웹 문서와 접근로그의 하이퍼링크 추출을 통한 웹 구조 마이닝

이성대* · 박휴찬**

Web Structure Mining by Extracting Hyperlinks from Web Documents and Access Logs

Seong Dae Lee* · Hyu Chan Park**

요 약

웹 사이트의 구조가 정확하게 주어진다면, 정보 제공자의 입장에서는 사용자의 행위 패턴이나 특성을 효과적으로 파악할 수 있어 보다 나은 서비스를 제공할 수 있고, 사용자의 입장에서는 더욱 쉽고 정확하게 유용한 정보를 찾을 수 있을 것이다. 하지만 웹상의 문서들은 빈발하게 수정되기 때문에 웹 사이트의 구조를 정확하게 추출하는 것은 상당한 어려움이 있다. 본 논문에서는 이러한 웹 사이트의 구조를 자동으로 추출하는 알고리즘을 제안한다. 제안하는 알고리즘은 두 단계로 구성된다. 첫 번째 단계는 웹 문서를 분석하여 그들 간의 하이퍼링크를 추출하고 이를 웹 사이트의 구조를 나타내는 방향 그래프로 표현한다. 하지만 플래시나 자바 애플릿에 포함된 하이퍼링크는 추출할 수 없는 한계가 있다. 두 번째 단계에서는 이러한 숨겨진 하이퍼링크를 추출하기 위하여 웹 사이트의 접근로그를 이용한다. 즉, 접근로그로부터 각 사용자의 클릭스트림을 추출한 후, 첫 번째 단계에서 생성한 그래프와 비교하여 숨겨진 하이퍼링크를 추출한다. 본 논문에서 제안한 알고리즘의 성능을 평가하기 위하여 다양한 실험을 수행하였고, 이러한 실험을 통하여 웹 사이트의 구조를 보다 정확하게 추출할 수 있음을 확인하였다.

ABSTRACT

If the correct structure of Web site is known, the information provider can discover users' behavior patterns and characteristics for better services, and users can find useful information easily and exactly. There may be some difficulties, however, to extract the exact structure of Web site because documents on the Web tend to be changed frequently. This paper proposes new method for extracting such Web structure automatically. The method consists of two phases. The first phase extracts the hyperlinks among Web documents, and then constructs a directed graph to represent the structure of Web site. It has limitations, however, to discover the hyperlinks in Flash and Java Applet. The second phase is to find such hidden hyperlinks by using Web access log. It fist extracts the click streams from the access log, and then extract the hidden hyperlinks by comparing with the directed graph. Several experiments have been conducted to evaluate the proposed method.

키워드
웹 마이닝, 웹 구조, 접근로그, 그래프

* 한국해양대학교 산학협력단
** 한국해양대학교 컴퓨터·제어·전자통신공학부 (교신저자)

접수일자 : 2007. 8. 10

Ⅰ. 서 론

컴퓨터 기술의 발달과 웹의 급속한 성장으로 웹 사이트의 사용자가 기하급수적으로 증가하였고, 사용자에게 다양한 정보를 실시간으로 제공하고 있다. 기존의 다양한 정보 서비스 시스템들은 웹을 기반으로 통합하거나 웹 사이트를 구축하여 질 높은 정보의 공유와 보다 안정성 있는 정보의 흐름을 보여주고 있다[1]. 이러한 웹 사이트들은 정보의 양이 방대해지고 웹 문서들의 잦은 생성, 삭제, 수정이 반복되면서 더욱 복잡한 구조의 양상을 띄게 되었으며 사용자가 원하는 최적의 정보에 접근하는 방법 또한 예측하기 힘든 구조를 가지게 되었다. 이러한 구조에서 사용자의 행동패턴과 웹 문서간의 관계를 예측할 수 있다면 개발자는 합리적이고 효율적인 시스템을 구축하여 보다 나은 서비스를 제공하여 할 수 있을 것이다.

따라서 웹 사이트의 구조를 체계적으로 분석할 수 있는 방법이 요구되고 있다. 이를 위해서는 통계적이고 경험적인 분석을 통하여 사용자들의 행동패턴을 관찰하고 실질적인 데이터를 수집하는 것이 요구된다[2]. 그리고 정확한 데이터를 수집하기 위해서 웹 문서의 분석과 데이터의 흐름을 분석하여 많은 링크를 추출하고 체계적으로 구조화해야 한다. 이를 통하여 사용자의 특성과 행동 패턴에 적합한 웹 사이트를 구성하고 웹 사용의 효율성을 증대시키는 연구가 있었다[3,4].

웹의 구조를 추출하기 방법은 주로 웹 문서들 간의 경로분석을 통하여 이뤄진다[5,6]. 그러나 이런 경로분석 방법들은 몇 가지 문제점들을 가지고 있다. 그 중에 가장 큰 문제점은 그래픽이나 프로그램인 경우 경로추적이 어렵다는 것이다. 즉, 근래의 웹은 텍스트 형태의 문서보다 그래픽이나 프로그램 위주로 구성되고 있다. 특히 애드온(add-on) 형태로 동작하는 플래시(Flash)나 자바 애플릿(Java applet) 등이 포함된 웹 문서들은 경로 추적이 어려워지면서 구조적 패턴 탐사가 힘들어졌다.

이를 해결하기 위하여 본 논문에서는 웹 문서뿐만 아니라 접근로그를 이용하여 웹 사이트의 구조를 추출하는 방법을 제안한다. 제안하는 방법은 크게 두 단계로 나누어진다. 첫 번째, 웹 문서의 태그구조 분석을 통해 하이퍼링크를 탐색하고 탐색된 링크의 우선 순서로 깊이 우선탐색(depth first search) 알고리즘을 적용함으로써 방향성 그래프(directed graph) 형태로 구조화한다. 하지만 이 방법만으로는 플래시와 애플릿 등에 숨겨진 하이퍼링크를 찾을 수 없다. 두 번째, 웹 접근로그를 이용하여 첫 번째 과정에서 만들어진 방향성 그래프를 보완한다. 접근로그 정제과정을 거쳐 세션별 클릭스트림(click stream)을 추출하여 큐(queue)와 스택(stack)을 이용해 숨겨진 하이퍼링크를 찾아내고 새로운 정점과 간선을 그래프에 추가한다.

본 논문의 2장에서는 기존의 연구에서 제시된 웹 마이닝과 접근로그, 전처리 과정, 클릭스트림에 대하여 논한다. 3장에서는 웹 문서에서 하이퍼링크를 추출하여 방향성 그래프를 생성하는 방법과 접근로그에서 클릭스트림을 추출하고 새로운 정점과 간선을 갱신하는 방법을 설명한다. 4장에서는 시스템의 구현 및 실험에 관하여 설명하고, 5장에서는 결론과 향후 연구과제를 제시한다.

Ⅱ. 관련 연구

본 논문과 관련이 있는 웹 마이닝과 웹 로그에 대하여 살펴본 후, 웹 마이닝을 위한 기초 기술인 웹 로그의 전처리 과정과 세션별 클릭스트림 추출 방법에 대하여 논한다.

2.1 웹 마이닝

웹 마이닝은 웹 환경에서 적용되어지는 데이터 마이닝 기술로서 웹으로부터 얻어지는 방대한 양의 정보를 필터링하고 분석하여 유용한 정보를 찾아내는 것이다. 웹 마이닝은 웹 내용 마이닝(Web content mining), 웹 사용 마이닝(Web usage mining), 웹 구조 마이닝(Web structure mining)으로 구분된다[7,8]. 먼저, 웹 내용 마이닝은 웹 사이트를 구성하고 있는 웹 페이지로부터 유용한 정보를 추출하는 기법이다. 즉, 웹 기반의 구조화되지 않은 데이터로부터 원하는 정보를 찾기 위한 데이터 마이닝 기법이다. 두 번째, 웹 사용 마이닝은 웹 서버에 접속한 사용자들의 행위를 저장하고 있는 로그를 이용하여 사용자들의 접속 유형을 탐색하는 기법이다. 마지막으로, 웹 구조 마이닝은 웹 문서들 간의 구조를 추출하는 것이다. 즉, 하이퍼링크를 통해 참조되거나 참조하는 문서들 간의 관계에 대한 정보를 탐색하는 기법이다. 이런 기법은 사용자의 행동패턴과 정보를 쉽고 빠르게 찾을 수 있

도록 하며 양질의 웹 서비스를 제공하는데 도움을 준다.

2.2 웹 로그

사용자가 웹 서버에 접속하게 되면 그 이후 클릭하는 모든 작업들이 웹 서버의 로그에 기록되게 된다. 기록되는 내용으로는 IP 주소와 데이터 크기, 접속시간, 상태코드, 프로토콜, 전송된 문서경로 등 다양한 정보가 함께 기록되고, 사용자가 요청하는 특정 웹 문서뿐만 아니라 해당 웹 문서와 관련된 이미지 파일과 이미지데이터 등에 대한 정보까지 웹 로그파일에 기록되게 된다. 따라서 웹 서버는 웹 서비스에 대한 요청과 제공에 대한 기록을 모두 웹 로그에 저장하게 된다. 이것은 웹 마이닝의 기초 자료로 활용될 수 있다. 본 논문에서는 웹 문서로부터 추출할 수 없는 숨겨진 하이퍼링크를 찾기 위한 자료로 사용한다.

웹 로그는 웹 서버에 따라 여러 형식으로 나누어지는데, CLF(Common Log Format)와 IIS (Internet Information Server)형식이 대표적이다. 본 논문에서는 CLF 형식의 웹 로그를 사용하였다. 웹 로그는 기록되는 내용에 따라 일반적인 사항을 기록하는 접근로그(access log), 에러나 접속 실패를 기록하는 에러로그(error log), 검색엔진 등 소개해 준 사이트를 기록하는 참조로그(referrer log), 사용하는 브라우저와 운영체제를 기록하는 에이전트로그(agent log)로 나눌 수 있다[9].

2.3 웹 로그의 전처리

웹 로그에는 마이닝에 꼭 필요한 정보만 있는 것은 아니며, 웹 문서에 포함된 이미지에 대한 데이터 등 불필요한 정보도 포함되어 있다. 따라서 웹 로그를 마이닝에 적합한 데이터 형태로 변환하는 전처리(preprocessing) 과정이 필요하다. 그림 1은 웹 로그의 전처리 과정을 나타내고 있다.

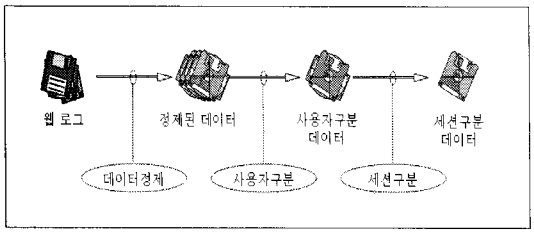


그림 1. 웹 로그의 전처리 과정
Fig. 1 Preprocessing of Web logs

그림 1에서처럼 데이터 정제(data cleaning)는 웹 서버에 기록된 웹 로그의 불필요한 정보를 필터링하는 과정이다. 사용자 구분(user identification)은 웹 사이트에 접근한 사용자를 구분하는 과정이다. 로컬 캐시, 방화벽, 프록시 서버 등 때문에 표준 웹 로그만으로는 실제 사용자를 정확히 구분하는데 한계가 있다. 이를 극복하기 위하여 웹 로그에 사용자 정보가 포함되도록 웹 서버를 수정하거나, 각각의 웹 문서에 로그를 저장하도록 프로그래밍하고 쿠키를 사용하는 등의 방법이 필요하다. 세션 구분(session identification)은 한 사용자가 웹 사이트에 접속하여 웹 탐색을 수행한 후 접속을 종료할 때까지의 일련의 행위이다. 따라서 실제 사용자의 웹 사용패턴을 탐색하기 위한 세션 구분은 데이터 마이닝의 입력 데이터로서 매우 중요한 의미를 가진다. 그러나 웹 로그만으로 세션을 구별하기에는 어려움이 있다. 사용자들은 로그아웃 없이 웹 브라우저를 종료하거나 브라우저의 '뒤로' 버튼으로 인해 웹 서버의 접근로그에 사용자의 행위가 기록되지 않는 경우가 발생할 수 있기 때문이다. 사용자 및 세션 구분에 사용되는 일반적인 방법들은 표 1에서 비교하였다[10,11].

표 1. 사용자 및 세션 구분을 위한 방법
Table. 1 Methods for user and session identification

방법	설명	보안	장·단점
IP주소, 에이전트	IP, 에이전트의 쌍	낮음	· 항상 사용 가능 · 추가 기술 불필요 · 유동 IP 식별 불가
세션 ID	링크에 동적으로 생성된 페이지 ID 삽입	낮음/보통	· 항상 사용 가능 · IP 주소에 독립적 · 반복 접근 인식 불가 · 동적 사이트 요구
등록	사용자가 직접 등록	보통	· 모든 브라우저 인식 · 등록을 원하지 않는 사용자 존재
쿠키	쿠키 활용	보통/높음	· 반복된 접근 인식 · 사용자의 거부
SW 에이전트	브라우저에 삽입된 에이전트	높음	· 정확한 사용 데이터 · 사용자의 거부
브라우저 수정	브라우저가 사용 데이터 기록	매우 높음	· 정확한 사용 데이터 · 사용자가 브라우저에 요구사항을 명시

2.4 클릭스트림

클릭스트림은 웹 사이트상에서의 사용자의 이동경로를 분석할 수 있는 모든 행위를 말한다. 이것은 웹 브

라우저를 통해 어떤 객체를 선택하거나 원하는 항목을 마우스를 이용하여 클릭하는 순차적인 흐름이다. 이것은 고객의 행동, 선호도, 구매성향 등에 대한 지식기반이 될 수 있으며, 사용자의 최근 행위를 반영하는 개인에 맞춰진 역동적인 컨텐츠 제공에 이용할 수 있는 자료가 된다. 본 논문에서는 클릭스트림을 이용하여 사용자 행동을 분석하고 경로를 파악하여 웹 문서에서 찾지 못한 숨겨진 하이퍼링크를 찾는데 사용한다.

Ⅲ. 웹 사이트의 구조 추출

웹 문서 내의 하이퍼링크는 각 문서를 연결하는 자료의 흐름이며, 이는 방향성이 있는 그래프로 표현할 수 있다. 본 논문에서는 웹 문서와 접근로그를 사용하여 웹 사이트의 구조를 추출하고 이를 방향성이 있는 그래프로 표현한다. 그림 2는 본 논문에서 제안하는 알고리즘의 흐름도이다.

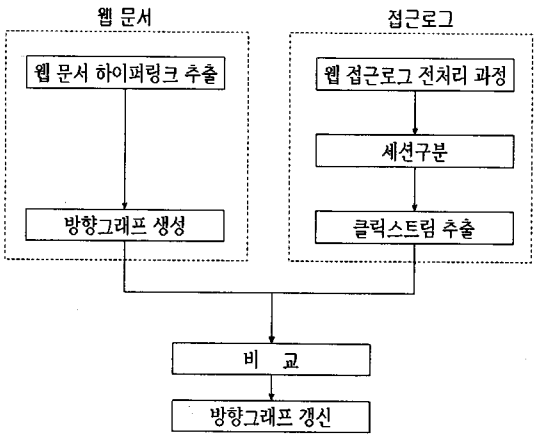


그림 2. 제안하는 알고리즘의 흐름도
Fig. 2 Flowchart of proposed algorithm

그림 2에서처럼 먼저 각 웹 문서 사이의 하이퍼링크를 추출하여 방향성 그래프로 표현한다. 하지만 웹 문서 내의 플래시와 같은 경우 해당 하이퍼링크는 존재하지만 플래시를 포함하고 있는 실제 웹 문서에는 하이퍼링크가 나타나지 않는다. 이렇게 숨겨진 하이퍼링크를 찾는 방법으로 본 논문에서는 접근로그의 전처리과정을 거쳐 세션별 클릭스트림을 추출하여 이미 생성된 방향성 그래프와 비교 분석하여 방향 그래프를 갱신한다.

3.1 웹 문서의 하이퍼링크 추출

웹 사이트의 구조를 추출하기 위하여 우선 웹 문서의 태그를 분석하여 하이퍼링크를 추출한다. 그림 3은 웹 사이트의 시작 웹 문서 내에 존재하는 하이퍼링크를 따라 웹 사이트 내부의 문서들을 재귀적으로 탐색하는 알고리즘이다. 시작 웹 문서로부터 얻어진 하이퍼링크를 추출하여 연결된 모든 문서들을 깊이우선탐색 알고리즘을 적용하여 탐색한다. 탐색 시 문서를 방문하는 순서대로 스택에 넣고 방문했음을 기록한다. 이 알고리즘은 깊이우선탐색을 하기 때문에 탐색 시 문서 d가 방문 될 때마다 그 문서와 인접하면서 방문하지 않은 모든 문서들을 순환적으로 방문할 수 있다. 그림 3의 알고리즘에서 isEquelSite()는 동일한 사이트인지를 하이퍼링크 주소를 통해 결정하는 함수이며, isVisited()는 하이퍼링크에 연결된 문서의 방문 여부를 검증하는 함수이다. AddDocument()는 문서 d의 하이퍼링크 h를 따라 그래프에 정점과 간선을 추가하는 함수이며, AddVisitedDocument()은 문서 d를 방문하였음을 표시하는 함수이다.

```
Algorithm FindHyperLinks
 input: Web document d
 output: Graph G=(V, E)

 G.V = 0; // 웹 사이트 그래프 정점 초기화
 G.E = 0; // 웹 사이트 그래프 간선 초기화

 Find_Hyperlink(document d) {
  AddVisitedDocument(d);
  // 문서 d에 있는 각 하이퍼링크 h에 대하여
   foreach hyperlink h in document d {
    if (isEquelSite(h) == true) {
     // 그래프 G에 d → h 링크 추가
      AddDocument(d, h);
     if (isVisited(h) == false)
       Find_Hyperlink(h);
     else
       return;
      }
   }
 }
```

그림 3. 하이퍼링크 추출 알고리즘
Fig. 3 Algorithm for extracting hyperlinks

이런 과정을 통하여 웹 문서로부터 하이퍼링크를 추출한 예제는 표 2와 같고, 이를 방향 그래프로 나타내면 그림 4와 같다.

표 2. 추출한 하이퍼링크 예제
Table. 2 An example of extracted hyperlinks

문서	→	문서
A.html	→	B.html
A.html	→	C.html
A.html	→	D.html
A.html	→	E.html
A.html	→	F.html
B.html	→	G.html
B.html	→	H.html
C.html	→	I.html

문서	→	문서
C.html	→	J.html
D.html	→	A.html
D.html	→	J.html
D.html	→	K.html
E.html	→	L.html
I.html	→	B.html
L.html	→	A.html

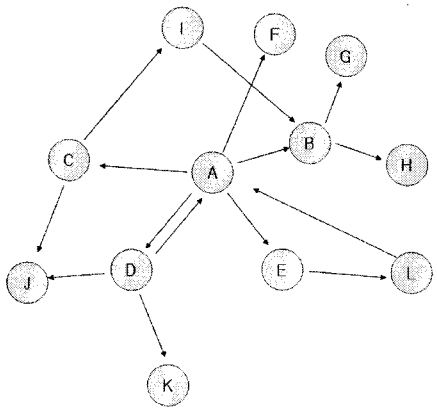


그림 4. 하이퍼링크의 그래프 표현
Fig. 4 Graph representation of hyperlinks

그림 4의 그래프는 플래시에 존재하는 하이퍼링크와 같이 웹 문서에서 찾을 수 없는 숨겨진 하이퍼링크가 표현되어 있지 않기 때문에 완벽한 웹 사이트의 구조를 나타내는 그래프가 아니다. 이를 보완할 수 있는 방법으로 본 논문에서는 접근로그로부터 획득할 수 있는 클릭스트림을 이용하여 새로운 하이퍼링크를 찾아 그림 5의 방향성 그래프를 보완한다.

3.2 접근로그 전처리

접근로그 데이터를 이용하여 새로운 정점과 간선들을 갱신하여야 한다. 그러기 위해서는 접근로그의 정제과정을 거친 후 사용자 및 세션을 구분하는 전처리 과정이 필요하다.

접근로그의 정제과정은 원시 데이터로부터 불필요한 항목을 필터링하는 과정이다. 웹 로그는 HTTP 프로토콜의 연결속성 때문에 해당 웹 문서뿐만 아니라 해당 웹 문서와 관련된 이미지파일, 미디어파일 등 사용자의 행위와 관련된 모든 파일들이 웹 로그에 기록된다. 데이터 정제는 사용자의 요청과 직접적인 관련이 없거나 분석에 있어서 불필요한 이미지파일과 미디어파일 등을 제거하는 작업으로, 요청된 페이지에 포함된 이미지나 비디오 클립 등이 이에 해당될 수 있다. 이처럼 불필요한 페이지에 대한 필터링은 웹 로그의 요청된 파일에 대한 확장자를 통해 판별할 수 있다. 그림 5는 이런 과정을 거쳐 정제된 접근로그의 예를 보여주고 있다.

```
210.114.57.238 - - [21/Sep/2006:11:34:39 +0900]
 "GET /index.html HTTP/1.1" 200 2363
210.114.57.238 - - [21/Sep/2006:11:34:40 +0900]
 "GET /bottom.html HTTP/1.1" 200 483
210.114.57.238 - - [21/Sep/2006:11:34:40 +0900]
 "GET /main.php HTTP/1.1" 200 2540
210.114.57.238 - - [21/Sep/2006:11:34:42 +0900]
 "GET /company1.html HTTP/1.1" 200 1637
210.114.57.238 - - [21/Sep/2006:11:36:59 +0900]
 "GET /aro.html HTTP/1.1" 200 2134
220.93.55.27 - - [21/Sep/2006:11:35:09 +0900]
 "GET /index.html HTTP/1.1" 200 2363
220.93.55.27 - - [21/Sep/2006:11:35:10 +0900]
 "GET /bottom.html HTTP/1.1" 200 483
220.93.55.27 - - [21/Sep/2006:11:35:10 +0900]
 "GET /main.php HTTP/1.1" 200 245
220.93.55.27 - - [21/Sep/2006:11:35:17 +0900]
 "GET /h_pressure.html HTTP/1.1" 200 2206
220.93.55.27 - - [21/Sep/2006:11:35:18 +0900]
 "GET /h_pressure_01.html HTTP/1.1" 200 2907
220.93.55.27 - - [21/Sep/2006:11:35:18 +0900]
 "GET /h_pressure_01a.html HTTP/1.1" 200 1232
220.93.55.27 - - [21/Sep/2006:11:37:22 +0900]
 "GET /aro.html HTTP/1.1" 200 2134
220.93.55.27 - - [21/Sep/2006:11:37:25 +0900]
 "GET /aro_01.html HTTP/1.1" 200 1226
220.93.55.27 - - [21/Sep/2006:11:37:25 +0900]
 "GET /arno_01b.html HTTP/1.1" 200 4071
24.69.255.237 - - [21/Sep/2006:12:22:34 +0900]
 "GET /wilden.html HTTP/1.1" 200 2126
24.69.255.237 - - [21/Sep/2006:12:22:40 +0900]
 "GET /wilden_01.html HTTP/1.1" 200 1376
24.69.255.237 - - [21/Sep/2006:12:22:41 +0900]
 "GET /wilden_01b.html HTTP/1.1" 200 4506
24.69.255.237 - - [21/Sep/2006:12:25:22 +0900]
 "GET /wilden_02.html HTTP/1.1" 200 1376
24.69.255.237 - - [21/Sep/2006:12:25:22 +0900]
 "GET /wilden_02b.html HTTP/1.1" 200 4491
```

그림 5. 정제된 접근로그의 예
Fig. 5 An example of access log

다음으로 사용자 및 세션 구분이 이루어져야 한다. 대부분의 웹 사용자들은 개인정보 유출 등의 문제로 인해 자신들의 익명성을 원하기 때문에 인증이 꼭 필요한 곳에서만 인증 절차를 거치게 된다. 또한 표준 웹 로그 데이터만으로는 사용자를 완전히 구분할 수 없으므로 정확한 사용자 구분을 위해 고유의 세션 ID를 부여한다. 세션 ID는 접속시간과 사용자 IP 주소를 구분하여 총 30자리로 설계하였다. 1-14번째 자리까지는 년 4자리, 월 2자리, 일 2자리, 시 2자리, 분 2자리, 초 2자리로 이루어져 있다. 15번째 자리는 구분자, 16-30번째 자리는 사용자의 IP 주소로 구성되어 있다. 세션은 사용자가 접속을 시작한 시점부터 접속을 종료한 시점까지를 말한다. 세션이 종료되기 전까지는 동일한 사용자에 의한 레코드로 처리하지만 사용자가 일정한 시간동안 아무런 응답이 없을 경우 새로운 세션 ID를 부여하고 다른 사용자로 구분하여야 한다.

3.3 클릭스트림 수집

클릭스트림은 웹 사이트에서 사용자의 이동경로를 분석할 수 있는 모든 행위 패턴정보를 말한다. 즉, 사용자가 웹사이트에 접속한 순간부터 웹 문서를 이동하기 위해 마우스를 이용하여 클릭 이벤트를 발생시키는 순차적인 흐름의 정보이다. 이런 클릭스트림을 수집하기 위해서는 접근로그의 정제 과정을 통해 세션별로 웹 문서 형식의 하이퍼링크를 수집한다. 클릭스트림은 웹 문서 이동경로와 문서 당 머문 시간으로 구분하고, 문서 이동 경로는 동일한 IP를 묶고 요청시간별로 정렬하여 해당 IP가 접근한 문서들의 이동 경로를 추출한다. 표 3은 그림 5의 정제된 접근로그 데이터를 통해 수집된 세션별 클릭스트림이다.

표 3. 세션별 클릭스트림
Table. 3 Click-streams of sessions

세션	세션 ID	클릭스트림
1	20060921113439_210.114.57.238	index.html
	20060921113440_210.114.57.238	bottom.html
	20060921113440_210.114.57.238	main.php
	20060921113442_210.114.57.238	company1.html
	20060921113659_210.114.57.238	aro.html
2	20060921113509_220.93.55.27	index.html
	20060921113510_220.93.55.27	bottom.html
	20060921113510_220.93.55.27	main.php
	20060921113517_220.93.55.27	h_pressure.html
3	20060921113518_220.93.55.27	h_pressure_01.html
	20060921113518_220.93.55.27	h_pressure_01a.html
	20060921113722_220.93.55.27	aro.html
	20060921113725_220.93.55.27	aro_01.html
	20060921113725_220.93.55.27	aro_01b.html
	20060921122234_24.69.255.237	wilden.html
	20060921122240_24.69.255.237	wilden_01.html
	20060921122241_24.69.255.237	wilden_01b.html
	20060921122522_24.69.255.237	wilden_02.html
	20060921122522_24.69.255.237	wilden_02b.html

3.4 숨겨진 하이퍼링크 탐색

실제 존재하지만 웹 문서의 하이퍼링크 추출을 통해서 발견하지 못한 하이퍼링크는 접근로그의 세션별 클릭스트림을 이용하여 탐색하게 된다. 이때 클릭스트림에 있는 문서들의 순서는 브라우저의 '뒤로' 버튼 사용으로 인해 정확한 하이퍼링크를 추출하는데 어려움이 있다. 이는 사용자의 '뒤로' 버튼 사용이 웹 서버의 접근로그에는 기록되지 않기 때문이다. 이를 해결하기 위한 방법으로 본 논문에서는 웹 문서 하이퍼링크 추출을 통해 생성된 그래프와 사용자의 클릭스트림을 이용하여 '뒤로' 버튼을 인지하고, 숨겨진 하이퍼링크를 탐색하여 문서와 하이퍼링크를 추가하는 알고리즘을 제안한다.

그림 6은 숨겨진 하이퍼링크를 탐색하는 알고리즘이다. 접근로그를 통한 문서와 하이퍼링크의 추가 방법은 클릭스트림을 큐에 순서대로 먼저 삽입(enqueue)하고 큐에 들어간 클릭스트림의 헤드를 삭제(dequeue)하여 스택에 삽입(push)한다. 큐의 헤드(head)와 스택의 탑(top)을 그래프와 비교한다. 동일한 간선이 있을 경우 큐를 삭제하고 스택에 삽입한다. 다음 큐의 헤드와 스택의 탑을 비교한다. 만약 동일한 간선이 없을 경우 스택에서 삭제(pop)하고 큐의 헤드와 스택의 탑을 다시 비교한다. 이런 과정을 새로운 정점이 나타날 때까지 반복한다. 새로운 정점이 나타나면 이를 해쉬 함수(hash function)를 이용하여 새로운 정점에 대한 패턴후보를 저장하고, 발생횟수를 증가시킨다. 패턴후보 중 짧은 경로이면서 발생횟수가 가장 많은 정점에 새로운 간선을 추가한다. 그림 6에서 *EdgeLengthPass()*는 새로운 정점에 대해 패턴후보의 길이가 짧은 간선을 찾는 함수이며, *EdgeMax()*는 새로운 정점에 대해 패턴후보 중 발생횟수가 많은 간선을 찾는 함수이다. *updateGraph()*는 새로운 정점과 간선

을 그래프에 추가하는 함수이다.

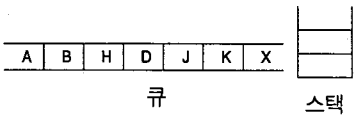
```
Algorithm SearchHiddenLinks
 input:   Click-stream Set CSet, Graph G=(V, E)
 output: Updated Graph G=(V, E)

SearchHiddenLink(CSet, G) {
 Queue Q;
 Stack S;
 HashSet NewEdgeSet {
  Stack s;
  Vertex nv;
  int count;
 }
 foreach c in CSet {
  Q.EnQueue(c); // 클릭스트림의 정점을 큐에 삽입
  S.push(Q.DeQueue());
  while (not Q.empty()) {
   V1 = S.top();
   V2 = Q.DeQueue();
   if (<V1 V2> ∈ E ) {
    S.push(V2);
   } else {
    if (V2 ∉ V) {
     V.addVertex(V2);
     // 새로운 정점, 패턴후보
     NewEdgeSet.hash(S, V2)++;
     break;
    } else {
     S.pop();
    }
   }
  }
  S.init();  // 스택 초기화
  Q.init();  // 큐 초기화
 }
 foreach e in NewEdgeSet {
  // e.nv에 대해 경로의 길이가 가장 짧은 간선 선택
  if (e.nv is EdgeLengthPass(e.s)) {
   // e.nv에 대해 가장 빈발한 간선 선택
   if (e.nv is EdgeMax(s.count))
    updateGraph(G<s.top(), e.nv>); // 그래프 추가
  }
 }
}
```

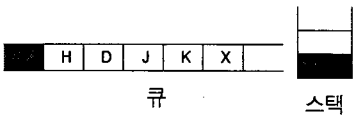
그림 6. 숨겨진 하이퍼링크 탐색 알고리즘
Fig. 6 Algorithm for searching hidden hyperlinks

예를 들어, 하나의 세션에 (A, B, H, D, J, K, X) 클릭 스트림이 있다면, 다음의 순서로 그림 5의 그래프를 보완한다.

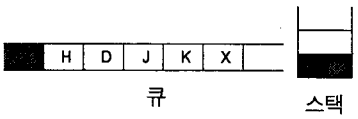
① 클릭스트림을 큐에 삽입한다.

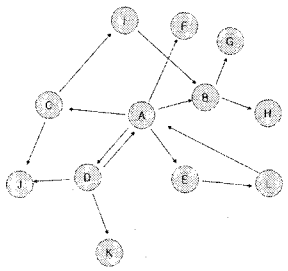


② 클릭스트림이 저장된 큐에서 헤드를 삭제하여 스택에 삽입한다.



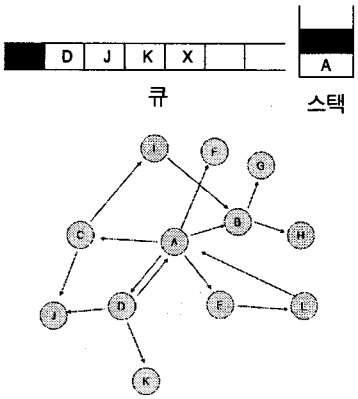
③ 큐의 헤드(B)와 스택의 탑(A)을 그림 5의 그래프와 비교하여 간선 유무를 확인한다.





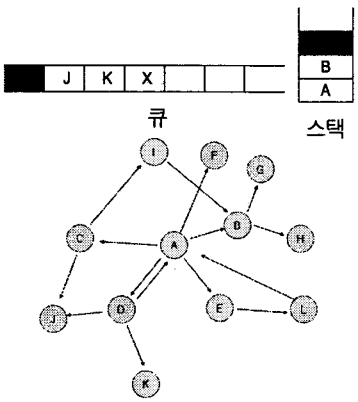
그래프

④ 간선이 존재하면 큐에서 삭제 후 스택에 삽입한다. A→B, B→H 간선이 존재하므로 큐에서 삭제하여 스택에 삽입한다.

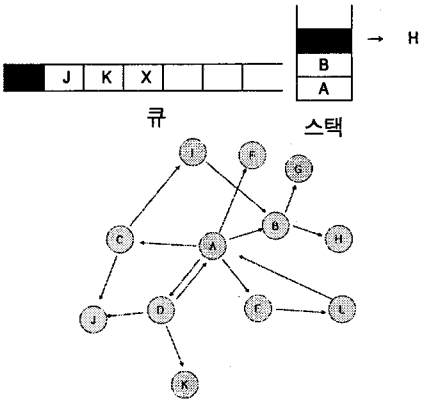


그래프

⑤ 간선이 존재하지 않을 때까지 ③ ~ ④ 과정을 반복한다. 만약 그래프와 비교하여 간선이 존재하지 않으면 스택에서 삭제한다. 예를 들어, H→D 간선은 존재하지 않으므로 H를 스택에서 삭제한다. 이러한 경우는 '뒤로' 버튼을 클릭한 상태로 인식한다.

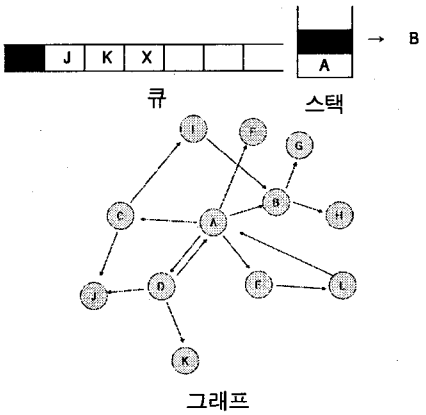


그래프



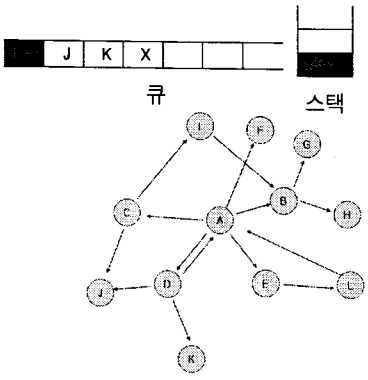
그래프

⑥ 새로운 정점이 나타날 때까지 위의 과정을 반복하고, 만약 큐가 비워질 때까지 새로운 정점이 나타나지 않는다면 새로운 정점이 없으므로 다음 세션 클릭스트림을 검사한다.

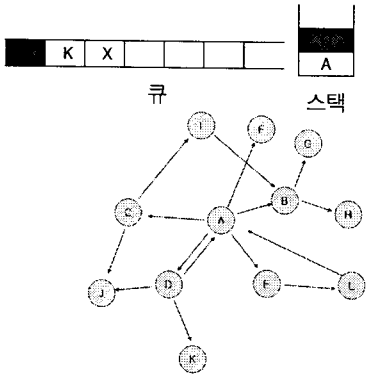


그래프

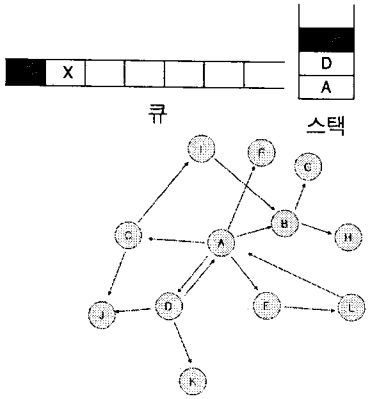
⑦ 새로운 정점이 탐색 되었을 때 새로운 정점과 스택의 내용을 해싱함수를 이용하여 저장하고 빈발수를 1 증가한다. 큐와 스택을 초기화한다.



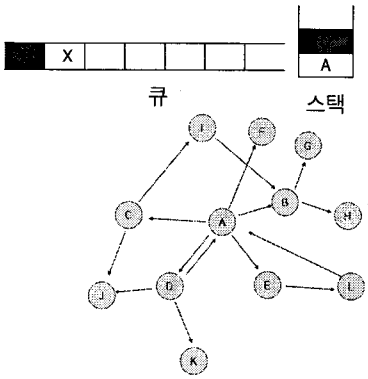
그래프



그래프

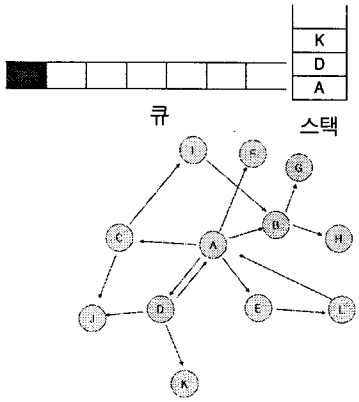


그래프



그래프

⑧ 위의 과정을 반복하여 세션별 클릭스트림을 모두 검사한다.



그래프

⑨ 위의 예제에서는 새로운 정점 X가 탐색되었을 때 얻은 패턴 후보는 A, D, K가 된다.

표 4는 10개의 세션 클릭스트림으로 알고리즘을 적용하였을 때 새로운 정점 X에 대한 세션별 패턴후보이다. 새로운 정점에 대한 패턴후보의 간선 추가는 패턴후보의 길이가 짧으면서, 발생 횟수가 가장 많은 정점에 새로운 간선을 추가한다. 그 이유는 사용자는 새로운 정점으로 이동하기위해 짧은 경로를 선택하는 사람이 있으며, 짧은 경로 중 빈발하게 사용되는 정점에 간선을 추가함으로써 뒤로 버튼을 최소화한 결과가 될 것이다. 실험에서 많은 클릭스트림을 사용하였을 때 새로운 정점에 대

해 신뢰할 수 있는 간선을 100% 찾을 수 있었으며, 탐색된 정점에 여러 간선이 존재할 경우 오류는 발생하지만 그중에 신뢰할 수 있는 간선은 꼭 존재함을 알 수 있었다. 위의 예제에서 새로 탐색된 정점에 대해서 패턴후보의 크기가 가장 짧은 정점 즉, 짧은 경로(A→D, A→F)이면서, 발생 횟수가 가장 많은 (A→D) 후보의 경로에 새로운 정점 X와 간선을 추가한다. 그림 8은 표 4에서 새로운 정점 X를 패턴후보 중 짧은 경로인 A→D의 정점 D에 새로운 간선 D→X를 추가한 결과이다.

표 4. 새로운 정점 탐색
Table. 4 Searching new vertices

세션	클릭스트림	패턴후보	새로운 정점
1	ABHDJKX	ADK	X
2	ACJDELDX	AD	X
3	ABDX	AD	X
4	ADX	AD	X
5	AFBHEDX	AD	X
6	ACJDELADX	AELAD	X
7	ADAX	ADA	X
8	AFBDKX	ADK	X
9	ABHGF	AF	X
10	ADKLX	∅	X

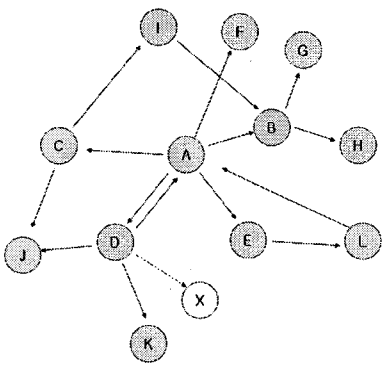


그림 8. 결과 그래프
Fig. 8 Result graph

Ⅳ. 실험 및 결과

본 논문에서 제안하는 알고리즘은 보안 문제로 인해 웹 로그를 수집할 수 없어 실제 웹 사이트를 대상으로 수행하기에는 어려움이 있다. 따라서 가상의 웹 사이트 구조를 임의로 생성한 후 몇몇 정점과 간선을 숨긴 상태에서 클릭스트림을 생성하여 실험하였다.

4.1 실험환경 및 평가기준

실험은 Intel Pentium Ⅳ 2.4G CPU와 1G 메인메모리, Microsoft Windows XP Professional을 운영체제로 하는 개인용 컴퓨터를 기반으로 수행하였다. DBMS는 Mysql, 프로그래밍 언어는 Java를 사용하였다.

웹 사이트를 모형화한 그래프는 100개의 정점과 1,787개의 간선을 갖는 방향성이 있는 그래프로 생성하였다. 접근로그에서 추출되는 클릭스트림은 앞에서 만든 방향그래프를 탐색하면서 사용자들이 웹을 항해하는 것과 같이 '뒤로' 버튼을 랜덤하게 추가하여 만들었으며, 총 50,000개의 클릭스트림 패턴을 사용하여 실험하였다.

실험에 대한 평가는 식 1의 정확률(precision), 식 2의 재현률(recall), 식 3의 정확도(accuracy), 식 4의 오류율(error)에 대하여 평가하였다. 표 5의 평가 분할표는 실제 그래프의 정점 및 간선과 본 논문의 알고리즘을 통해 찾은 그래프의 정점 및 간선을 의미한다.

표 5. 평가 분할표
Table 5. Contingency table for evaluation

찾은 구조 / 실제 구조	존재	비존재
존재	A	B
비존재	C	D

정확률(precision)은 본 논문에서 제안한 알고리즘을 통해 찾은 정점과 간선의 범주에서 실제 그래프와 일치하는 비율이며 식 1과 같다.

$$\text{정확률} : \frac{A}{A+B} \quad (1)$$

재현률(recall)은 실제 그래프의 정점과 간선 범주에서 찾은 그래프의 정점과 간선이 일치하는 비율을 나타내며 식 2와 같다.

재현률 : $\frac{A}{A+C}$ (2)

정확도(accuracy)는 전체 그래프에서 일치하게 분류된 정점과 간선의 수에 대한 비율이며 식 3과 같다.

정확도 : $\frac{A+D}{A+B+C+D}$ (3)

오류율(error)은 전체 그래프에서 잘못 분류된 정점과 간선의 수에 대한 비율이며 식 4와 같다.

오류율 : $\frac{B+C}{A+B+C+D}$ (4)

4.2 실험 및 결과

본 논문에서는 숨겨진 정점 및 신뢰성 있는 간선을 찾는 실험과, 숨겨진 정점 및 모든 간선을 찾는 실험을 수행하였다.

실험 1: 표 6은 숨겨진 정점 및 신뢰성 있는 간선을 탐색한 결과이다. 이 실험에서 사용한 실제 그래프의 정점은 100개이고 이중 숨겨진 정점은 20개이다. 본 논문에서 제안하는 알고리즘을 수행한 결과로 숨겨진 정점 20개는 모두 찾을 수 있었다. 그리고 실제 그래프의 간선은 1,787개이고 이들 중 숨겨진 간선은 358개이다. 알고리즘을 통해 찾아낸 숨겨진 간선은 100개이고 그중 일치하는 숨겨진 간선은 97개를 찾을 수 있었다. 숨겨진 간선은 하나의 정점에 여러 간선이 있을 경우 모든 간선을 찾는 것이 아니라 신뢰성이 보장된 최소 하나의 간선을 찾는다. 모든 간선을 찾지 못하는 이유는 정점을 기준으로 신뢰성 있는 간선 즉, 빈발도가 높으면서 패턴경로가 짧은 간선에 숨겨진 간선을 추가하기 때문에 전체 정점의 개수보다 많은 간선은 찾을 수 없다. 그리고 탐색한 숨겨진 정점에 대해서는 신뢰할 수 있는 간선을 찾을 수 있었다.

표 6. 실험 결과 1
Table 6. Experimental result 1

	정점	간선
실제	100	1,787
갱신	80	1,492
히든	20	358
추가	20	100
일치	20	97
불일치	0	3
정확률	1.00	0.97
재현률	1.00	0.27
정확도	1.00	0.27
오류률	0.00	0.58

실험 결과를 통해서 알 수 있듯이, 숨겨진 정점 20개 모두를 정확하게 찾았기 때문에 정점에 대한 정확률, 재현률, 정확도는 모두 1.0이다. 간선에 대한 정확률은 찾아낸 간선 100개 중 97개가 실제 숨겨진 간선과 일치하기 때문에 0.97로서 높은 수치를 보였다. 하지만 재현률과 정확도는 숨겨진 간선 358개 중 97개만을 찾았기 때문에 0.27로 낮은 비율을 보였다.

실험 2: 표 7은 숨겨진 간선을 모두 찾을 때까지 알고리즘을 반복하여 수행한 결과이다. 그러나 일치하는 모든 간선은 찾을 수 있지만 불일치하는 간선도 같이 찾을 수 있었다. 본 논문에서는 새로운 정점으로 이동하기위해 패턴후보 길이가 짧으면서 발생횟수가 많은 간선에 정점을 추가하기 때문에 만약 일치하지 않는 간선이 새롭게 만들어지면 그 간선으로 인해 새로운 정점까지의 최단 간선이 만들어질 수 있는 문제점이 생긴다. 표 7의 결과에서는 아홉 번째의 단계에서 모든 간선을 찾을 수 있었고, 열 번째 단계에서는 숨겨진 간선을 새롭게 탐색하지 못하였기 때문에 반복 수행을 멈추었다.

표 7. 실험 결과 2
Table 7. Experimental result 2

단계	간선/정점	실제	갱신	히든	추가	일치	불일치	정확률	재현률	정확도	오류율
1	간선	1787	1429	358	100	97	3	0.97	0.27	0.27	0.58
	정점	100	80	20	20	20	0	1.00	1.00	1.00	0.00
2	간선	1787	1529	358	94	89	5	0.96	0.52	0.51	0.33
	정점	100	80	20	0	0	0	1.00	1.00	1.00	0.00
3	간선	1787	1623	358	78	72	6	0.95	0.72	0.69	0.18
	정점	100	80	20	0	0	0	1.00	1.00	1.00	0.00
4	간선	1787	1701	358	69	51	18	0.91	0.86	0.79	0.12
	정점	100	80	20	0	0	0	1.00	1.00	1.00	0.00
5	간선	1787	1770	358	72	23	49	0.80	0.93	0.76	0.14
	정점	100	80	20	0	0	0	1.00	1.00	1.00	0.00
6	간선	1787	1842	358	79	15	64	0.71	0.97	0.69	0.18
	정점	100	80	20	0	0	0	1.00	1.00	1.00	0.00
7	간선	1787	1921	358	50	7	43	0.65	0.99	0.65	0.21
	정점	100	80	20	0	0	0	1.00	1.00	1.00	0.00
8	간선	1787	1971	358	22	3	19	0.63	1.00	0.63	0.23
	정점	100	80	20	0	0	0	1.00	1.00	1.00	0.00
9	간선	1787	1993	358	7	1	6	0.63	1.00	0.63	0.23
	정점	100	80	20	0	0	0	1.00	1.00	1.00	0.00
10	간선	1787	2000	358	0	0	0	0.00	0.00	0.00	1.00
	정점	100	80	20	0	0	0	1.00	1.00	1.00	0.00
평균	간선							0.80	0.81	0.62	0.24
	정점							1.00	1.00	1.00	0.00

실험 결괴에서 정확률은 정답 범주일거라고 예측된 간선이 실제로 정답일 확률이다. 1단계에서는 97%의 비율을 나타내지만 4단계 이후는 급격히 정확률이 떨어졌다. 이유는 일치하지 않는 간선의 추가로 인해 새로운 정점까지의 최단 간선이 만들어져 불일치하는 간선이 많은 추가되기 때문이다.

재현률은 정답 범주에 속하는 링크가 정답으로 분류될 확률이다. 1단계에서는 정점을 기준으로 찾을 수 있는 간선의 수가 100개 이하이기 때문에 358개를 모두 찾지 못하고 단계별로 추가를 하며 찾아간다. 4단계까지는 빠르게 증가하지만 4단계 이후는 일치하지 않는 간선들의 잡음으로 인해 불일치하는 간선이 많아져 낮은 증가율을 보이고, 9단계에서는 모든 숨겨진 간선을 찾은 결과를 보여준다.

정확도는 전체 실험 데이터에서 바르게 분류된 데이터 개수의 비율이다. 1단계에서는 정점을 기준으로 찾을 수 있는 간선의 수가 100개 이하이기 때문에 358개를 모두 찾지 못하므로 낮은 정확도를 보여주고 있다. 4단계까지는 신뢰성 있는 간선의 추가로 정확도가 빠르게 높아지지만, 4단계 이후부터는 일치하지 않은 간선들의 잡음으로 정확도가 줄어드는 것을 알 수 있다.

오류율은 전체 실험 데이터에서 잘못 분류된 데이터 개수의 비율이다. 1단계에서는 정점을 기준으로 찾을 수 있는 간선의 수가 100개 이하이기 때문에 358개를 모두 찾지 못한다. 그래서 많은 오류를 나타내고 있다. 4단계까지는 신뢰성 있는 간선의 추가로 오류가 빠르게 줄어들지만, 4단계 이후부터는 일치하지 않은 간선들을 찾음으로 조금씩 오류가 늘어남을 알 수 있다.

결과적으로 모든 간선을 찾을 때까지 숨겨진 링크 탐색 알고리즘을 반복 수행하는 것보다 신뢰할 수 있는 구간을 4단계까지로 하여 알고리즘을 반복 수행하는 것이 좀 더 신뢰할 수 있는 결과를 얻을 수 있다.

V. 결론 및 향후 과제

본 논문에서는 웹 사이트의 정확한 구조를 추출하기 위하여 웹 문서뿐만 아니라 웹 접근로그를 사용하는 알고리즘을 제시하였다. 먼저, 웹 문서의 태그분석을 통해 하이퍼링크를 추출하여 그래프로 표현한다. 다음으로, 웹 접근로그를 이용하여 미처 추출하지 못한 하이퍼링크를 찾아내고 새로운 정점과 간선을 그래프에 추가하는 방법을 제시하였다. 결과적으로 웹 문서의 태그 분석이라는 기존의 방식을 개선시켜 숨겨진 하이퍼링크를 찾아 추가하는 것이 가능하다. 이를 통해 다양한 웹 구조 개선 및 웹 마이닝을 위한 핵심 자료로 활용될 뿐 아니라 쉽고 빠른 검색을 사용자들에게 제공할 수 있으며 웹 문서의 효율적인 개발과 유지보수를 가능하게 할 것이다.

하지만, 접근로그를 통해 새로운 간선을 추가하거나 정점 하나에 여러 간선이 존재할 경우, 하나의 간선에 대한 경로 탐색은 신뢰성을 가지지만 그 이상의 간선에 대해서는 오류율이 증가하는 문제점이 발생했다. 향후에는 이 오류를 줄이기 위해 웹 구조에 대한 사용자별 패턴을 분석하는 연구가 필요할 것이다.

참고문헌

[1] Y. Zou and K. Kontogiannis, "Migrating and Specifying Services for Web Integration", Lecture Notes In Computer Science, Vol. 1999, pp. 253-270, 2000.

[2] M. Baglioni, U. Ferrara, A. Romei, S. Ruggieri and F. Turini, "Preprocessing and Mining Web Log Data for Web Personalization", Lecture Note in Computer Science, Vol. 2829, pp. 237-249, 2003.

[3] Y. Kosala and H. Blockeel, "Web Mining Research, A Survey", Newsletter of the Special Interest Group on Knowledge Discovery & Data Mining, Vol. 2, pp. 1-15, 2000.

[4] J. Huysmans, B. Baesens and J. Vanthienen, "Web Usage Mining: A Practical Study", Proceedings of the Twelfth Conference on Knowledge Acquisition and Management, pp. 86-99, 2004.

[5] R. Cooley, B. Mobasher and J. Srivastava, "Data Preparation for Mining World Wide Web Browsing Patterns", Knowledge and Information System, Vol. 1, pp. 1-26, 1999.

[6] M. S. Chen, "Efficient Data Mining for Path Traversal Pattern", IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No. 2, pp. 209-221, 1996.

[7] S. Chakrabarti, Mining the Web, Morgan Kaufmann, 2002.

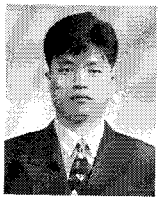
[8] Thuraisingham and M. Bhavani, Web Data Mining and Business Intelligence Analysis, CRC Press, 2003.

[9] 김진수, 김태용, 최준혁, 임기욱, 이정현, "사용자 로그 분석과 클러스터 내의 문서 유사도를 이용한 동적 추천 시스템", 한국정보과학회 정보과학논문지, 소프트웨어 및 응용 제31권 제5호, pp. 586-594, 2004.

[10] M. Koutri, N. Avouris and S. Daskalaki, "A Survey on Web Usage Mining Techniques for Web-Based Adaptive Hypermedia System", Information Resources Management Association, Vol. 13, pp. 125-149, 2005.

[11] 최영환, 이상용, "웹 마이닝을 위한 입력 데이터의 전처리과정에서 사용자구분과 세션보정", 한국정보과학회 정보과학회논문지, 소프트웨어 및 응용, 제30권 제9.10호, pp. 843-849, 2003.

저자소개



이 성 대(Seong-Dae Lee)

1999 한국해양대학교 컴퓨터공학과 (공학사)
2001 한국해양대학교 컴퓨터공학과 (공학석사)
2007 한국해양대학교 컴퓨터공학과 (공학박사)
1995~1996 미래정보CIM
2007~현재 한국해양대학교 산학협력단 전임연구원
※관심분야 : 데이터베이스, 데이터마이닝, 해양정보시스템, XML

박 휴 찬(Hyu-Chan Park)

1985 서울대학교 전자공학과 (공학사)
1987 한국과학기술원 전기및전자공학과 (공학석사)
1995 한국과학기술원 전기및전자공학과 (공학박사)
1987~1990 금성반도체
1997~현재 한국해양대학교 부교수
※관심분야 : 데이터베이스, 해양정보시스템, 데이터마이닝, XML