웹의 연결구조와 웹문서의 적합도를 이용한 효율적인 인터넷 정보추출

황 인 수*

Efficient Internet Information Extraction Using Hyperlink Structure and Fitness of Hypertext Document

Insoo Hwang*

Abstract

While the World-Wide Web offers an incredibly rich base of information, organized as a hypertext, it does not provide a uniform and efficient way to retrieve specific information. Therefore, it is needed to develop an efficient web crawler for gathering useful information in acceptable amount of time.

In this paper, we studied the order in which the web crawler visit URLs to rapidly obtain more important web pages. We also developed an internet agent for efficient web crawling using hyperlink structure and fitness of hypertext documents. As a result of experiment on a website, it is shown that proposed agent outperforms other web crawlers using BackLink and PageRank algorithm.

Keywords : Information Retrieval(IR), Web Crawler, Agent

논문접수일 : 2004년 2월 27일 논문게재확정일 : 2004년 10월 12일

* 전주대학교 정보기술공학부 부교수

1. 서 론

인터넷을 위한 인프라의 구축이 확대되고 인터넷의 활용이 생활화됨에 따라 웹이나 전자우편을 통해 엄청난 양의 정보가 제공되고 있으며, 그 종류도 뉴스, 광고, 커뮤니티 등으로 매우 다양하다. 그러나 개인이 필요로 하는 정보는 극히 일부분에 지나지 않기 때문에, 인터넷에서 사용자가 원하는 정보를 신속하게 검색하여 제공하는 인터넷 정보 에이전트의 중요성은 점점 더 증대되고 있다.

인터넷의 정보를 처리하는 에이전트는 정보검색 에이전트, 정보필터링 에이전트, 정보통합 에이전트, 정보추출 에이전트 등으로 분류된다[최중민, 2000]. 정보검색 에이전트는 인터넷으로부터 사용자가 원하는 정보를 찾아주는 역할을 하는 것으로서, 웹로봇, 크롤러, 스파이더, 웜(worm), 혹은 워커(walker) 등으로 부른다[김성진 외, 2003]. 정보필터링 에이전트는 인터넷에서 제공되는 자료중에서 사용자가 원하는 정보만을 필터링하거나 가공하는 역할을 하는 것으로서, 전자우편 에이전트를 예로 들 수 있다. 정보통합 에이전트는 여러 가지의 이질적인 정보원으로부터 정보를 검색하여 단일화된 형태로 통합하여 제공하는 역할을 하는 것으로서, 메타 검색엔진이나 가격비교 쇼핑시스템 등을 예로 들 수 있다.

끝으로, 정보추출 에이전트는 Wrapper라는 정보추출규칙을 이용하여 인터넷 HTML 문서로부터 사용자가 원하는 부분의 텍스트 정보를 추출하는 역할을 하는 것으로서, 전자우편주소 추출 에이전트를 예로 들 수 있다. 정보추출규칙은 정보원으로부터 원하는 정보를 추출하기 위한 규칙이나 프로그램을 의미하는 것으로[박상위, 2002], 웹문서마다 그 구성 방식이나 내용이 다르기 때문에 추출하고자 하는 정보의 종류나 웹문서의 구성에 따라 서로 다른 정보추출규칙을 적용해야 한다[Kushmerick *et al.*, 1997].

본 연구는 웹문서에 존재하는 링크를 따라 인터넷을 항해하면서 사용자가 요구하는 정보를 효율적으로 추출하는 정보추출 에이전트의 설계 및 구현을 목적으로 한다. 인터넷 정보는 웹문서에 텍스트의 형태로 존재하기 때문에 각 문서를 탐색한 후 HTML로 이루어진 텍스트로부터 추출해야 한다. 그러나 인터넷에 존재하는 모든 웹문서를 탐색하여 정보를 추출하는 것은 현실적으로 불가능하기 때문에 효율적인 탐색 전략이 요구된다.

최근의 연구에서는 웹문서의 중요도에 따라 탐색의 순서를 결정하는 Back Link, Forward Link 등 웹의 링크를 이용한 방법들이 제안되었으며[Cho *et al.* 1998], 구글(Google) 검색엔진은 PageRank라는 개념을 도입하여[Page *et al.* 1998 ; 김성진외 2002] 검색의 정확도를 높이고 있다. 그러나 이들 방법은 사전에 웹베이스를 구축하여 각 웹문서의 중요도를 계산해 놓기 때문에, 인터넷으로부터 실시간으로 웹문서를 탐색하거나 혹은 정보를 추출할 때에는 그 성과를 보장할 수 없다. 이에 따라 본 논문은 효율적인 탐색(web crawling)을 위하여 웹의 연결구조(hyperlink)와 웹문서(hypertext)의 적합도를 함께 이용하는 방안을 제시한다.

본 논문의 구성은 다음과 같다. 제2장에서는 Web Crawling 전략의 종류와 방법, 그리고 정보추출을 하는 예로서의 전자우편주소 추출을 위한 정보추출규칙의 설계에 대해 기술하며, 제3장에서는 정보추출 에이전트의 개발에 대해 기술하고, 제4장에서는 웹사이트를 대상으로 실시한 Web Crawling 및 정보추출 시뮬레이션의 결과를 기술한다. 끝으로, 제5장에서 본 연구의 결과와 한계점을 요약한 후 향후 연구방향을 제시한다.

2. Web Crawling과 정보추출

2.1 기본적인 Web Crawling

인터넷으로부터 정보를 추출하기 위해서는 인터넷에 존재하는 모든 링크를 순차적으로 탐색하여 웹문서를 수집해야 한다. 그러나 제한된 시간내에 인터넷에 존재하는 모든 웹문서를 탐색하는 것은 물리적으로 불가능한 일이기 때문에 보다 효율적인 웹문서 탐색전략이 요구된다. 현재 주로 이용되고 있는 기본적인 탐색전략으로는 너비우선 탐색(breadth first search), BackLink, Location Metric, 그리고 구글 검색엔진에서 사용하는 PageRank [Page *et al.* 1998] 기법 등이 있다[Cho *et al.* 1998].

2.1.1 Breadth First Search(BFS)

BFS는 네트워크 혹은 트리를 너비우선으로 탐색하는 기본적인 탐색기법으로서, 웹문서에서 발견한 링크를 큐(queue)에 입력한 후 순서대로 인출하여 탐색함으로써 구현된다. BFS는 문서의 내용과는 관계없이 웹의 링크를 따라 탐색을 수행하기 때문에 시작문서로부터의 연결횟수가 적을수록 높은 우선순위를 갖는다.

2.1.2 BackLink

BackLink는 웹문서를 참조하는 링크의 개수를 의미하는 것으로서, 중요한 문서일수록 많은 문서로부터 참조되는 것이 일반적이다. 따라서, BackLink의 개수가 많을수록 웹문서는 높은 우선순위를 갖는다. 그러나 인터넷의 모든 웹문서를 검색하지 않은 상태에서 자신을 참조하는 BackLink의 개수를 정확히 계산하는 것을 불가능하기 때문에, 탐색우선순위를 결정하기 위해서는 현재시점까지 파악된 BackLink의 개수를 이용한다.

2.1.3 Location Metric

Location Metric은 웹문서의 내용과는 관계없이 웹문서의 위치에 따라 중요도를 부여하는 것으로서, 특정 단어를 포함하는 URL 혹은 URL에 포함되어 있는 슬래시(/)의 개수나 폴더의 위치 등을 이용하여 웹문서의 중요도를 평가한다. 즉, 경영과 관련있는 자료를 검색할 경우 URL에 "business" 단어가 포함되어 있거나 혹은 ".com" 도메인을 갖는 URL에 높은 우선순위를 부여하거나, 슬래시(/)의 개수가 적을수록 중요한 문서로 판단하는 것이다. 참고로, 본 연구에서는 동일한 폴더에 존재하는 웹문서중에서 이미 검색된 웹문서들로부터 계산되는 적합도를 적용한다. 본 연구에서는 웹문서가 포함하고 있는 검색정보의 평균개수를 적합도록 사용하였다.

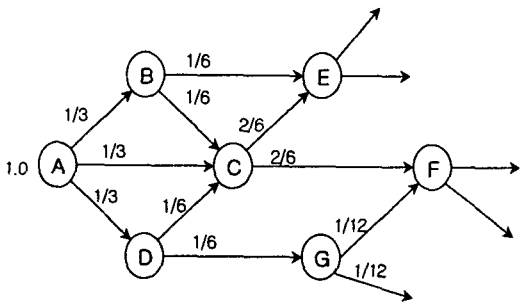
2.1.4 PageRank

위에서 설명한 BackLink는 웹문서를 참조하는 모든 링크에 동일한 가중치를 부여하기 때문에 자신을 참조하는 링크의 개수에 따라 우선순위가 결정된다. PageRank에서는 탐색을 시작하는 웹문서의 $PR(p_0)$을 1.0으로 설정한 후, 이를 후속 웹문서로 동일하게 분배하여 전파한다 [Page *et al.* 1998]. 따라서, 웹문서를 참조하는 각 링크는 해당 웹문서의 PageRank 값과 이 문서가 갖고 있는 링크의 개수에 따라 서로 다른 가중치를 갖는다. 이 방법은 구글 검색엔진에 적용되어 상당히 좋은 검색결과를 나타내는 것으로 알려져 있다. PageRank 알고리즘에서는 각 웹문서의 중요도를 다음 식에 따라 계산한다 [Cho *et al.* 1998].

$$PR(p) = (1-d) + d \times \sum_{i=1}^{n} PR(t_i)/C(t_i)$$

웹문서 p의 중요도 $PR(p)$은 p를 참조하는

다른 웹문서 t_i의 $PR(t_i)$를 해당 웹문서로부터 외부로 나가는 링크의 개수 $C(t_i)$로 나눈 값의 합으로 계산한다. 즉, 자신을 참조하는 웹문서의 중요도가 높고 이 문서에서 외부로 나가는 링크의 개수가 적을수록 $PR(p)$의 값은 커진다. 또한, d는 Damping Factor로서 0에서부터 1까지의 값을 가지며, 자신을 참조하는 웹문서들의 영향지수를 의미한다.



〈그림 1〉 네트워크에서 PageRank의 계산 예

<그림 1>은 Damping Factor가 1.0일 때, 초기값이 1.0인 PR(A)이 링크를 통하여 전체 네트워크로 전파되는 과정을 보여준다. 웹문서 A의 값이 B, C, D로 균등하게 분배되어 전파되면 각각 1/3의 값을 갖지만, C는 다시 B와 D로부터 중요도를 전파받기 때문에 PR(C)의 값은 4/6가 된다. 따라서, B, C, D 중에서 C가 가장 높은 중요도를 갖기 때문에 C를 먼저 방문하는 것이 합리적임 알 수 있다.

그러나 Web Crawling은 웹문서에 포함되어 있는 링크를 따라 이루어지므로, B와 D를 방문하지 않은 상태에서는 B→C와 D→C 링크의 존재여부를 알 수 없으므로 이 시점에서는 B, C, D가 동일한 중요도를 갖는 것으로 평가된다. 따라서, 네트워크를 검색하는 동안 계산되는 PageRank의 값은 전체 네트워크의 구조를 알고 있을 때의 PageRank 값과 상이하기 때문에 상대적으로 낮은 탐색성과를 나타내게 될 것이다.

2.2 웹의 연결구조를 이용한 Web Crawling

본 연구에서는 웹의 연결구조를 이용함과 동시에 웹문서가 포함하고 있는 검색정보의 개수에 따라 계산되는 유사도 혹은 적합도가 높은 웹문서를 우선적으로 탐색하도록 하였다. 여기서, 유사도 혹은 적합도는 웹문서가 검색질의에 적합한 정도를 나타내는 척도로서, Salton[1989]과 Yuwono[1995] 등의 정보검색에 많이 적용되어 왔다. 유사도를 계산하기 위해서는 각 문서를 n개의 단어로 구성되는 n차원의 벡터로 보고, 특정 질의 혹은 정보를 포함하는 정도에 따라 벡터를 구성하여 처리한다. 일반적으로는 문서에서 각 단어가 나타나는 횟수와 이 단어의 *idf(inverse document frequency)*를 곱한 값으로 계산한다. *idf*는 전체 웹에서 검색하는 단어가 나타나는 빈도의 역수로 계산되기 때문에, 모든 문서를 탐색하기 전에는 정확한 *idf*를 알 수 없다.

본 연구는 웹문서의 정확도를 평가하는 방법보다는 Web Crawling의 효율성에 초점을 맞추고 있기 때문에, 웹문서의 적합도를 계산하는 간단한 방법으로서 웹문서가 포함하고 있는 검색정보의 개수에 따라 다음과 같이 계산하였다.

$$F(p)^* = 1 - 1/e^m$$

여기서, m은 웹문서 p에 존재하는 검색정보의 개수를 나타낸다. $F(p)^*$는 웹문서 p를 탐색한 후에 사후적으로 결정되기 때문에 웹문서의 탐색을 위한 우선순위 결정에 사용될 수 없다. 따라서, PageRank 기법에서 제시한 중요도 전파기법을 도입하여 이 웹문서를 참조하는 웹문서들의 $F(t_i)^+$에 따라 계산한 $F(p)$의 값을 웹문서의 중요도로 사용하였다. 결과적으로, 본 연구에서 제시하는 $F(p)$는 웹문서의 적합도와

웹의 구조를 모두 반영한다.

$$F(p)=(1-d)+d\cdot\sum_{i=1}^{n}F(t_i)^{+}/C(t_i)$$

여기서, d는 RageRank 기법에서 사용한 Damping Factor이며, $F(t_i)^{+}$는 웹문서 t_i를 참조하는 웹문서들에 의해 사전적으로 계산되는 $F(t_i)$와 웹문서 t_i를 방문한 후에 결정되는 $F(t_i)$에 가중치 w를 부과하여 다음과 같이 계산한다.

$$F(t_i)^{+}=w\times F(t_i)+(1-w)\times F(t_i)^{*}$$

2.3 웹문서의 중요도 전파

전체 웹의 연결구조를 사전적으로 알고 있을 경우에는 Page[1998]가 제시한 벡터계산을 이용하거나 혹은 네트워크관련 이론을 이용하면 각 웹문서가 갖는 중요도를 쉽게 계산할 수 있다. 그러나 Web Crawling에서는 탐색을 마친 웹문서간의 연결구조만을 알 수 있기 때문에, 웹문서를 탐색하는 동안 새로운 링크가 발견되면 이미 계산이 완료된 웹문서인 경우에도 중요도를 보정해야 한다. <표 1>은 <그림 1>에 대해 PageRank 기법을 적용하여 탐색을 수행하는 과정을 기술한 것이다.

<표 1>에서 T0에 탐색을 시작하는 웹문서 A의 PR(A)는 1.0으로 초기화되어 있으며, T1에서 A의 탐색을 마치면 이 문서의 중요도 1.0을 B, C, D에 각각 0.333씩 할당한다. T2에는 B, C, D가 동일한 중요도를 가지므로 임의로 B를 탐색한다면, C는 원래의 값인 0.333에서 B→C로 전파되는 0.167을 합하여 0.5의 중요도를 갖게 되며, E는 0.167, 그리고 D는 원래의 값인 0.333을 그대로 유지한다. 이와 같은 과정을 반복함으로써 각 시점에 가장 높은 중요도를 갖는 웹문서들을 순차적으로 탐색한다.

여기서 중요도 전파의 필요성을 살펴보면 다음과 같다. <표 1>에서 보는 바와 같이 웹문서 C는 이미 T3에 중요도 0.5에 따라 탐색이 완료되었으나, T5에 D를 탐색하면 D→C 링크에 따라 D의 중요도가 C로 전파되고, 이는 다시 E와 F로 전파되므로 C, E, F의 값은 D가 전파하는 값에 따라 보정되어야 한다. 따라서, 거대한 네트워크에서 각 웹문서의 중요도를 정확히 계산하는 것은 거의 불가능하며, 네트워크에 상호 참조가 존재할 경우에는 무한루프에 빠지는 문제가 발생한다.

〈표 1〉 웹문서의 중요도와 네트워크 탐색의 과정

시점	문서	PR(A	PR(B	PR(C	PR(E	PR(D	PR(F	PR(G
T0		1.000						
T1	A		0.333	0.333		0.333		
T2	B			0.500	0.167	0.333		
T3	C				0.417	0.333	0.250	
T4	E					0.333	0.250	
T5	D			0.667	0.500		0.333	0.167
T6	F							0.167
T7	G						0.417	
...	...							

이에 따라, 본 연구에서는 중요도를 효율적으로 전파함으로써 정확도를 높일 뿐만 아니라, 무한루프에 빠지는 것을 방지하는 휴리스틱을 개발하였다. 이를 자바 프로그래밍 언어의 문법에 따라 자기호출함수로 구현하면 <그림 2>와 같다.

여기서, d는 Damping Factor로서 0.0에서 1.0까지의 값을 가질 수 있으나 본 연구에서는 네트워크의 연결구조를 보다 잘 반영하도록 비교적 큰 값인 0.9로 설정하였다. e는 다음링크로 전파되는 중요도의 한계치(threshold)로서, 웹문서가 현재 갖고 있는 중요도에 대한 증분의 비

율이 e를 초과하는 경우에만 전파한다. 이 값이 작으면 PageRank 계산의 정확도는 증가하지만 중요도 전파에 많은 시간이 소요되어 탐색효율이 저하되기 때문에 본 연구에서는 몇 차례의 반복적인 실험을 통해 0.001로 설정하였다.

또한, 웹문서간에 상호참조가 이루어질 경우에는 무한 루프에 빠지는 문제가 발생하므로, 본 연구에서는 후속 웹문서로 전파되는 증분량을 1/(링크의 개수 + 1)로 설정하여 무한루프를 해결하였다. 참고로, 아직까지 탐색하지도 않았거나 큐에 할당되지 않은 웹문서들은 중요도가 나중에 설정되므로 중요도를 전파시킬 필요가 없다. 특정 웹문서의 중요도 변화는 국부적으로만 영향을 미치기 때문에 효율적인 알고리즘의 구현이 가능할 뿐만 아니라, 새로운 웹문서의 추가에 따른 중요도 전파가 용이한 장점이 있다.

```
// 웹문서 p의 PageRank를 계산하는 메소드
public void computePageRank(Page p, double d, double e) {
  double  pageRank = 0 ;
  for (int i = 0 ; i < p.getPrev( ).size( ) ;  i++) {          // 페이지랭크값 계산.
     Page pp=(Page) p.getPrev( ).elementAt(i);
     pageRank += p.getPageRank( ) / pp.getNext( ).size( ) ;
  }
  pageRank = d * pageRank + (1-d) ;                            // damping factor 적용
  double diff = pageRank-p.getPageRank() ;                     // 기존값과의 변화(diff) 계산
  rankPropagation(p, diff, e) ;                                // 변화된 값을 후속 링크로 전파
}

// 웹문서 p의 중요도 증분을 후속 링크로 전파하는 메소드
public void rankPropagation(Page p, double diff, double e) {
   np.setPageRank(p.getPageRank( ) + diff) ;                   // PageRank 보정
   double propa = diff/(p.getNext( ).size( )+1) ;              // 다음 링크로 전파할 중요도 크기
   for (int i = 0 ; i < p.getNext( ).size( ) ;  i++) {         // 다음 링크로 연결된 각 문서에 대해
      Page np = (Page) p.getNext( ).elementAt(i) ;
      if (np.getPageRank( ) > 0 && Math.abs(propa)/np.getPageRank( ) > e)
        rankPropagation(np, propa, e) ;                        // 자기호출로 증분을 전파
   }
}
```

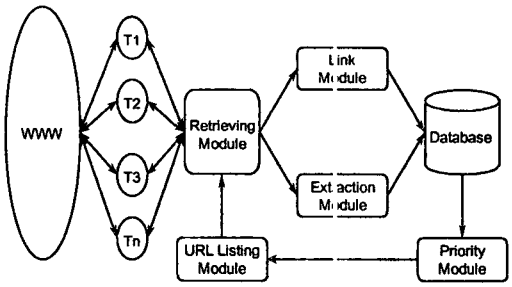
〈그림 2〉 웹문서의 중요도를 전파하는 메소드

3. 정보추출 에이전트의 구현

3.1 에이전트의 동작원리

본 연구에서는 후지쯔의 Primergy 230 펜티엄-Ⅲ 1 GHz Dual Processor에서 Windows 2000 Server를 운영체제로 하고, 프로그래밍 언어는 JAVA 1.4를 사용하였다. 데이터베이스는 Oracle8i를 사용하고, 프로그램과 데이터베이스는 JDBC로 연동하였다. 본 연구에서 개발한 인터넷 탐색 및 정보추출 에이전트의 구조를 그림으로 나타내면 <그림 3>과 같다.

Web Crawling은 시스템에서 제공하는 첫 번째 웹문서의 주소 혹은 Database에 저장되어 있는 링크 중에서 우선순위가 높은 링크로부터 시작한다. 그림에서 Retrieving Module은 링크의 URL에 존재하는 웹문서를 수집하는 모듈로서, Web Crawling을 효율적으로 수행할 수 있도록 자바의 멀티 쓰레드(multi-thread)로 구성하였다.



〈그림 3〉 정보추출 에이전트의 구조

Retrieving Module이 수집한 웹문서는 Link Module과 Extraction Module로 전달되는데, Link Module에서는 웹문서에 포함되어 있는 링크(outlink)를 추출하여 데이터베이스에 저장한다. Extraction Module은 웹문서에 포함되어 있는 정보를 추출하는 모듈로서, 본 연구에서는 웹문서로부터 정보를 추출하여 데이터베이스에 저장하는 정보추출규칙으로 구현하였다. 다음으로 Priority Module은 데이터베이스에 저장되어 있는 각 문서와 링크의 중요도를 계산하여 우선순위가 높은 링크를 URL Listing Module로 전송한다. URL Listing Module이 있는 링크들은 Retrieving Module의 각 쓰레드에 의해 호출되어 웹문서를 수집하는 과정을 반복한다.

3.2 정보추출규칙의 설계

HTML로 이루어진 웹문서로부터 정보를 추출하기 위해서는 해당하는 정보가 존재하는 양식을 파악한 후, 이에 적합한 정보추출규칙을 구성해야 한다. 앞에서도 설명한 바와 같이, 본 논문에서는 Web Crawling 전략의 효율성을 평가하는데 목적이 있기 때문에 정보추출의 대상을 정보추출규칙이 비교적 정형화되어 있는 전자우편주소로 하였다.

웹문서로부터 정보를 추출하기 위해서는 정보의 구성뿐만 아니라, 정보가 웹문서에 나타나는 형태를 파악해야 한다. 전자우편주소는 "gildong @kormail. net"과 같이 사용자의 아이디와 메일 서버의 주소로 구성되어 있으며, 웹문서에는 HTML의 <A> 태그를 이용하여 <A HREF = "mailto : gildong@kormail.net">홍길동</A>의 형태로 표현된다. 그러나 최근에는 전자우편관리 프로그램을 사용하기보다는 웹 브라우저에서 직접 메일을 작성하여 발송하는 폼메일이 일반화되어 있다. 폼메일의 경우에는 개발자가 폼메일을 발송하는 프로그램 및 주소변수를 어떻게 선언했는지에 따라서 URL을 지정하는 방법이 달라지지만, 일반적으로 <A HREF = "formMail. jsp?to=gildong@kormail.net">홍길동</A> 등의 형태를 갖는다.

이에 따라 본 연구에서는 인터넷 웹문서에 다양한 형태로 존재하는 전자우편주소를 효과적으로 추출할 수 있도록 정보추출규칙을 설계하였는데, 이를 Java 프로그래밍 언어의 메소드로 표현하면 <그림 4>와 같다. 즉, 웹문서에서 전자우편주소를 분리하기 위한 <> : " 등의 기호를 이용하여 HTML 문서를 토큰(token)으로 분리한 후, At(@) 기호 뒤에 점(.)을 갖는 토큰만을 전자우편주소로 인정한다.

```
// 전자우편 주소를 추출하는 메소드
public void extractMailFrom(String content, Vector
mail) {
  String separator = "\n\"\'<>: " ;
                                  // 문자열 분리 기준
  StringTokenizer st = new StringTokenizer(content,
  separator) ;                    // 문자열 분리
  while(st.hasMoreTokens( )) {
    String token=st.nextToken( ) ;
    if(token.indexOf("@") > 1 && token.indexOf("@")
                  + 2 < token.indexOf("."))
      mail.addElement(token) ;  // 전자우편주소 추가
  }
}
```

〈그림 4〉 전자우편주소 추출규칙의 예

4. 문제정의 및 시뮬레이션

4.1 문제정의

본 연구는 웹문서에 있는 링크를 따라 자율적으로 인터넷을 항해하면서 정보를 효율적으로 추출하는 정보검색 및 정보추출 에이전트의 개발을 목적으로 한다. 따라서, 에이전트의 적용범위는 인터넷 전영역이 될 수 있으나, 컴퓨터 시뮬레이션이 용이하도록 탐색의 범위를 국내 *P* 대학의 웹사이트로 한정하였으며, '~'로 연결되는 개인 웹사이트와 게시판은 탐색의 대상에서 제외하였다. 또한, 추출하는 정보는 웹문서에 포함된 모든 문자열을 대상으로 할 수 있으나, 앞에서 설명한 바와 같이 탐색전략별 비교가 용이하도록 전자우편주소를 추출하는 것을 목표로 하였다.

본 연구에서 제안하는 에이전트의 성과를 측정하기 위해 웹문서 검색 에이전트를 이용하여 사전적으로 조사한 *P* 대학 웹사이트의 깊이(depth)별 경로(directory), 웹문서의 내용, 그리고 검색정보의 개수를 표로 정리하면 <표 2>와 같다.

이 사이트에는 총 11,800개의 웹문서에 13,714개의 검색정보가 존재하므로 웹문서당 약 1.2개의 검색정보가 존재함을 알 수 있다. 그러나 한 개 이상의 검색정보를 갖고 있는 웹문서는 2,867개에 불과하기 때문에 검색정보를 갖고 있는 웹문서의 평균 검색정보의 개수는 약 4.8개이다. 또한, 총 39,218개의 링크가 존재하는 데, 이것은 하나의 웹문서에 약 3.3개의 링크가 존재함을 의미한다. 전체 문서 11,800개 중에서 링크(outlink)를 갖지 않는 단말노드(terminal node)가 전체 웹문서의 절반에 가까운 5,829개로서, 링크를 갖는 웹문서들은 평균적으로 약 6.6개의 링크를 갖는 것으로 나타났다.

결과적으로, 본 연구에서 대상으로 하는 웹사이트는 비교적 깊이가 깊은 트리 형태임을 알 수 있으며, 검색정보가 각 링크에 분산되어 있기 때문에 원하는 정보를 찾기까지 많은 시간이 소요될 것임을 예상할 수 있겠다.

〈표 2〉 http://www.pXxx.ac.kr 사이트의 분석결과

깊이	경로의 개수	문서의 개수	검색정보의 개수	문서당 정보개수
1	91	107	31	0.29
2	522	1,055	296	0.28
3	633	3,220	5,431	1.69
4	496	3,675	5,038	1.37
5	238	2,339	2,425	1.04
6+	106	1,404	493	0.35
합계	2,086	11,800	13,714	1.16

4.2 시뮬레이션 설계

본 연구에서 제시한 여러 가지의 탐색전략를 실제 웹사이트에 적용하여 시뮬레이션을 수행할 경우 해당 웹서버에 부담을 줄뿐만 아니라, 시뮬레이션에 많은 시간이 소요된다. 따라서 본 연구에서는 웹문서 검색 에이전트를 이용하여 *P* 대학 웹사이트의 각 웹문서의 주소, 검색정보의 개수, 웹문서간의 링크관계 등에 대한 자료를 수집하여 지역데이터베이스에 저장하였다.

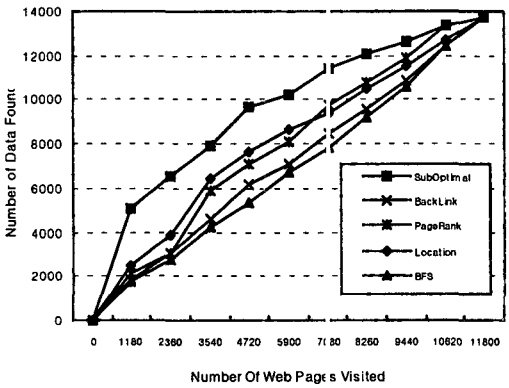
다음으로, 시뮬레이션의 수행은 지역 데이터베이스에 저장된 내용을 읽어서 웹문서, 링크 등을 객체로 설정한 후, 컴퓨터의 메모리상에서 모든 시뮬레이션이 수행되도록 구성하였다. 따라서, 컴퓨터의 메모리에 구성된 가상 인터넷을 이용함으로써 다양한 조건하에서 시뮬레이션을 반복적으로 수행할 수 있었다.

4.3 시뮬레이션 결과

4.3.1 기본 탐색 알고리즘의 성과 분석

탐색의 기본이 되는 넓이우선탐색(BFS)과 웹

의 연결구조를 이용한 BackLink 및 PageRank 등의 성과를 측정하는 시뮬레이션을 실시하였으며, 이의 결과를 그림으로 나타내면 <그림 5>와 같다.



〈그림 5〉 기본적인 탐색알고리즘의 정보추출 성과비교

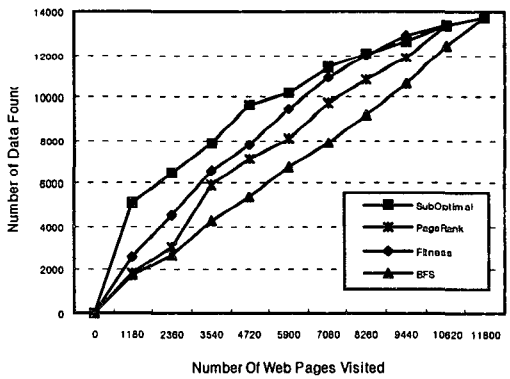
여기서, X축은 전체 웹문서중에서 검색을 마친 웹문서의 개수를 나타내며, Y축은 검색한 웹문서로부터 추출한 정보의 개수를 나타낸다. BFS는 웹문서에서 각 링크가 발견되는 순서에 따라 탐색을 수행하기 때문에 검색한 웹문서와 추출한 정보의 비율이 직선으로 나타남을 알 수 있다. 웹의 구조를 이용하는 BackLink는 BFS보다 약간 나은 성과를 나타냈으며, Google 등에서 사용되는 PageRank와 Location Metric은 BackLink에 비하여 상당히 향상된 성과를 보여주고 있다.

SubOptimal은 웹문서의 적합도, 즉 웹문서가 포함하고 있는 검색정보의 개수를 사전적으로 알고 있다고 가정할 때, 검색예정인 모든 웹문서 중에서 검색정보를 가장 많이 갖고 있는 웹문서를 우선적으로 검색했을 때의 성과를 나타낸다. 따라서, 전체 네트워크를 고려하지 않았기 때문에 사용하는 성과척도에 따라서 최적해가 될 수는 없으나, 최적해의 근사해로 사용될 수 있다.

SubOptimal이 비교적 완만한 경사를 갖는 것은 앞에서 언급한 바와 같이, 연구 대상으로 택한 웹사이트의 링크구조가 다단계의 트리구조를 갖기 때문에 하위의 웹문서를 탐색하기 위해서는 상위의 웹문서를 순차적으로 탐색하는 것이 불가피하기 때문으로 분석된다.

4.3.2 웹구조를 이용하는 탐색 알고리즘의 성과 분석

웹의 검색 및 정보추출의 효율성을 높이기 위해 웹의 구조와 문서의 적합도를 이용하는 탐색알고리즘에 대한 시뮬레이션을 실시한 결과 상대적으로 좋은 성과를 나타냈는데, 이를 그림으로 나타내면 <그림 6>과 같다.



〈그림 6〉 웹구조를 이용한 정보추출의 성과비교

웹문서의 적합도는 질의에 적합한 정도를 나타내는 것으로서, 본 연구에서는 웹문서에 포함된 검색정보의 개수를 이용하여 적합도를 계산하였다. 그러나 앞의 탐색전략에서도 기술한 바와 같이 웹문서의 적합도는 그 문서를 방문하기 전까지는 알 수 없기 때문에 이 문서를 참조하는 문서들의 적합도를 이용하여 계산하였다.

그림에서 보는 바와 같이, 본 연구에서 제시한 웹의 구조와 웹문서의 적합도를 이용한 탐색방법은 탐색의 모든 영역에서 PageRank 방법

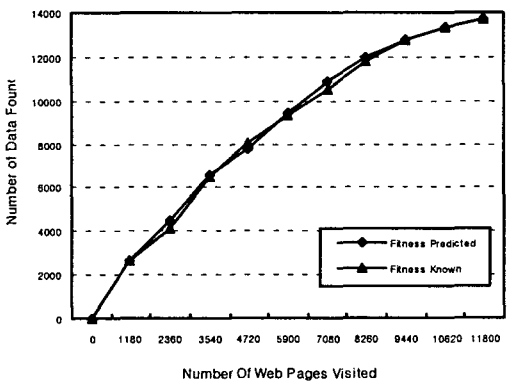
보다 우수한 성과를 나타냈으며, 탐색이 진행될수록 SubOptimal에 상당히 근접하는 것으로 나타났다.

4.3.3 중요도 예측의 정확도 검증

위에서 BackLink, PageRank, 그리고 Fitness 등과 같이 웹의 구조를 이용하여 탐색을 수행하는 알고리즘들은 전체 네트워크를 모두 검색하지 않으면 정확한 값을 계산할 수 없는 단점이 있다. 그러므로 탐색한 웹문서만을 이용하여 탐색할 웹문서들의 중요도를 계산해야 하는 Web Crawler는 사전에 웹문서를 모두 데이터베이스화하여 검색서비스를 제공하는 검색엔진에 비하여 성과가 저하될 수밖에 없다.

이에 따라, 본 연구에서는 사전에 웹의 구조를 모두 알고 있는 경우에 각 탐색전략을 적용한 결과와의 성과를 비교하는 추가적인 연구를 수행하였다.

<그림 7>은 이의 결과를 보여주고 있는데, 전체 웹의 구조를 사전에 알고 있는 경우와 그렇지 않은 경우의 탐색성과간에는 큰 차이가 없는 것으로 나타났다. 결과적으로, 본 연구에서 적합도를 예측하기 위해 사용한 성과지표는 충분한 정확도와 예측력을 갖는 것으로 판단된다.



〈그림 7〉 적합도에 대한 인지여부에 따른 성과비교

5. 결론 및 향후 연구계획

본 연구는 인터넷으로부터 정보를 검색 혹은 추출하는 여러 가지 에이전트 중에서 정보검색을 통한 정보추출 에이전트의 설계 및 구현에 초점을 맞추었으며, 정보추출을 위한 정보추출 규칙이 비교적 정형화되어 있는 전자우편주소 추출문제를 대상으로 하였다. Web Crawling의 순서를 결정하기 위한 방법으로 많이 사용되고 있는 BackLink 및 PageRank의 성과를 측정하였으며, 정보검색의 효율을 극대화하기 위한 방안으로서 웹구조와 문서의 적합도를 이용하는 방안을 연구하였다.

본 연구에서는 각 알고리즘의 효율성을 평가하기 위해 국내 P 대학의 웹사이트에 있는 모든 웹문서를 지역 데이터베이스에 다운받은 후 데이터베이스를 이용한 시뮬레이션을 실시하였다. 시뮬레이션 결과, 웹의 연결구조에 따라 문서의 적합도를 계산하여 검색을 수행한 경우 PageRank보다 훨씬 더 나은 성과를 나타냈으며, 부분적으로는 SubOptimal에 가까운 성과를 나타냈다. 다만, 탐색을 시작하는 초기에는 SubOptimal에 비교하여 상대적으로 낮은 성과를 나타냈는데, 이것은 검색된 웹문서의 개수가 작은 경우에는 웹의 구조를 이용한 중요도 전파가 충분히 이루어지지 않았기 때문으로 해석된다. 따라서, 초기의 검색성과를 높일 수 있는 알고리즘을 개발하여 본 연구에서 제시한 적합도 방식과 함께 결합하여 사용하는 방안에 대한 연구가 필요할 것으로 생각된다.

본 연구의 한계점으로는 인터넷을 대상으로 하는 시뮬레이션이기 때문에 보다 많은 다양한 사이트를 대상으로 하지 못했다는 점과 비교적 정형화된 전자우편주소 추출을 문제로 택하였다는 점을 들 수 있겠다. 따라서, 향후에는 다양

한 영역에서 다양한 정보를 보다 효율적으로 검색하여 추출하는 인공지능 기법의 도입방안에 대해 연구하고자 한다. 또한, 다양한 웹사이트로부터 정보를 효과적으로 추출을 위해 정보추출규칙을 지능적으로 설계하는 방안에 대해서도 연구하고자 한다.

참 고 문 헌

[1] 김성진, 이상호, "웹 로봇 구현 및 한국 웹 통계보고", 정보처리학회논문지C, 제10-C권 제4호, 2003년.

[2] 김성진, 이상호, 방지환, "페이지랭크 알고리즘 적용을 위한 구현 기술", 정보처리학회논문지D, 제9-D권 제5호, 2002년.

[3] 김은정, 배종민, "XLinks를 이용한 하이퍼텍스트 검색시스템", 정보처리학회논문지D, 제8권 제5호, 2001년.

[4] 박상위, 오정석, 이상호, "메타 검색엔진을 위한 HTML 문서변경탐지기의 설계 및 구현", 정보처리학회논문지D, 제9-D권 제3호, 2002년.

[5] 이동원, 현순주, "Hyperlink 구조와 Hypertext 분류방법을 이용한 Web Crawler", 한국정보처리학회 춘계학술발표논문집, 제9권 제1호, 2000년.

[6] 최중민, "에이전트의 개요와 연구방향", 정보과학회지, 제15권 제3호, 1997년, pp. 7-16.

[7] 최중민, "인터넷 정보추출 에이전트", 정보과학회지, 제18권 제5호, 2000년, pp. 48-53.

[8] 황인수, "적응적 탐색기법을 이용한 인터넷 정보추출 에이전트의 설계 및 구현", 산경논총, 제21권 제2호, 2003년.

[9] Ambite, J., N. Ashish G. Barish, C. Knoblock, S. Minton, P. Modi, I. Muslea, A. Philpot and S. Tejada, "ARIADNE : A System for Constructing Mediators for Internet Sources", *ACM SIGMOD International Conference on Management of Data*, 1998, pp. 561-563.

[10] Cho, J., H. Garcia-Molina and L. Page, "Efficient Crawling Through URL Ordering", *Proceedings of the Seventh International Web Conference*, 1998.

[11] Cho, J. and H. Gracia-Molina, "Parallel Crawlers", *26th Conference on VLDB*, 2002, pp. 124-135.

[12] Cohen, W., "A Web-based Information System that Reasons with Structured Collections of Text", *Second International Conference on Autonomous Agents*, 1997, pp. 400-407.

[13] Doorenbos, R., O. Etzioni and D. Weld, "A Scalable Comparison-Shopping Agent for the World Wide Web", *First International Conference on Autonomous Agents*, 1997, pp. 39-48.

[14] George Chang, Marcus J. Healey, James A.M. McHugh and Jason T.L. Wang, *Mining the World Wide Web : An Information Search Approach*, Kluwer Academic Publishers, 2000.

[15] Jennings, N., K. Sycara and M. Wooldridge, "A Roadmap of Agent Research and Development", *Autonomous Agents and Multi-Agent Systems*, Vol. 1, 1998, pp. 7-38.

[16] Kushmerick, N., "Gleaning the Web", *IEEE Intelligent Systems*, Vol. 14, No. 2, 1999, pp. 20-22.

[17] Kushmerick, N., D. Weld and R. Doorenbos, "Wrapper Induction for Information Extraction", *International Joint Conference on Artificial Intelligent*, 1997, pp.

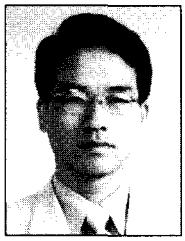
729-735.

[18] Page, L., S. Brin, R. Motwani and T. Winograd, "The PageRank Citation Ranking: Bring Order to the Web", *Technical Report*, Stanford University, Stanford, CA, 1998.

[19] Salton, G., *Automatic Text Processing*, Addison Wesley, Massachusetts, 1989.

[20] Yuwono, B., L. Lam, H. Ying and L. Lee, "A World Wide Web Resource Discovery System", *The Fourth International WWW Conference, Boston*, USA, December 1995.

저자소개



황 인 수

전주대학교 정보기술공학부 정보시스템 전공의 부교수로 재직중이다. 고려대학교 경영학과를 졸업하고 동 대학원에서 경영정보시스템을 전공하여 석사 및 박사학위를 취득하였으며, 산업연구원(KIET) 물류·유통연구센터의 연구원을 역임하였다. 주요 관심분야는 e-Business, CRM, 데이터마이닝, 웹 에이전트 등이다.

◈ 이 논문은 2004년 2월 27일 접수하여 1차 수정을 거쳐 2004년 10월 12일 게재확정되었습니다.