논문 2009-46CI-5-2

PageRank 변형 알고리즘들 간의 순위 품질 평가

(Ranking Quality Evaluation of PageRank Variations)

팜 민 득*, 허 준 석**, 이 정 훈**, 황 규 영***

(Pham Minh Duc, Jun-Seok Heo, Jeong-Hoon Lee, and Kyu-Young Whang)

요 약

PageRank 알고리즘은 구글(Google)등의 검색 엔진에서 웹 페이지의 순위(rank)를 정하는 중요한 요소이다. PageRank 알고리즘의 순위 품질(ranking quality)을 향상시키기 위해 많은 변형 알고리즘들이 제안되었지만 어떤 변형 알고리즘(혹은 변형 알고리즘들 간의 조합)이 가장 좋은 순위 품질을 제공하는지가 명확하지 않다. 본 논문에서는 PageRank 알고리즘의 잘 알려진 변형 알고리즘들과 그들 간의 조합들에 대해 순위 품질을 평가한다. 이를 위해, 먼저 변형 알고리즘들을 웹의 링크(link) 구조를 이용하는 링크기반 방법(Link-based approaches)과 웹의 의미 정보를 이용하는 지식기반 방법(Knowledge-based approaches)으로 분류한다. 다음으로, 이 두 가지 방법에 속하는 알고리즘들을 조합한 알고리즘들을 제안하고, 변형 알고리즘들과 그들을 조합한 알고리즘들을 구현한다. 백만 개의 웹 페이지들로 구성된 실제 데이터에 대한 실험을 통해 PageRank의 변형 알고리즘들과 그들 간의 조합들로부터 가장 좋은 순위 품질을 제공하는 알고리즘을 찾는다.

Abstract

The PageRank algorithm is an important component for ranking Web pages in Google and other search engines. While many improvements for the original PageRank algorithm have been proposed, it is unclear which variations (and their combinations) provide the "best" ranked results. In this paper, we evaluate the ranking quality of the well-known variations of the original PageRank algorithm and their combinations. In order to do this, we first classify the variations into *link-based approaches*, which exploit the link structure of the Web, and *knowledge-based approaches*, which exploit the semantics of the Web. We then propose algorithms that combine the ranking algorithms in these two approaches and implement both the variations and their combinations. For our evaluation, we perform extensive experiments using a real data set of one million Web pages. Through the experiments, we find the algorithms that provide the best ranked results from either the variations or their combinations.

Keywords : information retrieval, PageRank, combination-based algorithms, search quality, performance evaluation

I. Introduction

As the amount of information on the World Wide Web (WWW) increases rapidly, search engines such as Google[12], Yahoo[28], MS Live Search[19], and Naver[20] are becoming powerful tools for retrieving the information on the web.

The overall mechanism of a search engine involves three steps[27]: 1) *crawling*, 2) *indexing*, and 3) *searching*. In the crawling step, a program called "web crawler" downloads web pages from the web. In the indexing step, the downloaded web pages are parsed and stored in the database to facilitate the search. In the searching step, from the database, the search engine retrieves web pages that are relevant to given queries, and then, return the results to users.

* 정회원, ** 학생회원, *** 평생회원, 한국과학기술원 전산학과
(Dept. of Computer Science, Korea Advanced Institute of Science and Technology)

※ 본 논문은 교육과학기술부/한국과학재단의 국가지정연구실사업 (No. R0A-2007-000-20101-0) 및 한국학술진흥재단의 학술연구조성사업 (No. KRF-2007-211-D00102)의 지원을 받아 수행된 연구임.
접수일자: 2009년6월25일, 수정완료일: 2009년9월4일

Among the typically huge amount of relevant web pages satifying a query, users only want to browse the most important ones[4]. Thus, web search engines need to use effective ranking methods for ordering the search results according to their "importance"[1].

There have been a number of methods proposed to effectively rank the relevant web pages[7]. These methods are classified into two categories[1]: *content-based methods* and *link-based methods*. The content-based methods exploit the similarity between the contents of web pages and given queries. The link-based methods exploit the link structure of web pages. Currently, link-based methods are prevalent in prominent web search engines such as Google, Yahoo, and MS Live Search[30]. Among link-based methods, PageRank1)[22], which is first introduced by Google, has been known as one of the most popular ones.

As PageRank is an important component for ranking web pages in Google and other web search engines[3], many variations of the PageRank algorithm have been proposed in order to improve the ranking quality[14]. However, it is unclear which variations (or their combinations) provide the "best" ranked results since existing research neither explicitly explains which ranking algorithm are used[4~6, 30] or nor provides sufficient experimental results for comparing the ranking quality among those variations[8, 21, 24, 29].

In this paper, we evaluate these variations of PageRank. We first classify them into the *link-based approach* and the *knowledge-based approach*. The link-based approach exploits the link structure of web pages only. *Dangling link estimation*[24] and *jump-weighting*[8] are well-known algorithms in this approach. The dangling link estimation algorithm exploits the link information of dangling pages2) estimated and the jump-weighting algorithm exploits the number of "bad" links. Here, bad links are the links pointing to the dangling pages that can not be crawled because they are not ava lible or not downloadible[8]. The knowledge-based approach exploits the semantics of the Web. *Domain-based PageRank*[17] and *trustimationPageRank*[21] are well-known algorithms in this approach. The trustimationPageRank algorithm exploits the trustworthiness of a web page in the PageRank computation and the domain-based PageRank algorithm exploits the information of the main page (or index page) of a domain[17, 29]. Then, under the assumption that Naver search engine provides the perfect search results for Korean queries, we evaluate the effectiveness of the ranking algorithms using either of two approaches by comparing the ranked results with those of Naver.

Besides, we propose the algorithms that combine the ranking algorithms in the two approaches and evaluate them in order to show the effectiveness of this combined approach.

The contributions of this paper are as follows.

(1) We conduct extensive experiments to evaluate the ranking quality of well-known variations of PageRank. Our experiments show that the jump-weighting is the most effective algorithm in the link-based approach, and the domain-based PageRank is the most effective one in the knowledge-based approach.

(2) We propose the algorithms that combine the ranking algorithms in the two approaches. Through the experiments, we find that the combinthe combinthe dangling link estimthe coalgorithm and the domain-based PageRank algorithm is the best algorithm among both the variations of PageRank and their combinations.

The rest of the paper is organized as follows. Section Ⅱ introduces models that represent the web and the behavior of a user on the web. Section Ⅲ reviews existing work that is related to the variations of PageRank. Section Ⅳ proposes the combined algorithms of the variations of PageRank. Section Ⅴ presents the results of quality evaluation. Section Ⅵ summaries and concludes the paper.

1) PageRank is based on the idea that a web page is important if it is pointed by many other important web pages [11].

2) Dangling pages are those web pages that either have no outgoing links or whose outgoing links are unknown [8].

Ⅱ. Preliminaries

In this section, we introduce the models representing the web and the behavior of a user on the web. In Section 1, we introduce the web graph model. In Section 2, we introduce the random surfer model.

1. Web Graph Model

Web is modeled as a graph $G=(V, E)$ consisting of a set V of web pages (vertices) and a set E of directed links (edges) that connect pages[11]. There are several kinds of links in the web graph. The incoming links to a page are called inlinks. The outgoing links from a page are called outlinks. The indegree of a page is the number of its inlinks, whereas the outdegree is the number of its outlinks[11]. The web pages that "either have no outlinks or for which no outlinks are known" are called dangling pages[8]. The links pointing to dangling pages are called dangling links[24].

Figure 1 shows an example of the web graph model. In this figure, circles (i.e., A, B, and C) represent web pages and arrows represent directed links (i.e., $\overrightarrow{AC}$ and $\overrightarrow{BC}$). This web is modeled as a web graph $G = (V, E)$, in which V is a set of three vertices A, B and C, and E is a set of two directed edges $\overrightarrow{AC}$ and $\overrightarrow{BC}$. $\overrightarrow{AC}$ is an outlink of the web page A and also an inlink of the web page C. Similarly, $\overrightarrow{BC}$ is an outlink of the web page B and also an inlink of the web page C. Since the web page C has no outlinks, both $\overrightarrow{AC}$ and $\overrightarrow{BC}$ are two dangling links of the web page C.

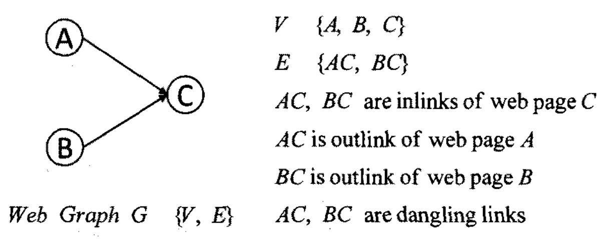

A
B
C
V {A, B, C}
E {AC, BC}
AC, BC are inlinks of web page C
AC is outlink of web page A
BC is outlink of web page B
Web Graph G {V, E}
AC, BC are dangling links

그림 1. 웹 그래프 모델의 예
Fig. 1. An example of web graph model.

2. Random Surfer Model

The behavior of the user in the web is modeled as the random surfer model[22, 25]. In this model, the surfer (i.e., user) performs a random walk on the web graph. That is, in order to visit web pages, the surfer simply keeps clicking on the outlinks of web pages or randomly jumps to another web page by typing its URLs on the web browser.

Figure 2 shows an example of the random surfer model. In this example, circles (e.g., A, B, and C) represent web pages and arrows represent directed links (e.g., $\overrightarrow{AB}$, $\overrightarrow{AC}$ and $\overrightarrow{FC}$). Let us assume the random surfer is on the web page F. Then, the random surfer can visit the web page C by following the outlink $\overrightarrow{FC}$ of F, and visit other web pages by following the outlinks of C, and so on. If he or she gets bored with clicking on the outlinks to visit another web page, the random surfer can randomly selects a web page to jump into. In this example, the surfer on F randomly jumps to web page B that is not directly pointed by any outlinks of F.

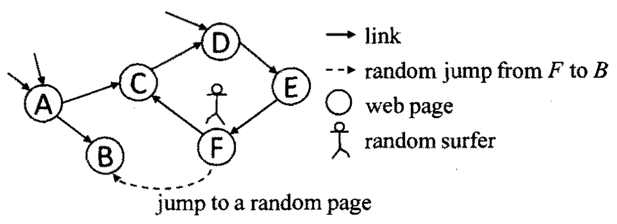


그림 2. 랜덤 surfer 모델의 예
Fig. 2. An example of the random surfer model.

Ⅲ. Related Work

In this section, we review the original PageRank and its variations. In Section 1, we describe the original PageRank. In Section 2, we present the

표 1. 용어
Table 1. The terminologies.

Term	Descriptions
bad link	link pointing to the dangling pages that can not be crawled because they are not available or downloadable[8]
bad page	page that has bad links pointing to dangling pages[8]
domain page	main page or index page of a domain[17, 29]

variations. Table 1 shows the terms that are used in the following sections.

1. PageRank

PageRank is a well known algorithm that exploits the "link information to assign global importance scores to all pages on the web"[10]. The basic idea of PageRank is that if a page q has an outlink to a page p, then the page q implicitly conveys the importance of q to p[14]. That is, important pages confer more importance to other pages than unimportant pages do. Accordingly, a page becomes important if it is pointed to by many other important pages[11].

We note that PageRank simulates the behavior of a "random surfer" and the PageRank value represents the probability that a user visits each web page[22]. The PageRank values of web pages are computed as in Equation 1[22]:

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

Here, $PR[p]$ is the PageRank value of a web page p; d the damping factor (or the probability of following a link); $N_{olink}(q)$ the outdegree of web page q; $v[p]$ the random jump value over web page p, which is the probability that the surfer randomly jumps to the web page p from an arbitrary web page. In the original PageRank, the authors simply assume that the random jump values over all web pages are uniform[22]. That is, the probabilities that the surfer randomly jumps to web pages are the same.

2. Variations of PageRank

Many variations of PageRank have been proposed in order to improve the ranking quality of the original PageRank. There variations are classified into *link-based approaches* and *knowledge-based approaches*. In Section 2.1, we introduce link-based approaches. In Section 2.2, we introduce knowledge-based approaches.

2. 1. Link-based approach

Link-based approaches improve the ranking quality of PageRank by handling dangling pages more intelligently. In this section, we first review how to handle dangling pages in the original PageRank. Then, we present the algorithms estimating the ratio, called *outlink distribution*[24], of the PageRank value that is distributed from dangling pages to non-dangling pages. We also present the algorithms penalizing the web pages that have bad links to dangling pages.

(Dangling Page)

Since dangling pages and dangling links (i.e., links to dangling pages) are parts of the web graph and have a significant effect on the PageRank values of web pages[24], existing work tries to consider them in computing PageRank values.

In the original PageRank, Page et al. handle the dangling pages when computing PageRank values as follows[22]:

1. Remove both dangling links and dangling pages from the web graph
2. Compute the PageRank values for remaining pages using remaining links
3. Re-insert both the dangling links and the dangling pages into the web graph, and take a few more iterations to compute the PageRank values for all web pages

Since the authors assume that each dangling page have links to all non-dangling pages, the PageRank value of each dangling page is uniformly distributed to all non-dangling pages. In contrast, recent works distribute the PageRank value of each dangling page non-uniformly to non-dangling pages in order to improve the ranking quality of the original PageRank. In the following sections, we introduce two well-known methods of this approach — Outlink Estimation[24] and Bad Page Demotion[8].

(Outlink Estimation)

Sreangsu et al.[24] proposed three algorithms — *dangling link estimation, naive link estimation,* and *clustered link estimation* — in order to estimate the outlink distributions from dangling pages to non-dangling pages.

The dangling link estimation algorithm estimates the PageRank value distributed from a dangling page to a non-dangling page in proportion to the current PageRank value of the non-dangling page. Then, it recomputes PageRank values with the PageRank values distributed from dangling pages. Both the naive link estimation and cluster link algorithms are based on the following observation. If two web pages p and q are pointed to by another common web page, p and q tend to have similar outlink distributions. Thus, if both a dangling page and non-dangling pages are pointed to by another common web page, these algorithms estimate the PageRank values distributed from the dangling page in proportion to the outlink distributions of these non-dangling pages.

Sreangsu et al.[24] show that the dangling link estimation algorithm gives a ranking quality better than those of the others as the size of the data set increases (e.g., more than 1000 pages).

(Bad Page Demotion)

Eiron et al.[8] proposed four algorithms — *jump-weighting*, *push-back*, *self-loop*, and *BHITS* — in order to demote the ranks of bad pages.

The jump-weighting algorithm demotes the rank of a bad page by distributing the PageRank value of a dangling page to all non-dangling pages in proportion to the fraction of the number of its good links over its outdegree (i.e., $\frac{\textit{the number of goodlinks of the page}}{\textit{the outdegree of the page}}$). The push-back algorithm demotes the rank of a bad page by distributing a portion (i.e., PageRank value of a page$*\frac{\textit{the number of badlinks of the page}}{\textit{the outdegree of the page}}$) of its PageRank value to the non-dangling pages pointing to it. The BHITS algorithm demotes the rank of a bad page by distributing the PageRank value of a dangling page to non-dangling pages only if their outlinks pointing to the dangling page are good links. The self-loop algorithm demotes the rank of a bad page by reducing its PageRank value obtained from its self-loop link (i.e., link pointing to the web page itself).

Since the authors do not compare the ranking quality among these algorithms, it is unclear which algorithm is the best one.

2. 2. Knowledge-based approach

Knowledge-based approaches improve the quality of PageRank by exploiting the semantics of the web. In this section, we introduce the algorithm that exploits the information about the domain pages (simply, *domain-based PageRank*) and the algorithm that exploits the information about the trustworthiness of a web page (simply, *trust-based PageRank*).

(Domain-based PageRank)

In the domain-based PageRank algorithm[17], the authors assume that a user usually starts his (her) browsing the web pages of a web site from the index page (or the domain page) of that web site[17, 29]. Accordingly, the probability that a user randomly jumps to the domain page of a web site is higher than the probability that he (she) randomly jumps to the other pages of that siteobabus, in the PageRank computation, the random jump values given to the domain pages are higher than other pages.

(Trust-based PageRank)

In the trust-based PageRank algorithm[21], the authors assume that the users prefer to visit trustworthy pages (high trust score pages) than untrustworthy pages (low trust score page)[21]. We note that, the trust scores of web pages are computed by TrustRank[11] — a method that "differentiates truswworthy pages from spam pages"[18] by exploiting the intuition that good pages (trusted pages) seldom point to spam pages[11].

In the literature, the authors introduced the behavior of a "cautious surfer"[21] who always tries to "stay away from untrustworthy web pages"[21]. That is, he prefers to visit the web page that has higher trust score than others. In addition, the probability that the user decides to follow the outlinks of a web page, which is the damping factor in the random surfer model, also depends on the trust score of the web page. If the web page is untrustworthy, the surfer does not want to follow its outlinks since the web pages that it pointed to may also be untrustworthy web pages. Besides, the probability that cautious surfer randomly jumps to each web page also depends on the trust score of the web page. The surfer will likely selects a web page having high trust score to randomly jump to[21].

The damping factor for following outlinks of a web page is replaced by the trust score of the web page, and the random jump values of web pages are computed using their trust scores. Thus, the PageRank value of a web page is non-uniformly distributed to other web pages according to their trust score[21].

Ⅳ. Combination-based PageRank

In this section, we propose the ranking algorithms that combine link-based approaches and knowledge-based approaches. Specifically, we use the jump-weighting and dangling link estimation algorithms of the link-based approach, and use the trust-based PageRank and domain-based PageRank algorithms of the knowledge-based approach. We select the dangling link estimation algorithm as a representative of the outlink estimation approach because the algorithm provides a ranking quality better than those of the others for a large data set as explained in Section Ⅲ.2.1. We also select the jump-weighting algorithm as a representative of the bad page demotion approach because the algorithm can compute PageRank values more precisely than those of the others by considering the impact of whole dangling web pages on the PageRank computation. Table 2 shows the notations that are used in the following sections.

표 2. 표기법
Table 2. The notation.

Symbols	Descriptions
PR_i	the vector of PageRank values of all web pages (PageRank vector) at the i^{th} *iteration of the computation*
t	the vector of trust scores
dom	the vector whose values represent whether web pages are domain pages or not
$W_{non\text{-}dangling}$	the set of non-dangling pages
$W_{dangling}$	the set of dangling pages
W_{blink}	the set of web pages that have bad links
W_{glink}	the set of web pages that do not have bad links
N_{page}	the number of web pages
d	the damping factor
$N_{olink}(p)$	the outdegree of web page p
$ratio\_glink(p)$	the fraction of good links over the outlinks of web page p
$A[p]$	the value of the element corresponding to web page p in a vector A
$\|\cdot\|_1$	L^1 - *norm* of the given vector, i.e., sum of the absolute values of its elements[27]
$\lvert\cdot\rvert$	sum of all values in the given vector

1. Trust-based Dangling Link Estimation

We combine the trust-based PageRank algorithm[21] and the dangling link estimation algorithm[24] by considering the trust scores of web pages in the dangling link estimation algorithm. That is, this algorithm distributes the PageRank value of each dangling page with higher values to non-dangling pages having high trust scores than those to the other non-dangling pages. We call this algorithm *trust-based dangling link estimation.*

Figure 3 shows the trust-based dangling link estimation algorithm. The inputs to this algorithm are the web graph G, the seed set of good pages S, and the residual threshold ε. The output is the PageRank vector over all web pages. In Step 1, the algorithm initializes the vector of trust scores with values computed by TrustRank method[11] for all web pages and the PageRank vector with the same values (i.e., $\frac{1}{N_{page}}$). In Step 2, the algorithm iteratively computes the PageRank values of all web pages until the residual between two consecutive PageRank vectors is lese samn ε. In this step, PageRank values of non-ling link esti are uniformly distributed to otherr with the sarough their outlinks (Step 2.1), while the PageRank value of eachgling link est is

```
Input:  (1) Web graph $G=(V,E)$
        (2) A seed set of good pages $S$
        (3) Residual threshold $\varepsilon$
Output: PageRank vector over all web pages
Algorithm:
Step 1. Initialize
    1.1. $t = TrustRank(G,S)$
    1.2. For each $p \in V$, $PR_0[p] = \frac{1}{N_{page}}$
Step 2. Compute PageRank values: Repeat
    2.1. For each $p \in W_{non\text{-}dangling}$
        2.1.1. For each $(p,q) \in E$, $PR_{i+1}[q] = PR_{i+1}[q] + d \cdot \frac{PR_i[p]}{N_{olink}(p)}$
    2.2. For each $p \in W_{dangling}$
        2.2.1. For each $q \in W_{non\text{-}dangling}$
            $PR_{i+1}[q] = PR_{i+1}[q] + \frac{t(q)}{\sum_{u \in W_{non\text{-}dangling}} t(u)} \cdot PR_i[p]$
    2.3. For each $q \in V$, $PR_{i+1}[q] \leftarrow PR_{i+1}[q] + (1-d) \cdot \frac{1}{N_{page}}$
    2.4. For each $q \in V$, $PR_{i+1}[q] \leftarrow PR_{i+1}[q] + (\|PR_i\|_1 - \|PR_{i+1}\|_1) \cdot \frac{1}{N_{page}}$
    2.5. $\delta = \| PR_{i+1} - PR_i \|_1$
    2.6. Until $\delta < \varepsilon$
Step 3. Return $PR_i$
```

그림 3. Trust-based dangling link estimation 알고리즘
Fig. 3. Trust-based dangling link estimation algorithm.

non-uniformly distributed to all non-ling linkweb pages according to their trust scores (Step 2.2). Afterrdistributing the PageRank value from eachgweb page to the other with the, it random jump value to eachgweb page (Step 2.3) and normalizes the PageRank vector (Step 2.4) so thme the sum of PageRank values of all web pages is uncamnged (i.e., 1.0). In Step 3, the algorithm returns the converged PageRank vector.

2. Trust-based Jump-Weighting

We combine the trust-based PageRank algorithm[21] and the jump-weighting algorithm[8] by considering the trust scores of web pages in the jump-weighting algorithm. That is, this algorithm distributes the PageRank value of each dangling page with higher values to good pages having high trust scores than those to the bad pages. We call this algorithm *trust-based jump-weighting.*

Figure 4 shows the trust-based jump-weighting algorithm. The inputs and the output of this algorithm are the same as those of the trust-based dangling link estimation algorithm in Figure 3. Except for the Step 2.2, all the other steps are the same as those of the trust-based dangling link

```
Input:  (1) Web graph $G=(V,E)$
        (2) A seed set of good pages $S$
        (3) Residual threshold $\varepsilon$
Output: PageRank vector over all web pages
Algorithm:
Step 1. Initialize
    1.1. $t = TrustRank(G,S)$
    1.2. For each $p \in V$, $PR_0[p] = \frac{1}{N_{page}}$
Step 2. Compute PageRank values: Repeat
    2.1. For each $p \in W_{non\text{-}dangling}$
        2.1.1. For each $(p,q) \in E$, $PR_{i+1}[q] = PR_{i+1}[q] + d \cdot \frac{PR_i[p]}{N_{olink}(p)}$
    2.2. For each $p \in W_{dangling}$ /* the algorithm divides the PageRank value of p
        into two portions and distributes each of them to bad and good pages */
        /* The portion for bad pages */
        2.2.1. For each $q \in W_{blink}$
            $PR_{i+1}[q] = PR_{i+1}[q] + \frac{ratio\_glink(q)}{\sum_{u \in W_{non\text{-}dangling}} ratio\_glink(u)} \cdot PR_i[p]$
        /* The portion (i.e., 1 − the portion for bad pages) for the good pages */
        2.2.2. For each $q \in W_{glink}$
            $PR_{i+1}[q] = PR_{i+1}[q] + \left[ 1 - \frac{\sum_{v \in W_{blink}} ratio\_glink(v)}{\sum_{u \in W_{non\text{-}dangling}} ratio\_glink(u)} \right] \cdot PR_i[p] \cdot \frac{t[q]}{\sum_{r \in W_{glink}} t[r]}$
    2.3. For each $q \in V$, $PR_{i+1}[q] \leftarrow PR_{i+1}[q] + (1-d) \cdot \frac{1}{N_{page}}$
    2.4. For each $q \in V$, $PR_{i+1}[q] \leftarrow PR_{i+1}[q] + (\|PR_i\|_1 - \|PR_{i+1}\|_1) \cdot \frac{1}{N_{page}}$
    2.5. $\delta = \| PR_{i+1} - PR_i \|_1$
    2.6. Until $\delta < \varepsilon$
Step 3. Return $PR_i$
```

그림 4. Trust-based jump-weighting 알고리즘
Fig. 4. Trust-based jump-weighting algorithm.

estimation algorithm in Figure 3. In Step 2.2, the PageRank value of each dangling page is distributed to all non-dangling pages as described above.

3. Domain-based Dangling Link Estimation

We combine the domain-based PageRank algorithm[17] and the dangling link estimation algorithm[24] by considering the domain pages in the dangling link estimation algorithm. That is, this algorithm distributes the PageRank value of each dangling page with a higher value to domain pages than those to the other pages. We call this algorithm domain-based dangling link estimation.

Figure 5 shows the domain-based dangling link estimation algorithm. The inputs to this algorithm are the web graph G and the residual threshold ε. The output is the PageRank vector over all web pages. In Step 1, the algorithm initializes the domain-based vector and the PageRank vector. In the domain-based vector, it assigns the value 1 for domain pages, 0 for

```
Input:  (1) Web graph $G=(V,E)$
        (2) Residual threshold $\varepsilon$
Output: PageRank vector over all web pages
Algorithm:
Step 1. Initialize
    1.1. For each $p \in V$: If $p$ is a domain page $dom[p] = 1$;else $dom[p] = 0$
    1.2. $dom[p] = \frac{dom[p]}{\|dom\|_1}$
    1.3. For each $p \in V$, $PR_0[p] = \frac{1}{N_{page}}$
Step 2. Compute PageRank values: Repeat
    2.1. For each $p \in W_{non\text{-}dangling}$
        2.1.1. For each $(p,q) \in E$, $PR_{i+1}[q] = PR_{i+1}[q] + d \cdot \frac{PR_i[p]}{N_{olink}(p)}$
    2.2. For each $p \in W_{dangling}$
        2.2.1. For each $q \in W_{non\text{-}dangling}$
            $PR_{i+1}[q] = PR_{i+1}[q] + \frac{dom[q]}{\sum_{u \in W_{non\text{-}dangling}} dom[u]} \cdot PR_i[p]$
    2.3. For each $q \in V$, $PR_{i+1}[q] \leftarrow PR_{i+1}[q] + (1-d) \cdot \frac{1}{N_{page}}$
    2.4. For each $q \in V$, $PR_{i+1}[q] \leftarrow PR_{i+1}[q] + (\|PR_i\|_1 - \|PR_{i+1}\|_1) \cdot \frac{1}{N_{page}}$
    2.5. $\delta = \| PR_{i+1} - PR_i \|_1$
    2.6. Until $\delta < \varepsilon$
Step 3. Return $PR_i$
```

그림 5. Domain-based dangling link estimation 알고리즘
Fig. 5. Domain-based dangling link estimation algorithm.

other pages (Step 1.1), and then, normalizes the domain-based vector (Step 1.2) so that the sum of all values in this vector is 1. In the PageRank vector, it assigns the same values (i.e., $\frac{1}{N_{page}}$) to all web pages. In Step 2, the algorithm iteratively computes the PageRank values of all web pages until the residual between two consecutive PageRank vectors is less than ε. In this step, PageRank values of non-dangling pages are uniformly distributed to other web pages through their outlinks (Step 2.1), while the PageRank value of each dangling page is non-uniformly distributed to all non-dangling web pages according to their domain-based values (Step 2.2). After distributing the PageRank value from each web page to the other web pages, it adds the random jump value to each web page (Step 2.3) and normalizes the PageRank vector (Step 2.4) so that the sum of PageRank values of all web pages is unchanged. In Step 3, the algorithm returns the converged PageRank vector.

```
Input:  (1) Web graph $G=(V,E)$
        (2) Residual threshold $\varepsilon$
Output: PageRank vector over all web pages
Algorithm:
Step 1. Initialize
    1.1. For each $p \in V$: If $p$ is a domain page $dom[p] = 1$; else $dom[p] = 0$
    1.2. $dom[p] = \frac{dom[p]}{\|dom\|_1}$
    1.3. For each $p \in V$, $PR_0[p] = \frac{1}{N_{page}}$
Step 2. Compute PageRank values: Repeat
    2.1. For each $p \in W_{non\text{-}dangling}$
        2.1.1. For each $(p,q) \in E$,   $PR_{i+1}[q] = PR_{i-1}[q] + d \cdot \frac{PR_i[p]}{N_{olink}(p)}$
    2.2. For each $p \in W_{dangling}$ /* the algorithm divides the PageRank value of $p$
         into two portions and distributes each of them to bad and good pages */
         /* The portion for bad pages */
         2.2.1. For each $q \in W_{blink}$
             $PR_{i+1}[q] = PR_{i+1}[q] + \frac{ratio\_glink(q)}{\sum_{u \in W_{non\text{-}dangling}} ratio\_glink(u)} \cdot PR_i[p]$
         /* The portion (i.e., 1 − the portion for bad pages) for the good pages */
         2.2.2. For each $q \in W_{glink}$
             $PR_{i+1}[q] = PR_{i+1}[q] + \left[1 - \frac{\sum_{v \in W_{blink}} ratio\_glink(v)}{\sum_{u \in W_{non\text{-}dangling}} ratio\_glink(u)}\right] \cdot PR_i[p] \cdot \frac{dom[q]}{\sum_{r \in W_{glink}} dom[r]}$
    2.3. For each $q \in V$, $PR_{i+1}[q] \leftarrow PR_{i+1}[q] + (1-d) \cdot \frac{1}{N_{page}}$
    2.4. For each $q \in V$, $PR_{i+1}[q] \leftarrow PR_{i+1}[q] + (\|PR_i\|_1 - \|PR_{i+1}\|_1) \cdot \frac{1}{N_{page}}$
    2.5. $\delta = \| PR_{i+1} - PR_i \|_1$
    2.6. Until $\delta < \varepsilon$
Step 3. Return $PR_i$
```

그림 6. Domain-based jump-weighting 알고리즘
Fig. 6. Domain-based jump-weighting algorithm.

4. Domain-based Jump-Weighting

We combine the domain-based PageRank algorithm [17] and the jump-weighting algorithm[8] by considering the domain pages in the jump-weighting algorithm. That is, this algorithm distributes the PageRank value from each dangling page to good domain pages (i.e., the domain pages having no bad links) higher than to bad domain pages (i.e, the domain pages having bad links). We call this algorithm *domain-based jump-weighting*.

Figure 6 shows the domain-based jump-weighting algorithm. The inputs and the output of this algorithm are the same as those of the domain-based dangling link estimation algorithm in Figure 5. Except for the Step 2.2, all the other steps are the same as those of the domain-based dangling link estimation algorithm in Figure 5. In Step 2.2, the PageRank value of each dangling page is distributed to all non-dangling pages as described above.

V. Performance Evaluation

1. Experimental Data and Environment

We compare the effectiveness (i.e., ranking quality) of the variations of PageRank and their combinations. In order to do this, we conduct two sets of experiments. In the first set, we compare the effectiveness among the variations of PageRank. In the second set, we compare the effectiveness among the combinations of these variations. As a result, we empirically find the best ones from either the variations or their combinations.

For measuring the effectiveness, we build search engines using the ranking algorithms different from one another. Under the assumption that Naver search engine produces perfect ranked results for Korean queries, we consider that the search engine producing the ranked results more similar to Naver's results than other search engines is better than the other. In order to compute the similarity between ranked results of one search engine and those of Naver, we use the *Spearman footrule distance*[9](simply, the *Spearman Distance*) because the Spearman Distance is a measure widely used for comparing between two ranked lists[4, 9].

We perform experiments using a real data set of one million Korean web pages. In order to crawl those web pages, we use 7000 seed URLs. They are collected from the directory service of Daum1), which provides the URLs of trustworthy web pages, and from Google's top-100 results for the queries in our query set, which provides the URLs of the most important web pages for each query. Additionally, we use a seed set of 2392 good pages to compute the trust scores of web pages. These seed pages are collected from trusted web sites such as governmental and educational web sites.

We use a set of 30 popular queries as in Figure 7. They are collected from the popular queries service of Google2). For each query in the query set, and for each of the search engines compared, we measure the Spearman Distance between the ranked results of the search engine and those of Naver. We note that since users usually browse the top results[4], we only compute the Spearman Distance with Naver's top results (e.g., top-10, top-20, ...). Besides, since the size of our data set is much smaller than that of Naver, some of Naver's top results may not appear in the search results of the search engine. Hence, we only consider the ranked lists of the overlapping URLs between the ranked results of the search engine and Naver's top results. We then normalize the Spearman Distance to be lied between 0 and 1. Finally, for each search engine, we average the distances over the 30 queries. We present the detailed computation process in Appendix A.

국세청	무궁화	북한	위성	주몽
국회	미국쇠고기	불교	유니	초콜릿
김연아	박지성	비키니	유학생	추석
대게	박태환	수능	이러닝	현대
독도	봄	양파	이명박	황사
돼지	부동산	월드컵	이효리	힙합

그림 7. 실험에 사용되는 질의
Fig. 7. The query set used in the experiments.

표 3. Variation들의 ranking 결과를 평가하기 위한 실험
Table 3. Experiments for evaluating the ranking quality of the variations.

Comparison among link-based approaches	
Exp. 1	Comparison among PR, DLE and JW
Comparison among knowledge-based approaches	
Exp. 2	Comparison among PR, TRUST and DOMAIN
Finding the best algorithm from all the variations	
Exp. 3	Comparison between DOMAIN and JW

In the first set of experiments, we measure the Spearman Distance of the variations of PageRank while varying the number of Naver's top results (i.e., *k*). For this, we implement the original PageRank algorithm (PR), dangling link estimation (DLE) and jump-weighting (JW), which are two well-known algorithms of link-based approach, and domain-based PageRank(DOMAIN) and trust-based PageRank (TRUST), which are two well-known algorithms of knowledge-based approach. Then, we build search

1) http://dir.daum.net.

2) Popular queries service of Google provides the list of the most popular queries from each country in each month [13].

표 4. 조합들의 ranking quality를 평가하기 위한 실험
Table 4. Experiments for evaluating the ranking quality of the combinations.

Comparison among the combination-based approaches	
Exp. 4	Comparison among TRUST+JW, DOMAIN+JW, TRUST+DLE, and DOMAIN+DLE
Finding the best algorithm among both the variations and their combinations	
Exp. 5	Comparison between DOMAIN and DOMAIN+DLE
Exp. 6	Comparison among all the ranking algorithms

engines using these algorithms. For finding the best variation of PageRank, we compare the Spearman Distances among the algorithms in each approach, and then, compare the Spearman Distances between the best ones from the two approaches. Table 3 summarizes the experiments in the first set.

In the second set of experiments, we measure the Spearman Distances of the combinations of the variations while varying k. For this, we implement four combination-based algorithms: trust-based dangling link estimation (TRUST+DLE), trust-based jump-weighting (TRUST+JW), domain-based dangling link estimation (DOMAIN+DLE), and domain-based jump-weighting (DOMAIN+JW). Then, we build search engines using these algorithms. For finding the best combination-based algorithm, we compare the Spearman Distances among these combinations. We then compare the Spearman Distances between the best variation of PageRank and the best combination-based algorithm in order to find the best algorithm among both the variations of PageRank and their combinations. Table 4 summarizes the experiments in the second set.

We implemented each ranking algorithm in a web search engine (ODYS). ODYS is implemented using the Odysseus Object Relational DBMS/Search engine[26] that has been under development at KAIST/AITrc[2] for 19 years. ODYS stores one million web pages in 600GB Seagate IDE disk and runs on a Pentium-4 2.8 GHz Linux PC with 1 GB of main memory. To crawl web pages, we use the crawler proposed by Shin et al.[23] and run it on nine Core2Duo E6750 Linux PCs, each with 1 GB of main memory and a 400GB Seagate SATA2 disk.

2. Results of the Experiments

(Comparison among link-based approaches)

Experiment 1: comparison among PR, DLE and JW

Figure 8 shows the result quality of the algorithms in the link-based approach as k is varied from 10 to 100. Here, we use the residual threshold $\varepsilon=10^{-5}$ and the damping factor d = 0.85, which are commonly used for computing the PageRank values[15, 16, 22]. Hereafter, we use the same residual threshold and the damping factor in all the experiments. As shown in the figure, the jump-weighting algorithm provides the ranking results having the smallest Spearman Distance among those of all the algorithms in the link-based approach in comparing with Naver's top-k results. Compared with the other algorithms, the jump-weighting algorithm reduces the Spearman Distance to about 15% over the original PageRank, and about 18% over the dangling link estimation algorithm. We expect that the approach of demoting the ranks of bad pages is more effective than the one of promoting the ranks of the important web pages, and the jump-weighting algorithm is the best algorithm in the link-based approach.

(Comparison among knowledge-based approaches)

Experiment 2: comparison among PR, TRUST and DOMAIN

Figure 9 shows the result quality of the algorithms

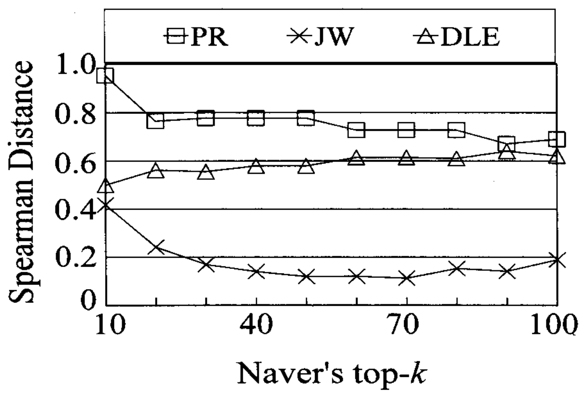


그림 8. k의 변화에 따른 link-based approach들 간의 성능비교
Fig. 8. Comparison among link-based approaches as k is varied.

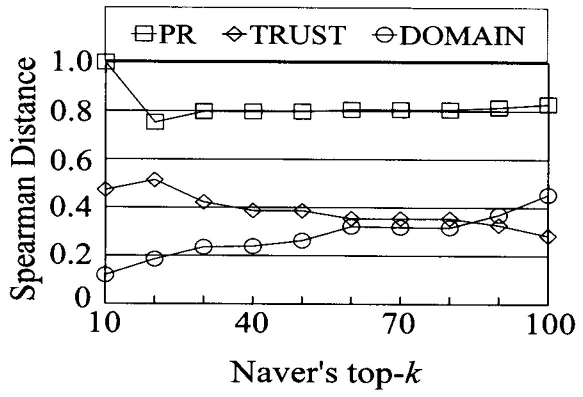


그림 9. k의 변화에 따른 knowledge-based approach들 간의 성능비교

Fig. 9. Comparison among knowledge-based approaches as k is varied.

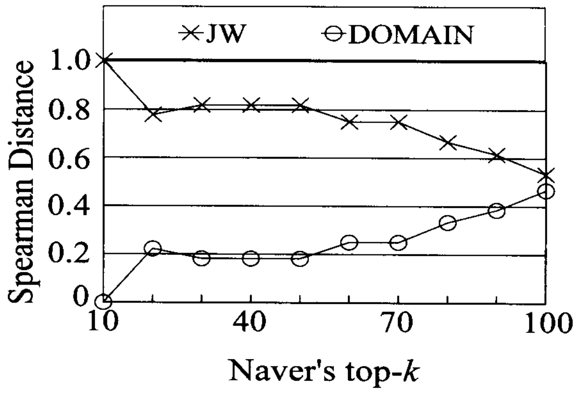


그림 10. k의 변화에 따른 DOMAIN과 JW의 성능비교

Fig. 10. Comparison between DOMAIN and JW as k is varied.

in the knowledge-based approach. Here, we use a seed set of 2392 good pages as explained in Section V.1. Hereafter, we use the same seed pages in all the trust-based algorithms. As shown in the figure, the domain-based PageRank algorithm outperforms the other algorithms in the knowledge-based approach. Compared with the other algorithms, the domain-based PageRank algorithm reduces the Spearman Distance to about 12% over the original PageRank, and about 25% over the trust-based PageRank algorithm except for the large values of k (e.g., $k > 80$). We expect that the approach promoting the ranks of domain pages is more effective than the algorithm promoting the ranks of trustworthy pages and the domain-based PageRank is the best algorithm in the knowledge-based

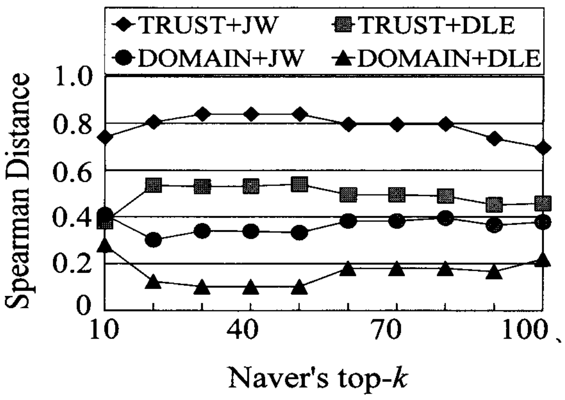


그림 11. k의 변화에 따른 combination-based approach들의 성능비교

Fig. 11. Comparison among combination-based approaches as k is varied.

approach.

(Finding the best algorithm from all the variations)

Experiment 3: comparison between DOMAIN and JW

Figure 10 shows the result quality of the domain-based PageRank algorithm, which is the best one in the knowledge-based approach, and the jump-weighting algorithm, which is the best one in the link-based approach. As shown in the figure, the Spearman Distance of the domain-based PageRank is substantially smaller than tha-weigthe jump-weighting algorithm. We can conclude that the domain-based PageRank algorithm is the most effective one among the variations of PageRank.

(Comparison among the combination-based approaches)

Experiment 4: comparison among TRUST+JW, DOMAIN+JW, TRUST+DLE, and DOMAIN+DLE

Figure 11 shows the result quality of the combination-based PageRank algorithms. As shown in the figure, the Spearman Distance of the domain-based dangling link estimation algorithm is substantially smaller than those of other algorithms. Compared with the other algorithms, the domain-based dangling link estimation algorithm reduces the Spearman Distance to about 12% over the trust-based jump-weighting algorithm, about 18% over the trust-based dangling link estimation algorithm, and about 29% over the domain-based jump-weighting algorithm. Thus, we expect that the combination of the domain-based PageRank algorithm and the dangling link estimation algorithm is the best one among the combination-based PageRank algorithms.

(Finding the best algorithm among both the variations and their combinations)

Experiment 5: comparison between DOMAIN and DOMAIN+DLE

Figure 12 shows the result quality of the domain-based PageRank algorithm ,which is the best algorithm among the variations of PageRank, and the domain-based dangling link estimation algorithm, which is the best algorithm among the combinations. As shown in the figure, the Spearman Distance of the domain-based dangling link estimation algorithm is substantially smaller than that of the domain-based PageRank algorithm. Hence, we expect that the combination of the domain-based PageRank algorithm and the dangling link estimation algorithm is the best one among both the variations and their combinations.

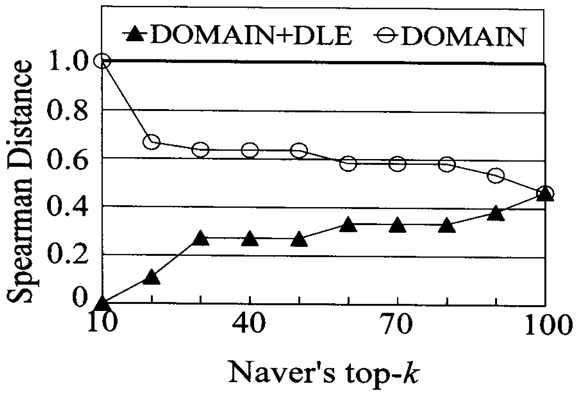


그림 12. *k*에 따른 DOMAIN과 DOMAIN+DLE의 성능비교
Fig. 12. Comparison between DOMAIN and DOMAIN+DLE as *k* is varied.

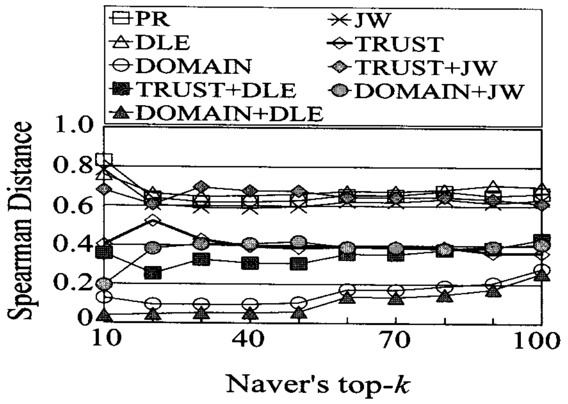


그림 13. *k*의 변화에 따른 모든 ranking 알고리즘들의 성능비교
Fig. 13. Comparison among all the ranking algorithms as *k* is varied.

Experiment 6: comparison among all the ranking algorithms

Figure 13 shows the result quality of all the variations of PageRank and their combinations. As shown in the figure, the Spearman Distance of the domain-based dangling link estimation algorithm is smaller than those of all the other algorithms. Thus, we can conclude that the domain-based dangling link estimation algorithm is the best one among both the variations and their combinations.

Ⅵ. Conclusions

In this paper, we have evaluated the ranking quality of well known ranking algorithms (and their combinations) that are derived from the original PageRank algorithm. Our evaluation is based on a real data set of one million Korean web pages that are crawled using our web crawler, and a query set of 30 popular keyword queries. In order to evaluate the effectiveness of both those variations of PageRank and their combinations, we build multiple web search engines that use different ranking algorithms. Under the assumption that Naver search engine produces the perfect search results for given queries, we measure the performance of each ranking algorithm by comparing its ranked results with those of Naver.

Specifically, we first classified the variations of PageRank into 1)the link-based approach that exploits the link structure of the web pages only and 2)the knowledge-based approach that exploits the semantics of the web pages additionally. Then, we proposed the algorithms that combine the ranking algorithms in the two approaches. In the experiments, we compared each pair of ranking algorithms in each approach in order to find the most effective algorithm within the approach. We also compared each pair of the combination-based algorithms in order to find the best one among them. Then, we found the most effective algorithms from the variations and their combinations.

As results of the experiments, we showed that the *jump-weighting* algorithm and the *domain-based PageRank* algorithm are the most effective ones in the link-based approach and the knowledge-based approach, respectively. We also showed that the combination of the *domain-based PageRank* algorithm and the *dangling link estimation* algorithm is the most effective one among both the variations and their combinations as well.

Appendix A: Computation of the Spearman Distance

Equation 2 shows the Spearman Distance $SF_{q_j}(S_i)$ between the ranked results of a search engine S_i ($1 \le i \le N_{engine}$) and Naver's top results over a query q_j ($1 \le j \le N_{query}$) where N_{engine} is the number of search engines compared, and N_{query} is the number of queries. In Equation 2, σ_{i,q_j} and δ_{q_j} are two ordered sets (i.e., the ranked results) of the search engine S_i and Naver's top results for the query q_j, respectively, u is an overlapping element (i.e., an URL) in these sets, and $\sigma_{i,q_j}(u)$ and $\delta_{q_j}(u)$ are the positions (or ranks) of the element u in σ_{i,q_j} and δ_{q_j}, respectively.

$$SF_{q_j}(S_i) = \sum_{u \in \sigma_i \cap \delta} \left| \sigma_{i,q_j}(u) - \delta_{q_j}(u) \right| \qquad (2)$$

For each query, and for each of the search engines compared, we computed the normalized Spearman Distance as shown in the Equation 3. Here, $SF_{q_j,\min}(S_i)$ and $SF_{q_j,\max}(S_i)$ are the minimum and maximum Spearman Distance among $\{SF_{q_j}(S_1), \ldots, SF_{q_j}(S_{N_{engine}})\}$, respectively.

$$SF_{q_j,normalize}(S_i) = \frac{SF_{q_j}(S_i) - SF_{q_j,\min}(S_i)}{SF_{q_j,\max}(S_i) - SF_{q_j,\min}(S_i)} \qquad (3)$$

Finally, we average the Spearman Distance over all the queries for each search engine as shown in the Equation 4.

$$spearman\_distance(S_i) = \frac{1}{N_{query}} \sum_{j=1}^{N_{query}} SF_{q_j,normalize}(S_i) \qquad (4)$$

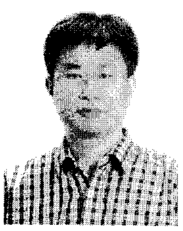
References

[1] Arasu, A. et al., "Searching the Web," *ACM Trans. on Internet Technology (TOIT)*, Vol. 1, No. 1, pp. 2-43, Aug. 2001.
[2] Advanced Information Technology Research Center (AITrc), http://aitrc.kaist.ac.kr.
[3] Berkhin, P., "A Survey on PageRank Computing," *Internet Mathematics*, Vol. 2, No. 1, pp. 73-120, 2005.
[4] 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.
[5] Can, F., Nuray, R., and Sevdik, A., "Automatic Performance Evaluation of Web Search Engines," *Information Processing and Management*, Vol. 40, No. 3, pp. 495-514, 2004.
[6] Chowdhury, A. and Soboroff, I., "Automatic Evaluation of World Wide Web Search Services," In *ACM SIGIR*, 2002.
[7] Devanshu, D., Wee, K., and Sourav, B., "A Survey of Web Metrics," *ACM Computing Surveys*, Vol. 34, No. 4, pp. 469-503, Dec. 2002.
[8] Eiron, N., McCurley, K., and Tomlin, J., "Ranking the Web Frontier," In *Proc. 13th Int'l Conf. on World Wide Web (WWW)*, pp. 309 - 318, May 2004.
[9] Fagin, R., Kumar, R., and Sivakumar, D., "Comparing Top k Lists," *SIAM J. DISCRETE MATH*, Vol. 17, No. 1, pp. 134-160, 2003.
[10] Gyongyi, Z., Berkhin, P., and Garcia-Molina, H., "Web spam taxonomy," In *AIRWeb*, 2005.
[11] Gyongyi, Z., Garcia-Molina, H., and Jan, P., "Combating Web Spam with TrustRank," In *VLDB*, 2004.
[12] Google Search, http://www.google.com.
[13] Google Popular Queries Service, http://www.google.com/intl/en/press/intl-zeitgeist.html.
[14] Haveliwala, T. H., "Topic-sensitive PageRank," In *WWW*, 2002.
[15] Kamvar, S., Haveliwala, T., and Golub, G., "Adaptive Methods for the Computation of Pagerank," *Linear Algebra and its Applications*, Vol. 386, pp. 51-66, 2004.

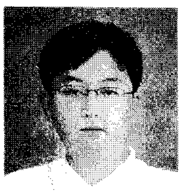
[16] Haveliwala, T. and Kamvar, S., The Second Eigenvalue of the Google Matrix, Technical Report, Dept. of Computer Science, Stanford Univ., 2003.
[17] Kamvar, S. et al., Exploiting the Block Structure of theWeb for Computing Pagerank, Technical Report, Dept. of Computer Science, Stanford Univ., 2003.
[18] Krishnan, V. and Raj, R., "Web Spam Detection With Anti-TrustRank," In *AIRWeb*, 2006.
[19] MS Live Search, http://www.live.com.
[20] Naver, http://www.naver.com.
[21] Nie, L., Wu, B., and Davison, B., Incorporating Trust into Web Search, Technical Report, Lehigh University, Dec. 2006.
[22] Page, L., et al., The PageRank Citation Ranking: Bringing Order to the Web, Technical Report SIDL-WP-1999-0120, Department of Computer Science, Stanford University, 1998.
[23] Shin, E. et al., "Implementation of a Parallel Web Crawler for the Odysseus Large-Scale Search Engine," *Journal of The Korean Institute of Information Scientist and Engineers(KIISE): Computing Practice and Letters*, Vol. 14, No. 6, pp. 567-581, Aug. 2008.
[24] Sreangsu, A., and Joydeep, G., "Outlink Estimation For PageRank Computation Under Missing Data," In *WWW*, 2004.
[25] Wang, Y. and Dewitt, D., "Computing PageRank in a Distributed Internet Search System," In *VLDB*, 2004.
[26] Whang, K. et al., "Odysseus: a High-Performance ORDBMS Tightly-Coupled with IR Features," In *ICDE*, 2005.
[27] Wikipedia, The free encyclopedia, http://www.wikipedia.org.
[28] Yahoo! Seach, http://www.yahoo.com.
[29] Yi, Z. et al., "XRank: Learning More from Web User Behaviors," In *CIT*, 2006.
[30] Yoshida, Y. et al., "What's Going on in Search Engine Rankings," In *AINAW*, 2008.

저 자 소 개

팜 민 득(정회원)
2006년 Hanoi University of Technology(HUT), Information Technology (학사)
2009년 한국과학기술원 전산학과 (석사)
2004년 MTSD, S/W 개발자
2004년~2005년 Tulap Group 개발팀장
2004년~2005년 HUT R&D Lab 인턴
2005년~2006년 HUT Library and Information Network Center 인턴
2006년~2007년 Vietnam Integrated Solutions S/W 개발자
<주관심분야 : 정보검색(IR), 텍스트 마이닝>



허 준 석(학생회원)
1995년 서울시립대학교 전산통계학과 (학사)
1997년 서울시립대학교 전산통계학과 (석사)
2009년 한국과학기술원 전산학과 (박사)
1997년~2002년 대우통신㈜ (현, 머큐리㈜) 선임연구원
2002년~2003년 한국과학기술원 첨단정보기술연구센터 연구원
2009년~현재 한국과학기술원 전산학과 연수연구원
<주관심분야 : 정보검색(IR), 주기억장치 데이터베이스, 공간데이터베이스>



이 정 훈(학생회원)
1995년 경북대학교 컴퓨터공학과 (학사)
1997년 경북대학교 컴퓨터공학과 (석사)
2003년~현재 한국과학기술원 전산학과 박사과정
1997년~2002년 한국정보통신㈜ 주임연구원
2002년~2003년 한국과학기술원 첨단정보기술연구센터 연구원
<주관심분야 : 센서네트워크, Peer-to-Peer>

황 규 영(평생회원)
1973년 서울대학교 전자공학과 (학사)
1975년 한국과학원 전기 및 전자학과 (석사)
1982년 Stanford University EE/CSL (석사)
1984년 Stanford University EE/CSL (박사)
1975년~1978년 국방과학연구소 선임연구원
1983년~1990년 IBM T.J.Waston Research Center, Research Staff Member
1992년~1994년 한국정보과학회 데이터베이스 연구회 (SIGDB) 운영위원장
2008년~현재 한국과학기술원 특훈교수
2007년 한국정보과학회 회장
2007년~현재 Fellow, IEEE
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 Trustee, The VLDB Endowment
2007년~2009년 Chair, Steering Committee, DASFAA
1990년~현재 한국과학기술원 전산학과 교수
<주관심분야 : 데이터베이스 시스템, 멀티미디어, 검색엔진, GIS>