논문 2011-48CI-6-15

시드 정제 기술을 이용한 웹 스팸 필터링의 품질 향상

(Improving the Quality of Web Spam Filtering by Using Seed Refinement)

무하마드 아티프 쿠레시[*], 윤 태 섭[*], 이 정 훈[**], 황 규 영[***]

(Muhammad Atif Qureshi, Tae-Seob Yun, Jeong-Hoon Lee, and Kyu-Young Whang)

요 약

웹 스팸은 중요하지 않은 웹 페이지들의 중요도를 승격시키기 때문에 웹 검색 결과의 품질에 중대한 영향을 준다. 따라서 웹 검색 엔진은 웹 스팸을 제거할 필요가 있다. 웹 스팸 필터링은 스팸 페이지들, 즉 웹 스팸에 기여하는 웹 페이지들을 식별하는 것이며, 잘 알려진 웹 스팸 필터링 알고리즘으로는 Trust Rank, Anti-Trust Rank, Spam Mass, 그리고 Link Farm Spam이 있다. 이러한 알고리즘들의 결과 품질은 입력 시드(input seed)에 따라 달라진다. 따라서 입력 시드를 정제(refinement)함으로써, 웹 스팸 필터링의 품질을 향상 시킬 수 있다. 본 논문에서는 잘 알려진 네 가지 알고리즘에 대한 시드를 정제하는 기술을 제안한다. 다음으로, 이러한 기술을 원(original) 알고리즘에 각각 적용하는 방법으로 알고리즘을 수정한다. 이를 수정된 웹 스팸 필터링 알고리즘이라고 부른다. 본 논문에서는 또한, 웹 스팸 필터링을 좀 더 향상시키기 위한 전략을 제안한다. 이 전략에서는 수정된 알고리즘들을 수행 순서상의 적절한 위치에 배치함으로써 알고리즘들의 상호간 지원을 통해 전체적으로 성능을 향상시키는 가능성을 고려한다. 마지막으로, 실험에서는 시드 정제의 효과를 보인다. 이를 위해, 먼저, 수정된 알고리즘의 웹 스팸 필터링 품질이 원 알고리즘의 품질보다 더 우수함을 보인다. 다음으로, 웹 스팸 필터링 알고리즘들이 수행되는 순서의 조합 중 가장 성능이 우수한 조합이 가장 뛰어난 잘 알려진 알고리즘과 비교하여 정확도(*precision*)를 유지하면서 파라미터의 전형적인 값 범위 내에서 재현율(*recall*)은 최대 1.38배까지 높게 향상됨을 보인다.

Abstract

Web spam has a significant influence on the ranking quality of web search results because it promotes unimportant web pages. Therefore, web search engines need to filter web spam. web spam filtering is a concept that identifies spam pages — web pages contributing to web spam. TrustRank, Anti-TrustRank, Spam Mass, and Link Farm Spam are well-known web spam filtering algorithms in the research literature. The output of these algorithms depends upon the input seed. Thus, refinement in the input seed may lead to improvement in the quality of web spam filtering. In this paper, we propose seed refinement techniques for the four well-known spam filtering algorithms. Then, we modify algorithms, which we call *modified spam filtering algorithms,* by applying these techniques to the original ones. In addition, we propose a strategy to achieve better quality for web spam filtering. In this strategy, we consider the possibility that the modified algorithms may support one another if placed in appropriate succession. In the experiments we show the effect of seed refinement. For this goal, we first show that our modified algorithms outperform the respective original algorithms in terms of the quality of web spam filtering. Then, we show that the best succession significantly outperforms the best known original and the best modified algorithms by up to 1.38 times within typical value ranges of parameters in terms of *recall* while preserving *precision.*

Keywords : 웹 스팸 필터링, 입력 시드 정제, 링크 스팸, 성능

Ⅰ. Introduction

World Wide Web(WWW) is a huge information resource, and it doubles in less than two years[1]. Thus, it would be difficult to find information from WWW without search assistance. A web search engine is a search system for retrieving relevant web

[*] 학생회원, [**] 학생회원-교신저자, [***] 평생회원, 한국과학기술원 전산학과
(Dept. of Computer Science, Korea Advanced Institute of Science and Technology)
※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2011-0020498).
접수일자: 2011년7월20일, 수정완료일: 2011년11월1일

pages for the user's queries from the WWW[2]. Google[3], Yahoo[4], MS Bing[5], and Naver[6] are popular examples of web search engines.

A web search engine usually returns a huge amount of relevant web pages for the user's query[7, 8]. However, the user wants to browse the most important ones[7]. Thus, the web search engine arranges the relevant web pages in the order of their importance[1]. For this the web search engine utilizes a ranking method[1].

Link-based ranking methods are prevalent in popular web search engines[2, 9~10] such as Google, Yahoo, and MS Bing[10]. These methods exploit the link structure of web for ranking the search results[1]. However, the methods suffer from link spam[11], which is the type of web spam that takes advantage of the link structure of web in order to boost importance of one or more unimportant web pages[11, 12]. In order to filter out link spam, many link spam filtering algorithms have been proposed[13~16]. However, the algorithms do not perform well if the seed given to the algorithms is not good because they are dependent upon the seed. Thus, if the seed is well refined, the quality of the spam filtering algorithms will get improved. So far, much research has been done on the link spam filtering algorithms. However, research on seed refinement techniques has been less than adequate.

In this paper, we propose input seed refinement techniques for four well-known web spam filtering algorithms i.e., TrustRank[14], Anti-TrustRank[15], Spam Mass[13], and Link Farm Spam[16]. The contributions of the paper are as follows. First, we propose the modified algorithms for four web spam filtering algorithms by making use of additional input seed sets. Specifically, the four web spam filtering algorithms use at most one type of seed set(either for spam or non-spam). However, we modify these algorithms to use both types of input seed sets(i.e., seed sets for spam and non-spam) to detect more web spam. Next, we propose a strategy that arranges the execution sequence of our modified algorithms in order to achieve better quality of web spam filtering. Finally, we conduct extensive experiments to show the effect of seed refinement. We first show the quality improvement of our algorithms compared to the corresponding original ones. Then, we evaluate the best succession among our algorithms.

The rest of this paper is organized as follows. In Section Ⅱ, we introduce the web graph model, PageRank, and link spam. In Section Ⅲ, we introduce the four well-known web spam filtering algorithms. In Section Ⅳ, we explain our modifications in the four well-known algorithms and investigate successions among them. In Section Ⅴ, we show the results of our evaluation. In Section Ⅵ, we conclude the paper.

Ⅱ. Preliminary

In this section, we introduce a graph model for web: web graph model. Then, we explain the PageRank algorithm, which is a popular link-based ranking algorithm[17]. Finally, we explain link spam.

2.1. Web Graph Model

Web can be modeled as a directed graph $G=(V,E)$ consisting of a set V of web nodes (vertices) and a set E of directed links(edges)[14]. Directed links are classified into *inlinks* and *outlinks*. *Inlinks* are those incoming to a web node, and *outlinks* are those outgoing from a web node[14]. Fig.1 shows an example of a web graph. In this figure A, B, and C represent web nodes while the arrows represent the links. $\overrightarrow{AB}$ and $\overrightarrow{BC}$ are the outlinks of the web nodes A and B, respectively. $\overrightarrow{AB}$ and $\overrightarrow{BC}$ are the inlinks of the web nodes B and C, respectively.

The web graph can be classified into two classes:

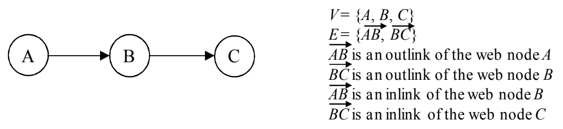

A
B
C
$V=\{A,B,C\}$
$E=\{\overrightarrow{AB},\overrightarrow{BC}\}$
$\overrightarrow{AB}$ is an outlink of the web node A
$\overrightarrow{BC}$ is an outlink of the web node B
$\overrightarrow{AB}$ is an inlink of the web node B
$\overrightarrow{BC}$ is an inlink of the web node C

그림 1. 웹 그래프의 예
Fig. 1. An example of a web graph.

page-level and domain-level web graphs[13, 15~16]. In a page-level web graph, a web node represents page information(e.g., cnn.com/index.html), and a directed link represents a link(i.e., URL) contained in a web page. In a domain-level web graph, a domain(e.g., cnn.com) of web pages is represented by a web node, as a link represents all the inlinks or outlinks to or from this domain from or to another domain[18]. In addition, algorithms on a domain-level web graph are more scalable compared to a page-level web graph, and the well-known web spam filtering algorithms[13~14, 16] use domain-level web graphs.

2.2. PageRank

PageRank[2, 17] is a well-known link-based ranking algorithm that exploits the link information to assign global importance score to the entire web[14, 17, 19]. The basic idea of PageRank is that a web page is important if it is inlinked by many other pages. The PageRank score of a web page is computed as in Eq.(1)[17]:

$$PR[p] = d \cdot \sum_{q:(q,p)\in E} \frac{PR[q]}{N_{outlink(q)}} + (1-d) \cdot v[p] \qquad (1)$$

In Eq.(1), $PR[p]$ denotes the PageRank score of the web page p; d is the damping factor, which is the probability of following an outlink; $N_{outlink(q)}$ is the number of outlinks of the web page q; $v[p]$ is the probability that a user randomly jumps from p to any arbitrary web page. The probability is uniform and is defined as reciprocal to the total number of web pages[17]. The PageRank algorithm can be applied to rank domains by using a domain-level web graph in place of a page-level web graph[13].

2.3. Link Spam

Web spam is a deliberate action performed in order to boost a web page's ranking without improving its real merit[11, 15, 20]. Link spam is an action that changes the link structure of web in order to boost a web page's ranking[11]. Fig.2 shows an example of link spam.

In this figure, there are eleven domains from $D1$ to

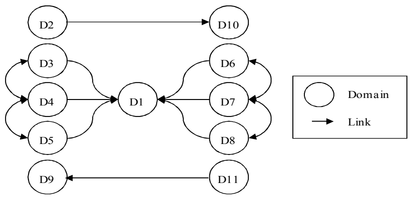


그림 2. 링크 스팸의 예
Fig. 2. An example of link spam.

$D11$. The domains $D3$ to $D8$ contain pages that outlink to pages of the domain $D1$. This link structure is created in order to provide undue advantage to the pages of the domain $D1$, i.e., to make those pages look important because they are inlinked by many pages of different domains. The domains $D1$ and $D3$ to $D8$ are involved in web spam. However, the domains $D2$, $D9$, $D10$, and $D11$ are not participating in web spam. The domains that are involved in web spam are known as spam domains while the rest are known as non-spam domains.

Ⅲ. Related Work

In this section, we review four well-known web spam filtering algorithms. In Section 3.1 we present a brief overview of classification(i.e., seed generators and spam detectors) of the four well-known algorithms. In Section 3.2 we describe TrustRank that promotes non-spam domains and Anti-TrustRank that demotes spam domains as seed generators. In Section 3.3 we describe Spam Mass and Link Farm Spam, which are spam detectors.

3.1. Overview

Web spam filtering is an action to identify web spam. To achieve this goal, existing work makes use of the link structure of web, manually declared spam or non-spam domains(simply, the input seed set), and additional information for declaring spam or non-spam domains. Research on web spam filtering is classified into two approaches: one of evaluating badness(or goodness) of domains by using only the input seed set and the other of identifying spam domains by exploiting additional properties of web spam. It is well known that, though the former can identify web spam, the quality of the results is insufficient[13, 15]. The latter detects more spam domains than the former because it adopts techniques uncovering boosting activity explained in Section 2.3. However, the former may help generate the refined input seed set, which can be used as the input of the latter. We expect that the refined input may help improve the quality of web spam filtering. Based on this expectation, we classify web spam filtering algorithms into two types of algorithms: seed generation algorithms(simply, seed generators) and spam detection algorithms(simply, spam detectors).

3.2. Seed Generation Algorithms

3.2.1. Trust Rank

TrustRank[14] exploits the outlink information of trusted domains, which are defined as well-known non-spam domains such as .gov and .edu. TrustRank begins by taking as the input a seed set of non-spam domains. Then, it propagates trust scores of the non-spam domains to the outlinks of the domains while attenuating by the damping factor as defined in Section 2.2. Finally, a threshold value is chosen, and all domains whose trust scores fall above this value are declared as new non-spam domains.

3.2.2. Anti-Trust Rank

Anti-TrustRank[15] exploits the inlink information of the spam domains that are provided as the input seed. Anti-TrustRank propagates anti-trust scores of the spam domains to their inlinks(i.e., in the reverse direction) while attenuating by the damping factor. Finally, a threshold value is chosen, and all domains whose anti-trust scores fall above this value are declared as new spam domains.

3.3. Spam Detection Algorithms

3.3.1. Spam Mass

Spam Mass[13] exploits both the scores coming from spam and non-spam domains[11]. The spam score is estimated by subtracting the non-spam score from the overall score. TrustRank is used to calculate the non-spam score, and the overall score is calculated by PageRank. The basic assumption of this algorithm is that a spam domain usually gets a high score from suspicious domains, which are not trusted by TrustRank. Under the assumption, this algorithm declares a domain that receives excessively higher PageRank score compared to the trust score as a spam domain.

3.3.2. Link Farm Spam

Link Farm Spam[16] exploits bidirectional links and outlinks of domains. That is, if a domain has many bidirectional links or many outlinks to spam domains, the domain is declared as a spam domain. Link Farm Spam begins by finding bidirectional links among domains and marks a domain as a spam if the number of bidirectional links of the domain is equal to or greater than a given threshold. Then, the algorithm attempts to find more spam domains by observing outlinks and marks a domain as a spam if the number of its outlinks to spam domains is equal to or greater than another threshold. We denote the threshold dealing with bidirectional links by *limitBL* and that dealing with outlinks by *limitOL*.

Ⅳ. Input Seed Refinement for Web Spam Filtering Algorithms

In this section, we propose modifications of four web spam filtering algorithms. We also propose a strategy for determining the succession(i.e., the execution sequence) of our modified algorithms in order to get better quality of web spam filtering.

4.1. Overview

The objective of seed refinement for web spam filtering is to improve the quality of web spam filtering. Specifically, the objective is to maximize the correct detections over the total detections(simply, *precision*[21]), to maximize the fraction of the correct detections over entire spam or non-spam population(simply, *recall*[21]), or both.

In Section Ⅲ, we have observed that the existing spam filtering algorithms depend on the input seed set. However, they utilize only one type of the input seed set belonging to either spam or non-spam seed domain. Thus, if both types are provided as input seed sets to the algorithms, it may improve the quality of web spam filtering. Furthermore, an output from one algorithm can become the input to the other algorithm. Thus, the effective succession between these algorithms may lead to further improvement.

4.2. Seed Generation Algorithms

4.2.1. Modified TrustRank

TrustRank takes as the input seed set a set of trusted domains. This algorithm may promote spam domains since trusted(i.e., non-spam) domains can outlink to spam domains. For example, a trusted university domain may outlink to a student's domain which may in turn outlink to a honey pot*. In Fig.3, the domain 1 represents the university's domain, the domain 3 represents the student's domain, and the domains 5 and 6 are the part of a honey pot.

In Fig.3, we observe that the spam domains 5 and 6 receive high trust scores because the domain 3 is deceived by the domain 5. In order to overcome this potential shortcoming caused by outlinking from a non-spam domain to a spam domain, we add another seed set of known spam domains, which serves as an

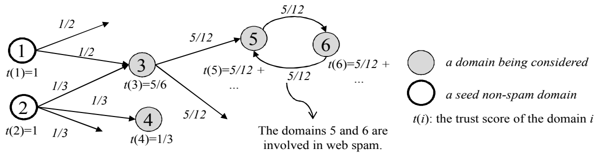


그림 3. 스팸 도메인에 높은 신뢰 점수를 부여하는 TrustRank의 예
Fig. 3. An example of TrustRank giving high trust scores to spam domains.

* Honey pot is a set of pages which provide some useful information(e.g., Unix documentation pages) but have hidden outlinks to spam pages[20].

```
Input:
        a seed set of non-spam domains $N$
        a seed set of spam domains $S$
        the threshold $cutoff$
        the difference threshold $\varepsilon$
        web graph $G=(V, E)$
Output:
        a set of non-spam domains $O_N$
        trust score vector of all domains $T_{ordered}$
Algorithm:
1.  FOR EACH $d \in V$
2.    IF $d \in N$ THEN
3.      $T_0[d] = \frac{1}{size(N)}$
4.    ELSE
5.      $T_0[d] = 0$
6.  $i = 0$
7.  DO
8.    FOR EACH $d \in V$
9.      FOR EACH $(d, q) \in E$
10.       IF $q \notin S$ THEN
11.         $T_{i+1}[q] = T_{i+1}[q] + damp \cdot \frac{T_i[d]}{N_{outlink}(d)}$
12.   FOR EACH $d \in V$
13.     $T_{i+1}[d] = T_{i+1}[d] + (1 - damp).T_i[d]$
14.   $\Delta = | T_{i+1} - T_i |$
15.   $i = i + 1$
16. UNTIL $\Delta < \varepsilon$
17. $T_{ordered}$ = Sort $T_{i+1}$ by trust scores in the descending order
18. $O_N$ = the set of domains with highest trust scores within $cutoff$
19. Remove domains whose trust score = 0 from $O_N$
20. RETURN $O_N$, $T_{ordered}$
```

그림 4. 수정된 TrustRank 알고리즘
Fig. 4. Modified TrustRank algorithm.

exception list, so that the links to the spam domains do not contribute to the trust scores of the domains.

Fig.4 shows the algorithm of Modified TrustRank. Inputs to the algorithm are a set of non-spam domains *N*, a set of spam domains *S*, the threshold value *cutoff*, the difference threshold *ε*, and web graph *G*. Hereafter, we use these inputs for other algorithms as well unless we explicitly specify different inputs. The output is a set of non-spam domains O_N and the trust score vector of all domains T_{orderd}. The threshold *cutoff* is used for determining the non-spam domains at the end of the algorithm[14, 15]. This threshold is defined relative to the size of the non-spam input seed set so that top *cutoff* percent of domains with the high trust scores are declared as non-spam domains. For example, if *cutoff* = 100%, the number of domains declared as non-spam domains is equal to that of the non-spam seed set under the ideal assumption that every domain declared as non-spam has a non-zero trust score because the score is affected by trust scores propagated from the non-spam seed set. We define this case as the base case for TrustRank and

Modified TrustRank. In fact, the size of domains declared as non-spam domains is less than or equal to that of the non-spam seed set because we remove the domains whose trust score = 0 from the domains declared as non-spam. Compared with original TrustRank algorithm, the modified algorithm additionally gets the set of spam domains S as an input seed set and uses it for preventing scores of non-spam domains from propagating to spam domains(lines 10-11). The algorithm first initializes the trust scores of all domains by assigning a uniform value $1/size(N)$ to every non-spam domain N and zero to the rest of domains(lines 1-5). Then, it iteratively calculates trust scores until the difference between the two consecutive trust score vectors is less than ε(lines 7-16). Specifically, the algorithm uniformly distributes trust scores of domains to their outlinks pointing to all domains except spam domains while taking the damping factor into account(lines 8-11). Here, $N_{outlink}(d)$ represents the number of outlinks of the domain d(line 11). Then, the algorithm assigns the random jump value to all domains according to their trust scores(lines 12-13). All the domains are arranged in the descending order of their trust scores(line 17), and then, the algorithm picks domains with the highest trust scores that are determined by *cutoff* defined earlier and adds the domains to O_N(line 18). Then, the domains with a zero trust score are removed from O_N(line 19). Finally, the algorithm returns O_N and $T_{ordered}$(line 20).

4.2.2. Modified Anti-TrustRank

Anti-TrustRank takes as the input seed set a set of spam domains. The algorithm may demote non-spam domains that point to spam domains such as honey pots as mentioned in Section 4.2.1.

Fig.5 shows the algorithm of Modified Anti-TrustRank. The output is a set of spam domains O_S and the anti-trust score vector of all domains $AT_{ordered}$. The threshold *cutoff* is used for determining the spam domains at the end of the algorithm[15]. This threshold is defined relative to the size of the spam input seed set so that top *cutoff* percent of domains with the high anti-trust scores are declared as spam domains. For example, if *cutoff* = 100%, the number of domains declared as spam domains is equal to that of the spam seed set under the assumption similar to what we made in Section 4.2.1 except that this assumption is based on the spam seed set and anti-trust score. We define this case as the base case for Anti-TrustRank and Modified Anti-TrustRank. Compared with original Anti-TrustRank algorithm, the modified algorithm additionally gets the set of non-spam domains N as an input seed set and uses it for preventing anti-trust scores of spam domains from propagating to non-spam domains(lines 10-11). The proposed algorithm first initializes the anti-trust scores of all domains by assigning a uniform value $1/size(S)$ to every spam domain S and zero to the rest of domains(lines 1-5). Then, it iteratively calculates anti-trust scores until the difference between the two consecutive anti-trust score vectors is less than ε (lines 7-16). Specifically, the algorithm uniformly distributes anti-trust scores of domains to their inlinks pointing to all domains except non-spam domains while taking the damping factor into

```
Input:
        a seed set of non-spam domains $N$
        a seed set of spam domains $S$
        the threshold $cutoff$
        the difference threshold $\Delta$
        web graph $G = (V, E)$
Output:
        a set of spam domains $O_S$
        anti-trust score vector of all domains $AT_{ordered}$
Algorithm:
1.  FOR EACH $d \in V$
2.      IF $d \in S$ THEN
3.          $AT_0[d] = \frac{1}{size(S)}$
4.      ELSE
5.          $AT_0[d] = 0$
6.  $i = 0$
7.  DO
8.      FOR EACH $d \in V$
9.          FOR EACH $(q, d) \in E$
10.             IF $q \notin N$ THEN
11.                 $AT_{i+1}[q] = AT_{i+1}[q] + damp \cdot \frac{AT_i[d]}{N_{inlink}(d)}$
12.     FOR EACH $d \in V$
13.         $AT_{i+1}[d] = AT_{i+1}[d] + (1 - damp) \cdot AT_i[d]$
14.     $\Delta = | AT_{i+1} - AT_i |$
15. $i = i + 1$
16. UNTIL $\Delta < \varepsilon$
17. $AT_{ordered}$ = Sort $AT_{i+1}$ by anti-trust scores in the descending order
18. $O_S$ = the set of domains with the highest anti-trust score within $cutoff$
19. Remove domains whose anti-trust score = 0 from $O_S$
20. RETURN $O_S$, $AT_{ordered}$
```

그림 5. 수정된 Anti-TrustRank 알고리즘
Fig. 5. Modified Anti-TrustRank algorithm.

account(lines 8-11). Here, $N_{inlink}(d)$ represents the number of inlinks of the domain d(line 11). Then, the algorithm assigns the random jump value to all domains according to their anti-trust scores(lines 12-13). All the domains are arranged in the descending order of their anti-trust scores(line 17), and then, the algorithm picks domains with the highest anti-trust scores that are determined by *cutoff* and adds the domains to O_S(line 18). Then, domains with a zero anti-trust score are removed from O_S(line 19). Finally, the algorithm returns O_S and $AT_{ordered}$(line 20).

4.3. Spam Detection Algorithms

4.3.1. Modified Spam Mass

Spam Mass uses TrustRank for determining spam domains so it suffers from the same problem as explained in Section 4.2.1. Thus, we substitute Modified TrustRank in place of TrustRank for performing Modified Spam Mass. Fig.6 shows the algorithm of Modified Spam Mass. Inputs to the algorithm are a set of non-spam domains N, set of spam domains S, the threshold value *topPR*, the threshold value *relativeMass*, the difference threshold ε, and web graph G. The output is a set of spam domains O_S. The threshold *topPR* represents the minimum PageRank score of a domain so that the domain is considered as a candidate for web spam[13]. The threshold *relativeMass* is defined as the ratio of the spam score to the overall score as explained in Section 3.3.1. It is used for deciding a domain as a spam domain so that, if the domain receives excessively higher spam score compared to non-spam score, the domain is a candidate for web spam[13]. While original Spam Mass algorithm calls original TrustRank for computing trust scores for all domains, the modified algorithm first initializes two vectors of trust scores and PageRank scores by using Modified TrustRank and PageRank, respectively(lines 1-2). Then, a domain is declared as a spam if it meets two thresholds constraints, *topPR* and *relativeMass*(lines 3-6). First, a domain should have at least *topPR* PageRank score to be considered as a spam(line 4). Second, the domain also should have a fractional value higher than or equal to *relativeMass*(line 5). Here, the fractional value of a domain is defined as the ratio of the difference between its PageRank score and its trust score to the PageRank score. If a domain satisfies the two constraints, the domain is added to O_S(line 6). Finally, algorithm returns O_S(line 7).

```
Input:
a seed set of non-spam domains $N$
a seed set of spam domains $S$
the threshold $topPR$
the threshold $relativeMass$
the difference threshold $\Delta$
web graph $G= (V, E)$
Output:
a set of spam domains $O_S$
Algorithm:
1. $O_N, T$ = Modified TrustRank($N, S, cutoff,, \Delta, G$)
2. $P$ = PageRank($\Delta, G$)
3. FOR EACH $d \in V$
4.     IF $P[d] \geq topPR$ THEN
5.         IF $\frac{P[d]-T[d]}{P[d]} \geq relativeMass$ THEN
6.             $O_S \leftarrow O_S \cup \{d\}$
7. RETURN $O_S$
```

그림 6. 수정된 Spam Mass 알고리즘
Fig. 6. Modified Spam Mass algorithm.

4.3.2. Modified Link Farm Spam

Link Farm Spam does not take any input seed. However, we argue that the input seed set of spam and non-spam domains would improve the quality of its spam filtering. That is, a seed set of non-spam

```
Input:
        a seed set of non-spam domains $N$
        a seed set of spam domains $S$
        the threshold $limitBL$
        the threshold $limitOL$
        web graph $G= (V, E)$
Output:
        a set of spam domains $O_S$
Algorithm:
1. $O_S \leftarrow S$
2. FOR EACH $d \in V$
3.     IF $d \notin N$ THEN
4.         $I = inDomain(d) - N - \{d\}$
5.         $O = outDomain(d) - N - \{d\}$
6.         IF $size(I \cap O) \geq limitBL$ THEN
7.             $O_S \leftarrow O_S \cup \{d\}$
8. DO
9.     $O_{old} \leftarrow O_S$
10.        FOR EACH $d \in V$
11.            IF $d \notin N$ THEN
12.                $O = outDomain(d) \cap O_S$
13.                IF $size(O) \geq limitOL$ THEN
14.                    $O_S \leftarrow O_S \cup \{d\}$
15. UNTIL $size(O_S) > size(O_{old})$
16. RETURN $O_S$
```

그림 7. 수정된 Link Farm Spam 알고리즘
Fig. 7. Modified Link Farm Spam algorithm.

domains allows us to minimize wrong spam detections such as well-known trusted domains(i.e., .gov, .edu, etc.), and a seed set of spam domains would help in identifying more spam domains.

Fig.7 presents the algorithm of Modified Link Farm Spam. Inputs to the algorithm are a set of non-spam domains N, a set of spam domains S, the threshold value *limitBL*(defined in Section 3.3.2), the threshold value *limitOL*(defined in Section 3.3.2), and web graph G. The output is a set of spam domains O_S. Compared with original Link Farm Spam algorithm, the modified algorithm additionally gets the two sets of non-spam domains N and spam domains S as input seed sets and uses them (lines 1-11). The algorithm first initializes O_S with S(line 1). Then, the algorithm considers domains as spam domains if they have many bidirectional links with the domains that are not included in non-spam domains given as the seed(lines 2-7). Here, non-spam domains are not considered as spam domains although they have many bidirectional links(line 3). *inDomain*(d) represents the set of domains pointing to the domain d(line 4), and *outDomain*(d) represents the set of domains pointed by the domain d(line 5). The domain d is considered as a spam domain(line 7) if the constraint for *limitBL* is satisfied(line 6). After finding spam domains O_S due to bidirectional links, the algorithm additionally declares the domains that have many outgoing links to O_S as spam domains(lines 9-14). This process(lines 9-14) continues until no more spam domain can be found(line 15). Finally, the algorithm returns O_S(line 16).

4.4. Successions of Web Spam Filtering Algorithms

In this section, we present successions among Modified TrustRank(*MTR*), Modified Anti-TrustRank (*MATR*), Modified Spam Mass(*MSM*), and Modified Link Farm Spam(*MLFS*). We first introduce the global view of successions among web spam filtering algorithms; then, we present the possible successions.

4.4.1. Global View of Successions

Seed generators are not pure spam detection algorithms; instead, they promote and demote spam and non-spam domains as discussed in Section 3. Therefore, we believe that *MTR* and *MATR* can help generate refined input seed sets. Since spam detectors require input seed sets with better quality for their operations, seed generators could provide good input seed sets if they come in succession. In the seed generators, our concern is how to precisely determine the seed sets of spam and non-spam domains whereas, in the spam detectors, our concern is how to correctly detect spam domains.

Our strategy is to execute the succession of seed generators(*MTR* and *MATR*) and that of spam detectors(*MSM* and *MLFS*) in turn. The input to the former succession of seed generators is lists of spam and non-spam domains, which are manually labeled. We call these domains the *manual spam and non-spam seed domains*. The output of the former succession is the input of the latter succession for spam detectors. We call the output the *refined spam and non-spam domains*. Finally, the output from the latter succession is a list of spam domains detected. We call these spam domains the *detected spam domains*. In Sections 4.2.2 and 4.2.3, we explain the successions inside each class.

4.4.2. Possible Successions inside the Seed Generator

The possible successions between *MTR* and

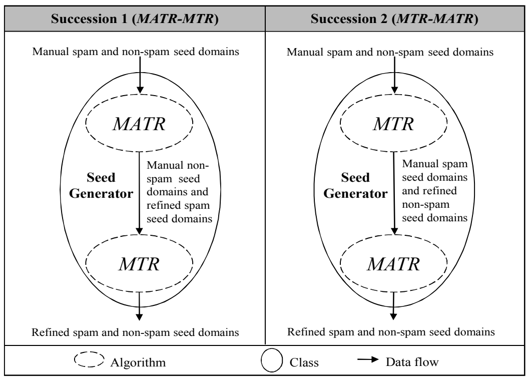

Succession 1 (MATR-MTR)
Succession 2 (MTR-MATR)
Manual spam and non-spam seed domains
MATR
Seed Generator
Manual non-spam seed domains and refined spam seed domains
MTR
Refined spam and non-spam seed domains
Manual spam and non-spam seed domains
MTR
Seed Generator
Manual spam seed domains and refined non-spam seed domains
MATR
Refined spam and non-spam seed domains
Algorithm
Class
Data flow

그림 8. 시드 생성기내에서의 가능한 연속
Fig. 8. Possible successions inside the seed generator.

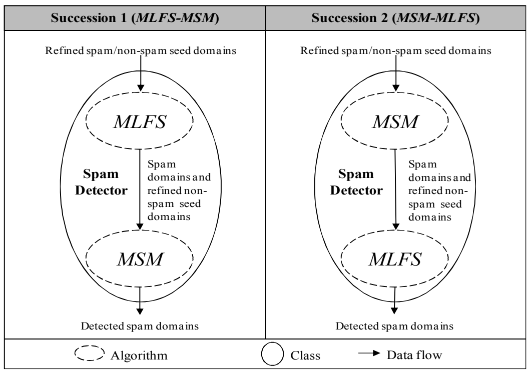


그림 9. 스팸 탐지기내에서의 가능한 연속
Fig. 9. Possible successions inside the spam detector.

MATR for the seed generator are shown in Fig.8. In Succession 1, we run *MATR* followed by *MTR* (*MATR-MTR*); in Succession 2, we run *MTR* followed by *MATR*(*MTR-MATR*). Under both successions, the manual spam seed domains are refined by *MATR* while the manual non-spam seed domains are refined by *MTR*.

4.4.3. Possible Successions inside the Spam Detector

The possible successions between *MSM* and *MLFS* for the spam detector are shown in Fig.9. In Succession 1, we run *MLFS* followed by *MSM*(*MLFS-MSM*); in Succession 2, we run *MSM* followed by *MLFS*(*MSM-MLFS*). Under both successions, we use the refined spam and non-spam seed domains as the input seed set. Here, we perform two more tests using single algorithms, i.e., *MLFS* and *MSM*, using the refined spam and non-spam seed domains as the input seed set, to investigate the effect of the succession of spam detectors. Table 1 shows the two types of tests for the spam detector.

표 1. 스팸 탐지기를 위한 테스트
Table 1. Tests for the spam detector.

Successions		Single Algorithms	
MLFS-MSM	*MSM-MLFS*	*MLFS*	*MSM*

Ⅴ. Performance Evaluation

In this section, we evaluate the effectiveness of our modifications over the original four web spam filtering algorithms. We also evaluate the effectiveness of the successions of the modifications and find the best one from those successions.

5.1. Experimental Data and Environment

We use two sets of experiments. In the first set, we show the effect of refining seed for four web spam filtering algorithms[13, 14, 15, 16] by comparing the original algorithms with our modified algorithms. In the second set, we show the effect of arranging the execution sequence among our modified algorithms. To achieve this goal, we get all the possible successions of the modified algorithms on the basis of the classification explained in Section 4.4. Then, we find the best succession. Finally, we show the

표 2. 실험에서 비교한 알고리즘
Table 2. Algorithms compared in the experiments.

	Symbol	Description
Original algorithms	*TR*	TrustRank
	ATR	Anti-TrustRank
	SM	Spam Mass
	LFS	Link Farm Spam
Modified algorithms	*MTR*	Modified TrustRank
	MATR	Modified Anti-TrustRank
	MSM	Modified Spam Mass
	MLFS	Modified Link Farm Spam
Successions of the modified algorithms	*MATR-MTR*	Modified Anti-TrustRank followed by Modified TrustRank
	MTR-MATR	Modified TrustRank followed by Modified Anti-TrustRank
	MLFS-MSM	Modified Link Farm Spam followed by Modified Spam Mass
	MSM-MLFS	Modified Spam Mass followed by Modified Link Farm Spam

표 3. 실험 요약
Table 3. Summary of the experiments.

Sets of the Experiments	Experiments		Parameters	
Comparisons for showing the effect of refining seed set	Exp. 1	Comparison between *TR* and *MTR*	$cutoff_{Tr}$	0% - 160%
			$ratio_{Top}$	10%, 50%, 100%
			damp	0.85
	Exp. 2	Comparison between *ATR* and *MATR*	$cutoff_{ATr}$	0% - 1600%
			$ratio_{Top}$	10%, 50%, 100%
			damp	0.85
	Exp. 3	Comparison between *SM* and *MSM*	*relativeMass*	0.7 – 1.0
			topPR	10%, 50%, 100%
			damp	0.85
	Exp. 4	Comparison between *LFS* and *MLFS*	*limitBL*	2 - 7
			limitOL	2 – 7
Comparisons for showing the effect of ordering executions	Exp. 5	Finding the best succession for the seed generator	$cutoff_{Tr}$	50% - 160%
			$cutoff_{ATr}$	50% - 350%
			damp	0.85
	Exp. 6	Finding the best succession for the spam detector	$cutoff_{Tr}$	110%
			$cutoff_{ATr}$	182%
			relativeMass	0.7 – 0.99
			topPR	100%
			limitBL	2
			limitOL	2
			damp	0.85
	Exp. 7	Comparison among the best succession, the best known algorithm, and best modified algorithm	$cutoff_{Tr}$	110%
			$cutoff_{ATr}$	182%
			relativeMass	0.7 – 0.99
			topPR	100%
			limitBL	2
			limitOL	2
			damp	0.85

quality improvement in web spam filtering of the best succession over the best modified and the best known original algorithms. Table 2 summarizes the algorithms that we compare in the experiments, and Table 3 summarizes the experiments.

In all the experiments, we use the public data set of UK-2006 domains[22]. The data set consists of 7,473 domains labeled as either spam or non-spam while the rest of 3,929 domains are unlabeled. The labeled data set is classified into two disjoint sets in order to perform evaluation[22]: Seed Set is the input seed set for the algorithms shown in Table 2, and Test Set is the universal set of domains that are used for computing the quality(i.e., *precision* and *recall*) of the outputs obtained from the algorithms. In addition, in order to enhance the quality of web spam filtering, we label more spam and non-spam domains by using the well-known labeling rule[11, 13~15, 22], and then, add them to Seed Set. That is, we label domains that contain spam terms in their domain name(e.g., mp3, mortgage, and sex) as spam domains. We also label trusted administrative and educational domains(e.g., .ac.uk, .gov.uk, and .police.uk) as non-spam domains.

Table 4 summarizes the characteristics of the data set in terms of domains and web pages. This data set has been prevalently used for web spam filtering[9, 11, 22~25]. Table 5 shows Seed Set and Test Set for the data set. Here, "Before Additional Labeling" represents the original seed set shown in[22]. "After Additional Labeling" represents the input seed set augmented by using the rules explained above. Hereafter, labeled spam domains of "After Additional Labeling" in Seed Set are called Spam Seed Set, and labeled non-spam domains of "After Additional Labelling" in Seed Set are called Non-Spam Seed Set. We conduct all the experiments using a Linux 2.6 system with a Pentium Core2Duo 3.0 GHz processor and 3.0 GBytes of main memory.

표 4. 도메인과 웹 페이지 데이터 집합의 특성[22]
Table 4. Characteristics of the data set in terms of domains and web pages[22].

	Domains		Web pages	
Labeled	Spam	1,924	Total	77.9 Million
	Non-spam	5,549		
Unlabeled	Unknown	3,929		
	Total	11,402		

표 5. 시드 및 테스트 데이터 집합의 분류
Table 5. Classification of the data set as Seed Set and Test Set.

	Seed Set		Test Set [7]
	Before Additional Labeling [7]	After Additional Labeling [3, 12, 13, 17]	
Labeled Spam Domains	674	737	1,250
Labeled Non-Spam Domains	4,948	7,306	601

5.2. Experimental Measures and Parameters

In order to evaluate the quality of the algorithms shown in Table 2, we use two well-known measures: *precision* and *recall*[21]. In all the algorithms other than *TR* and *MTR*, *precision* means how accurately an algorithm detects spam domains from Test Set(i.e., the ratio of the number of spam domains collected from Test Set to that of domains collected from Test Set), and *recall* means how large a portion of spam domains the algorithm detects from Test Set(i.e., the ratio of the number of spam domains collected from Test Set to that of spam domains in Test Set). In *TR* and *MTR*, because the two algorithms detect non-spam(i.e. trusted) domains, *precision* and *recall* are defined in the same way as explained above except that they are defined in terms of the non-spam domains. Table 6 summarizes input parameters for the experiments.

표 6. 실험 파라미터
Table 6. Parameters used in the experiments.

Parameters	Description
damp	It is a parameter used in *TR*, *MTR*, *ATR*, and *MATR* for representing the probability of following an outlink.
$ratio_{Top}$	It is the ratio for determining the input seed set in *TR*, *MTR*, *ATR*, and *MATR*. Specifically, from Spam(or Non-Spam) Seed Set, we retrieve domains whose PageRank scores are larger than or equal to the PageRank score of top-$ratio_{Top}$% domain among the entire domains, and then, use the retrieved domains as the input seed set.
$cutoff_{Tr}$	It is the *cutoff* threshold explained in Section 4.2.1. It is used in *TR* and *MTR* for determining a domain as a candidate of being non-spam.
$cutoff_{ATr}$	It is the *cutoff* threshold explained in Section 4.2.2. It is used in *ATR* and *MATR* for determining a domain as a candidate of being spam.
relativeMass	It is the threshold used in *SM* and *MSM* for determining a domain as a spam such that, if the ratio of the spam score(i.e., PageRank—trust score) of a domain to the overall score(i.e., PageRank score) of the domain is larger than or equal to *relativeMass*, the domain is a candidate for being web spam.
topPR	It is the threshold used in *SM* and *MSM* for determining the candidates for being web spam by comparing the PageRank score of a domain to be within the top percentage(i.e., *topPR*%) of PageRank scores.
limitBL	It is the threshold used in *LFS* and *MLFS* for determining a domain as a spam if the number of bidirectional links of the domain is equal to or greater than this threshold.
limitOL	It is the threshold used in *LFS* and *MLFS* for determining a domain as a spam if the number of outlinks of the domain pointing to spam domains is equal to or greater than this threshold.

5.3. Experimental Results

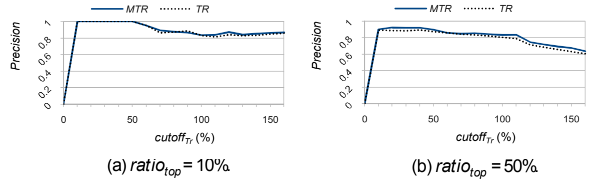
In Section 5.3.1, we show the results of the comparisons between the original web spam filtering algorithms and our modified algorithms. In Section 5.3.2, we show the results of the comparisons between the possible successions of our algorithms.

5.3.1. Comparisons between original and modified algorithms

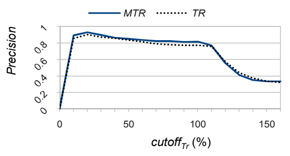
Exp. 1: comparison between *TR* and *MTR*

Figs.10 and 11 show the results as $ratio_{Top}$ is varied: 10%, 50%, and 100%. Here, we choose the value range of $cutoff_{Tr}$ so that *TR* and *MTR* declare all domains in Test Set as candidates of being non-spam domains at the high end of the value range (i.e., we choose the value range of $cutoff_{Tr}$ by considering the base case for TR and MTR as mentioned in Section 2). We use Spam Seed Set and Non-Spam Seed Set. Then, as explained in Table 6, we obtain the input seed sets for the trusted domains from Non-Spam Seed Set as $ratio_{Top}$ is varied. Hereafter, we use the two seed sets, Spam Seed Set and Non-Spam Seed Set, as the input seed sets unless otherwise specified. We set the damping factor to 0.85, which is considered as the standard[14, 26]. Hereafter, the damping factor is fixed for every experiment that needs it.

Figs.10 and 11 show that *MTR* performs slightly better than *TR*. In both Figs.10 and 11, *precision* and *recall* of *MTR* is overall higher than those of *TR* in the range where $cutoff_{Tr} \leq 110\%$. From the starting point where $cutoff_{Tr} > 110\%$, *precision* start decreasing sharply to the lowest as shown in Fig.10(c). The lowest point of *precision* is the point where almost every domain is marked as non-spam due to the high value of $cutoff_{Tr}$. Thus, from this point, increase in the value $cutoff_{Tr}$ would bear no change in *recall* while only decreasing *precision*. Therefore, all the points where *precision* is the lowest are insignificant for comparison.

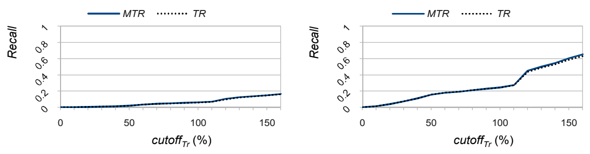


(a) $ratio_{top}$ = 10% (b) $ratio_{top}$ = 50%

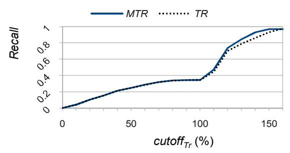


(c) $ratio_{top}$ = 100%

그림 10. $ratio_{Top}$의 변화에 따른 정확도
Fig. 10. *Precision* as $ratio_{Top}$ is varied.



(a) $ratio_{top}$ = 10% (b) $ratio_{top}$ = 50%



(c) $ratio_{top}$ = 100%

그림 11. $ratio_{Top}$의 변화에 따른 재현율
Fig. 11. *Recall* as $ratio_{Top}$ is varied.

From Figs.10 and 11 we observe $ratio_{Top}$=100% provides higher *recall* and comparable *precision* compared to other $ratio_{Top}$ values. Thus, hereafter, we fix the value of $ratio_{Top}$ as 100%. When $ratio_{Top}$=100% as shown in Fig.10(c), we observe that the effective $cutoff_{Tr}$ value is 110% since from that point onwards *precision* sharply decreases. When $cutoff_{Tr}$=110% in Figs.10(c) and 11(c), the *precision* of *MTR* is better than that of *TR* while the *recall* of *MTR* is the same as that of *TR*: the *precision* of *MTR* is 0.83, the *precision* of *TR* is 0.79, and their *recalls* are 0.27. Thus, we conclude that *MTR* is better than *TR*.

Exp. 2: comparison between *ATR* and *MATR*

Figs.12 and 13 show the results comparing the *ATR* with the *MATR*. Here, we choose the value range of $cutoff_{ATr}$ so that *ATR* and *MATR* declare all domains in Test Set as candidates of being spam domains at the high end of the value range (i.e., we choose the value range of $cutoff_{ATr}$ by considering the base case for ATR and MATR as mentioned in

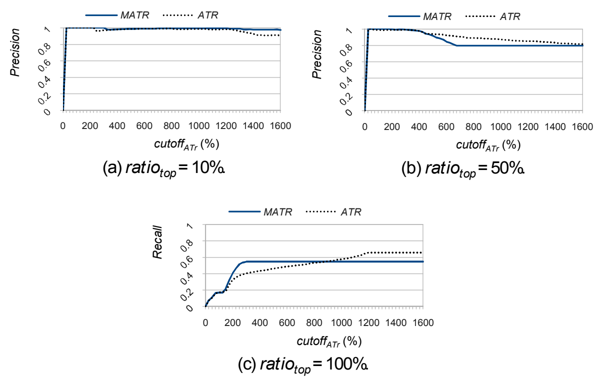


그림 12. $ratio_{Top}$의 변화에 따른 정확도
Fig. 12. *Precision* as $ratio_{Top}$ is varied.

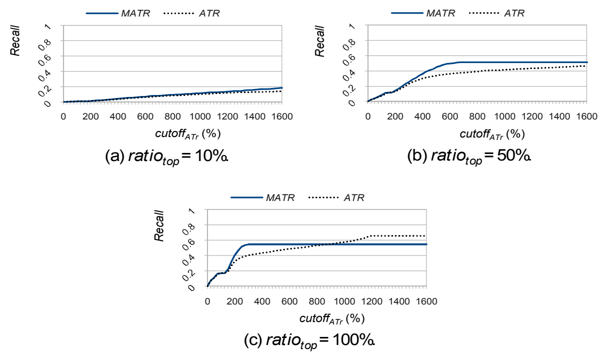


그림 13. $ratio_{Top}$의 변화에 따른 재현율
Fig. 13. *Recall* as $ratio_{Top}$ is varied.

Section 2). As explained in Table 6, we obtain the input seed sets for spam domains from Spam Seed Set as $ratio_{Top}$ is varied. In both Figs.12 and 13 we observe that *MATR* reaches the maximum *recall* much earlier than *ATR*. That is why the point of sharp decrease in *precision*(and increase in *recall*) also occurs earlier in *MATR* compared to *ATR*, as can be seen in above Figs.12 and 13(b) and (c). For a similar reason to that of Experiment 1, before this sharp decrease in *precision*, we observe comparable *precision* and better *recall* for *MATR* compared to *ATR*. We also see that, due to the same reason as explained in Experiment 1, the lowest points of *precision* shown in Fig.12(c) are insignificant for comparison.

Hereafter, we fix the value of $ratio_{Top}$ at 100% for same reason as mentioned in Experiment 1. When $ratio_{Top}$=100% as shown in Fig.12(c), we observe that the effective $cutoff_{ATr}$ value is 182% since from that point onwards *precision* sharply decreases. When $cutoff_{ATr}$=182% in Figs.12(c) and 13(c), the *recall* of *MATR* is better than that of *ATR* while the *precision* of *MATR* is the same as that of *ATR*: the *recall* of *MATR* is 0.34, the *recall* of *ATR* is 0.24, and their *precisions* are 0.99. Thus, we conclude that *MATR* is better than *ATR*.

Exp. 3: comparison between *SM* and *MSM*

Figs.14 and 15 show the results comparing *SM* with *MSM*. In each figure, (a)-(c) show the effect of spam detection as *topPR* is varied: 70%, 85%, and 100%. The original paper[13] of *SM* chooses an arbitrary low *topPR* value(approximately 1.2%) since the authors assume that spam domains have high(i.e., top-1.2%) PageRank values with high probability. However, we choose high values of *topPR* to investigate all domains and precisely determine whether or not a domain is a spam domain. Here, we vary *relativeMass* from 0.7 to 1.0. As explained in Section 3.3.1 and Table 6, we can have a chance to

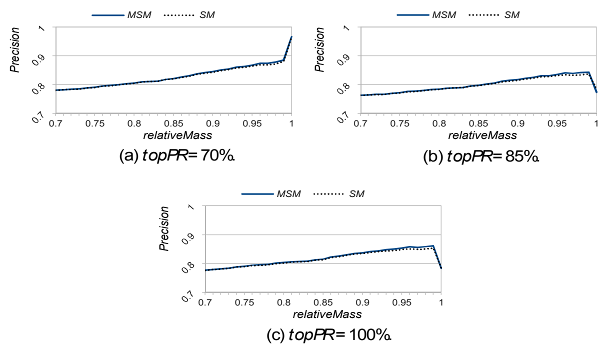


그림 14. *topPR*의 변화에 따른 정확도
Fig. 14. *Precision* as *topPR* is varied.

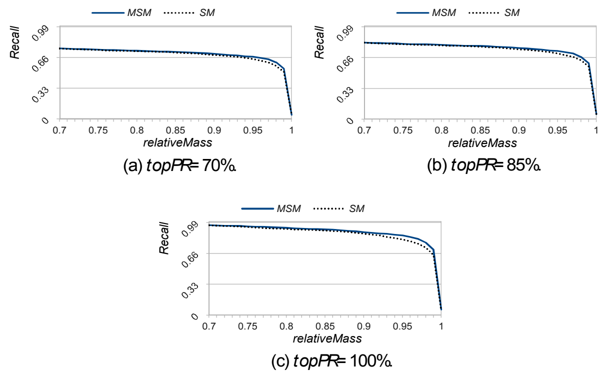


그림 15. *topPR*의 변화에 따른 재현율
Fig. 15. *Recall* as *topPR* is varied.

precisely detect actual spam domains as the value of *relativeMass* approaches 1.0 since the value represents the maximum effective degree of spam domains contributing to a domain. We do not consider the case that the value is smaller than 0.7 because, at this range, *precision* decreases while *recall* preserves or increases trivially. Existing work also considers the value that is larger than 0.7(specifically, 0.98 [13]).

Figs.14 and 15 show that *MSM* performs slightly better than *SM*. In both Figs.14 and 15 the *precision* and *recall* of *MSM* is higher than or equal to those of *SM* for all the points. We also see that *MSM* shows better quality than *SM* as *relativeMass* increases. Considering *relativeMass* at 0.98 as in [13], we observe both the *precision* and *recall* of *MSM* are better than that of *SM*: the *precision* of *MSM* is 0.86, the precision of *SM* is 0.85, the *recall* of *MSM* is 0.77, and the *recall* of *SM* is 0.72.

Hereafter, we fix the value of *topPR* at 100% since we observe higher *recall* and comparable *precision* compared to other values of *topPR*. Suppose that *topPR* and *relativeMass* are set to 100% and 1.0, respectively. Then, as explained in Sections 4.3.1, both of the two algorithms consider all domains (excluding the domains within Spam Seed Set) as non-spam domains. Since, hereafter, we fix the value of *topPR* at 100%, we do not set *relativeMass* to 1.0.

Exp. 4: comparison between *LFS* and *MLFS*

Fig.16 shows the results comparing the *LFS* with the *MLFS* as *limitBL* and *limitOL* are varied. These two experimental parameters are taken pairwise on

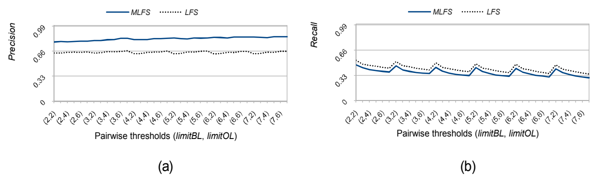


그림 16. *LFS*와 *MLFS*간의 탐지된 스팸 도메인의 품질 비교

Fig. 16. Comparison on the quality of the detected spam domains between *LFS* and *MLFS*.

the x-axis with ranging values from 2 to 7 for each parameter[16].

Fig.16 shows that *MLFS* performs much better than *LFS* in terms of *preicison* and reasonably comparable in terms of *recall*. We observe, *precision* for *MLFS* is higher than that of *LFS* in all the points. Moreover, we see the range of difference(0.14 to 0.22) in *precision* of two algorithms is larger than that(0.04 to 0.06) in *recall*. In addition, the highest point of *recall* in *MLFS* beats many points of *recall* for *LFS*. Overall, *MLFS* provides higher *precision* compared to that of *LFS* even when both of the two algorithms offer similar *recalls*. Thus, *MLFS* is overall better than *LFS*

From the experiment, we observe that *MLFS* has best reading at *limitBL*=2 and *limitOL*=2 since at that point *recall* is the highest compared to that at the other points, and *precision* is comparable to that at the rest of the points. At this point *MLFS* has relatively higher *precision* and comparable *recall* than those of *LFS*: the *precision* of *MLFS* is 0.78, the precision of *LFS* is 0.63, the *recall* of *MLFS* is 0.46, and the *recall* of *LFS* is 0.52.

In summary, we show that all the modified algorithms provide generally better quality than the respective original algorithms. In order to find the best original algorithms for detecting web spam, we compare *precisions* and *recalls* among *ATR*, *SM*, and *LFS*. In this comparison, we do not take *TR* into account because *TR* outputs non-spam domains as explained in Section 5.2. We find *SM* as the best algorithm among the three original algorithms since its *recall* is much higher while its *precision* is relatively comparable to the rest of the original algorithms as observed in Experiments 2 – 4. Similarly, we find *MSM* as the best one among the three modified algorithms *MATR*, *MSM*, and *MLFS*.

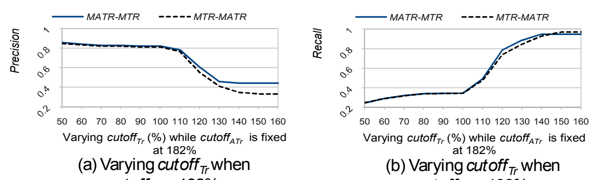
3.2. Comparisons for successions

In this section, we discuss the successions among the *MTR*, *MATR*, *MSM*, and *MLFS*. First, we perform succession tests for the seed generator and the spam detector, respectively. Then, we show the best succession of algorithm between the seed generator and the spam detector. Finally, we compare the best succession with the best known original and the best modified algorithms.

Exp. 5: the best succession for the seed generator

We conduct experiments to find the best succession of *MTR* and *MATR*. Here, we conduct the experiment to show the quality of the refined non-spam seed. We also conduct the other experiment to show the quality of the refined spam seed. In the experiment for the refined non-spam seed, we vary the $cutoff_{Tr}$ from 50% to 160% while we fix $cutoff_{ATr}$ at 182% as the best point determined in Experiment 2. Similarly, for the refined spam seed we vary the $cutoff_{ATr}$ from 50% to 350% while we fix $cutoff_{Tr}$ at 110% as the best point determined in Experiment 1. Here, we choose the value ranges of those two parameters so that the results obtained at points within the ranges are meaningful. That is, we choose the ranges so that, near to the maximum value of the range, *recall*(or *precision*) reaches to one (or zero) or bears no change while only *precision* decreases.

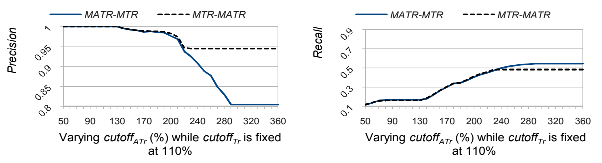
From Fig.17, we observe *MATR-MTR* is better than *MTR-MATR* in terms of *precision* and comparable in terms of *recall* for non-spam seed generation. From Fig.18, we observe *MTR-MATR* is better than *MATR-MTR* in terms of *precision* and comparable in terms of *recall* for spam seed generation. Thus, we select *MATR-MTR* and *MTR-MATR* as the best successions of seed generators for non-spam seed and spam seed, respectively.



(a) Varying $cutoff_{Tr}$ when $cutoff_{Atr}$ = 182%. (b) Varying $cutoff_{Tr}$ when $cutoff_{Atr}$ = 182%.

그림 17. 스팸 생성기의 연속들 간의 정제된 비 스팸 시드의 품질 비교

Fig. 17. Comparison on the quality of the refined non-spam seed between successions of the seed generators.



(a) Varying $cutoff_{ATr}$ when $cutoff_{Tr}$ = 110%. (b) Varying $cutoff_{ATr}$ when $cutoff_{Tr}$ = 110%.

그림 18. 스팸 생성기의 연속들 간의 정제된 스팸 시드의 품질 비교

Fig. 18. Comparison on the quality of the refined spam seed between successions of the seed generators.

Exp. 6: the best succession for the spam detector

We conduct experiments to find the best succession of spam detectors *MSM* and *MLFS*. As explained in Table 1, we perform experiments for two possible successions and two single algorithms. In this experiment, we use refined seed sets produced from the seed generators as the input seed sets for all the algorithms. Specifically, as observed in Experiment 5, we use the refined non-spam seed produced by *MATR-MTR*, which is the best choice of the seed generation for the non-spam seed. We also use the refined spam seed produced by *MTR-MATR*, which is the best choice of the seed generation for the spam seed. We vary *relativeMass* of *MSM* to show the tendency for comparison from 0.7 to 0.99 in the same way as in Experiment 3. We also exclude *relativeMass*=1.0 as observed in Experiment 3. As explained in Experiment 3, we fix *topPR*=100% in order to get spam domains as the results of investigating the entire domains. Moreover, as the best result of *MLFS* in Experiment 4, we fix *limitBL* and *limitOL* thresholds at 2 and 2,

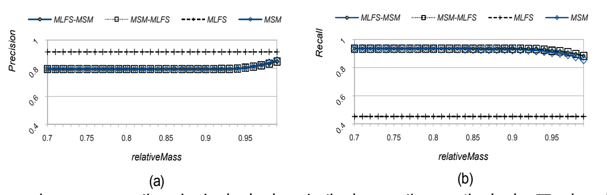


그림 19. 스팸 탐지기간의 정제된 스팸 도메인의 품질 비교

Fig. 19. Comparison on the quality of the detected spam domains among the spam detectors.

respectively.

Fig.19 show that *MLFS* has very low *recall* compared to the rest of algorithms. We see that *MLFS-MSM* is nearly identical to *MSM-MLFS* in both *precision* and *recall*. We also see that those two algorithms(i.e., *MLFS-MSM* and *MSM-MLFS*) are almost comparable to *MSM* in terms of *precision* and are slightly better than *MSM* in terms of *recall*. Specifically, in Fig.19(b), when *relativeMass*=0.99, *MLFS-MSM* and *MSM-MLFS* have 0.88 of *recall* while *MSM* has 0.86 of *recall*. Moreover, those three algorithms have the same value of *precision* as 0.85 when *relativeMass*=0.99. Thus, either *MLFS-MSM* or *MSM-MLFS* is best. Consequently, we choose *MLFS-MSM* as the best succession of spam detectors without loss of generality.

In summary, we find *MATR-MTR* and *MTR-MATR* as the best choice of the seed generation for non-spam and spam seeds, respectively. We also see that *MLFS-MSM*, which uses non-spam and spam seeds produced by the best choice of the seed generation as input seed sets, is the best for the spam detection. Hereafter, the best found succession of the best seed generator(i.e., *MATR-MTR* for the non-spam seed and *MTR-MATR* for the spam seed) followed by the best spam detector (i.e., *MLFS-MSM*) is called *SuccessionBest*.

Exp. 7: comparison among the best succession, the best known algorithm, and the best modified algorithm

We conduct experiments to find quality improvement of the best succession against the best modified and original(i.e., known) algorithms for the detection of web spam. From the Experiments 2 – 4, we observe that *SM* is the best original algorithm among the three original algorithms. We also observe that *MSM* is the best modified algorithm among the modified algorithms. Moreover, we observe that *MATR-MTR*(for the non-spam seed) and *MTR-MATR*(for the spam seed), which are followed by *MLFS-MSM*, is the best succession of the

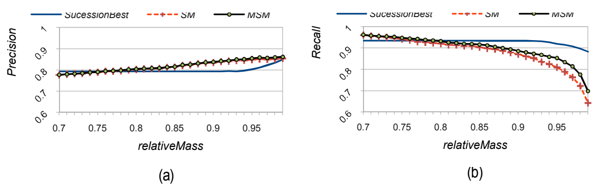


그림 20. *SuccessionBest*, *SM* 및 *MSM*간의 탐지된 스팸 도메인의 품질 비교

Fig. 20. Comparison on the quality of the detected spam domains among *SuccessionBest*, *SM*, and *MSM*.

modified algorithms(*SuccessionBest*). In this experiment, we use Spam Seed Set and Non-Spam Seed Set as the input seed sets for all the algorithms. In the same way as in Experiment 6, we vary the threshold *relativeMass* from 0.7 to 0.99, fix *topPR* at 100%, and keep *limitBL* and *limitOL* at 2 and 2, respectively. We use 110% $cutoff_{Tr}$ and 182% $cutoff_{ATr}$ as in Experiment 6.

In Fig.20, we see that *SuccessionBest* performs better than or similar to both *SM* and *MSM* in terms of *recall* while *SuccessionBest* is relatively comparable to the rest in terms of *precision*. We also see that *SuccessionBest* provides better quality as *relativeMass* increases. Specifically, in Fig.20, when *relativeMass*=0.99, we obtain the *precisions* 0.85, 0.86, and 0.85 for *SM*, *MSM*, and *SuccessionBest*, respectively. We also obtain the *recalls* 0.64, 0.70, and 0.88 for *SM*, *MSM*, and *SuccessionBest*, respectively. Thus, *SuccessionBest* outperforms other two algorithms by up to 1.38 times in *recall* while comparable to other two algorithms in *precision*. Consequently, we conclude that *SuccessionBest* is more effective than the rest for web spam filtering.

Ⅵ. Conclusions

In this paper, we have proposed seed refinement techniques for four well-known web spam filtering algorithms: TrustRank, Anti-TrustRank, Spam Mass, and Link Farm Spam. We enrich the input seed set by using both types of the input seed sets. These techniques are helpful in maximizing *recall* with *precision*. We also propose a strategy for the succession of the modified algorithms. We classify them into two classes: seed generators and spam

detectors. Modified TrustRank(*MTR*) and Modified Anti-TrustRank(*MATR*) are seed generators while Modified Spam Mass(*MSM*) and Modified Link Farm Spam(*MLFS*) are spam detectors. Moreover, we perform experiments between modified and original algorithms and also among the successions of modified algorithms in order to discover the best one. Our experimental results show that all the modified algorithms generally perform better than the original ones, and *MSM* is the best modified algorithm among them for web spam detection. We also show that the best quality among the successions is achieved by the *MATR* followed by *MTR* for non-spam seed generator and *MTR* followed by *MATR* for spam seed generator, which is then followed by *MLFS* and *MSM*(*SuccessionBest*). The best succession outperforms *SM* and *MSM* by up to 1.38 times in *recall* and is comparable to these two algorithms in *precision*: the *recalls* of *SM*, *MSM*, and *SuccessionBest* are 0.64, 0.70, 0.88, respectively; the precisions of *SM*, *MSM*, and *SuccessionBest* are 0.85, 0.86, and 0.85, respectively.

참 고 문 헌

[1] Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A., and Raghavan, S., “Searching the Web,” *ACM Transactions on Internet Technology(TOIT)*, Vol. 1, No. 1, pp. 2-43, Aug. 2001.
[2] Brin, S. and Page, L., “The Anatomy of a Large-Scale Hypertextual Web Search Engine,” In *Proc. 7th Int'l Conf. on World Wide Web (WWW)*, pp. 107-117, Brisbane, Australia, Apr. 14-18, 1998.
[3] Google Search, http://www.google.com.
[4] Yahoo! Seach, http://www.yahoo.com.
[5] MS Bing, http://www.bing.com.
[6] Naver, http://www.naver.com.
[7] Bar-Ilan, J., Mat-Hassan, M., and Levene, M., “Methods For Comparing Rankings of Search Engine Results,” *Computer Networks*, Vol. 50, No. 10, pp. 1448-1463, 2006.
[8] Baeza-Yates, R., and Ribeiro-Neto, B., “*Modern Information Retrieval*,” Addison Wesley, May 1999.
[9] Castillo, C., Donato, D., Gionis, A., Murdock, V., and Silvestri, F, “Know Your Neighbors: Web Spam Detection Using the Web Topology,” In *Proc. 30th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval*, Amsterdam, The Netherlands, July 2007.
[10] Yoshida, Y., Ueda, T., Tashiro, T., Hirate, Y., and Yamana, “What's Going on in Search Engine Rankings,” In *Proc. 22nd Int'l Conf. on Advanced Information Networking and Applications (AINAW)*, pp. 1199-1204, Okinawa, Japan, Mar. 2008.
[11] Becchetti, L., Castillo, C., Donato, D., Baeza-YATES, R., and Leonardi, S., “Link Analysis for Web Spam Detection,” *ACM Transactions on Web (TWEB)*, Vol. 2, No. 1, pp. 1-42, Mar. 2008.
[12] Henzinger, M. R., Motwani, R., and Silverstein, C., “Challenges in Web Search Engines,” *SIGIR Forum*, Vol. 36, No. 2, pp. 11-22, Sept. 2002.
[13] Gyongyi, Z., Berkhin, P., Garcia-Molina, H., and Pedersen, J., “Link Spam Detection Based on Mass Estimation,” In *Proc. 32th Int'l Conf. on Very Large Data Bases (VLDB)*, pp. 439-450, Seoul, Korea, Sept. 2006.
[14] Gyongyi, Z., Garcia-Molina, H., and Jan, P., “Combating Web Spam with TrustRank,” In *Proc. 30th Int'l Conf. on Very Large Data Bases (VLDB)*, pp. 576-587, Toronto, Canada, Aug. 2004.
[15] Krishnan, V. and Raj, R., “Web Spam Detection With Anti-TrustRank,” In *2nd Int'l Workshop on Adversarial Information Retrieval on the Web*, pp. 37-40, Washington, USA, Aug. 2006.
[16] Wu, B., Davison,B., “Identifying Link Farm Spam Pages,” In *Proc. Special interest tracks and posters of the 14th international conference on World Wide Web (WWW)*, pp. 820-829, Chiba, Japan, May 2005.
[17] Page, L., Brin, S., Motwani, R., and Winograd, T., The PageRank Citation Ranking: Bringing Order to the Web, Technical Report SIDL-WP-1999-0120, Department of Computer Science, Stanford University, 1998.
[18] Jiang, X., Xue, G., Song, W., Zeng, H., Chen, Z., and Ma, W., “Exploiting PageRank at Different Block Level,” In *Proc. 5th Int'l Conf. on Web Information Systems Engineering (WISE)*, pp. 241-252, Brisbane, Australia, Nov. 2004.
[19] Duc, P.M., Heo, J., Lee, J., and Whang, K., “Ranking Quality Evaluation of PageRank Variations,” *Journal of the Institute of Electronics Engineers of Korea* (in English), Vol.46, No.5, pp.14-28, Sept. 2009.
[20] Gyongyi, Z., Berkhin, P., Garcia-Molina, H., “Web spam taxonomy,” In *1st Int'l Workshop on*

Adversarial Information Retrieval on the Web (AIRWeb), pp. 39-47, Chiba, Japan, May 2005.

[21] Salton, G., and McGill, M.J., "*Introduction to Modern Information Retrieval*," McGraw-Hill, 1983.

[22] Castillo, C., Donato, D., Becchetti, L., Boldi, P., Leonardi, S., Santini, M., and Vigna, S., "A Reference Collection for Web Spam," *SIGIR Forum*, Vol. 40, No. 2, Dec. 2006.

[23] Abernethy, J., Chapelle, O., and Castillo, C., "Web Spam Identification through Content and Hyper-links," In Proc. 4th Int'l Workshop on Adversarial Information Retrieval on the Web (AIRWeb), pp. 41-44, Beijing, China, Apr. 2008.

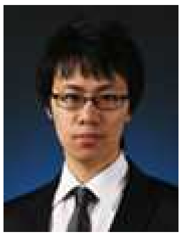
[24] Chung, Y., Toyoda, M., and Kitsuregawa, M., "A study of link farm distribution and evolution using a time series of web snapshots," In *Proc. 5th Int'l Workshop on Adversarial Information Retrieval on the Web(AIRWeb)*, pp. 9-16, Madrid, Spain, Apr. 2009.

[25] Zhou, D., Burges, C. J. C., and Tao, T., "Transductive Link Spam Detection," In *Proc. 3rd Int'l Workshop on Adversarial Information Retrieval on the Web(AIRWeb)*, pp. 21-28, Alberta, Canada, May 2007.

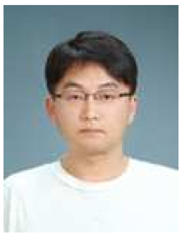
[26] Langville , A.N, and Meyer, C.D., "*Google PageRank and Beyond: The Science of Search Engine Rankings*," Princeton University Press, Princeton, 2006.

저 자 소 개

Muhammad Atif Qureshi(학생회원)
2007년 University of Karachi (UoK), 전산학과(학사)
2011년 한국과학기술원 전산학과 (석사)
2006년~2008년 UnHuman Productions, 선임연구원
2007년~2009년 Technology Orbit, S/W 개발자
2008년 University of Karachi, 전산학과, visiting faculty
<주관심분야 : 정보검색(IR), 웹 검색>



윤 태 섭(학생회원)
2008년 경북대학교 컴퓨터공학과 (학사)
2010년 한국과학기술원 전산학과 (석사)
2010년~현재 한국과학기술원 전산학과 박사과정
<주관심분야 : 검색엔진, 정보검색(IR)>



이 정 훈(학생회원)
1995년 경북대학교 컴퓨터공학과 (학사)
1997년 경북대학교 컴퓨터공학과 (석사)
2010년 한국과학기술원 전산학과 (박사)
2010년 9월~2011년 3월 한국과학기술원 연수연구원(Post Doc.)
2011년 4월~현재 경북대학교 IT대학 컴퓨터학부 연수연구원(Post Doc.)
<주관심분야 : 정보검색(IR), 센서네트워크, 그래프 데이터베이스>

황 규 영(평생회원)
1973년 서울대학교 전자공학과 (학사)
1975년 한국과학원 전기 및 전자 학과(석사)
1982년 Stanford University EE/CSL(석사)
1984년 Stanford University EE/CSL(박사)
1975년~1978년 국방과학연구소 선임연구원
1983년~1991년 IBM T.J. Waston Research Center, Research Staff Member
1992년~1994년 한국정보과학회 데이터베이스 연구회(SIGDB) 운영위원장
2009년~현재 Fellow, ACM
2007년~현재 Fellow, IEEE
2007년 한국정보과학회 회장
2007년~2009년 Coordinating Editor-in-Chief, The VLDB Journal(2003년~2009년 Editor-in-Chief)
2002년~2005년 Associate Editor, IEEE Trans. on Knowledge and Data Engineering
1990년~2003년 Associate Editor, The VLDB Journal
1990년~1993년 Associate Editor, The IEEE Data Engineering Bulletin
1996년~2004년, 2010~현재 Trustee, The VLDB Endowment
2006년 General Chair, Int'l Conf. on Very Large Data Bases(VLDB)
2007년~2009년 Chair, Steering Committee, DASFAA
1990년~현재 한국과학기술원 전산학과 교수
2008년~현재 한국과학기술원 특훈교수
<주관심분야 : 데이터베이스 시스템, 멀티미디어, 검색엔진, GIS>