페이지랭크 알고리즘 적용을 위한 구현 기술

김 성 진[†] · 이 상 호[††] · 방 지 환[†††]

요 약

1998년에 등장한 구글 검색 사이트(http://www.google.com)에 처음 소개된 페이지랭크 알고리즘은 웹 문서들의 연결 구조에 기반하여 문서들간의 순위를 부여하는 방법이다. 페이지랭크 알고리즘은 상용 검색 엔진에서 구현되어 사용되고 있으나, 상업상의 이유들로 인하여 구현 기법에 관한 연구 결과는 거의 발표되지 않고 있다. [4, 8]에서 소개된 페이지랭크 알고리즘의 구현 기법은 웹 문서들의 페이지랭크 값을 산출하기에 충분하지 않다. 본 논문은 페이지랭크 알고리즘의 구현 기법[4, 8]을 설명하고, 이를 적용하는데 필요한 입/출력 자료 구조 및 4가지 주요 구현 기술을 제시한다. 본 논문은 실제 웹 문서의 페이지랭크 값을 산출하는 시스템을 예로 들어 페이지랭크 알고리즘을 적용하는 방법에 대한 이해를 돕도록 하였다.

Implementation Techniques to Apply the PageRank Algorithm

Sung Jin Kim[†] · Sang Ho Lee[††] · Ji Hwan Bang[†††]

ABSTRACT

The Google search site (http://www.google.com), which was introduced in 1998, implemented the PageRank algorithm for the first time. PageRank is a ranking method based on the link structure of the Web pages. Even though PageRank has been implemented and being used in various commercial search engines, implementation details did not get documented well, primarily due to business reasons. Implementation techniques introduced in [4, 8] are not sufficient to produce PageRank values of Web pages. This paper explains the techniques[4, 8], and suggests major data structure and four implementation techniques in order to apply the PageRank algorithm. The paper helps understand the methods of applying PageRank algorithm by means of showing a real system that produces PageRank values of Web pages.

키워드 : 정보검색(information retrieval), 랭킹 알고리즘(ranking algorithm), 페이지랭크(PageRank)

1. 서 론

1998년에 등장한 구글 검색 사이트는 효과적인 검색 성능으로 많은 사람들에게 주목을 받기 시작하였다. 특히 구글 검색 시스템은 페이지랭크[1, 4, 6, 8]라 불리는 랭킹 알고리즘을 사용하여 각 문서들에 대한 순위를 부여함으로써 사용자들에게 만족할 만한 성능을 제공해 주었다[1]. 페이지랭크 알고리즘은 각 문서들간의 연결 구조에 기반한 랭킹 방법으로서, 많은 문서로부터 참조되거나 중요한 문서로부터 참조되는 문서가 높은 페이지랭크 값을 가진다.

페이지랭크 알고리즘은 효과적인 랭킹방법으로서 많은 웹 검색 시스템들은 페이지랭크 알고리즘을 자신의 시스템에 적용하길 원한다. 그러나, 그 구현 기법에 관한 연구는 상업상의 여러 이유들로 인하여 현재 거의 발표되지 않고 있다[1]. 본 논문은 웹 검색 시스템이 페이지랭크 알고리즘을 적용하는데 필요한 구현 지침을 제공하는 것을 목적으로 한다. 본 논문은 페이지랭크 알고리즘에 대한 이해를 돕고, 알고리즘 적용에 선결되어야할 주요 문제점들과 해결책을 제시한다.

웹 문서에 대한 페이지랭크 값 산출은 크게 두 단계로 구성된다. 첫 번째 단계는 웹 문서들을 분석하여 페이지랭크 계산에 알맞은 자료 구조로 문서들의 연결 정보를 표현한다. 두 번째 단계는 첫 번째 단계로부터 생성된 연결 정보를 이용하여 페이지랭크 값을 계산한다. [4, 8]은 두 번째 단계에 대하여 자세히 설명하고 있으나 페이지랭크 알고리즘을 적용하기 위하여 필요로 하는 전처리 과정에 대한 언급이 없다. 실제로 두 번째 단계의 수행은 짧은 시간내에 이루어진다[1, 6]. 반면, 첫 번째 단계는 페이지랭크 값의 산출을 위하여 대부분의 시간이 소요될 뿐 아니라, 최종 페이지랭크 값에 많은 영향을

※ 본 연구는 한국과학재단 목적기초연구(R-01-2000-00403)와 숭실대학교 교내학술연구비 지원으로 수행되었음.
† 준 회 원 : 숭실대학교 대학원 컴퓨터학과
†† 정 회 원 : 숭실대학교 컴퓨터학부 교수
††† 정 회 원 : (주) 소프트텔레웨어 연구원
논문접수 : 2001년 12월 11일, 심사완료 : 2002년 8월 19일

미친다. 따라서, 본 논문은 첫 번째 단계의 구현에 필요한 기술들을 주로 논한다.

본 논문은 페이지랭크 알고리즘을 이용하여 모든 웹 문서에 대한 페이지랭크 값을 산출하는 LAS(Link Analysis System, 연결 분석 시스템)를 소개한다. 본 논문은 LAS의 구체적인 입/출력 자료 구조를 보인다. LAS는 각 구성요소의 이해가 쉽고 분리가 용이하도록 파일 기반의 인터페이스를 갖는다. 따라서, 개발자는 페이지랭크 알고리즘을 적용하는 시스템의 설계와 구현에 LAS의 전체(혹은 일부)를 중요한 참고 자료로 삼을 수 있다.

본 논문의 구성은 다음과 같다. 제 2장은 페이지랭크 알고리즘을 소개하고, 이를 이용하여 효율적으로 각 문서의 페이지랭크 값을 산출하는 방법에 관하여 다룬다. 제 3장은 페이지랭크 알고리즘의 적용을 위해 필요한 4가지 주요 기술들에 관하여 논한다. 제 4장은 페이지랭크 알고리즘을 적용하는 실제 시스템의 예를 보인다. 제 5장은 향후 연구분야를 밝히고 끝맺음을 한다.

2. 페이지랭크 알고리즘

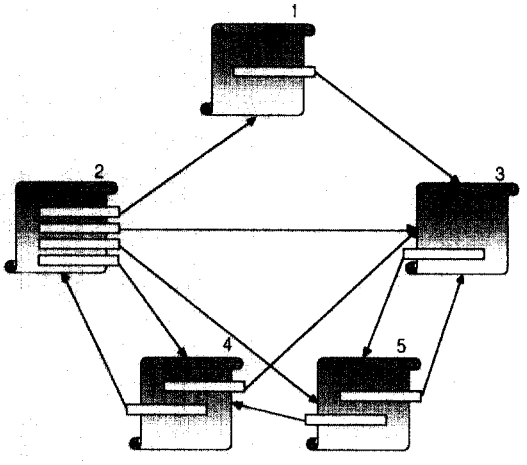
각 문서들의 상대적인 중요도는 매우 주관적이다. 구글 시스템[1, 3]은 이러한 상대적 중요도를 객관화하기 위하여 페이지랭크 알고리즘을 사용하였다. 페이지랭크 알고리즘은 웹 문서들의 참조관계에 기반하여 문서들의 중요도를 산출한다. 웹에서의 하이퍼링크(hyperlink)는 참조하는 문서의 저자가 참조되는 문서를 추천하는 것으로 간주된다. 일반적으로 중요 문서들은 많은 웹 문서들로부터 참조되고, 중요 문서들로부터 참조되는 문서는 중요 문서일 가능성이 높다.

한 웹 문서 v가 있을 때, F_v는 문서 v가 가리키는 문서들의 집합이고, B_v는 문서 v를 가리키는 문서들의 집합이라고 하자. N_v는 문서 v에서 가리키는 문서들의 개수를 나타내고, $N_v = |F_v|$ 일 때, 문서 v에 대한 페이지랭크 값 $PR(v)$는 식 (1)과 같이 정의된다.

$$PR(v) = \sum_{u \in B_v} \frac{PR(u)}{N_u} \quad (1)$$

웹 상의 모든 문서와 하이퍼링크는 노드(node)와 에지(edge)를 가지는 방향성 그래프로 나타날 수 있다[7]. 페이지랭크 알고리즘은 웹 문서들간의 상호 연결 관계를 하나의 거대한 그래프로 간주하고 각 문서들의 페이지랭크 값을 산출한다. 웹 상에 존재하는 모든 페이지의 개수가 N개라면, M은 웹 그래프 G에 대한 $N \times N$ 정방 행렬이다. 만약 페이지 j로부터 페이지 i로의 하이퍼링크가 존재한다면 행렬 M의 원소 m_{ij}는 $(1/N_j)$의 값을 가진다. 그리고 나머지 원소들의 값은 0이 된다. 모든 문서들에 대한 페이지랭크 값은 $Rank$ 벡터(즉, $N \times 1$ 행렬)로서 나타난다. $rank(i)$는 $Rank$ 벡터의 i번째 값을 나타내고 또한 문서 i의 페이지 랭크 값을 나타낸다.

(그림 1)과 같은 작은 웹을 가정하자. 예제 웹에는 총 5개의 문서가 존재한다. 각 문서간의 연결관계는 직선으로 나타나 있다. (그림 2)의 정방 행렬 M은 (그림 1)의 문서에 대한 참조관계 그래프 G를 나타낸 것이다. 문서 5는 문서 3과 4의 두 개 문서를 가리키므로 N_5는 2이다. 행렬의 원소 m_{35}와 m_{45}는 $(1/2)$으로 나타나고, 나머지 원소 m_{15}, m_{25}, m_{55}는 0으로 나타난다. (그림 2)의 $Rank$ 벡터는 각 문서들의 페이지랭크 값을 나타낸다. 문서 5는 문서 2와 3으로부터 참조된다. 따라서, 문서 5의 페이지랭크 값은 문서 2와 3의 페이지랭크 값으로부터 산출된다. 식 (1)에 따르면, 문서 5의 페이지랭크 값은 $(PR(2)/4)$와 $(PR(3)/1)$의 합이 된다. 즉, 식 (1)에 의하여 모든 문서에 대한 페이지랭크 값은 구하는 것과 행렬 M과 $Rank$ 벡터의 곱은 동일한 연산이다.



(그림 1) 예제 웹

	M						Rank		Rank
	1	2	3	4	5				
1	0	1/4	0	0	0		0.025		0.025
2	0	0	0	1/2	0		0.100		0.100
3	1	1/4	0	1/2	1/2	=	0.325	=	0.325
4	0	1/4	0	0	1/2		0.200		0.200
5	0	1/4	1	0	0		0.350		0.350

(그림 2) 예제 웹의 행렬 M과 페이지랭크 값

Rank 벡터는 행렬 M을 곱한 전후의 값이 같다. 따라서, *Rank* 벡터는 행렬 M의 고유 벡터(Eigen vector)[2]이다. 크기가 크지 않은 행렬 M이 주어졌을 때 그 행렬에 대한 고유 벡터를 구하는 방법은 잘 알려져 있다[2]. 구글은 고유 벡터 산출을 위하여 거듭제곱법[2]을 사용한다. 거듭제곱법을 이용한 페이지랭크 값 산출 방법은 다음과 같다. 우선 모든 원소의 합이 *1*인 임의의 벡터를 행렬 M에 곱하고 그 결과로 나온 벡터를 거듭하여 행렬 M에 곱한다. 행렬-벡터 곱셈의 결과는 어느 시점에 이르러 더 이상 변하지 않고 이때의 결과 벡터를 최종적인 *Rank* 벡터로 인식한다.

메인 메모리 내에서의 빠른 행렬-벡터 곱셈 연구는 과학 기술 분야에서 많은 연구가 되어 왔다[5, 9]. 그러나, 페이지랭크 알고리즘에 사용되는 행렬 M은 메모리에 저장되기에 너무 크다. 따라서, 페이지랭크 값 산출은 디스크 입/출력 중심의 행렬-벡터 곱셈 알고리즘을 필요로 한다. 근래에 하드웨어 사양이 좋아지고 가격이 급속도로 하락함에 따라 대용량의 메모리를 탑재한 시스템이 늘어났다. 대부분의 시스템은 행렬-벡터 곱셈 연산 중 *Rank* 벡터를 메모리에 적재 가능하므로, 본 논문은 *Rank* 벡터를 메모리에 적재하고 행렬 M을 디스크에 기반하는 구현 방법만을 소개한다. *Rank* 벡터가 메모리에 적재되지 못하는 경우는 *Rank* 벡터를 일정한 단위로 나누어 계산하는 방법이 있으나 이에 대한 자세한 알고리즘은 [4]를 참조하기 바란다.

일반적으로 행렬 M은 희소 행렬이다. 따라서, 원소 값이 *0*인 원소는 디스크에 저장하지 않음으로써 디스크의 저장 공간을 줄일 수 있다. (그림 3)은 행렬 M이 디스크에 저장된 구조를 나타낸다. 즉, (그림 3)은 행렬 M의 원소 중 *0*이 아닌 원소의 값만을 저장하고 있다.

원본 노드	링크 수	목적 노드
1	1	3
2	4	1, 3, 4, 5
3	1	5
4	2	2, 3
5	2	3, 4

(그림 3) 행렬 M을 위한 효율적 저장구조

(그림 3)의 구조로 행렬 M을 저장한 파일을 'Links'라고 할 때, (그림 4) [4]는 'Links'를 이용하여 효율적으로 행렬 M과 *Rank* 벡터를 거듭제곱하는 알고리즘이다. 우선 두 개의 소수형 벡터 *Source*와 *Dest*를 생성한다. *Source*의 모든 원소는 *(1/N)*로 초기화된다. *Dest*는 행렬 M과 *Source*의 곱셈 결과를 저장하기 위하여 사용된다. *residual*은 *Source*와 *Dest*의 차이를 나타내는 변수이다. 행렬 M과 *Source*를 곱할수록 *residual*은 작아진다. 타우(τ)는 0부터 1사이의 상수이다. (그림 4)는 *residual*이 임계값 타우보다 작은 값을 가질 때 *Source*와 *Dest*가 같은 벡터라고 가정하고, 이때의 *Source*를 *Rank* 벡터로 간주한다. (그림 4)의 두 번째 while 문은 행렬 M과 *Source*의 곱셈을 처리한다. 'Links' 파일로부터 읽은 원본 노드, 링크 수, 목적 노드들은 *source*, *n*, $dest_n$에 저장된다. 행렬 M에서 *source* 번째 열은 *0*이 아닌 원소가 *n*개 있다. $dest_n$은 *0*이 아닌 원소들의 위치를 나타낸다. for 문은 행렬 M의 한 열에 대한 곱셈을 처리한다. 행렬 M의 모든 열에 대한 곱셈이 수행되면(즉, 두 번째 while 문에 대한 수행이 끝나면), *residual*이 계산되고, *Dest*는 *Source*로 치환된다.

```
∀s, Source[s] = 1/N          // Source 벡터 초기화
residual = 1                 // residual 변수 초기화
while (residual > τ) {
    ∀d, Dest[d] = 0          // Dest 벡터 초기화

    // 'eof()'는 파일의 끝일 경우 '참'을 반환한다.
    while ( not Links.eof () ) {

        // source : 원본 노드, n : 링크 수, destn : 목적노드
        Links.read ( source, n, dest1, dest2, ···, destn )

        // 원본 노드의 현재 페이지랭크 값을 목적 노드에 분배한다.
        for j = 1 ··· n
            Dest[ destj ] = Dest[ destj ] + Source[ source ] / n
    }

    // '|| 벡터 ||'는 벡터 내 모든 원소들의 합이다.
    residual = || Source − Dest ||
    Source = Dest
}
```

(그림 4) 페이지랭크 값 계산 알고리즘

3. 주요 구현 기술

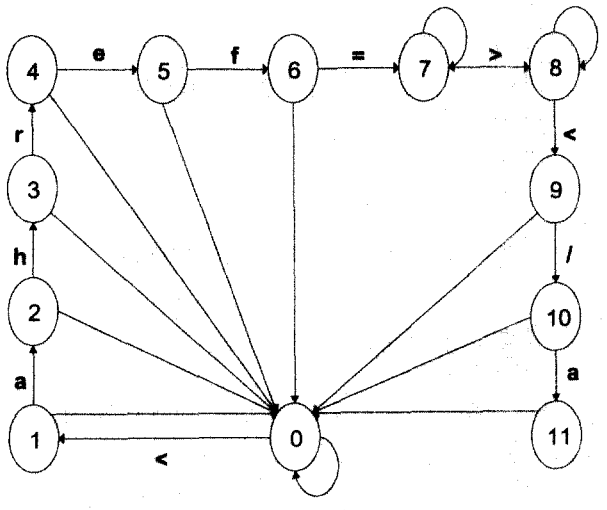
본 장은 페이지랭크 알고리즘을 적용하여 웹 문서들의 페이지랭크 값 산출을 위하여 요구되는 기술을 소개한다. 본 장은 페이지랭크 알고리즘을 적용하는 시스템에서 반드시 고려되어야 하는 사항과 그에 따른 해결 방법을 제시한다.

3.1 연결 URL 추출

페이지랭크 알고리즘은 웹 문서들간의 참조 관계를 이용하여 문서들의 중요도를 산출한다. 따라서, 모든 웹 문서들에 대해 자신이 참조하는 문서들을 파악하는 작업은 페이지랭크 값 산출을 위한 필수 작업이다. 웹 문서는 자신이 참조하는 웹 문서에 대한 URL을 문서의 내용중에 포함하고 있다.

HTML로 작성된 웹 문서의 경우에 URL들은 태그 내에 나타나거나 태그 밖에서 나타난다. 태그 안에 나타난 URL은 실제로 다른 문서를 참조하는데 사용되므로 연결 URL이라 하고, 태그 밖에서 나타난 URL은 웹 브라우저 상에서 보이기만 할 뿐 실제로 다른 참조하지 않으므로 비-연결 URL이라고 한다.

연결 URL은 일반적으로 "<a href>" 태그 내에 존재한다. 시스템은 유한상태기계(Finite State Machine)를 사용하여 "<a href>" 태그로부터 연결 URL을 추출할 수 있다. 문서에서 한 문자가 읽히면, 읽힌 문자는 유한상태기계로 보내지고 그에 대한 처리가 결정된다. 유한상태기계는 자신이 가지는 상태에 근거하여 입력받은 문자가 웹 문서 내용의 일부인지 연결 URL의 일부인지를 판단할 수 있다. 한번 처리 결정된 문자는 다시 사용되지 않으므로, 유한상태기계는 한 문서에 대한 읽기 수를 최소화한다. 즉, 유한상태기계의 사용은 시스템이 단 한번의 문서 읽기로써 문서내의 연결 URL 추출을 가능하도록 한다. 웹 상에는 수많은 문서들이 존재하므로 한 문서에 대한 읽기 수를 최소화하는 것은 시스템의 성능을 크게 개선시킨다. 또한, 유한상태기계는 HTML로 작성된 문서의 문법확인 작업을 유연하게 처리할 수 있다. 예를 들어, 많은 공백문자를 포함한 태그의 인식이나 구문상의 오류 인식을 쉽게 한다.



(그림 5) "<a href>" 태그를 위한 유한 상태도

"<a href>" 태그에서 연결 URL을 추출하기 위한 유한상태기계는 0부터 11까지의 총 12개 상태를 가진다. 본 유한상태기계는 하나의 문자를 입력받고, 상태는 입력받은 문자에 따라 전이된다. (그림 5)는 유한상태기계의 상태도를 나타낸다. 원은 하나의 상태를 나타내고, 원안의 숫자는 상태 번호를 나타낸다. 직선은 하나의 상태에서 전이 가능한 상태를 가리킨다. 직선에 나타난 문자들은 상태들간의 전이가 발생하는 조건을 나타낸 것이다. 만약, 하나의 상태에서 조건에 맞는 문자가 입력되면 '문자 있는 직선'을 따라 상태가 전이하지만, 그 외의 문자를 입력받으면 '문자 없는 직선'을 따라 상태가 전이된다. 예를 들어, 상태 5에 있는 유한 상태 기계가 'f'를 입력받으면, 상태 6으로 전이되고, 그 외의 문자가 입력되면 상태 0으로 전이된다. 본 유한상태기계를 사용하는 시스템은 웹 문서에서 한 글자씩 읽어 유한상태기계로 입력한다. 유한상태기계의 상태가 7에서 8로 전이되면, 시스템은 상태가 7인 동안에 읽었던 문자들을 모아서 하나의 문자열을 만든다. 하나의 문서를 읽는 동안 생성된 문자열들은 각각 연결 URL이 된다.

문서간의 연결은 "<a href>"이외의 태그에서 발생할 수도 있다. 다양한 태그로부터의 연결 URL 추출을 가능하도록 하기 위해, 하나 이상의 유한상태기계가 사용될 수 있다. 예를 들어, "<frame>" 태그나 "<form>" 태그로부터 연결 URL을 추출하는 유한상태기계가 추가될 수 있다.

현재 웹 상에 존재하는 많은 문서들은 스크립트 언어를 포함한다. 또한, 스크립트 언어를 이용한 문서 연결이 빈번하게 발생한다. 스크립트 언어의 사용은 보다 다양한 태그로부터의 문서 연결을 가능하도록 한다. (그림 6)은 스크립트 언어를 포함한 실제 웹 문서의 예를 보인 것이다. (그림 6)에서 문서 연결은 하나의 태그에서 이루어지지 않고 두 개의 태그를 통해서 이루어진다. 즉, 연결 URL "http://orion.ssu.ac.kr/"은 스크립트 태그에서 정의되고, 실제 문서의 연결은 이미지 태그에서 이루어진다. 연결 URL이 정의된 태그와 실제 적용된 태그의 관계를 찾는 일은 많은 추가 비용을 초래한다. 웹 문서들의 페이지랭크값 산출은 문서들간의 연결 정보만이 사용되므로 연결 URL이 정의된 태그와 실제 적용된 태그의 관계

```
<HTML>
<HEAD>
        <TITLE> Load URL </TITLE>
        <SCRIPT LANGUAGE = "JavaScript"> <!--
        function loadPage() {
                location.href = "http://orion.ssu.ac.kr/"
        }
        // --> </SCRIPT>
</HEAD>
<BODY>
                                ...
        <IMG SRC = "./images/database.jpg" ONCLICK
                        = loadPage()>
                                ...
</BODY>
</HTML>
```

(그림 6) 스크립트를 포함한 HTML 문서 예

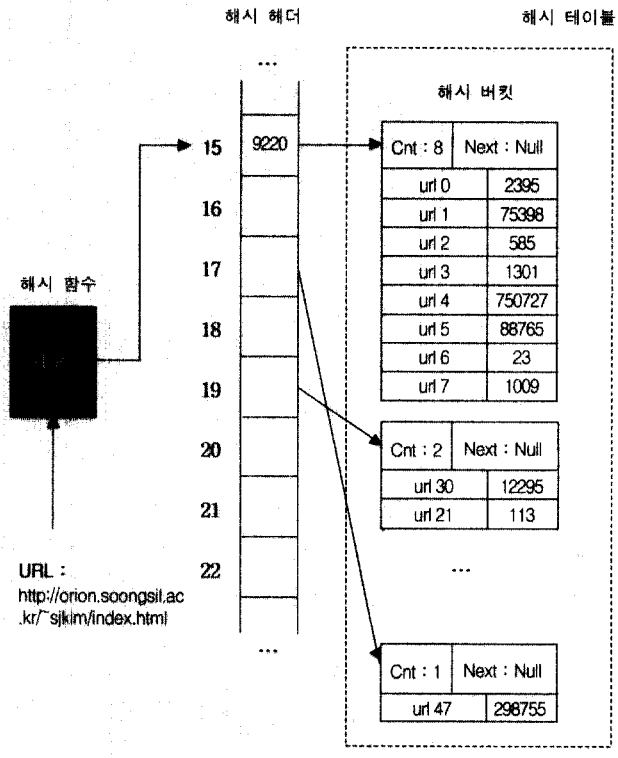
를 찾는 일은 무의미하다. 시스템은 스크립트 태그로부터 정의된 연결 URL을 추출하고 실제 연결이 이루어지는 태그는 고려하지 않음으로써 유한상태 기계를 사용할 수 있다.

3.2 URL / 문서번호 변환

웹 문서들은 URL과 시스템 내에서의 고유 문서번호를 가진다. 문서번호는 시스템이 보다 적은 저장 공간으로써 웹 문서들을 구분할 수 있도록 하고, 문서들간의 연결 정보 관리를 용이하게 한다. 웹 문서에서 추출된 연결 URL들은 문서번호로 변환되고, 문서들 간의 연결 관계는 (그림 3)과 같은 구조로 파일에 저장된다. URL/문서번호 변환을 위하여 모든 URL에 대한 문서번호 정보를 메모리에 저장하는 것은 디스크 입/출력을 피하는 최선의 방법이다. 하지만, 웹에는 메모리의 저장 한계를 넘는 URL/문서번호 쌍이 존재하므로 URL에 대한 문서번호는 디스크 상에 관리되어야 한다. URL/문서번호 변환은 매우 빈번하게 이루어지는 작업으로서 매 변환 작업의 수행마다 URL/문서번호 목록 전체를 참조하는 것은 상당한 디스크 입/출력 오버헤드(overhead)를 초래한다. 결국, 최소의 디스크 입/출력으로써 연결 URL에 대한 문서번호를 반환할 수 있는 디스크 기반의 자료 구조가 필요하다.

해시는 O(1) 비용의 파일 접근으로써 원하는 자료를 찾도록 하고, 자료에 대해 항상 일정한 접근 시간을 기대할 수 있다[10]. 디스크에 기반한 효율적인 URL/문서번호 변환을 위하여, 본 논문은 해시 함수, 해시 헤더, 해시 테이블로 구성된 해시 색인 구조를 제안한다. (그림 7)은 해시 색인 구조를 나타낸다. 해시 테이블은 모든 문서에 대한 URL과 그에 따른 문서번호의 쌍을 여러 개의 해시 버킷으로 구성한다. 하나의 해시 버킷은 같은 해시 키 값을 가지는 URL과 그에 따른 문서번호, 해시 버킷이 저장하고 있는 URL 개수를 포함한다. 같은 해시 키를 가지는 URL의 개수가 버킷이 허용하는 개수를 초과하면, 새로운 해시 버킷이 할당된다. 기존 버킷에 저장되지 못한 URL은 새로 추가된 버킷에 저장되고, 기존 버킷은 새로 추가된 버킷의 위치 정보를 저장한다. 해시 헤더는 해시 키 값에 따른 버킷의 위치 정보를 저장한다. 마지막으로, 해시 함수는 URL을 입력받아서 그에 따른 해시 키를 생성한다. (그림 7)의 각 해시 버킷은 추가로 할당된 버킷이 없음을 의미한다.

해시 색인 사용하여 연결 URL에 대한 문서번호를 검색하는 예는 (그림 7)에 나타난다. 연결 URL이 "http://orion.soongsil.ac.kr/~sjkim/index.html"일 때, 해시 함수에 의해 얻어진 해시 키 값은 15이다. 해시 헤더는 해시 키 15에 대한 해시 버킷의 위치가 9220임을 나타낸다. (그림 7)에서 해시 버킷은 8개의 URL과 문서번호 쌍에 대한 정보를 가지고 있다. 또한



(그림 7) 해시 색인 예

추가된 해시 버킷을 가리키는 필드가 'Null'이므로 해시 키 15를 가지는 URL은 더 이상 없다. 버킷 내의 "url4"가 "http://orion.soongsil.ac.kr/~sjkim/index.html"라면, 입력받은 URL에 대한 문서번호는 750727이다.

3.3 상대/절대 URL 변환

문서에서 추출된 연결 URL은 해시 색인을 참조하여 문서번호로 변환된다. 해시 색인에 나타난 URL은 절대 URL로 표현되므로, 상대 URL로 표현된 연결 URL은 절대 URL로 변환되어야 한다. 웹 문서들은 같은 사이트내의 문서를 상대 URL로 참조하는 경우가 많으므로 상대/절대 URL 변환 작업은 빈번하게 발생한다. 본 논문은 상대/절대 URL 변환을 위하여 URL을 프로토콜, 도메인, 디렉토리, 파일의 네 가지 요소로 구분한다. 연결 URL은 네 가지 구성 요소의 결합 형태에 근거하여 <표 1>과 같이 분류된다.

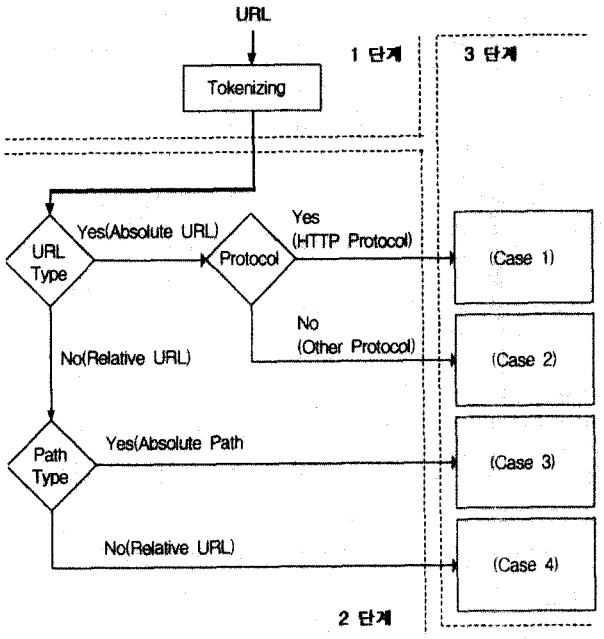
형식 1, 3, 5의 URL은 특정 파일을 명시하지 않는다. 따라서, URL이 가리키는 실제 문서는 연결되는 사이트에 따라 다르다. 형식 1의 URL은 일반적으로 'index.html' 또는 'index.htm'의 파일을 가리키게된다. 형식 1, 2, 3, 4는 절대 URL이고, 형식 5, 6, 7은 상대 URL이다. 절대 URL과 상대 URL의 판단은 프로토콜 포함 여부에 기준한다. 프로토콜을 포함한 URL은 절대 URL이고, 프로토콜을 포함하지 않은 URL은 상대 URL이다. 상대 URL은 도메인을 명시하지 않는다.

〈표 1〉 연결 URL의 구성

형식	구 성 방 식	예
1	프로토콜, 도메인	http://orion.soongsil.ac.kr/
2	프로토콜, 도메인, 파일	http://orion.soongsil.ac.kr/index.html
3	프로토콜, 도메인, 디렉토리	http://orion.soongsil.ac.kr/~sjkim/
4	프로토콜, 도메인, 디렉토리, 파일	http://orion.soongsil.ac.kr/~sjkim/index.html
5	디렉토리	../sam/html/
6	디렉토리, 파일	/sam/sam.html
7	파일	sam.html

상대/절대 URL 변환 과정은 (그림 8)에 나타나 있다. (그림 8)의 변환 과정은 크게 다음의 세 단계로 나누어 볼 수 있다.

- 1 단계 : 연결 URL을 토큰(token)한다.
- 2 단계 : 연결 URL의 형식을 파악한다.
- 3 단계 : 절대 URL을 생성한다.



(그림 8) 상대/절대 URL 변환 과정

1 단계는 슬래시('/')와 EOS(End of String) 문자를 기준으로 하여 URL을 보다 작은 의미 단위로 나눈다. 본 논문에서 이 의미 단위는 토큰이라고 명명된다. 즉, 하나의 URL 문자열이 여러 개의 토큰 문자열로 나뉘어진다. 토큰 문자열의 마지막 문자는 슬래시 또는 EOS이다.

2 단계는 토큰들로부터 연결 URL의 형식을 파악한다. (그림 8)의 'URL Type' 비교는 연결 URL이 절대 URL인지 상대 URL인지를 파악한다. 만약 두 번째 토큰이 '/'이면 연결 URL은 절대 URL이다. 'Protocol' 비교는 절대 URL의 프로토콜을 검사한다. 'HTTP' 프로토콜만을 고려한다면, 첫 번째 토큰이 "http:/" 인지 여부를 검사한다. (그림 8)의 'Path Type' 비교는 상대 URL의 디렉토리 경로가 절대 경로인지 상대 경로인지 파악한다. 만약 첫 번째 토큰이 '/'이면 절대 경로이고, 그 외는 상대 경로이다.

3 단계는 연결 URL의 형식에 따라 절대 URL을 생성한다. (그림 8)에서 'case 1'과 'case 2'는 연결 URL이 절대 URL이므로 해당 URL을 변환할 필요가 없다. 연결 URL을 포함한 문서의 URL을 부모 URL이라고 하면, 상대 URL은 부모 URL에 따라 다르게 절대 URL을 생성한다. 'case 3'은 연결 URL이 상대 URL이고 절대 경로인 경우이다. 연결 URL의 프로토콜과 도메인은 부모 URL의 프로토콜과 도메인 값과 같다. 따라서, 생성된 절대 URL은 프로토콜과 도메인의 부모 URL의 값을 사용하고, 디렉토리와 파일은 자신의 값을 사용한다. 'case 4'는 연결 URL이 상대 URL이고 상대 경로인 경우이다. 절대 URL 생성을 위하여 프로토콜과 도메인은 부모 URL의 값을 사용하고 파일은 연결 URL 자신의 값을 사용한다. 디렉토리는 부모 URL의 디렉토리에 연결 URL의 디렉토리가 연결되어 생성된다.

3.4 대/소문자 구분

대부분의 URL은 알파벳으로 구성되어 있다. 한 알파벳 문자는 대문자와 소문자로 표현 가능하기 때문에, 연결 URL은 모두 대문자로 표현되거나 소문자로 표현될 수 있고, 대/소문자들의 조합으로 표현될 수도 있다. 하나의 연결 URL이 문서번호로 변환되기 위해서는 해시 색인에서 해당 연결 URL에 대한 문서번호를 저장하고 있어야 한다. 하지만 하나의 연결 URL은 대/소문자의 조합에 따라 수많은 종류의 문자열로서 표현 가능하기 때문에, 해시 테이블에 모든 조합의 문자열에 대한 문서번호를 저장하는 것은 사실상 불가능하다.

아래 네 가지 경우의 URL들을 고려해 보자.

- http://www.kdlp.co.kr/sam/index.htm
- http://www.kDLP.co.kR/sam/index.htm

- http://www.kdlp.co.kr/Sam/index.htm
- http://www.kdlp.co.kr/sam/Index.htm

위의 URL들은 대/소문자를 구분하지 않으면 모두 같다. 첫 번째 URL은 모두 소문자로 이루어진 URL이다. 두 번째 URL은 도메인 이름에 대문자를 포함하고 있다. 세 번째 URL은 디렉토리 이름에 대문자를 포함하고 있다. 네 번째 URL은 파일이름에 대문자를 포함하고 있다. 실제적으로 도메인 이름은 대/소문자를 구분하지 않으므로 두 번째 URL과 첫 번째 URL은 같다. 디렉토리 및 파일 부분은 웹을 서비스하는 기계(Machine)의 환경에 따라 대/소문자를 구분이 다르므로, 세 번째 URL과 네 번째 URL은 경우에 따라 첫 번째 URL과 다를 수 있다.

대/소문자 구분을 고려한 두 URL의 비교에는 다음과 같은 세 가지 방법이 있다. 첫째, '정확 매치'는 대/소문자를 엄밀히 구분하여 두 개의 URL을 비교하는 것이다. 둘째, '실제 매치'는 실제 웹 상에서 같은 URL로서 인식될 수 있는지의 여부를 가지고 비교하는 것이다. 셋째, '의미 매치'는 대/소문자를 구분하지 않고 두 개의 URL을 비교하는 것이다.

'정확 매치'는 상대적으로 가장 빠른 처리가 가능하고 쉽게 구현 가능한 방법이다. URL 문자열에 대한 추가적인 처리가 필요치 않고 연결 URL과 해시 요약 파일 내의 URL을 단순히 비교하는 과정만이 필요하기 때문이다. 그러나, 이 방법은 실제적으로 같은 URL임에도 불구하고 같지 않은 URL이라고 판명되는 경우가 빈번히 발생하므로 효과적이지 못하다. 즉, 연결 URL에 대한 문서번호가 해시 요약 파일 내에 있음에도 불구하고 없는 URL로 판단한다.

'실제 매치'를 사용하여 연결 URL과 해시 요약 파일내의 URL을 비교하기 위해서는 매 연결 URL마다 네트웍 접속을 통한 확인이 필요하다. 이는 심각한 성능 문제를 초래할 수 있다. 대안으로서 빈번하게 참조하는 기계에 대한 정보를 사전에 포함하고 있을 수도 있으나, 이러한 방법도 부가적인 정보의 처리가 필요하고 그러한 정보를 저장해야 하는 부담이 발생한다. 따라서, '실제 매치'는 구현상의 복잡성을 초래시킬 수 있다.

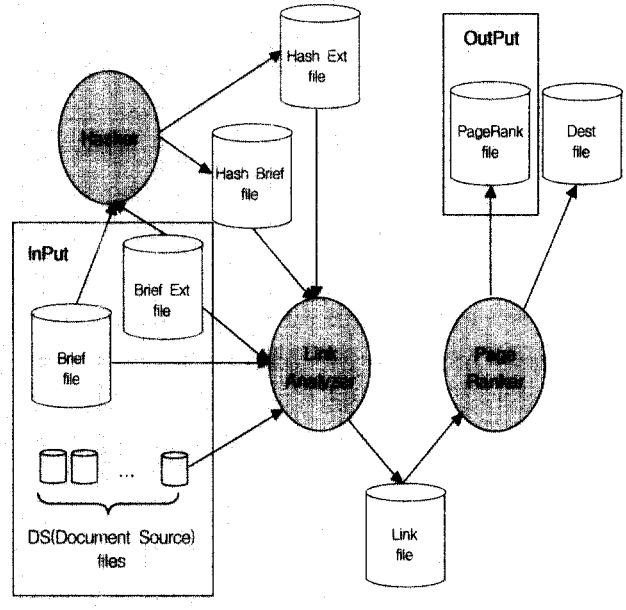
본 논문은 '의미 매치'의 방법을 사용하여 연결 URL과 해시 요약 파일내의 URL을 비교하는 방법을 권한다. 대/소문자 구분만 틀린 두 URL이 실제로 서로 다른 URL인 경우는 매우 드물다. 따라서, '의미 매치'는 연결 URL과 해시 요약 파일내의 URL이 대/소문자 구분만 틀린 경우에 연결 URL에 문서 작성자의 오류가 포함된 것으로 간주한다. '의미 매치'는 해당 웹 문서의 작성자가 참조하려는 문서에 대한 의도를 감안하여 연결 URL과 해시 색인에 존재하는 URL을 대/소문자 구분하지 않고 비교한다.

4. 페이지랭크 알고리즘 적용 시스템

본 장은 페이지랭크 알고리즘을 적용하여 웹 문서에 대한 페이지랭크 값을 산출하는 LAS 시스템을 소개한다. 본 장은 LAS의 개략적인 구성 요소, 자료 구조, 작업 흐름을 소개하여 웹 문서에 대한 페이지랭크 값을 산출하는 새로운 시스템을 개발하거나 기존의 시스템에 페이지랭크 알고리즘을 적용하고자 할 때 참고가 되도록 한다.

LAS는 방대한 자료를 처리하므로, 입/출력에 관계된 모든 자료를 디스크 상의 파일에 기반하여 처리한다. LAS는 네트웍을 통하여 실제 웹에 접근하지 않고, 파일로 웹 문서들을 입력받는다. 파일 기반 전략은 LAS의 확장성과 이식성을 높인다. 즉, 시스템의 구조 변경이나 추가적인 자원 없이 처리할 문서 수를 증가시킬 수 있고, 인터페이스를 간단하게 하여 다른 시스템에서 쉽게 LAS를 사용할 수 있다. LAS의 성능은 디스크 입/출력의 성능에 좌우되므로, 각 파일들은 효율적인 디스크 입/출력이 가능한 구조를 가져야 한다. LAS는 HTML로 작성된 문서들이 사용자의 작성 오류를 포함하는 경우가 많은 것을 감안하여 페이지랭크 값을 산출한다.

LAS의 기본적인 구성요소는 (그림 9)와 같다. 원은 작업 프로세스를 나타내고 디스크모양의 도형은 파일 하나를 나타낸다. 직선은 프로세스와 파일간의 입/출력 관계를 나타낸다. 모든 프로세스는 파일 형태의 자료를 입력받아 처리한 후 파일형태의 결과를 출력한다. 목적(Dest) 파일은 임시 파일이므



(그림 9) LAS의 구성 요소

로 직선이 점선으로 표시되었다. (그림 9)에서 페이지랭크 값을 산출하는 과정은 크게 연결 파일 작성 단계와 페이지랭크 값 계산 단계로 구분할 수 있다.

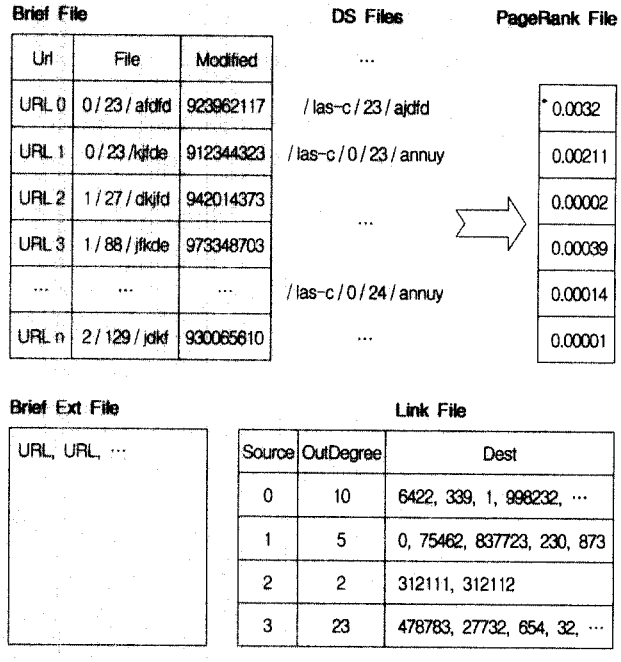
해쉬어(Hasher)는 요약 파일과 요약 확장 파일로부터 적당한 해시 함수를 가지고 해시헤더와 해시 테이블을 구성한다. 해시 요약(Hash Brief) 파일은 해쉬어에 의해 생성된 해시 헤더와 해시 테이블을 저장한다. 해시 색인을 이용한 검색 과정은 3장에서 설명되었다. 해시 확장(Hash Ext) 파일은 해시 요약 파일의 허용 길이를 초과하는 URL들을 저장한 파일이다. 긴 URL은 해시 확장 파일에 저장되고, 해시 요약 파일에는 URL 대신 해시 확장 파일에서 실제 URL 존재하는 위치 정보가 기록된다.

연결 분석기(Link Analyzer)는 제 3장에서 제시된 기술들을 사용하여 각 문서들간의 연결관계를 분석하고 연결 파일을 생성한다. 페이지랭커(PageRanker)는 연결 파일로부터 각 문서들에 대한 중요도(페이지랭크 값)를 산출한다. 페이지랭커는 제 2장에서 소개된 (그림 4)의 알고리즘을 사용하여 페이지랭크 값을 계산한다. 페이지랭크 파일은 각 문서에 대한 페이지랭크 값을 저장하고 있다. 목적(Dest) 파일은 임시 파일로서 페이지랭크 파일과 동일한 값을 갖는다. 페이지랭크(PageRank) 파일과 목적 파일은 (그림 4)의 *Source*와 *Dest*에 해당한다.

하나의 웹 문서는 디스크에 하나의 파일로 존재한다. 디스크에 저장된 하나의 파일을 문서 원본(DS, Document Source) 파일이라 한다. 문서 원본 파일에 대한 여러 가지 요약 정보는 요약(Brief) 파일에 기록된다. 예를 들어, 요약 파일은 각 문서 원본 파일에 대한 URL, 파일의 경로 및 이름, 마지막 문서 변경 날짜를 기록하고 있다. LAS는 요약 파일내의 파일 경로와 이름을 통하여 문서 원본 파일에 접근한다. LAS는 한 문서에 대한 요약 정보 크기를 최소화하여, 하나의 디스크 블록에 많은 문서 요약 정보가 포함되도록 한다. 따라서, 요약 파일은 긴 URL의 저장을 위하여 URL 저장 공간을 크게 할당하지 않는다. 이는 URL 저장 공간이 낭비되는 것을 방지한다. LAS는 요약 파일에 기록될 URL의 길이가 일정 값보다 크면 해당 URL을 요약 확장(Brief Ext) 파일에 저장한다. 결과적으로 LAS는 문서 원본 파일들, 요약 파일, 요약 확장 파일을 입력으로 하여 문서들의 페이지랭크 값을 계산한다.

(그림 10)은 주요 파일들에 대한 자료 구조와 자료의 예를 보인 것이다.

요약 파일의 레코드 개수는 문서 원본 파일 개수와 같다. 즉, 요약 파일의 레코드는 하나의 문서 원본 파일에 대한 정보를 나타낸다. 요약 파일은 'Url', 'File', 'Modified' 필드를 갖는다. 'Url' 필드는 92바이트의 크기를 가지고 해당 문서의



(그림 10) 연결 분석 시스템의 기본 파일 구조

실제 URL을 기록하고 있다. 92바이트보다 긴 URL은 확장 요약 파일 내에 기록되어 지고, 요약 파일의 'Url' 필드는 확장 요약 파일 내의 저장된 위치를 기록한다. 'File' 필드는 32바이트의 크기를 가지고, 디스크 에 저장된 문서 원본 파일의 경로와 파일 이름을 나타낸다. 'Modified' 필드는 4바이트의 길이를 가지고, 문서에 대한 마지막 변경 시간을 나타낸다. 변경 시간은 1970년 1월 1일 자정부터 현재까지의 경과시간을 밀리(milli) 초 단위로 나타낸다. 요약 파일내의 3개 필드는 모두 고정길이 필드이므로, 요약 파일의 레코드는 128바이트의 고정 길이 레코드이다. 문서 요약 파일내의 각 레코드는 문서번호에 대해 정렬된다. 따라서, 문서번호는 각 레코드의 바이트 오프셋(offset)을 통해 알 수 있다. (그림 10)의 요약 파일에 따르면, 0번 문서에 대한 요약 정보가 요약 파일내의 바이트 오프셋 0에 기록되어 있다. 0번 문서의 URL은 "URL0"이고, 실제 디스크상의 "0/23/ajdfd"라는 파일로 기록되어 있으며, 이 문서의 마지막 변경시간은 1999년 4월 13일이다.

연결 파일은 요약 파일에 명시된 문서들간의 연결 정보가 기록된 파일이다. 연결 파일은 'Source', 'OutDegree', 'Dest' 필드를 가진다. 'Source' 필드는 해당 레코드가 표현하고 있는 문서에 대한 문서번호를 나타낸다. 'OutDegree' 필드는 해당 문서에 존재하는 링크의 총 개수를 나타낸다. 'Dest' 필드는 해당 문서로가 가리키는 문서들의 번호를 나타낸다. 즉, 'OutDegree' 필드는 해당 문서 번호가 가지는 목적 문서 번호의 개수를 의미한다. (그림 10)의 연결 파일에 따르면, 1번 문

서는 총 5개의 링크를 가지고 0, 230, 873, 75462, 837723번 문서를 가리키는 것을 의미한다. 'Source' 필드와 'OutDegree' 필드는 4바이트의 고정 길이 필드이고, 'Dest' 필드는 가변 길이 필드로서 가리키는 목적 문서가 많을수록 커진다. 결과적으로 연결 파일의 레코드는 가변 길이 레코드가 된다.

페이지랭크 파일은 각 문서의 페이지랭크 값들의 집합을 저장한다. 하나의 페이지랭크 값은 0부터 1사이의 수로 나타나고 4바이트의 고정 길이를 가진다. 저장된 페이지랭크 값은 문서 번호에 대해 정렬되어 있으므로, n번째 문서에 대한 페이지랭크 값을 알기 위해서, $(n*4)$ 바이트 위치의 값을 참조할 수 있다.

5. 결론 및 향후연구

본 논문은 페이지랭크 알고리즘 구현에 사용되는 중요 기능들을 제시하고 설명하였다. 또한, 본 논문은 페이지랭크 알고리즘을 구현한 실제 시스템(LAS)을 보임으로써 사용자에게 페이지랭크 알고리즘을 적용하는 것에 대한 이해를 돕도록 하였다. LAS는 문서 원본 파일들, 요약 파일, 요약 확장 파일을 입력으로 하여 문서 원본 파일들의 페이지랭크 값을 산출하는 독립된 시스템이다. 두 개의 750Mhz CPU와 512 메가바이트의 메모리를 탑재한 리눅스 시스템에서 천 만개 문서 원본 파일에 대한 페이지랭크 값 산출 작업은 약 2일이 소요되었다. 연결 파일로부터 페이지랭크 값을 계산하는 작업은 약 20분 정도가 소요되었고, 대부분의 시간은 연결 파일을 생성하는 작업에 소비되었다.

[4, 8]에서 소개된 페이지랭크 알고리즘은 웹 그래프 G를 행렬 M으로 나타낼 때, 같은 컬럼에 속하는 원소들의 합이 *1*이 되어야 하는 제약 조건을 가진다. 이러한 제약조건은 행렬-벡터 곱셈의 연산 중에 결과 벡터의 원소 합이 항상 *1*이 되도록 보장한다. 만약 제약 조건이 위배되면, 결과 벡터의 원소 합은 계속해서 줄어든다. 즉, 문서들은 자신이 가져야할 페이지랭크 값보다 작은 값을 갖는다. 제약조건을 만족하기 위해서는 웹의 모든 문서가 적어도 하나 이상의 연결 URL을 포함하여야 한다. 이는 실제 웹 환경에서 지켜지기 힘든 제약조건이다. 따라서, LAS는 보다 개선된 알고리즘[6]을 사용하여 각 문서의 페이지랭크 값을 산출하였다. [6]에서 소개된 알고리즘은 연결 URL을 포함하지 않는 문서 원본 파일들이 존재하여도 모든 문서들의 페이지랭크 값의 합이 작아지지 않도록 하였다.

LAS가 신뢰할만한 페이지랭크 값을 산출하기 위해서는 다음과 같은 사항이 전제되어야 한다. 첫째, LAS가 입력받는 문서들은 최신의 문서이어야 한다. 둘째, LAS는 실제 웹 상에 존재하는 가능한 모든 문서들을 입력받아야 한다. 셋째, 입력받는 문서들은 중복이 제거되어야 한다. 웹 로봇은 웹 상의 모든 문서들을 저장하고 이에 대한 접근 방법 및 통계정보를 제공하는 시스템이다. 웹 로봇은 LAS의 전제조건을 만족시킴으로서 LAS가 신뢰할만한 페이지랭크 값을 산출할 수 있도록 해야 한다. 웹 로봇에 대한 연구는 향후 과제로 남긴다.

참 고 문 헌

[1] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," In Proceedings of World Wide Web Conference, 1998.

[2] R. L. Burden and J. D. Faires, "Numerical Analysis," Seventh Edition, BROOKS/COLE, 2001.

[3] Google Search Engine : http://www.google.com.

[4] T. H. Haveliwala, "Efficient Computation of PageRank," Unpublished manuscript, Stanford University, 1999.

[5] E.-J. Im and K. Yelick, "Optimizing Sparse Matrix Vector Multiplication on SMPS," In Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing, 1999.

[6] S. J. Kim and S. H. Lee, "An Improved Computation of the PageRank Algorithm," In Proceedings of the 24th BCS-IRSG European Colloquium on IR Research, pp.73-85, 2002.

[7] J. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "The Web as a Graph : Measurements, Models and Methods," Invited survey at the International Conference on Combinatorics and Computing, 1999.

[8] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking : Bringing Order to the Web," Unpublished manuscript, Stanford University, 1998.

[9] S. Toledo, "Improving the Memory-system Performance of Sparse-matrix Vector Multiplication," In IBM Journal of Research and Development, Vol.41. 1997.

[10] B. Zoellick and M. J. Folk, "File Structures," Second Edition, Addison-Wesley, 1991.

김 성 진

e-mail : lace@nownuri.net

1998년 숭실대학교 소프트웨어 공학과 졸업 (학사)

2000년 숭실대학교 대학원 컴퓨터학과(석사)

2002년~현재 숭실대학교 컴퓨터학과 대학원 박사과정 수료

관심분야 : 인터넷 데이타베이스, 데이타베이스 시스템 성능평가

이 상 호

e-mail : shlee@computing.soongsil.ac.kr

1984년 서울대학교 전산공학과 졸업(학사)

1986년 미국 노스웨스턴대 전산학과(석사)

1989년 미국 노스웨스턴대 전산학과(박사)

1990년~1992년 한국전자통신 연구원, 선임 연구원

1999년~2000년 미국 조지 메이슨대 소프트웨어 정보 공학과 교환 교수

1992년~현재 숭실대학교 컴퓨터학부 부교수

관심분야 : 인터넷 데이타베이스, 데이타베이스 시스템 성능 평가 및 튜닝

방 지 환

e-mail : viper122@kornet.net

2000년 숭실대학교 컴퓨터학부 졸업(학사)

2002년 숭실대학교 대학원 컴퓨터학과 졸업 (석사)

2002년~현재 (주) 소프트텔레웨어 연구원

관심분야 : 인터넷 데이터베이스, 웹 에이전트