Workshop on Web Information and Data Management, Hong Kong, China*, 1651596: ACM, New York, NY, USA. doi: 10.1145/1651587.1651596.

Zhang, Xianchao, You Wang, Nan Mou, and Wenxin Liang. 2011. "Propagating Both Trust and Distrust with Target Differentiation for Combating Web Spam" *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI-11), San Francisco, California*, conf/aaai/ZhangWML11: AAAI Press.

http://dblp.uni-trier.de/db/conf/aaai/aaai2011.html#ZhangWML11.

Zhang, Yan, Qiancheng Jiang, Lei Zhang, and Yizhen Zhu. 2009. "Exploiting Bidirectional Links: Making Spamming Detection Easier" *Proceedings of the 18th ACM conference on Information and knowledge management, Hong Kong, China*, 1646244: ACM. doi: 10.1145/1645953.1646244.

Zhiyang, Jia, Li Weiwei, Gao Wei, and Xia Youming. 2012. "Research on Web Spam Detection Based on Support Vector Machine" *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*, doi: 10.1109/csnt.2012.117.

Appendices

APPENDIX A - ASYMPTOTIC NOTATION

Θ *-notation*

For a given function $g(n)$, we denote by $\Theta(g(n))$ the set of functions if there exist positive constant c_1, c_2, and n_0 such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0$.

We say that $g(n)$ is an asymptotic tight bound for $f(n)$.

O *-notation*

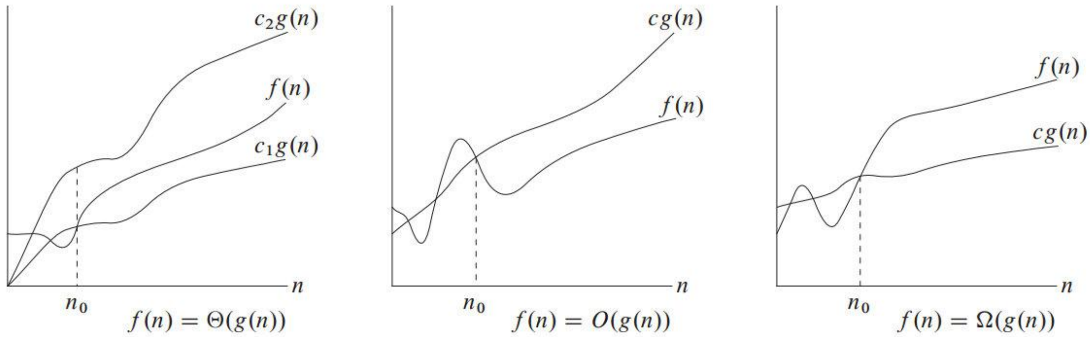
For a given function $g(n)$, we denote by $O(g(n))$ the set of functions if there exist positive constant c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$.

We say that $g(n)$ is an asymptotic upper bound for $f(n)$.

Ω *-notation*

For a given function $g(n)$, we denote by $\Omega(g(n))$ the set of functions if there exist positive constant c and n_0 such that $0 \le cg(n) \le f(n)$ for all $n \ge n_0$.

We say that $g(n)$ is an asymptotic lower bound for $f(n)$.



Graphic examples of the Θ, O, and Ω notations.

Taken from Cormen et al., Introduction to Algorithms (Cormen et al. 2001).

APPENDIX B – ADJACENCY-MATRIX REPRESENTATION

For the adjacency-matrix representation of a graph $G = (\upsilon, \varepsilon)$, we assume that the vertices are numbered $1,2,...,|\upsilon|$ in some arbitrary manner. Then the adjacency-matrix representation of a graph G consists of a $\upsilon \times \upsilon$ matrix $A = (a_{ij})$ such that

$$a_{ij} = \begin{cases} 1 & if\,(i,j) \in \varepsilon \\ 0 & otherwise \end{cases}$$

An adjacency matrix can represent a weighted graph. If $G = (\upsilon, \varepsilon)$ is a weighted graph with edge-weight function ω, we can simply store the weight $\omega(u,v)$ of the edge $(u,v) \in E$ as the entry in row u and column v of the adjacency matrix. If an edge does not exist, we can store a NIL value as its corresponding matrix entry, though for many problems it is convenient to use a value such as 0 or ∞.

Taken from Cormen et al., Introduction to Algorithms (Cormen et al. 2001) .

APPENDIX C - CHAPTER 3 RESULTS

WEBSPAM-UK2006

*HR – HostRank, TR50 – TrustRank 50, TR75 – TrustRank 75, TR100 – TrustRank 100, NS – Non-spam, S – spam

A. Number of non-spam and spam hosts in each bucket

Index	***HR***		***TR50***		***TR75***		***TR100***	
	NS	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***
1	354	19	373	0	373	0	373	0
2	362	11	368	5	369	4	373	0
3	297	76	364	9	372	1	368	5
4	185	188	362	11	364	9	373	0
5	315	58	347	26	359	14	359	14
6	338	35	320	53	352	21	357	16
7	310	63	311	62	321	52	352	21
8	313	60	293	80	301	72	307	66
9	310	63	271	102	278	95	302	71
10	307	66	286	87	278	95	259	114
11	301	72	262	111	270	103	265	108
12	306	67	242	131	237	136	252	121
13	292	81	237	136	223	150	216	157
14	275	98	221	152	221	152	207	166
15	253	120	194	179	201	172	214	159
16	219	154	205	168	170	203	159	214
17	167	206	201	172	208	165	182	191
18	157	216	252	121	246	127	226	147
19	198	175	219	154	191	182	191	182
20	290	96	221	165	215	171	214	172

B. Incremental summation of non-spam hosts for all buckets

Index	***HR***	***TR50***	***TR75***	***TR100***
1	354	373	373	373
2	716	741	742	746
3	1013	1105	1114	1114
4	1198	1467	1478	1487
5	1513	1814	1837	1846
6	1851	2134	2189	2203
7	2161	2445	2510	2555
8	2474	2738	2811	2862
9	2784	3009	3089	3164
10	3091	3295	3367	3423
11	3392	3557	3637	3688
12	3698	3799	3874	3940
13	3990	4036	4097	4156
14	4265	4257	4318	4363
15	4518	4451	4519	4577
16	4737	4656	4689	4736
17	4904	4857	4897	4918
18	5061	5109	5143	5144
19	5259	5328	5334	5335
20	5549	5549	5549	5549

C. Average promotion level for non-spam hosts and number of non-spam hosts being promoted over HostRank

	Average promotion level for non-spam hosts			***Number of non-spam hosts being promoted***		
Index	***TR50***	***TR75***	***TR100***	***TR50***	***TR75***	***TR100***
1	0.000	0.000	0.000	0	0	0
2	1.000	1.000	1.000	21	16	12
3	1.320	1.179	1.171	25	28	41
4	1.150	1.140	1.146	40	43	41
5	1.189	1.193	1.245	212	218	220
6	1.364	1.392	1.396	231	240	240
7	1.471	1.498	1.569	225	241	239
8	1.609	1.783	1.878	233	240	237
9	1.682	1.841	1.991	214	233	235
10	1.886	2.034	2.287	219	236	247
11	1.884	2.050	2.286	164	200	213
12	2.011	2.164	2.387	183	220	235
13	2.543	2.532	2.639	175	205	227
14	2.543	2.659	2.617	127	170	201
15	5.183	4.504	3.988	109	131	165
16	4.000	3.526	3.081	81	97	124
17	2.415	2.528	2.394	53	53	66
18	3.746	3.492	3.409	63	65	66
19	2.698	2.655	2.490	149	148	149
20	2.330	2.387	1.950	115	119	119

D. Evaluated non-spam host represented in pages level

Index	***HR***	***TR50***	***TR75***	***TR100***
1	4633746	5772732	5911439	6045772
2	7878928	9405003	9538061	9339090
3	10707594	12453809	12656345	12648555
4	11860496	15018597	15006317	14940319
5	14333206	17326257	17267232	17165337

E. Propagation coverage

	TR 50	***TR 75***	***TR 100***
$Sn(\upsilon_E)$	8564.00	8766.00	8922.00
$Sn(\upsilon_N)$	4223.00	4388.00	4519.00
$Sn(\upsilon_S)$	1374.00	1381.00	1384.00
η_N	86.20	90.01	92.84
η_S	13.80	9.99	7.16

WEBSPAM-UK2007

*HR – HostRank, TR100 – TrustRank 100, TR150 – TrustRank 150, TR200 – TrustRank 200, NS – Non-spam, S – spam

A. Number of non-spam and spam hosts in each bucket

	HR		***TR100***		***TR150***		***TR200***	
Index	***NS***	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***
1	465	9	470	4	466	8	470	4
2	467	8	469	6	469	6	467	7
3	466	10	465	9	471	3	468	7
4	460	14	468	7	469	6	469	5
5	466	8	457	17	461	14	461	15
6	447	27	460	16	455	20	460	15
7	447	28	453	21	451	23	451	23
8	450	24	460	14	453	21	447	27
9	444	30	448	26	458	16	450	24
10	451	24	437	37	443	31	452	22
11	445	29	438	36	434	40	441	33
12	448	26	437	37	438	37	432	43
13	449	26	446	29	439	35	434	40
14	449	25	443	31	442	32	445	29
15	440	34	452	22	450	24	452	22
16	441	33	445	29	449	25	450	24
17	444	30	446	28	443	31	443	31
18	435	39	446	28	445	29	443	31
19	445	29	422	53	424	51	427	48
20	421	48	418	51	420	49	418	51

B. Incremental summation of non-spam hosts for all buckets

Index	***HR***	***TR100***	***TR150***	***TR200***
1	465	470	466	470
2	932	939	935	937
3	1398	1404	1406	1405
4	1858	1872	1875	1874
5	2324	2329	2336	2335
6	2771	2789	2791	2795
7	3218	3242	3242	3246
8	3668	3702	3695	3693
9	4112	4150	4153	4143
10	4563	4587	4596	4595
11	5008	5025	5030	5036
12	5456	5462	5468	5468
13	5905	5908	5907	5902
14	6354	6351	6349	6347
15	6794	6803	6799	6799
16	7235	7248	7248	7249
17	7679	7694	7691	7692
18	8114	8140	8136	8135
19	8559	8562	8560	8562
20	8980	8980	8980	8980

C. Average promotion level for non-spam hosts and number of non-spam hosts being promoted over HostRank

	Average promotion level for non-spam hosts			***Number of non-spam hosts being promoted***		
Index	***TR100***	***TR150***	***TR200***	***TR100***	***TR150***	***TR200***
1	0.000	0.000	0.000	0	0	0
2	1.000	1.000	1.000	57	56	57
3	1.067	1.115	1.159	89	78	82
4	1.177	1.193	1.282	124	119	124
5	1.319	1.343	1.468	182	181	173
6	1.472	1.560	1.673	161	159	156
7	1.674	1.779	1.873	175	172	173
8	1.871	1.957	2.139	178	188	180
9	2.078	2.105	2.163	167	172	172
10	2.175	2.220	2.283	177	182	180
11	2.469	2.493	2.581	194	207	210
12	2.605	2.536	2.602	177	192	206
13	2.887	2.809	2.787	204	215	221
14	2.927	2.850	2.709	206	214	220
15	3.014	3.221	3.063	209	208	222
16	2.870	2.859	2.773	247	241	247
17	2.668	2.663	2.486	238	249	253
18	3.567	3.054	3.083	217	222	206
19	1.548	1.431	1.475	343	288	280
20	1.000	1.000	1.000	109	109	109

D. Evaluated non-spam host represented in pages level

Index	***HR***	***TR100***	***TR150***	***TR200***
1	5433512	5762322	6048337	6221612
2	9514864	9798875	10029708	9981872
3	12113172	12601227	12790826	12721565
4	14570132	15149195	15107725	15151087
5	16374587	16952324	17088503	17095291

E. Propagation coverage

	TR100	***TR150***	***TR200***
$Sn(\upsilon_E)$	73790	75629	76759
$Sn(\upsilon_N)$	6603	6780	6858
$Sn(\upsilon_S)$	242	249	260
η_N	98.98	99.15	99.26
η_S	1.02	0.85	0.74

APPENDIX D - CHAPTER 4 RESULTS

WEBSPAM-UK2006

*TR50 – TrustRank 50, TP50 – TPRank 50, SM – Spam Mass 50, TPSM – TP Spam Mass 50

A. Number of non-spam and spam hosts in each bucket

Index	***TR50***		***TP50***		***SM50***		***TPSM50***	
	NS	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***
1	373	0	372	1	204	169	194	179
2	368	5	370	3	142	231	127	246
3	364	9	369	4	213	160	78	295
4	362	11	364	9	232	141	84	289
5	347	26	363	10	189	184	153	220
6	320	53	364	9	264	109	178	195
7	311	62	364	9	215	158	210	163
8	293	80	366	7	222	151	253	120
9	271	102	357	16	254	119	300	73
10	286	87	345	28	297	76	339	34
11	262	111	310	63	325	48	364	9
12	242	131	257	116	340	33	361	12
13	237	136	215	158	355	18	365	8
14	221	152	211	162	338	35	369	4
15	194	179	181	192	344	29	367	6
16	205	168	189	184	341	32	360	13
17	201	172	134	239	345	28	362	11
18	252	121	85	288	319	54	356	17
19	219	154	123	250	324	49	355	18
20	221	165	210	176	286	100	374	12

B. Incremental summation of non-spam hosts for TR50 and TP50 and spam hosts for SM50 and TPSM50 for all buckets.

	Non-spam		***Spam***	
Index	***TR50***	***TP50***	***SM50***	***TPSM50***
1	373	372	169	179
2	741	742	400	425
3	1105	1111	560	720
4	1467	1475	701	1009
5	1814	1838	885	1229
6	2134	2202	994	1424
7	2445	2566	1152	1587
8	2738	2932	1303	1707
9	3009	3289	1422	1780
10	3295	3634	1498	1814
11	3557	3944	1546	1823
12	3799	4201	1579	1835
13	4036	4416	1597	1843
14	4257	4627	1632	1847
15	4451	4808	1661	1853
16	4656	4997	1693	1866
17	4857	5131	1721	1877
18	5109	5216	1775	1894
19	5328	5339	1824	1912
20	5549	5549	1924	1924

C. Average promotion level for non-spam hosts and spam hosts, and number of non-spam and spam hosts being promoted

TP50 over TR50		
Index	***Average promotion level for non-spam hosts***	***Number of non-spam hosts being promoted***
1	0.000	0
2	1.000	14
3	1.040	25
4	1.217	23
5	1.308	13
6	1.455	22
7	1.760	25
8	1.786	42
9	1.836	55
10	2.034	89
11	2.733	101
12	3.233	103
13	3.371	116
14	3.430	142
15	4.617	133
16	5.874	175
17	7.228	171
18	8.388	224
19	2.705	139
20	1.900	20

TPSM50 over SM50		
Index	***Average promotion level for spam hosts***	***Number of spam hosts being promoted***
1	0.000	44
2	1.000	22
3	1.000	23
4	1.000	19
5	1.462	17
6	1.906	17
7	2.496	12
8	2.959	7
9	3.021	11
10	3.826	6
11	3.804	2
12	4.871	2
13	6.000	3
14	6.886	0
15	8.125	5
16	8.417	5
17	8.286	0
18	10.038	1
19	10.383	0
20	9.959	0

D. Evaluated host represented in pages level

Index	***TR50***	***TP50***	***SM50***	***TPSM50***
1	5772732	5105405	151418	92751
2	9405003	8617492	1548226	1545822
3	12453809	11389610	2496185	3922350
4	15018597	13548654	2963797	5340260
5	17326257	15972170	3510497	6723305
6	19436425	18080304	3701269	7640583
7	21183616	19746985	4329449	8539854
8	23057903	21699476	4854187	9318171
9	24886572	23498479	5513986	9680861
10	26594747	25371805	5962484	9929897

E. Propagation coverage

	TR50	***TP50***
$Sn(S)$	8564.00	10183.00
$Sn(S_G)$	4223.00	5242.00
$Sn(S_B)$	1374.00	1553.00
η_G	86.20	98.02
η_B	13.80	1.98

WEBSPAM-UK2007

* TR100 – TrustRank 100, TP100 – TPRank 100, SM100 – Spam Mass 100, TPSM100 – TP Spam Mass 100

A. Number of non-spam and spam hosts in each bucket

Index	***TR100***		***TP100***		***SM100***		***TPSM100***	
	NS	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***
1	470	4	474	0	423	51	421	53
2	469	6	472	2	415	60	432	43
3	465	9	472	3	443	31	431	43
4	468	7	469	5	454	20	444	30
5	457	17	465	10	442	32	450	24
6	460	16	467	7	452	22	445	29
7	453	21	457	18	440	34	436	38
8	460	14	449	26	442	33	433	41
9	448	26	445	29	454	20	424	51
10	437	37	439	35	435	39	444	31
11	438	36	444	30	428	47	440	34
12	437	37	438	37	445	29	436	38
13	446	29	430	44	446	28	457	17
14	443	31	438	36	462	13	463	14
15	452	22	444	30	470	5	469	5
16	445	29	445	29	470	4	471	3
17	446	28	444	30	471	4	471	3
18	446	28	438	36	466	8	474	0
19	422	53	434	41	469	5	470	4
20	418	51	416	53	453	16	469	0

B. Incremental summation of non-spam hosts for TR100 and TP100 and spam hosts for SM100 and TPSM100 for all buckets.

	Non-spam		***Spam***	
Index	***TR100***	***TP100***	***SM100***	***TPSM100***
1	470	474	51	53
2	939	946	111	96
3	1404	1418	142	139
4	1872	1887	162	169
5	2329	2352	194	193
6	2789	2819	216	222
7	3242	3276	250	260
8	3702	3725	283	301
9	4150	4170	303	352
10	4587	4609	342	383
11	5025	5053	389	417
12	5462	5491	418	455
13	5908	5921	446	472
14	6351	6359	459	486
15	6803	6803	464	491
16	7248	7248	468	494
17	7694	7692	472	497
18	8140	8130	480	497
19	8562	8564	485	501
20	8980	8980	501	501

C. Average promotion level for non-spam hosts and spam hosts, and number of non-spam and spam hosts being promoted

TP 100 over TR 100		
Index	***Average promotion level for non-spam hosts***	***Number of non-spam hosts being promoted***
1	0.000	0
2	1.000	27
3	1.075	53
4	1.220	59
5	1.273	77
6	1.688	109
7	2.022	139
8	2.654	153
9	2.746	138
10	2.110	154
11	1.973	186
12	1.643	210
13	1.796	230
14	1.644	222
15	1.547	201
16	1.743	237
17	1.746	130
18	1.078	77
19	1.386	210
20	1.421	19

TPSM50 over SM50		
Index	***Average promotion level for spam hosts***	***Number of spam hosts being promoted***
1	0.000	0
2	1.000	5
3	1.000	21
4	1.000	7
5	1.583	12
6	1.100	10
7	1.500	14
8	1.364	11
9	2.000	8
10	2.138	29
11	2.161	31
12	2.714	21
13	2.957	23
14	2.556	9
15	3.200	5
16	3.250	4
17	4.750	4
18	4.375	8
19	5.600	5
20	5.875	16

D. Evaluated host represented in pages level

	Non-spam		***Spam***	
Index	***TR100***	***TP100***	***SM100***	***TPSM100***
1	5762322	5893117	5278	5288
2	9798875	9487775	6684	6436
3	12601227	12041690	7385	7286
4	15149195	14092532	8013	8866
5	16952324	16163390	9342	9310
6	18311165	17929892	9519	9532
7	19565437	19312567	16486	19081
8	20322920	20280246	19305	20145
9	21056879	20846941	20188	155152
10	21581252	21627362	63341	157259

E. Propagation coverage

	TR100	***TP100***
$Sn(S)$	73790	95192
$Sn(S_G)$	6603	7900
$Sn(S_B)$	242	353
η_G	98.981	99.536
η_B	1.019	0.464

APPENDIX E - CHAPTER 5 RESULTS

WEBSPAM-UK2006

*ATR – Anti-TrustRank, WATR – Weighted Anti-TrustRank, WU – Wu et al. Distrust, WWU – Weighted Wu et al. Distrust, NIE – Nie et al. Distrust, WNIE – Weighted Nie et al. Distrust

A. Number of spam hosts in each bucket

Index	***ATR***		***WATR***		***WU***		***WWU***		***NIE***		***WNIE***	
	NS	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***
1	30	343	45	328	31	342	52	321	32	341	45	328
2	193	180	97	276	152	221	110	263	135	238	106	267
3	311	62	258	115	282	91	244	129	292	81	256	117
4	345	28	313	60	329	44	326	47	319	54	306	67
5	358	15	334	39	346	27	320	53	346	27	331	42
6	351	22	349	24	331	42	354	19	354	19	351	22
7	351	22	360	13	365	8	354	19	329	44	360	13
8	341	32	363	10	359	14	359	14	362	11	361	12
9	346	27	362	11	341	32	360	13	350	23	360	13
10	342	31	358	15	353	20	358	15	350	23	361	12
11	343	30	362	11	350	23	358	15	354	19	360	13
12	342	31	360	13	346	27	364	9	354	19	360	13
13	311	62	355	18	344	29	354	19	352	21	359	14
14	335	38	356	17	349	24	341	32	345	28	354	19
15	297	76	331	42	327	46	334	39	331	42	334	39
16	194	179	220	153	202	171	236	137	202	171	219	154
17	163	210	130	243	146	227	129	244	146	227	130	243
18	210	163	210	163	210	163	210	163	210	163	210	163
19	192	181	192	181	192	181	192	181	192	181	192	181
20	194	192	194	192	194	192	194	192	194	192	194	192

B. Incremental summation of spam hosts for all buckets.

Index	***ATR***	***WATR***	***WU***	***WWU***	***NIE***	***WNIE***
1	343	328	342	321	341	328
2	523	604	563	584	579	595
3	585	719	654	713	660	712
4	613	779	698	760	714	779
5	628	818	725	813	741	821
6	650	842	767	832	760	843
7	672	855	775	851	804	856
8	704	865	789	865	815	868
9	731	876	821	878	838	881
10	762	891	841	893	861	893
11	792	902	864	908	880	906
12	823	915	891	917	899	919
13	885	933	920	936	920	933
14	923	950	944	968	948	952
15	999	992	990	1007	990	991
16	1178	1145	1161	1144	1161	1145
17	1388	1388	1388	1388	1388	1388
18	1551	1551	1551	1551	1551	1551
19	1732	1732	1732	1732	1732	1732
20	1924	1924	1924	1924	1924	1924

C. Average promotion level for spam hosts, and number of spam hosts being promoted

	Average promotion level for non-spam hosts			***Number of non-spam hosts being promoted***		
Index	***WATR over ATR***	***WWU over WU***	***WNIE over NIE***	***WATR over ATR***	***WWU over WU***	***WNIE over NIE***
1	0.000	0.000	0.000	0	0	0
2	1.000	1.000	1.000	85	76	88
3	1.578	1.510	1.093	45	49	43
4	1.833	1.897	1.457	18	29	35
5	1.929	2.095	2.000	14	21	13
6	3.381	1.528	2.182	21	36	11
7	3.944	2.667	2.500	18	3	42
8	5.033	4.429	4.300	30	7	10
9	5.708	4.600	3.733	24	25	15
10	6.407	5.067	6.176	27	15	17
11	6.160	7.474	4.462	25	19	13
12	7.667	5.450	4.714	27	20	14
13	7.153	4.150	4.417	59	20	12
14	6.250	5.500	3.765	28	10	17
15	4.107	5.800	3.421	28	30	19
16	2.750	2.444	1.231	24	36	13
17	1.000	1.000	1.000	48	48	48
18	0.000	0.000	0.000	0	0	0
19	0.000	0.000	0.000	0	0	0
20	0.000	0.000	0.000	0	0	0

D. Evaluated host represented in pages level

Index	***ATR***	***WATR***	***WU***	***WWU***	***NIE***	***WNIE***
1	1313291	1349790	1622823	1295424	1653512	1273941
2	2456555	2901785	2728892	2613922	2822220	2872809
3	2841133	3780350	3336290	3681108	3438744	3821501
4	3063894	4282150	3739390	4124931	3929221	4296223
5	3204105	4461991	3945059	4454984	3995253	4448293
6	3370887	4570601	4049277	4546177	4166130	4566529
7	3508984	4615278	4094948	4632867	4282662	4639370
8	3754869	4727802	4282774	4749903	4369946	4694204
9	4061543	4792877	4506142	4806178	4445917	4797769
10	4214193	4857809	4631553	4921995	4713938	4919528

WEBSPAM-UK2007

*ATR – Anti-TrustRank, WATR – Weighted Anti-TrustRank, WU – Wu et al. Distrust, WWU – Weighted Wu et al. Distrust, NIE – Nie et al. Distrust, WNIE – Weighted Nie et al. Distrust

A. Number of spam hosts in each bucket

Index	***ATR***		***WATR***		***WU***		***WWU***		***NIE***		***WNIE***	
	NS	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***
1	384	89	369	104	377	96	370	103	376	97	372	101
2	448	25	441	32	456	17	444	29	454	19	440	33
3	462	11	453	20	454	20	448	25	462	12	453	20
4	468	6	458	16	456	18	459	16	455	19	460	15
5	460	13	458	16	466	7	460	14	455	18	462	11
6	459	15	466	7	470	3	465	8	465	8	462	11
7	460	13	471	2	471	2	462	11	466	7	467	6
8	465	8	465	8	455	18	464	9	460	13	467	6
9	460	13	471	3	456	17	470	3	467	6	469	5
10	463	10	466	7	465	8	469	4	462	11	465	8
11	462	11	462	11	461	12	469	4	466	7	462	11
12	466	7	464	9	463	10	467	6	464	9	464	9
13	455	19	469	4	464	10	464	9	460	14	471	2
14	452	21	451	22	450	23	453	20	452	21	450	23
15	439	34	439	34	439	34	439	34	439	34	439	34
16	430	43	430	43	430	43	430	43	430	43	430	43
17	433	40	433	40	433	40	433	40	433	40	433	40
18	430	43	430	43	430	43	430	43	430	43	430	43
19	436	38	436	38	436	38	436	38	436	38	436	38
20	448	42	448	42	448	42	448	42	448	42	448	42

B. Incremental summation of spam hosts for all buckets.

Index	***ATR***	***WATR***	***WU***	***WWU***	***NIE***	***WNIE***
1	89	104	96	103	97	101
2	114	136	113	132	116	134
3	125	156	133	157	128	154
4	131	172	151	173	147	169
5	144	188	158	187	165	180
6	159	195	161	195	173	191
7	172	197	163	206	180	197
8	180	205	181	215	193	203
9	193	208	198	218	199	208
10	203	215	206	222	210	216
11	214	226	218	226	217	227
12	221	235	228	232	226	236
13	240	239	238	241	240	238
14	261	261	261	261	261	261
15	295	295	295	295	295	295
16	338	338	338	338	338	338
17	378	378	378	378	378	378
18	421	421	421	421	421	421
19	459	459	459	459	459	459
20	501	501	501	501	501	501

C. Average promotion level for spam hosts, and number of spam hosts being promoted

	Average promotion level for non-spam hosts			***Number of non-spam hosts being promoted***		
Index	***WATR over ATR***	***WWU over WU***	***WNIE over NIE***	***WATR over ATR***	***WWU over WU***	***WNIE over NIE***
1	0.000	0.000	0.000	0	0	0
2	1.000	1.000	1.000	14	9	10
3	1.500	1.273	1.125	8	11	8
4	2.167	1.625	1.750	6	8	12
5	2.800	2.500	2.500	10	6	10
6	2.929	1.000	2.833	14	1	6
7	2.583	3.000	2.800	12	2	5
8	4.250	4.059	3.615	4	17	13
9	5.077	5.000	2.000	13	16	4
10	4.400	6.000	3.500	5	3	8
11	5.000	3.857	3.250	7	7	4
12	5.333	3.125	2.000	3	8	7
13	2.800	6.556	2.600	15	9	10
14	2.500	2.500	2.000	2	4	2
15	0.000	0.000	0.000	0	0	0
16	0.000	0.000	0.000	0	0	0
17	0.000	0.000	0.000	0	0	0
18	0.000	0.000	0.000	0	0	0
19	0.000	0.000	0.000	0	0	0
20	0.000	0.000	0.000	0	0	0

D. Evaluated host represented in pages level

Index	*ATR*	*WATR*	*WU*	*WWU*	*NIE*	*WNIE*
1	929200	970876	947715	892901	960910	910721
2	1005378	1069127	1026238	1025691	1020807	1063244
3	1012441	1101626	1058704	1077870	1057994	1079379
4	1037990	1115851	1075044	1139643	1076417	1110244
5	1064091	1120334	1112379	1181418	1117387	1116103
6	1080294	1148771	1115044	1188373	1126445	1120537
7	1099606	1149016	1115320	1190689	1134084	1130236
8	1107402	1190866	1131098	1207496	1136708	1190857
9	1139103	1191074	1164160	1207522	1184168	1206430
10	1198849	1216639	1203615	1207579	1205119	1206584

APPENDIX F - CHAPTER 6 RESULTS

WEBSPAM-UK2006

*ATR – Anti-TrustRank, ATR8 – Anti-TrustRank 8th iteration DSP, WU – Wu et al. Distrust, WU8 – Wu et al. Distrust 8th iteration DSP, NIE – Nie et al. Distrust, NIE8 – Nie et al. Distrust 8th iteration DSP

A. Number of spam hosts in each bucket

Index	***ATR***		***ATR8***		***WU***		***WU8***		***NIE***		***NIE8***	
	NS	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***
1	30	343	27	346	31	342	37	336	33	340	48	325
2	193	180	153	220	152	221	134	239	133	240	97	276
3	311	62	268	105	282	91	242	131	293	80	215	158
4	345	28	338	35	329	44	331	42	318	55	285	88
5	358	15	341	32	346	27	353	20	347	26	338	35
6	351	22	341	32	331	42	348	25	354	19	347	26
7	351	22	346	27	365	8	335	38	329	44	363	10
8	341	32	349	24	359	14	353	20	362	11	364	9
9	346	27	342	31	341	32	357	16	351	22	362	11
10	342	31	337	36	353	20	356	17	349	24	367	6
11	343	30	355	18	350	23	366	7	354	19	367	6
12	342	31	354	19	346	27	350	23	354	19	362	11
13	311	62	354	19	344	29	354	19	352	21	365	8
14	335	38	351	22	349	24	349	24	345	28	370	3
15	297	76	348	25	327	46	338	35	331	42	351	22
16	194	179	186	187	202	171	204	169	202	171	206	167
17	244	129	244	129	227	146	227	146	227	146	227	146
18	242	131	242	131	242	131	242	131	242	131	242	131
19	138	235	138	235	138	235	138	235	138	235	138	235
20	135	251	135	251	135	251	135	251	135	251	135	251

B. Incremental summation of spam hosts for all buckets.

Index	***ATR***	***ATR8***	***WU***	***WU8***	***NIE***	***NIE8***
1	343	346	342	336	340	325
2	523	566	563	575	580	601
3	585	671	654	706	660	759
4	613	706	698	748	715	847
5	628	738	725	768	741	882
6	650	770	767	793	760	908
7	672	797	775	831	804	918
8	704	821	789	851	815	927
9	731	852	821	867	837	938
10	762	888	841	884	861	944
11	792	906	864	891	880	950
12	823	925	891	914	899	961
13	885	944	920	933	920	969
14	923	966	944	957	948	972
15	999	991	990	992	990	994
16	1178	1178	1161	1161	1161	1161
17	1307	1307	1307	1307	1307	1307
18	1438	1438	1438	1438	1438	1438
19	1673	1673	1673	1673	1673	1673
20	1924	1924	1924	1924	1924	1924

C. Average promotion level for spam hosts, and number of spam hosts being promoted

	Average promotion level for non-spam hosts			***Number of non-spam hosts being promoted***		
Index	***ATR8 over ATR***	***WU8 over WU***	***NIE8 over NIE***	***ATR8 over ATR***	***WU8 over WU***	***NIE8 over NIE***
1	0.000	0.000	0.000	0	0	0
2	1.000	1.000	1.000	32	45	73
3	1.074	1.524	1.244	27	42	41
4	1.417	1.313	1.472	12	32	36
5	2.000	1.550	2.200	7	20	20
6	2.900	1.333	2.722	20	6	18
7	3.063	6.000	3.048	16	1	42
8	4.087	3.250	5.000	23	8	10
9	4.700	4.500	4.864	20	22	22
10	5.333	3.091	6.500	24	11	22
11	4.304	7.105	5.278	23	19	18
12	6.040	3.895	5.167	25	19	18
13	4.768	2.400	7.211	56	15	19
14	5.065	1.900	6.550	31	10	20
15	3.486	4.100	6.185	35	20	27
16	3.375	1.500	2.765	24	4	17
17	1.000	0.000	0.000	1	0	0
18	0.000	0.000	0.000	0	0	0
19	0.000	0.000	0.000	0	0	0
20	0.000	0.000	0.000	0	0	0

D. Evaluated host represented in pages level

Index	***ATR***	***ATR8***	***WU***	***WU8***	***NIE***	***NIE8***
1	1313291	1298996	1622823	1588293	1653314	1539283
2	2456555	2694632	2728892	2856597	2832581	3003471
3	2841133	3535509	3336290	3841061	3438744	4171748
4	3063894	3845866	3739390	4132319	3929393	4619347
5	3204105	4024198	3945059	4279553	3995253	4944029
6	3370887	4316557	4049277	4488813	4166130	5240632
7	3508984	4448721	4094948	4591742	4282662	5284677
8	3754869	4529346	4282774	4676556	4369946	5352723
9	4061543	4832617	4506142	4780965	4440319	5536531
10	4214193	5004972	4631553	4939534	4713938	5540404

WEBSPAM-UK2007

*ATR – Anti-TrustRank, ATR9 – Anti-TrustRank 9th iteration DSP, WU – Wu et al. Distrust, WU7 – Wu et al. Distrust 7th iteration DSP, NIE – Nie et al. Distrust, NIE9 – Nie et al. Distrust 9th iteration DSP

A. Number of spam hosts in each bucket

Index	***ATR***		***ATR9***		***WU***		***WU7***		***NIE***		***NIE9***	
	NS	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***	***NS***	***S***
1	382	92	377	97	376	98	371	103	373	101	367	107
2	449	25	453	21	459	15	437	38	458	16	437	37
3	465	9	461	13	453	22	472	2	458	17	453	21
4	467	8	466	9	458	17	463	11	458	17	452	24
5	464	10	460	14	467	7	455	19	458	16	465	9
6	457	18	458	17	470	4	463	12	466	8	462	12
7	460	14	464	10	468	6	467	7	467	7	471	4
8	466	8	460	14	457	17	465	9	462	12	466	8
9	462	12	462	13	461	13	468	7	465	9	470	4
10	466	8	465	9	465	9	465	9	468	6	470	4
11	464	10	466	8	464	10	472	2	464	10	473	1
12	466	8	465	9	464	10	462	12	465	9	471	3
13	456	19	468	6	463	12	465	9	461	14	469	5
14	453	21	452	22	452	22	452	22	454	20	451	23
15	439	35	439	35	439	35	439	35	439	35	439	35
16	431	43	431	43	431	43	431	43	431	43	431	43
17	435	39	435	39	435	39	435	39	435	39	435	39
18	432	42	432	42	432	42	432	42	432	42	432	42
19	436	39	436	39	436	39	436	39	436	39	436	39
20	430	41	430	41	430	41	430	41	430	41	430	41

B. Incremental summation of spam hosts for all buckets.

Index	***ATR***	***ATR9***	***WU***	***WU7***	***NIE***	***NIE9***
1	92	97	98	103	101	107
2	117	118	113	141	117	144
3	126	131	135	143	134	165
4	134	140	152	154	151	189
5	144	154	159	173	167	198
6	162	171	163	185	175	210
7	176	181	169	192	182	214
8	184	195	186	201	194	222
9	196	208	199	208	203	226
10	204	217	208	217	209	230
11	214	225	218	219	219	231
12	222	234	228	231	228	234
13	241	240	240	240	242	239
14	262	262	262	262	262	262
15	297	297	297	297	297	297
16	340	340	340	340	340	340
17	379	379	379	379	379	379
18	421	421	421	421	421	421
19	460	460	460	460	460	460
20	501	501	501	501	501	501

C. Average promotion level for spam hosts, and number of spam hosts being promoted

	Average promotion level for non-spam hosts			***Number of non-spam hosts being promoted***		
Index	***ATR9 over ATR***	***WU7 over WU***	***NIE9 over NIE***	***ATR9 over ATR***	***WU7 over WU***	***NIE9 over NIE***
1	0.000	0.000	0.000	0	0	0
2	1.000	1.000	1.000	4	7	11
3	1.000	1.222	1.143	4	9	7
4	1.000	1.667	1.714	2	3	7
5	1.333	2.000	2.625	3	7	8
6	2.000	3.333	2.300	5	3	10
7	1.000	4.000	4.400	3	1	5
8	1.600	2.000	4.000	5	1	4
9	1.800	2.167	4.833	5	6	6
10	1.000	4.500	5.125	1	12	8
11	3.000	4.154	2.500	4	13	2
12	2.500	4.750	8.333	6	4	6
13	4.250	3.000	6.400	4	4	5
14	3.600	1.500	5.375	5	2	8
15	2.000	3.000	8.500	1	1	2
16	3.167	4.250	7.556	6	4	9
17	3.455	3.333	8.143	11	6	7
18	1.333	1.000	2.000	3	1	3
19	0.000	0.000	0.000	0	0	0
20	0.000	0.000	0.000	0	0	0

D. Evaluated host represented in pages level

Index	***ATR***	***ATR9***	***WU***	***WU7***	***NIE***	***NIE9***
1	937106	932239	973094	995268	1005285	1003788
2	1009426	1032782	1027670	1064206	1022486	1068011
3	1014809	1034819	1079967	1064527	1079508	1120374
4	1039525	1060942	1096300	1078740	1112970	1193032
5	1065252	1078895	1133635	1148592	1138692	1195269
6	1084178	1085877	1136315	1158840	1150000	1220140
7	1103333	1087342	1139671	1168950	1155144	1228255
8	1147566	1169866	1153311	1185852	1157171	1258372
9	1163808	1225869	1183984	1228032	1189461	1258837
10	1200058	1226594	1194601	1242814	1205620	1309149

APPENDIX G - CHAPTER 7 RESULTS

WEBSPAM-UK2006

**HD - Number of hidden neurons, AUC - Area under the receiver operating characteristic curve*

Feature Set A (24 Content Features)

HD	AUC	HD	AUC	HD	AUC
2	0.7953	11	0.7975	20	0.8000
3	0.7972	12	0.7993	21	0.8004
4	0.7953	13	0.7987	22	0.8006
5	0.7967	14	0.7996	23	0.8003
6	0.7972	15	0.8006		
7	0.7970	16	0.7997		
8	0.7978	17	0.7987		
9	0.7986	18	0.8003		
10	0.7993	19	0.7999		

Feature Set B (96 Full Content Features)

HD	AUC	HD	AUC	HD	AUC	HD	AUC	HD	AUC
2	0.7953	21	0.8688	40	0.8718	59	0.8704	78	0.8710
3	0.7972	22	0.8705	41	0.8714	60	0.8689	79	0.8734
4	0.7953	23	0.8691	42	0.8705	61	0.8730	80	0.8710
5	0.7967	24	0.8704	43	0.8708	62	0.8721	81	0.8740
6	0.7972	25	0.8706	44	0.8713	63	0.8716	82	0.8719
7	0.7970	26	0.8686	45	0.8708	64	0.8719	83	0.8738
8	0.7978	27	0.8694	46	0.8707	65	0.8715	84	0.8721
9	0.7986	28	0.8699	47	0.8693	66	0.8723	85	0.8728
10	0.7993	29	0.8705	48	0.8708	67	0.8710	86	0.8730
11	0.8677	30	0.8698	49	0.8692	68	0.8654	87	0.8740
12	0.8674	31	0.8705	50	0.8726	69	0.8690	88	0.8733
13	0.8681	32	0.8699	51	0.8709	70	0.8728	89	0.8711
14	0.8687	33	0.8698	52	0.8705	71	0.8717	90	0.8720
15	0.8694	34	0.8712	53	0.8733	72	0.8713	91	0.8722
16	0.8684	35	0.8702	54	0.8700	73	0.8736	92	0.8761
17	0.8675	36	0.8692	55	0.8702	74	0.8726	93	0.8715
18	0.8692	37	0.8695	56	0.8700	75	0.8723	94	0.8736
19	0.8684	38	0.8710	57	0.8715	76	0.8695	95	0.8707
20	0.8689	39	0.8708	58	0.8722	77	0.8737	96	0.8730

Feature Set C (41 Link-based Features)

HD	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***
2	0.8353	11	0.8321	20	0.8314	29	0.8284	38	0.8273
3	0.8333	12	0.8316	21	0.8315	30	0.8289	39	0.8274
4	0.8346	13	0.8325	22	0.8314	31	0.8264	40	0.8272
5	0.8329	14	0.8317	23	0.8301	32	0.8275	41	0.8272
6	0.8336	15	0.8310	24	0.8278	33	0.8288		
7	0.8336	16	0.8299	25	0.8276	34	0.8273		
8	0.8320	17	0.8301	26	0.8282	35	0.8273		
9	0.8318	18	0.8319	27	0.8305	36	0.8276		
10	0.8327	19	0.8314	28	0.8278	37	0.8282		

Feature Set D (138 Transformed Link-based Features)

HD	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***
2	0.8380	30	0.8270	58	0.8265	85	0.8262	113	0.8288
3	0.8276	31	0.8281	59	0.8282	86	0.8268	114	0.8264
4	0.8267	32	0.8270	60	0.8253	87	0.8301	115	0.8283
5	0.8282	33	0.8234	61	0.8285	88	0.8257	116	0.8296
6	0.8275	34	0.8245	62	0.8275	89	0.8294	117	0.8279
7	0.8257	35	0.8271	63	0.8281	90	0.8298	118	0.8267
8	0.8279	36	0.8249	64	0.8300	91	0.8304	119	0.8263
9	0.8266	37	0.8274	65	0.8280	92	0.8277	120	0.8318
10	0.8213	38	0.8289	66	0.8285	93	0.8284	121	0.8309
11	0.8269	39	0.8261	67	0.8285	94	0.8277	122	0.8280
12	0.8246	40	0.8327	68	0.8291	95	0.8270	123	0.8271
13	0.8254	41	0.8245	69	0.8292	96	0.8295	124	0.8275
14	0.8268	42	0.8239	70	0.8322	97	0.8280	125	0.8284
15	0.8216	43	0.8259	71	0.8283	98	0.8277	126	0.8299
16	0.8302	44	0.8249	72	0.8239	99	0.8263	127	0.8270
17	0.8228	45	0.8250	73	0.8264	100	0.8287	128	0.8320
18	0.8236	46	0.8242	74	0.8264	101	0.8271	129	0.8291
19	0.8256	47	0.8254	75	0.8255	102	0.8289	130	0.8271
20	0.8212	48	0.8261	76	0.8275	103	0.8277	131	0.8311
21	0.8240	49	0.8291	77	0.8272	104	0.8297	132	0.8281
22	0.8290	50	0.8278	78	0.8309	105	0.8290	133	0.8278
23	0.8264	51	0.8248	79	0.8283	106	0.8277	134	0.8261
24	0.8259	52	0.8261	80	0.8325	107	0.8269	135	0.8316
25	0.8249	53	0.8277	81	0.8278	108	0.8262	136	0.8282
26	0.8247	54	0.8282	82	0.8263	109	0.8293	137	0.8314
27	0.8289	55	0.8297	82	0.8297	110	0.8289	138	0.8291
28	0.8237	56	0.8290	83	0.8267	111	0.8323		
29	0.8255	57	0.8306	84	0.8265	112	0.8289		

Feature Set A + C (65 Features)

HD	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***
2	0.8688	15	0.8693	28	0.8690	41	0.8676	54	0.8686
3	0.8745	16	0.8706	29	0.8684	42	0.8676	55	0.8666
4	0.8773	17	0.8694	30	0.8689	43	0.8682	56	0.8671
5	0.8760	18	0.8682	31	0.8690	44	0.8673	57	0.8675
6	0.8754	19	0.8673	32	0.8663	45	0.8674	58	0.8666
7	0.8735	20	0.8672	33	0.8663	46	0.8680	59	0.8689
8	0.8716	21	0.8676	34	0.8660	47	0.8688	60	0.8688
9	0.8715	22	0.8680	35	0.8677	48	0.8675	61	0.8670
10	0.8724	23	0.8691	36	0.8675	49	0.8676	62	0.8684
11	0.8716	24	0.8675	37	0.8669	50	0.8669	63	0.8678
12	0.8706	25	0.8676	38	0.8664	51	0.8679	64	0.8677
13	0.8711	26	0.8676	39	0.8673	52	0.8678	65	0.8677
14	0.8700	27	0.8691	40	0.8671	53	0.8674		

Feature Set B + D (234 Features)

HD	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***
2	0.8847	49	0.8871	96	0.8863	143	0.8889	190	0.8868
3	0.8894	50	0.8864	97	0.8899	144	0.8863	191	0.8878
4	0.8878	51	0.8850	98	0.8876	145	0.8874	192	0.8887
5	0.8798	52	0.8899	99	0.8870	146	0.8864	193	0.8874
6	0.8770	53	0.8868	100	0.8885	147	0.8888	194	0.8876
7	0.8816	54	0.8870	101	0.8879	148	0.8869	195	0.8871
8	0.8798	55	0.8880	102	0.8858	149	0.8897	196	0.8869
9	0.8831	56	0.8881	103	0.8870	150	0.8854	197	0.8892
10	0.8775	57	0.8882	104	0.8844	151	0.8888	198	0.8894
11	0.8787	58	0.8864	105	0.8866	152	0.8867	199	0.8886
12	0.8828	59	0.8901	106	0.8902	153	0.8905	200	0.8897
13	0.8803	60	0.8883	107	0.8885	154	0.8878	201	0.8876
14	0.8817	61	0.8869	108	0.8854	155	0.8872	202	0.8884
15	0.8855	62	0.8875	109	0.8868	156	0.8889	203	0.8878
16	0.8827	63	0.8854	110	0.8880	157	0.8885	204	0.8893
17	0.8842	64	0.8886	111	0.8872	158	0.8885	205	0.8871
18	0.8836	65	0.8861	112	0.8880	159	0.8880	206	0.8874
19	0.8859	66	0.8879	113	0.8845	160	0.8870	207	0.8870
20	0.8846	67	0.8848	114	0.8878	161	0.8885	208	0.8858
21	0.8854	68	0.8874	115	0.8902	162	0.8877	209	0.8868
22	0.8880	69	0.8861	116	0.8874	163	0.8878	210	0.8900
23	0.8814	70	0.8873	117	0.8855	164	0.8871	211	0.8876
24	0.8837	71	0.8886	118	0.8883	165	0.8878	212	0.8869

25	0.8866	72	0.8853	119	0.8866	166	0.8875	213	0.8891
26	0.8829	73	0.8861	120	0.8894	167	0.8879	214	0.8893
27	0.8856	74	0.8865	121	0.8890	168	0.8880	215	0.8890
28	0.8836	75	0.8863	122	0.8892	169	0.8885	216	0.8861
29	0.8875	76	0.8878	123	0.8896	170	0.8865	217	0.8885
30	0.8881	77	0.8880	124	0.8880	171	0.8891	218	0.8873
31	0.8851	78	0.8869	125	0.8878	172	0.8886	219	0.8898
32	0.8866	79	0.8889	126	0.8856	173	0.8874	220	0.8888
33	0.8824	80	0.8843	127	0.8867	174	0.8870	221	0.8854
34	0.8865	81	0.8870	128	0.8864	175	0.8851	222	0.8872
35	0.8875	82	0.8863	129	0.8879	176	0.8883	223	0.8909
36	0.8867	83	0.8882	130	0.8874	177	0.8873	224	0.8870
37	0.8869	84	0.8851	131	0.8867	178	0.8870	225	0.8881
38	0.8844	85	0.8882	132	0.8869	179	0.8866	226	0.8861
39	0.8876	86	0.8879	133	0.8869	180	0.8887	227	0.8882
40	0.8897	87	0.8877	134	0.8903	181	0.8896	228	0.8884
41	0.8861	88	0.8872	135	0.8870	182	0.8904	229	0.8867
42	0.8848	89	0.8862	136	0.8885	183	0.8881	230	0.8887
43	0.8841	90	0.8870	137	0.8873	184	0.8889	231	0.8881
44	0.8888	91	0.8863	138	0.8880	185	0.8855	232	0.8898
45	0.8844	92	0.8849	139	0.8876	186	0.8881	233	0.8870
46	0.8856	93	0.8887	140	0.8881	187	0.8883	234	0.8878
47	0.8858	94	0.8872	141	0.8874	188	0.8881		
48	0.8851	95	0.8867	142	0.8883	189	0.8864		

WEBSPAM-UK2007

**HD* - Number of hidden neurons, AUC - Area under the receiver operating characteristic curve*

Feature Set A (24 Content Features)

HD	***AUC***	***HD***	***AUC***	***HD***	***AUC***
2	0.6590	11	0.7083	20	0.7087
3	0.6730	12	0.7047	21	0.7200
4	0.6796	13	0.7106	22	0.7127
5	0.6888	14	0.7033	23	0.7078
6	0.6862	15	0.7153		
7	0.6936	16	0.7154		
8	0.6995	17	0.7137		
9	0.7012	18	0.7137		
10	0.7055	19	0.7188		

Feature Set B (96 Full Content Features)

HD	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***
2	0.6695	21	0.7514	40	0.7557	59	0.7547	78	0.7533
3	0.6806	22	0.7498	41	0.7546	60	0.7459	79	0.7552
4	0.6854	23	0.7440	42	0.7416	61	0.7502	80	0.7537
5	0.6914	24	0.7471	43	0.7494	62	0.7623	81	0.7503
6	0.7341	25	0.7475	44	0.7596	63	0.7532	82	0.7438
7	0.7224	26	0.7565	45	0.7543	64	0.7534	83	0.7598
8	0.7297	27	0.7453	46	0.7515	65	0.7502	84	0.7485
9	0.7419	28	0.7454	47	0.7561	66	0.7517	85	0.7638
10	0.7399	29	0.7491	48	0.7535	67	0.7636	86	0.7567
11	0.7440	30	0.7410	49	0.7537	68	0.7293	87	0.7543
12	0.7415	31	0.7429	50	0.7514	69	0.7551	88	0.7503
13	0.7508	32	0.7596	51	0.7542	70	0.7532	89	0.7518
14	0.7383	33	0.7525	52	0.7514	71	0.7402	90	0.7507
15	0.7495	34	0.7521	53	0.7561	72	0.7548	91	0.7498
16	0.7361	35	0.7503	54	0.7480	73	0.7639	92	0.7571
17	0.7339	36	0.7535	55	0.7586	74	0.7482	93	0.7461
18	0.7427	37	0.7548	56	0.7487	75	0.7566	94	0.7413
19	0.7482	38	0.7579	57	0.7587	76	0.7525	95	0.7531
20	0.7438	39	0.7457	58	0.7612	77	0.7477	96	0.7532

Feature Set C (41 Link-based Features)

HD	AUC	HD	AUC	HD	AUC	HD	AUC	HD	AUC
2	0.6247	11	0.6162	20	0.6233	29	0.6195	38	0.6248
3	0.6218	12	0.6142	21	0.6236	30	0.6192	39	0.6234
4	0.6087	13	0.6143	22	0.6196	31	0.6242	40	0.6229
5	0.6025	14	0.6184	23	0.6224	32	0.6206	41	0.6229
6	0.6103	15	0.6191	24	0.6164	33	0.6191		
7	0.6027	16	0.6193	25	0.6216	34	0.6218		
8	0.6125	17	0.6180	26	0.6242	35	0.6191		
9	0.6149	18	0.6130	27	0.6212	36	0.6226		
10	0.6169	19	0.6193	28	0.6226	37	0.6240		

Feature Set D (138 Transformed Link-based Features)

HD	AUC	HD	AUC	HD	AUC	HD	AUC	HD	AUC
2	0.6583	30	0.6536	58	0.6641	85	0.6690	113	0.6663
3	0.6553	31	0.6549	59	0.6637	86	0.6686	114	0.6679
4	0.6561	32	0.6539	60	0.6673	87	0.6599	115	0.6701
5	0.6602	33	0.6559	61	0.6724	88	0.6650	116	0.6741
6	0.6280	34	0.6575	62	0.6614	89	0.6630	117	0.6696
7	0.6258	35	0.6583	63	0.6594	90	0.6713	118	0.6720
8	0.6394	36	0.6513	64	0.6679	91	0.6684	119	0.6715
9	0.6256	37	0.6498	65	0.6639	92	0.6687	120	0.6671
10	0.6384	38	0.6325	66	0.6678	93	0.6692	121	0.6660
11	0.6371	39	0.6330	67	0.6657	94	0.6604	122	0.6709
12	0.6290	40	0.6549	68	0.6632	95	0.6691	123	0.6740
13	0.6330	41	0.6524	69	0.6603	96	0.6699	124	0.6671
14	0.6352	42	0.6612	70	0.6607	97	0.6685	125	0.6674
15	0.6407	43	0.6610	71	0.6580	98	0.6669	126	0.6695
16	0.6400	44	0.6609	72	0.6658	99	0.6740	127	0.6682
17	0.6458	45	0.6630	73	0.6665	100	0.6680	128	0.6643
18	0.6491	46	0.6527	74	0.6697	101	0.6639	129	0.6698
19	0.6430	47	0.6672	75	0.6642	102	0.6664	130	0.6696
20	0.6411	48	0.6677	76	0.6568	103	0.6690	131	0.6720
21	0.6467	49	0.6629	77	0.6696	104	0.6632	132	0.6561
22	0.6442	50	0.6566	78	0.6700	105	0.6660	133	0.6699
23	0.6517	51	0.6560	79	0.6654	106	0.6720	134	0.6699
24	0.6523	52	0.6616	80	0.6620	107	0.6669	135	0.6661
25	0.6492	53	0.6642	81	0.6615	108	0.6658	136	0.6677
26	0.6534	54	0.6587	82	0.6596	109	0.6708	137	0.6678
27	0.6502	55	0.6532	82	0.6762	110	0.6652	138	0.6624
28	0.6550	56	0.6724	83	0.6628	111	0.6630		
29	0.6559	57	0.6581	84	0.6641	112	0.6722		

Feature Set A + C (65 Features)

HD	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***
2	0.7451	15	0.7536	28	0.7290	41	0.7298	54	0.7179
3	0.7343	16	0.7526	29	0.7442	42	0.7246	55	0.7388
4	0.7220	17	0.7411	30	0.7328	43	0.7337	56	0.7481
5	0.7374	18	0.7447	31	0.7361	44	0.7275	57	0.7424
6	0.7464	19	0.7464	32	0.7305	45	0.7463	58	0.7347
7	0.7375	20	0.7234	33	0.7336	46	0.7470	59	0.7304
8	0.7229	21	0.7506	34	0.7257	47	0.7532	60	0.7364
9	0.7248	22	0.7436	35	0.7299	48	0.7306	61	0.7416
10	0.7255	23	0.7356	36	0.7345	49	0.7521	62	0.7274
11	0.7326	24	0.7402	37	0.7272	50	0.7166	63	0.7249
12	0.7302	25	0.7397	38	0.7367	51	0.7289	64	0.7331
13	0.7340	26	0.7226	39	0.7371	52	0.7193	65	0.7423
14	0.7325	27	0.7375	40	0.7457	53	0.7275		

Feature Set B + D (234 Features)

HD	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***	***HD***	***AUC***
2	0.7843	49	0.7545	96	0.7440	143	0.7516	190	0.7421
3	0.7807	50	0.7625	97	0.7548	144	0.7374	191	0.7256
4	0.7640	51	0.7572	98	0.7441	145	0.7401	192	0.7325
5	0.7648	52	0.7503	99	0.7433	146	0.7516	193	0.7412
6	0.7722	53	0.7392	100	0.7447	147	0.7378	194	0.7419
7	0.7627	54	0.7406	101	0.7547	148	0.7273	195	0.7445
8	0.7631	55	0.7479	102	0.7339	149	0.7320	196	0.7229
9	0.7574	56	0.7375	103	0.7355	150	0.7433	197	0.7373
10	0.7561	57	0.7473	104	0.7386	151	0.7159	198	0.7270
11	0.7539	58	0.7399	105	0.7591	152	0.7437	199	0.7412
12	0.7698	59	0.7359	106	0.7534	153	0.7302	200	0.7443
13	0.7634	60	0.7465	107	0.7358	154	0.7451	201	0.7368
14	0.7624	61	0.7549	108	0.7406	155	0.7406	202	0.7346
15	0.7740	62	0.7551	109	0.7381	156	0.7267	203	0.7491
16	0.7537	63	0.7377	110	0.7368	157	0.7292	204	0.7230
17	0.7635	64	0.7495	111	0.7337	158	0.7308	205	0.7383
18	0.7574	65	0.7532	112	0.7352	159	0.7361	206	0.7341
19	0.7731	66	0.7362	113	0.7538	160	0.7385	207	0.7376
20	0.7624	67	0.7435	114	0.7390	161	0.7340	208	0.7396
21	0.7538	68	0.7409	115	0.7512	162	0.7452	209	0.7256
22	0.7643	69	0.7418	116	0.7326	163	0.7345	210	0.7340
23	0.7597	70	0.7511	117	0.7511	164	0.7537	211	0.7377
24	0.7661	71	0.7510	118	0.7400	165	0.7492	212	0.7383

25	0.7448	72	0.7474	119	0.7437	166	0.7280	213	0.7452
26	0.7558	73	0.7493	120	0.7451	167	0.7442	214	0.7344
27	0.7727	74	0.7429	121	0.7536	168	0.7488	215	0.7340
28	0.7637	75	0.7728	122	0.7421	169	0.7315	216	0.7458
29	0.7629	76	0.7532	123	0.7429	170	0.7448	217	0.7344
30	0.7583	77	0.7397	124	0.7366	171	0.7474	218	0.7186
31	0.7560	78	0.7518	125	0.7460	172	0.7307	219	0.7391
32	0.7545	79	0.7422	126	0.7388	173	0.7436	220	0.7467
33	0.7503	80	0.7545	127	0.7427	174	0.7290	221	0.7426
34	0.7628	81	0.7332	128	0.7611	175	0.7423	222	0.7323
35	0.7482	82	0.7414	129	0.7417	176	0.7310	223	0.7340
36	0.7725	83	0.7314	130	0.7500	177	0.7449	224	0.7415
37	0.7590	84	0.7399	131	0.7423	178	0.7480	225	0.7252
38	0.7646	85	0.7411	132	0.7409	179	0.7432	226	0.7463
39	0.7509	86	0.7393	133	0.7418	180	0.7353	227	0.7157
40	0.7546	87	0.7375	134	0.7325	181	0.7609	228	0.7433
41	0.7506	88	0.7336	135	0.7265	182	0.7327	229	0.7375
42	0.7606	89	0.7369	136	0.7566	183	0.7495	230	0.7330
43	0.7684	90	0.7486	137	0.7400	184	0.7430	231	0.7421
44	0.7544	91	0.7285	138	0.7460	185	0.7218	232	0.7249
45	0.7580	92	0.7369	139	0.7377	186	0.7546	233	0.7291
46	0.7533	93	0.7403	140	0.7514	187	0.7440	234	0.7194
47	0.7583	94	0.7373	141	0.7309	188	0.7302		
48	0.7665	95	0.7322	142	0.7435	189	0.7739		