웹 마이닝 시스템을 위한 페이지 로깅 시스템

윤 선 희† · 오 해 석††

요 약

웹은 그 양적인 면이나 복잡도에 있어 현재 놀라운 속도로 급성장하고 있다. 이와 함께 웹사이트 설계나 웹서버 설계와 같은 작업은 물론 단순히 웹사이트를 검색하는 작업에 있어서도 그 복잡도가 크게 증가했다. 이러한 설계 작업들에 있어서 중요한 입력 요소로는 웹사이트가 어떻게 사용되고 있는가에 대한 정확한 데이터가 필수적으로 요구된다. 본 연구에서는 웹 마이닝 시스템에서 요구하는 사용자의 웹페이지 이용 정보 즉 사용자 세션(user sessions)을 효과적으로 획득할 수 있는 '페이지 로깅 시스템(Page Logging System : PLS)'을 제안한다. 페이지 로깅 시스템은 사용자의 웹페이지 탐색 정보를 획득하는 페이지 로거(page logger)와 획득한 데이터를 이용하여 사용자 세션 파일을 생성하는 로그 처리기(log processor), 그리고 웹사이트의 HTML 페이지에 페이지 로거 애플릿을 삽입하는 코드로 구성된다. 제안한 PLS는 기존의 웹 마이닝 시스템에서 많은 시간과 비용을 수반했던 데이터 전처리 작업의 일부를 제거한다. 특히 사용자가 각 페이지를 탐색한 시간(access length)을 직접 획득함으로써 트랜잭션 구분 단계를 단순화시킨다. 또한 PLS는 기존의 웹서버 로그로부터 사용자 세션을 획득함에 있어 가장 문제가 되었던 로컬 캐쉬(local cache) 및 ISP가 제공하는 프록시 서버 사용으로 인하여 야기되는 문제 등을 해결한다.

Page Logging System for Web Mining Systems

Sun-Hee Yoon† · Hae-Seok Oh††

ABSTRACT

The Web continues to grow fast rate in both a large scale volume of traffic and the size and complexity of Web sites. Along with growth, the complexity of tasks such as Web site design, Web server design, and of navigating simply through a Web site have increased. An important input to these design tasks is the analysis of how a Web site is being used. This paper proposes a Page Logging System(PLS) identifying reliably user sessions required in Web mining systems. PLS consists of 'Page Logger' acquiring all the page accesses of the user, 'Log Processor' producing user sessions from these data, and statements to incorporate a call to the Page Logger applet. Proposed PLS abbreviates several preprocessing tasks which spends a lot of time and efforts that must be performed in Web mining systems. In particular, it simplifies the complexity of transaction identification phase through acquiring directly the amount of time a user stays on a page. Also PLS solves local cache hits and proxy IPs that create problems with identifying user sessions from Web server log.

키워드 : 웹 마이닝(Web Mining), 데이터 전처리, 사용자 세션(user session), 페이지 로깅 시스템(Page Logging System: PLS), 페이지 로거(page logger), 로그 처리기(log processor)

1. 서 론

웹은 현재 그 크기나 복잡도 면에서 급속도로 팽창하고 있다. 웹사이트 설계나 웹서버 설계 등이 그와 함께 복잡해졌고, 또한 사용자에 대한 데이터를 분석하는 일이 수작업으로는 불가능하게 되었다. 이와 같은 작업을 수행하는데 있어서 웹사이트가 어떻게 사용되고 있는가에 대한 분석이 중요한 입력으로 사용된다. 이러한 분석은 페이지 접근 빈도 수와 같은 간단한 통계치에서부터 웹사이트를 어떠한 순서로 방문하였나에 대한 정보에 이르기까지 다양하다[1].

대부분의 웹 마이닝 시스템들은 W3C에서 규정하고 있는 Common Log Format(CLF) 혹은 Extended Common Log Format(ECLF)[2] 형태의 로그 파일을 이용하여 마이닝 작업을 수행하게 된다. 물론 CLF 혹은 ECLF 로그로부터 획득한 데이터를 웹 마이닝 시스템의 입력으로 사용하기까지는 여러 가지 전처리 단계를 거쳐야 한다[3].

일반적으로 웹 마이닝 시스템의 효율성은 마이닝 알고리즘의 입력으로 사용하는 데이터의 정확도에 달려있다[4, 5]. 그러나 CLF 혹은 ECLF 로그만을 이용하는 경우에는 여러 단계의 전처리 과정을 거쳐야 함은 물론 부정확한 데이터를 얻을 수 있다. 요구되는 전처리 단계들로는 데이터 클리닝(data cleaning), 사용자구분(user identification), 세션구분(session identification), 패스완성(path completion), 트랜잭션 구분(transaction identification) 등이 있다[3]. 또한 부정확한 데이터를 얻게 되는 이유는 사용자측의 로컬 캐쉬(local cache)의 사용, 혹은 ISP가 제공하는 프록시 서버(proxy ser-

† 정 회 원 : 미림 전산고등학교 교사
†† 정 회 원 : 숭실대학교 컴퓨터학과 교수
논문접수 : 2001년 2월 6일, 심사완료 : 2001년 9월 4일

ver)의 사용으로 기인할 수 있다. 사용자 대부분이 로컬 캐쉬를 사용하기 때문에 한번 접근한 페이지를 그 세션에 다시 접근하는 경우는 그 페이지에 대한 기록이 로그 파일에 남지 않아 정확한 데이터를 얻을 수 없다. 또한 사용자가 프록시 서버를 사용하는 경우 사용자의 IP조차도 구분할 수 없는 경우가 발생한다[6, 7]. 특히 이 두 가지 문제는 웹 마이닝 시스템의 입력으로 사용되는 '사용자 세션(user session)'을 구분하는데 가장 장애가 되는 요소들이다[8]. 사용자 세션(user session)은 W3C We Characterization Activity (WCA)[9]에서 정의한 용어로 하나 이상의 웹 서버에 걸쳐서 사용자가 클릭하여 발생한 모든 로그들을 사용자별로 구분한 것을 의미한다.

위에서 언급한 문제들을 해결하기 위해 전처리 과정을 효과적으로 수행하기 위한 연구들[3, 10]과 로컬 캐쉬, 혹은 프록시 서버 사용 문제 등을 해결하기 위한 연구들[3, 8]이 제안되었다. 로컬 캐쉬 문제를 해결하기 위해 제안된 방법으로는 대부분 쿠키(cookies)를 이용하거나 캐쉬 버스팅(cache busting)을 사용하는 것이었다[3]. 쿠키는 사이트 방문객들을 자동으로 추적하는데 사용되는 마커(marker)이다. 캐쉬 버스팅은 페이지가 보여질 때마다 항상 새롭게 다운받아 보여지도록 하는 것이다. 그러나 Pitkow[8]에 잘 제시되어 있듯이 이러한 방법들의 대부분이 완벽하게 문제를 해결하지는 못한다. 쿠키는 사용자가 제거할 수 있으며, 캐쉬 버스팅 또한 캐싱에 의한 빠른 속도를 얻을 수 없기 때문에 사용자에 의해 쉽게 제거될 수 있다. 정확한 사용자 세션을 얻기 위한 또 다른 방법으로는 폼(form)을 이용하는 것으로 사용자로 하여금 사용자 정보를 등록하도록 하는 방법이 있다. 그러나 이 방법 역시 사용자의 의지에 의존하는 것으로 사용자가 개인 정보 유출을 우려하여 거짓 정보를 입력한다거나 등록하지 않는 경우에는 정확한 사용자 정보를 얻을 수 없다는 문제를 안고 있다.

본 연구에서는 앞서 언급한 정확한 사용자 세션을 획득하는데 걸림돌이 되는 로컬 캐쉬와 프록시 서버 사용 문제를 효과적으로 해결함은 물론 여러 단계를 거쳐 수행해야 하는 전처리 과정을 단순화시켜 주는 '페이지 로깅 시스템(Page Logging System : PLS)'을 제안한다. PLS는 웹사이트에 접근한 사용자의 정확한 로그 정보를 획득하기 위해 서버 측에 삽입된 자바기반 애플릿 프로그램이다.

PLS는 페이지 로거(Page Logger)와 로그 프로세서(Log Processor)로 구성되며, 추가로 웹사이트의 각 페이지에 자바 애플릿을 호출하는 문장을 삽입하는 코드가 요구된다. 물론 이 코드의 삽입은 Mod Layout[11]을 이용하면 자동으로 쉽게 삽입할 수 있다. 페이지 로거는 웹사이트의 각 페이지에 삽입된 애플릿 호출 문장이 실행되면 사용자(클라이언트) 브라우저에 내려가서 사용자의 정확한 정보와 접근한 페이지에서의 행동 정보를 획득하여 로그 프로세서에 전달한다. 이러한 정보들로는 ECLF에서 제공하는 모든 항목들 외에도 추가적으로 ISP에서 제공하는 프록시 서버 사용자의 정확한 IP 주소, 사용자가 각 페이지를 본 시간(viewing time), 로컬 캐쉬의 사용 유무에 관계없이 사용자가 "Back" 버튼을 눌렀을 때의 해당 페이지의 URL 등이 포함된다. 특히 ISP에서 제공하는 프록시 서버 사용자의 IP 주소도 인증에 답하면 정확하게 획득할 수 있다. 또한 자바 애플릿 호출 코드를 각 웹페이지의 끝 부분에 위치시킴으로써 실제로 사용자가 페이지 뷰(pageview)를 본 시간을 근사시킨다. 페이지 뷰는 W3C WCA에서 정의한 용어로 한 사용자가 한번의 마우스 클릭으로 보게 되는 페이지의 모든 파일을 의미한다. 로그 프로세서는 페이지 로거로부터 전달된 정보를 데이터베이스에 저장한다.

웹 마이닝 시스템에서 사용자에 관한 정확한 정보는 그 시스템의 성능 향상에 큰 영향을 미친다. 앞서 언급한 바와 같이 PLS는 사용자의 의지와는 무관하게 그 사용자에 관한 정확한 정보를 획득하게 된다. 이외에도 PLS는 다음과 같은 특징들을 가진다 : 1) 클라이언트 측의 브라우저를 전혀 수정하지 않아도 되고, 2) 현재의 HTTP 프로토콜을 그대로 유지하며, 3) 서버 측에서 애플릿 호출 코드 삽입을 완전히 자동으로 수행한다(이것은 대단위 웹사이트의 경우 큰 시간적 부담을 줄인다).

본 연구는 모두 5장으로 구성되어 있으며 그 내용은 다음과 같다. 2장에서는 사용자 세션을 획득하기 위하여 수행되는 여러 전처리 단계들에 대하여 살펴본다. 3장에서는 본 연구에서 제안한 페이지 로깅 시스템을 상세히 설명하고, 4장에서는 제안한 시스템의 구현 환경에 대하여 살펴본다. 마지막 5장에서는 결론 및 향후 연구 방향에 대하여 검토한다.

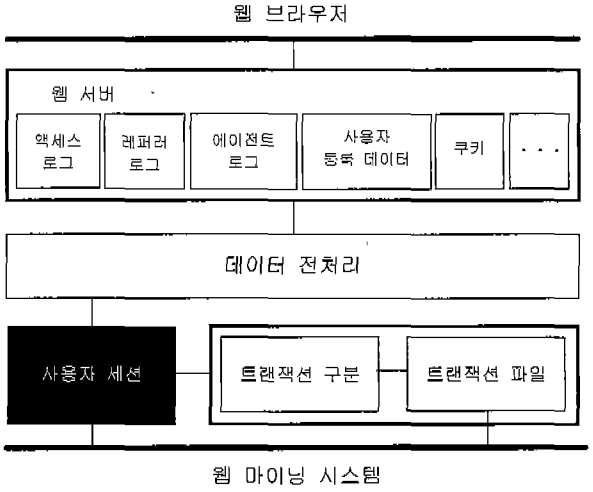
2. 사용자 세션 획득을 위한 전처리

일반적으로 웹 마이닝 시스템의 효율성은 그 입력 데이터의 정확도에 크게 영향을 받기 때문에 대부분 다음과 같은 전처리 과정을 거치게 된다 : 데이터 클리닝, 사용자 구분, 세션 구분, 패스 완성. 그러나 PLS 로그 정보는 이러한 전처리 단계의 일부를 제거 혹은 단순화시켜 준다. 전처리 대상으로는 웹서버 로그, 쿠키, 사용자 등록 데이터 등이 있고, 이 외에도 사이트 파일이나 사전 분석을 통한 웹 이용 통계 데이터 등이 있을 수 있다. 웹 마이닝 시스템에서는 이러한 모든 데이터를 통합·처리하여 사용자 세션을 생성하여 입력 데이터로 사용하게 된다.

ECLF 로그에 대한 전처리의 필요성으로는 다음과 같은 이유들이 있다 : 1) 웹서버 로그로부터 얻은 데이터의 경우에 한 페이지에 접근한 사용자가 그 페이지에서 본 모든 데이터를 각각의 항목으로 로그 파일에 기록하기 때문에 불필요한 정보까지도 포함하게 되는 것과, 2) 사용자 대부분이 로컬 캐쉬를 사용하기 때문에 한번 접근한 페이지를

그 세션에 다시 접근하는 경우는 그 페이지에 대한 기록이 로그 파일에 남지 않아 정확한 데이터를 얻을 수 없고, 3) 사용자가 프록시 서버를 사용하는 경우에는 사용자의 IP 주소를 구분할 수 없는 경우가 발생하게 되는 문제가 있다.

(그림 1)은 일반적인 웹 마이닝 시스템을 보여주는 것으로서, 웹 브라우저, 웹서버, 데이터 전처리, 사용자 세션, 트랜잭션 파일 및 웹 마이닝 시스템으로 구성된다. (그림 1)에서 마이닝 시스템의 목적이 연관 규칙(association rules)을 생성하는 것이라면 먼저 트랜잭션 구분 단계를 거쳐서 트랜잭션 파일을 얻어야 한다.



(그림 1) 일반적인 웹 마이닝 시스템

2.1 데이터 클리닝(Data Cleaning)

서로 관련이 없는 데이터가 웹 마이닝 시스템의 입력으로 사용될 경우 시스템의 성능에 커다란 영향을 주기 때문에 이를 제거하는 것은 매우 중요하다고 할 수 있다. 이것은 웹 마이닝 시스템뿐만 아니라 웹 분석 툴에서도 마찬가지이다. 웹 마이닝 시스템의 가장 주된 목적은 사용자의 행위를 파악하는 것이기 때문에 사용자가 명시적으로 요청하지 않은 파일 요청을 웹 마이닝 시스템의 입력으로 사용하는 것은 의미가 없다.

그러나 HTTP 프로토콜은 웹서버에 요청된 모든 파일에 대한 독립된 연결을 요구하기 때문에, 특정 페이지를 열기 위한 사용자의 요청은 HTML 파일뿐만 아니라 그래픽이나 스크립트 등도 각각의 독립된 항목으로 로그 파일에 기록된다. 대부분의 경우에 HTML 파일만이 유용하기 때문에 사용자 세션에는 이 항목만을 유지하게 된다.

따라서 사용자의 행위를 파악하는데 도움이 되지 못하는 항목들인 그래픽이나 스크립트에 대한 항목들을 로그 파일에서 제거해야 한다. 이러한 항목들을 제거하기 위한 한 가지 방법은 URL에서 .gif, jpg, GIF, JPG, map 등의 접미사를 가진 항목들을 제거하는 것이다. 또한 count.cgi와 같은 스크립트도 제거할 수 있다. 그러나 이 방법은 분석하고자 하는 서버의 형태에 따라 달라져야 한다. 예를 들어, 그래픽 저장소로 운영되는 웹사이트의 경우에 모든 그래픽에 관련된 항목들을 제거해 버리면 정확한 분석 결과를 얻을 수 없을 것이기 때문이다. 제안하는 PLS에서는 해당 페이지 뷰에 관한 로그 정보만을 기록하기 때문에 데이터 클리닝 단계가 불필요하다.

2.2 사용자 구분(User Identification)

이미 언급한 바와 같이 로그 파일로부터 사용자를 구분하는 작업은 로컬 캐쉬나 프록시 서버 사용 등으로 인하여 크게 복잡해진다. 사용자와의 상호작용에 의존하는 방법은 이러한 문제를 해결하는 가장 쉬운 방법일 것이다. 그러나 웹 로그에 근거한 방법들에도 사용자를 구분하는데 이용할 수 있는 휴리스틱(heuristics)이 있다. [12]에서는 IP 주소가 같은 경우라도 에이전트 로그(agent log)에 변화가 있다면 다른 사용자로 간주하는 방법을 사용한다. 사용자 구분을 위한 또 다른 휴리스틱으로는 각 사용자에 대한 브라우징 패스를 획득하기 위하여 엑세스 로그뿐만 아니라 레퍼러 로그(referrer log)와 사이트 구조(site topology)를 같이 이용하는 것이다. 즉 사용자가 탐색한 어떤 페이지로부터도 도달할 수 없는 페이지가 요청된 경우이면 같은 IP 주소를 가지는 다른 사용자로 가정한다.

그러나 위의 방법은 단지 사용자를 구분하기 위한 휴리스틱일 뿐이다. 만일 같은 종류의 시스템 상에서 같은 브라우저를 사용하는 같은 IP 주소를 가지는 2명의 사용자가 같은 웹페이지 집합을 탐색하고 있다면 한 명의 사용자로 혼동될 수 있다. 이와 반대로 2개의 다른 브라우저를 사용하거나 사이트 링크를 사용하지 않고 직접 URL을 입력하여 탐색한다면 이 경우는 여러 명의 사용자로 잘못 구분될 수 있다.

2.3 세션 구분(Session Identification)

세션 구분의 목적은 각 사용자의 페이지 탐색을 개별적인 세션으로 구분하는 것이다. 이것을 해결하기 위한 가장 쉬운 방법은 시간제한에 대한 임계값(timeout)을 두는 것이다. 만일 페이지 요청 사이의 시간이 정해진 임계값을 초과하면 그 사용자는 새로운 세션을 시작한 것으로 간주된다. 많은 상업적인 제품들은 30분을 임계값으로 사용한다.[13]은 실험적인 데이터에 근거하여 25.5분의 임계값을 얻었다. 일단 사이트 로그가 분석되면 특정 웹사이트에 적절한 임계값을 세션 구분 알고리즘에 다시 피드백할 수 있다.

2.4 패스완성(Path Completion)

유일한 사용자 세션을 구분함에 있어서 또 다른 문제는 엑세스 로그에 기록되지 않은 중요한 엑세스가 있는 가 하는 점이다. 어떤 페이지 요청이 사용자가 요청한 마지막 페이지에 직접 연결되지 않은 경우일 때 레퍼러 로그를 이용

하여 해결할 수 있다. 그 페이지가 사용자의 현재의 요청기록에 들어 있다면, 이것은 대부분의 브라우저에서 이용할 수 있는 "Back" 버튼을 누른 경우로 간주한다. 이미 한번 가져온 페이지는 브라우저의 캐쉬에 저장되었다가 다음 탐색 시에 캐쉬에 있는 내용이 다시 보여지기 때문에 서버의 엑세스 로그에는 기록되지 않는다. 레퍼러 로그가 명확하지 않을 때는 사이트 구조를 이용하여 누락된 페이지들을 사용자 세션 파일에 추가시킬 수는 있다. 그러나 문제는 추가된 각 페이지 탐색에 대한 탐색시간을 추정하기 위한 알고리즘이 요구된다는 점이다. PLS에서는 이 단계의 전처리 과정을 제거할 수 있다(3.5절 참조).

3. 제안한 페이지 로깅 시스템(PLS)

제안한 PLS는 사용자가 웹사이트의 각 웹페이지를 방문할 때마다 관련 정보를 수집하는 페이지 로거와 페이지 로거로부터 수집된 자료를 처리하여 정확한 사용자 세션 파일을 생성하는 로그 처리기, 그리고 각 웹페이지에 삽입되는 자바 애플릿 호출 코드로 구성된다.

3.1 구조

기존의 방법들에서는 사용자가 웹사이트에 접근하면 웹서버의 로그 파일에 방문 자료가 기록되고, 이 로그 파일의 자료를 이용하여 데이터 클리닝, 사용자구분, 세션구분, 패스완성과 같은 전처리 과정을 통하여 사용자 세션을 획득하게 된다. 본 연구에서 제안한 PLS는 기존의 방법에서 필수적으로 요구되었던 전처리 과정의 일부를 제거함은 물론 로컬 캐쉬 및 프록시 서버 사용으로 인한 문제점도 해결한다.

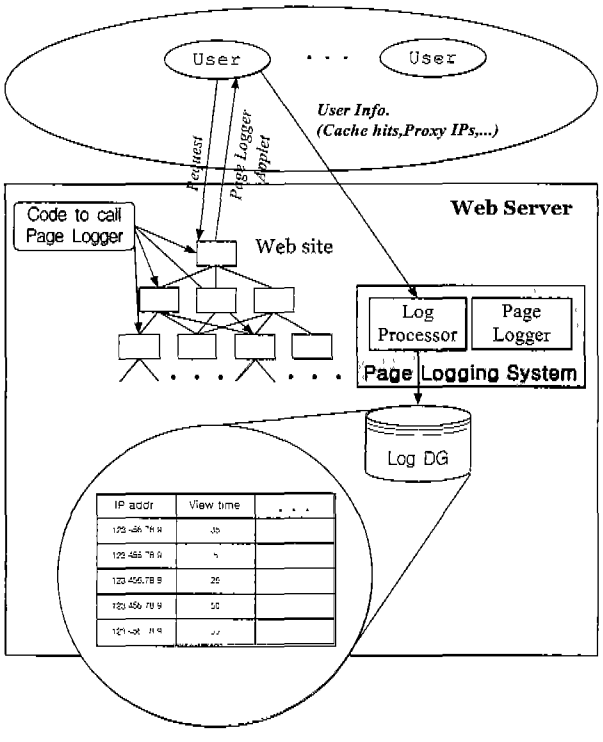
PLS는 처음부터 특정 페이지의 각각의 구성요소에 대한 접근을 기록하지 않음으로써 데이터 클리닝 단계를 제거하며(2~1절 참조), 각 페이지마다 페이지 로거가 존재하여 현재 탐색된 페이지가 새로운 것이든 캐쉬에 존재하던 것이든 항상 탐색될 때마다 하나의 로그 항목으로 기록되기 때문에 기존의 방법에서 여러 부수적인 정보(레퍼러 로그, 사이트 구조 등)를 근거로 하여 반드시 수행해야만 하던 패스완성 단계를 제거한다(2~4절 참조). 또한 연관 규칙 생성이 목적인 웹 마이닝 시스템에서 각 웹페이지의 실제 탐색시간(access length)을 직접 획득하기 때문에 트랜잭션 구분 과정을 단순화시켜 준다(3.5절 참조).

PLS는 서버 사이트가 아닌 클라이언트 측에서 사용자 정보를 획득하게 된다. 따라서 사용자에 관한 보다 더 정확한 정보를 얻을 수 있다. Cunha[14] 등도 클라이언트 측에서 보다 정확한 정보를 얻는 방법을 제안하였는데, 그들의 방법은 웹브라우저를 수정해야 하는 불편함을 가진다. 그러나 제안한 PLS는 클라이언트 측에서의 어떤 수정도 요구하지 않으며(브라우저를 포함하여), 사용자와의 어떤 협의도 요구하지 않는다. 또한 HTTP 프로토콜의 현재 상태에 대해 그 어떤 수정도 할 필요가 없다. 단지 서버 사이트의 웹페이지에 자바 애플릿 호출 코드를 삽입하면 되는데, 이것 또한 쉽게 자동으로 삽입할 수 있기 때문에 문제되지 않는다.

(그림 2)는 제안한 PLS의 구조를 보여준다. PLS는 페이지 로거, 로그 프로세서, 그리고 자바 애플릿 호출 코드로 구성된다. PLS가 동작되기 위해서는 먼저 웹사이트의 HTML 페이지들에 다음과 같은 애플릿 호출 코드가 삽입되어야 한다.

```
<applet code = "pageLogger.class" width = "1" height = "1"
                MAYSCRIPT>
<param name = "cookie" value = "userInfo">
</applet>
```

이 작업은 Mod Layout을 이용하면 자동으로 웹사이트의 모든 HTML 페이지에 삽입할 수 있다. Mod Layout은 SSI(Server Side Includes)를 이용하지 않고서도 웹사이트의 문서에 원하는 내용을 자동으로 삽입할 수 있는 프로그램이다.



(그림 2) PLS의 구조

PLS의 동작원리는 다음과 같다. 사용자가 웹사이트에 접근하면 해당 페이지의 애플릿 코드(페이지 로거)가 호출되어 클라이언트의 브라우저에 다운되어 사용자에 관한 정보를 획득하여 PLS의 로그 프로세서에 전달하게 된다. 전달되는 정보로는 CLF 혹은 ECLF 로그 정보뿐만 아니라 캐

쉬 히트(cache hits)(3~3절 참조), ISP가 제공하는 프록시 서버 사용자 IP(3~4절 참조) 등을 포함한다. 이러한 정보를 전달받은 로그 프로세서는 사용자가 실제로 해당 페이지를 본 시간(view time) 등을 계산하여 데이터베이스에 저장하게 된다. 이처럼 데이터베이스에 저장된 내용을 사용자 구분과 세션 구분을 수행하면 웹 마이닝 시스템의 목적에 따라 다양하게 포맷하여 사용할 수 있는 사용자 세션을 생성할 수 있다.

3.2 알고리즘

제안한 PLS의 수행절차는 다음과 같다. 사용자가 웹사이트의 한 페이지에 접근하면 페이지 로거가 클라이언트 브라우저에서 호출되어 사용자가 브라우저를 종료하거나 다른 페이지로 이동한 것인 가를 확인하여, 이동하지 않은 경우에는 해당 페이지를 본 시간을 계산하고, 사용자가 브라우저를 종료하거나 다른 페이지로 이동하면 클라이언트의 쿠키, 카운터 등 로깅에 필요한 정보들을 PLS의 로그 처리기에 전달한다. 로그 처리기에서는 클라이언트 측의 페이지 로거로부터 전달된 정보(cache hits, proxy IPs, viewing time,…)와 함께 CLF 혹은 ECLF 로그를 필터링하여 페이지 접근 정보 데이터베이스(Log DB)에 저장한다.

3.2.1 페이지 로거

페이지 로거는 웹사이트의 모든 페이지에 삽입되어 사용자가 각 페이지에 접근될 때마다 클라이언트의 브라우저에 로드(load)되어 사용자에 관한 정보를 획득하는 자바 애플릿이다. 페이지 로거는 CLF 혹은 ECLF 로그와 같이 서버 측에서 획득하는 것이 아니라 클라이언트 측에서 직접 획득하기 때문에 보다 정확한 사용자 정보를 얻을 수 있다는 장점을 가진다. 또한 현재의 HTTP 프로토콜이나 패킷 구조(packet structure)를 전혀 수정하지 않아도 되며, 단지 웹사이트의 모든 페이지에 페이지 로거 호출 코드만 삽입하면 된다. (그림 3)은 페이지 로거의 수행절차를 보여준다.

```
단계 1.  사용자가 웹페이지 접근
단계 2.  start() 실행
    2.1 cookies(user ID,…) 획득
    2.2 현재 시간 저장
단계 3.  현재 페이지를 떠나는 시점에서 stop() 실행
단계 4.  stop()에서
    4.1 단계 2.2에서 획득한 시간을 통해 viewing time을 계산
단계 5.  로그 프로세서 호출
    5.1 단계 2.1의 cookies와 단계 4의 viewing time 전달
단계 6.  페이지 로거 종료
```

(그림 3) 페이지 로거

3.2.2 로그 프로세서

로그 프로세서는 웹 환경변수를 통하여 CLF 혹은 ECLF 로그를 획득하고, 또한 페이지 로거로부터 전달된 쿠키, 로컬 IP, viewing time 등을 데이터베이스에 저장한다. (그림 4)는 로그 프로세서의 수행절차를 보여준다.

```
단계 1. 페이지 로거에 의한 호출
단계 2. 데이터베이스 open()
단계 3. 환경변수를 통한 CLF 혹은 ECLF 로그 획득 및 페이지
        로거로부터 획득된 정보에 대한 질의어 생성
단계 4. 질의어 실행 및 데이터베이스 테이블에 저장
단계 5. 데이터베이스 close()
```

(그림 4) 로그 프로세서

3.3 로컬 캐쉬

대부분의 클라이언트 브라우저는 최근에 접근되었던 페이지들을 캐쉬한다. 사용자가 이미 접근했던 페이지로 다시 되돌아가면 웹서버 로그에는 그 페이지에 대한 기록이 남지 않는다. 이러한 로컬 캐쉬에 대한 기록을 획득하기 위한 여러 방법들이 제안되었다.

가장 단순한 방법으로 HTML 페이지의 만료시간(expiration time)을 아주 짧게 지정하여 브라우저가 항상 서버로부터 모든 페이지를 검색하도록 하는 방법이다. HTTP 프로토콜의 *Expires* 헤더 필드를 사용하면 된다. 즉 이 필드의 값을 영(zero) 값으로 주거나 해당 페이지가 검색된 후 즉시 만료되도록 잘못된 날짜 포맷을 지정해주면 가능하다. 또 다른 방법으로 브라우저의 캐쉬 크기를 영(zero) 값으로 지정하는 방법이 있다. 그러나 이 방법은 사용자의 의지가 개입되어야 하며 강요할 수는 없다. 이 두 가지 방법의 단점은 속도 향상을 위해 사용되는 캐쉬의 기능을 활용하지 않음으로써 성능이 크게 떨어진다는 점이다.

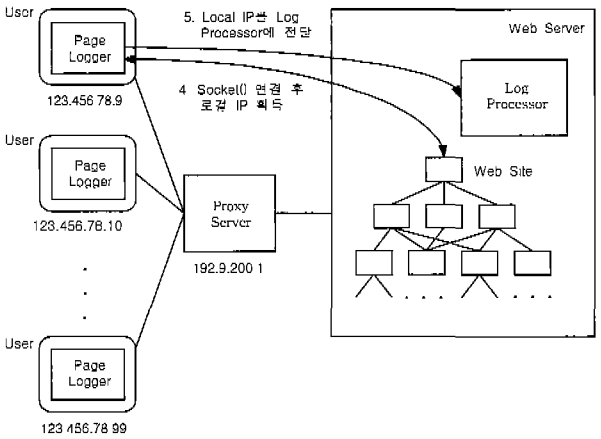
다른 부류의 해결 방법으로는 웹사이트의 구조(topology)를 이용하여 캐쉬 히트(cache hits)된 페이지를 찾아내는 방법이다. 이 방법은 패스의 링크들이 일관되어야 한다는 사실, 즉 각 링크의 마지막 페이지가 다음 링크의 시작 페이지와 같아야 한다는 가정에 근거한다. 그러나 이 방법 역시 여러 가지 상황들, 실제 웹사이트가 위의 가정에 위배되게 구성되어 있다든지(예를 들어, 한 페이지로 오는 링크가 여러 페이지로부터 가능한 경우), 혹은 사용자가 직접 URL을 이용하여 접근하는 경우 등에는 적용되지 않는다.

위의 모든 방법들과는 대조적으로 제안한 PLS는 간단하고 정확하게 캐쉬 히트를 찾아낸다. 즉 3.1절에서 언급한 바와 같이 웹사이트의 모든 웹페이지에 페이지 로거를 삽입하므로써 단순 명료하게 캐쉬 히트를 찾아낸다. 이 방법은 사용자가 어떤 루트를 통하여 해당 페이지에 접근한 것인지, 즉 링크를 통한 것인지, "Back" 버튼을 이용한 것인지, 아니면 직접 URL을 기입하여 접근한 것인지 등과는 무관하게 캐쉬 히트를 찾아낼 수 있다.

3.4 프록시 서버

ISP가 제공하는 프록시 서버는 클라이언트 브라우저와

웹서버 사이에서 동작한다. 프록시 레벨 캐쉬 기능은 네트워크 트래픽 부담뿐만 아니라 사용자의 웹페이지 로드 시간도 단축시키는 기능을 제공한다. 그러나 이는 사용자 세션을 획득하는데 있어서 로컬 캐쉬보다 훨씬 많은 문제를 야기한다. CLF 혹은 ECLF 로그에서 프록시 서버로부터의 모든 요청은 같은 IP를 가진다. 또한 프록시 서버 수준의 캐쉬 기능으로 인하여 서버로부터의 하나의 요청이 실제로 여러 명의 사용자에 의한 것처럼 보일 수 있다. 이 문제를 해결하기 위해 쿠키(cookies)를 사용하면 가능하다. 그러나 이 방법 역시 사용자의 협력 없이는 불가능하며, 사용자의 프라이버시(privacy)에 좌우된다.



(그림 5) 로컬 IP 획득

PLS는 (그림 5)와 같이 로컬 IP 주소 획득을 수행한다. 기본적으로 프록시 서버를 사용하는 모든 사용자의 IP는 ECLF 로그에 사용자의 실제 IP(123.456.78.9,…)가 기록되는 것이 아니라 프록시 서버 IP(192.9.200.1)가 기록된다. PLS에서는 이 문제를 해결하기 위해 페이지 로거와 웹사이트의 루트 페이지를 socket()으로 연결하여 사용자의 실제 IP를 획득하게 된다. (그림 6)은 프록시 서버에서의 로컬 IP 획득 과정을 보여준다.

단계 1. 사용자가 웹페이지 접근
단계 2. 페이지 로거 load, 실행
  2.1 사용자가 사인
단계 3. 현재 페이지를 떠나는 시점에서 페이지 로거 stop() 실행
단계 4. stop()에서 웹사이트 루트 페이지와 socket() 연결
  4.1 Local IP 획득
단계 5. 로그 프로세서 호출
  5.1 Local IP 전달
단계 6. 페이지 로거 종료

(그림 6) 프록시 서버에서의 로컬 IP 획득

(그림 6)에서 보면 사용자가 웹사이트의 한 페이지에 접근하면(단계 1), 그 페이지에 삽입되어 있는 페이지 로거 호출 코드에 의해 사용자 브라우저로 페이지 로거가 호출되어 실행된다(단계 2). 사용자가 인증에 "예"로 확인하면 사용자의 로컬 IP를 획득할 수 있다. 사용자가 현재 페이지를 떠나는 시점에서 페이지 로거의 stop() 함수가 실행된다(단계 3). stop()에서 모든 웹사이트에 존재하는 루트 페이지와 socket()을 연결한다(단계 4). 일단 socket()을 연결하면 웹사이트의 외부, 즉 웹서버의 다른 곳에 위치한 페이지 로거에서도 사용자의 브라우저에 접근할 수 있게 된다. 따라서 로컬 IP를 획득할 수 있으며(단계 4.1), 획득한 로컬 IP를 로그 프로세서에 전달하고(단계 5.1) 페이지 로거를 종료한다.

3.5 PLS에 의한 트랜잭션 구분

PLS에 의해 획득한 DB를 이용하여 쉽게 사용자 세션 파일을 획득할 수 있다. 이 사용자 세션은 이제 마이닝 알고리즘인 순차 패턴(sequential pattern) 마이닝이나 클러스터링의 입력으로 사용될 수 있다. 그러나 연관성 규칙 마이닝 알고리즘을 위해서는 트랜잭션 구분을 수행해야 한다. 일반적으로 웹 마이닝 시스템의 목적이 연관 규칙 생성이라면 트랜잭션 구분을 통하여 획득한 트랜잭션 파일을 마이닝 알고리즘의 입력으로 사용한다.

트랜잭션 구분의 일반적인 모델은 다음과 같다. L을 사용자 세션 파일 항목 집합이라 하자. 세션 항목 $l \in L$은 사용자 IP 주소 $l.ip$, 사용자 ID인 $l.id$, 탐색된 페이지의 URL $l.url$, 탐색시각 $l.time$을 포함한다. 사용자 ID를 얻을 수 없는 경우는 사용자 세션 파일에 임의로 정해진 ID를 사용한다. 일반적인 트랜잭션 t는 다음처럼 정의할 수 있다.

$$t = \langle ip_t, uid_t, \{(l_1^t.url, l_1^t.time), \cdots, (l_m^t.url, l_m^t.time)\}\rangle \quad (1)$$

여기서 $1 \le k \le m$, $l_k^t \in L$, $l_k^t.ip = ip_t$, $l_k^t.uid = uid_t$ 이다.

그러나 PLS에서는 다음과 같이 실제적인 데이터 획득에 근거하여 트랜잭션을 구분한다. L을 사용자 세션 파일 항목 집합이라 하자. 세션 항목 $l \in L$은 사용자 IP 주소 $l.ip$, 사용자 id인 $l.id$, 탐색된 페이지의 URL $l.url$, 탐색시간 $l.accesslength$를 포함한다. 사용자 id를 얻을 수 없는 경우는 사용자 세션 파일에 임의로 정해진 id를 사용한다. PLS에 의한 트랜잭션 구분에서 트랜잭션 t_{pls}는 다음과 같이 정의한다.

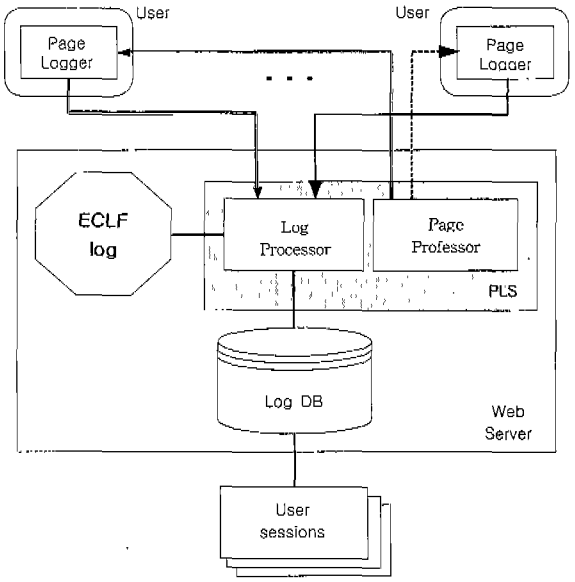
$$t_{pls} = \langle ip_{t_{pls}}, uid_{t_{pls}}, \{(l_1^{t_{pls}}.url, l_1^{t_{pls}}.accesslength), \cdots, (l_m^{t_{pls}}.url, l_m^{t_{pls}}.accesslength)\}\rangle \quad (2)$$

여기서 $1 \le k \le m$, $l_k^{t_{pls}} \in L$, $l_k^{t_{pls}}.ip = ip_{t_{pls}}$ $l_k^{t_{pls}}.uid = uid_{t_{pls}}$ 이다.

일반적인 모델과 PLS의 트랜잭션 구분 과정에서의 차이는 일반적인 모델에서는 계산을 통한 추정이지만 PLS에서는 그 값을 직접 획득함으로써 정확성과 단순성을 제공해 준다.

4. 시스템 구현

본 연구에서 구현한 PLS의 개발 환경은 Unix 환경에서 Java 언어와 데이터베이스(MySQL)를 이용하여 개발하였다. PLS는 WWW 환경에서 확장된 웹 서버 로그를 획득하기 위해 엔진처럼 수행되는 시스템으로 사용자 인터페이스는 요구되지 않는다. 단지 데이터베이스에 저장된 결과를 보기 위해 MySQL의 SQL문을 이용하는 경우가 있을 수 있다. (그림 7)은 구현한 PLS의 구성도이다. 페이지 로거가 사용자 정보를 획득하여 로그 프로세서에 전달하게 되고, 로그 프로세서는 적절한 포맷으로 Log DB를 생성하게 된다.



(그림 7) PLS 구성도

(그림 8)은 PLS에 의해 획득된 Log DB의 스키마를 보여준다.

(그림 8)의 idx는 인덱스를 의미하며, Userid는 웹사이트

user_session
idx
Userid
IPAddress
login_time
accesslength
URL
Agent
referer

(그림 8) User session 스키마

의 일부에 사용자가 login하는 페이지가 존재할 경우 획득할 수 있는 정보이다. Userid는 필요할 경우 쉽게 획득할 수 있는 것으로 여기서는 이를 생략한다. IPAddress는 사용자 IP 주소를 의미하는데, 사용자가 ISP가 제공하는 프록시 서버를 이용하는 경우에도 사용자가 보안 인증에 '예'로 응답할 경우 정확한 IP 주소를 획득할 수 있다. login_time은 사용자가 웹사이트의 해당 페이지에 접근한 시간이며, accesslength는 그 페이지를 실제로 본 시간(PageViewtime)이다. URL은 해당 페이지의 주소를 의미하는데, 여기서는 하나의 웹사이트에 적용하는 시스템이기 때문에 단순하게 해당 페이지 이름만 기술하도록 한다. Agent는 사용자의 웹브라우저를 의미하며, referer는 해당 페이지를 접근하기 위해 참조한 바로 직전의 페이지를 의미한다. (그림 9)는 PLS에 의해 획득된 Log DB의 일부를 보여준다.

5. 결 론

본 연구는 웹 마이닝에서 요구하는 사용자 세션을 획득함에 있어서 CLF 혹은 ECLF 로그만으로는 많은 어려움이 따르는 문제들, 즉 복잡한 전처리 단계들, 로컬 캐쉬 혹은 프록시 서버 사용 문제 등을 해결하기 위한 확장된 웹 서버 로그 획득 시스템(PLS)을 구현하였다. PLS는 기존의 웹

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3079 to server version: 3.23.37

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> select * from user_session;
```

idx	Userid	IPAddress	login_time	accesslength	URL	Agent	referer
42	a	203.253.21.116	2001-09-01 13:30:57	2	/S.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/N.html
39	a	203.253.21.116	2001-09-01 13:30:46	3	/A.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/
41	a	203.253.21.116	2001-09-01 13:30:54	2	/N.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/E.html
40	a	203.253.21.116	2001-09-01 13:30:51	5	/E.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/A.html
38	a	203.253.21.116	2001-09-01 13:30:42	339	/	Mozilla/4.0 (compatible; MSIE 5.0; Win32	
37	a	61.157.244.253	2001-08-26 17:38:47	17	/B.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/A.html
36	a	61.157.244.253	2001-08-26 17:38:29	6	/B.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/B.html
35	a	61.157.244.253	2001-08-26 17:38:22	5	/B.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/A.html
34	a	61.157.244.253	2001-08-26 17:38:19	4	/A.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/
33	a	61.157.244.253	2001-08-26 17:38:12	6	/	Mozilla/4.0 (compatible; MSIE 5.0; Win32	
31	a	203.253.21.116	2001-08-25 12:02:03	3	/D.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/A.html
30	a	203.253.21.116	2001-08-25 12:02:00	13	/A.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/
32	a	203.253.21.116	2001-08-25 12:08:46	402	/K.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/D.html
29	a	203.253.21.116	2001-08-25 12:01:47	3	/C.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/A.html
28	a	203.253.21.116	2001-08-25 12:01:43	5	/I.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/C.html
27	a	203.253.21.116	2001-08-25 12:01:38	3	/Q.html	Mozilla/4.0 (compatible; MSIE 5.0; Win32	/I.html

(그림 9) PLS에 의해 획득된 Log DB의 일부

마이닝 시스템에서 많은 시간과 비용을 수반했던 데이터 전처리 작업을 단순화시키고, 캐쉬 히트, 프록시 서버 IP 등의 문제를 해결하여 정확한 사용자 세션을 획득함으로써 마이닝 알고리즘에 더 정확한 입력을 제공하게 되어 효율적인 마이닝 작업을 수행할 수 있도록 한다.

제안한 PLS를 이용하여 획득한 정확한 사용자 세션은 여러 마이닝 알고리즘을 위해 여러 가지 형태로 포맷되어 사용될 수 있다. 즉 PLS를 이용하여 획득한 사용자 세션을 근거로 하여 여러 마이닝 알고리즘에 적절한 포맷을 수행한다면 정확한 입력을 제공해주는 효과로 인하여 효율적인 데이터 마이닝 작업을 수행할 수 있을 것으로 판단된다. 본 연구의 향후 연구 과제는 PLS를 크고 작은 여러 실제 웹 사이트에 적용하여 획득한 사용자 세션을 여러 마이닝 알고리즘에 적용해 보고, CLF 혹은 ECLF 로그를 이용한 마이닝 시스템과의 비교를 통하여 PLS의 정확한 사용자 세션이 마이닝 시스템의 성능 향상에 크게 기여할 수 있도록 개선하는 것이다.

참 고 문 헌

[1] Alex G. Buchner, Maurice D. Mulvenna, "Discovering Internet Marketing Intelligence through Online Analytical Web Usage Mining," SIGMOD Record, (4) 27, 1999.

[2] http : //www.w3.org/Daemon/User/Config/Logging.html.

[3] R. Cooley, B. Mobasher, and J. Srivastava, "Data preparation for mining World Wide Web browsing patterns," Journal of Knowledge and Information Systems, (1) 1, 1999.

[4] B. Mobasher, N. Jain, E. Han, and J. Srivastava, "Web Mining : Pattern discovery from World Wide Web Transactions," Technical Report TR96-050, Univ. of Minnesota, Dept. of Computer Science, Minneapolis, 1996.

[5] R. Cooley, B. Mobasher, and J. Srivastava, "Web Mining : Information and pattern discovery on the World Wide Web," In International Conference on Tools with Artificial Intelligence, pp.558-567, Newport Beach, CA, 1997.

[6] R. Cooley, B. Mobasher, and J. Srivastava, "Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns," Proc. of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop(KDEX-97), Nov. 1997.

[7] Osmar R. Zaiane, Man Xin, Jiawei Han, "Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs," School of Computing Science, Simon Fraser University, 1998.

[8] J. Pitkow, "In search of reliable usage data on the WWW," In Sixth International World Wide Web Conference, pp. 451-463, 1997.

[9] World Wide Web Committee(W3C) Web usage Characterization Activity(WCA), http : //www.w3c.org/WCA, 1999.

[10] M. S. Chen, J. S. Park, and P. S. Yu, "Data mining for path traversal patterns in a web environment," In Proceedings on the 16th International Conference on Distributed Computing Systems, pp.385-392, 1996.

[11] http : //software.tangent.org/modlayout/.

[12] P. Pirolli, J. Pitkow, and R. Rao, "Silk from a sow's ear : Extracting usable structures from the Web," In Proc. of 1996 Conference on Human Factors in Computing Systems(CHI-96), 1996.

[13] L. Catledge and J. Pitkow, "Characterizing browsing behaviors on the World Wide Web," Computer Networks and ISDN Systems, 27(6), 1995.

[14] C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of www client-based traces," Technical Report TR-95-010, Boston University, CS Dept., Boston, MA02215, 1995.

윤 선 희

e-mail : sunniyoon@hanmail.net

1986년 숭실대학교 전자계산학과(공학사)

1988년 숭실대학교 전자계산학과 (공학석사)

1998년 숭실대학교 전자계산학과 박사 과정 수료

1992년~현재 미림 전산고등학교 교사

관심분야 : 데이터마이닝, 웹컴퓨팅, 밀티미디어 통신, 멀티미디어 응용 등

오 해 석

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

1975년 서울대학교 응용수학과(공학사)

1981년 서울대학교 계산통계학과 (이학석사)

1989년 서울대학교 계산통계학과 (이학박사)

1983년~현재 숭실대학교 컴퓨터학과 교수

1996년~1999년 숭실대학교 부총장 역임

관심분야 : 멀티미디어 통신, 웨이블릿 영상코딩, 멀티미디어 응용 등