웹 마이닝을 위한 입력 데이타의 전처리과정에서 사용자구분과 세션보정

(User Identification and Session completion in Input Data Preprocessing for Web Mining)

최 영 환 † 이 상 용 ††
(Young-hwan Choi) (Sang-yong Lee)

요 약 웹 이용 마이닝은 거대한 웹 로그들을 이용하여 웹 사용자의 이용 패턴을 분석하는 데이타 마이닝 기술이다. 이러한 웹 이용 마이닝 기술을 사용하기 위해서는 전처리 과정 중의 사용자와 세션을 정확하게 구분해야 하는데, 표준 웹 로그 형식의 로그 파일만으로는 완전히 구분할 수 없다. 사용자와 세션을 구분하기 위해서는 로컬캐시, 방화벽, ISP, 사용자 프라이버시, 쿠키 등과 같은 많은 문제들이 있지만, 이 문제를 해결하기 위한 명확한 방법은 아직 없다. 특히, 로컬캐시 문제는 웹 마이닝 시스템의 입력으로 사용되는 사용자 세션을 구분하는데 가장 어려운 문제이다.

본 연구에서는 참조 로그와 에이전트 로그, 그리고 액세스 로그 등의 서버측 클릭스트림 데이타만을 이용하여 로컬캐시 문제를 해결하고, 사용자 세션을 구분하고 세션을 보정하는 휴리스틱 방법을 제안한다.

키워드 : 웹 이용 마이닝, 데이타 전처리, 사용자 세션, 세션구분, 세션보정, 휴리스틱

Abstract Web usage mining is the technique of data mining that analyzes web users' usage patterns by large web log. To use the web usage mining technique, we have to classify correctly users and users session in preprocessing, but can't classify them completely by only log files with standard web log format. To classify users and user session there are many problems like local cache, firewall, ISP, user privacy, cookey etc., but there isn't any definite method to solve the problems now. Especially local cache problem is the most difficult problem to classify user session which is used as input in web mining systems.

In this paper we propose a heuristic method which solves local cache problem by using only click stream data of server side like referrer log, agent log and access log, classifies user sessions and completes session.

Key words : Web Usage Mining, Data Preprocessing, User session, Session Identify, Session Complete, Heuristic

1. 서 론

WWW은 트래픽의 양과 웹 사이트의 복잡도와 크기가 늘어나면서 눈부시게 발전하고 있다. 웹 사이트의 디자인, 웹 서버 디자인 등은 이러한 성장과 더불어 더욱 복잡해지고 있다. 데이타 마이닝은 많은 양의 데이타에 함축적으로 들어 있는 지식이나 패턴을 찾아내는 기술이라고 정의할 수 있다. 데이타 마이닝은 사용자가 방문한 웹 페이지와 사용자의 특징을 보관하고, 이 데이타를 분석하여 각각의 사용자에 맞는 웹 페이지를 동적으로 생성해줄 수 있다. 더욱이 웹 액세스의 다차원 웹 로그 분석을 이용한 트랜드 분석을 통해 웹에서 어떤 일이 일어나고 있는지에 대해서도 대략적인 정보를 제공해줄 수 있다[1].

웹 마이닝은 거대한 웹 데이타 저장소의 로그들을 이용하여 웹 사용자의 사용 패턴을 분석하는 데이타 마이닝 기술이다[2]. 일반적으로 웹 마이닝 시스템의 효율성은 마이닝 시스템의 입력으로 사용되는 데이타의 정확도에 달려있다[3]. 대부분의 웹 마이닝 시스템들은 W3C에서 규정하고 있는 Common Log Format(CLF) 혹은 Extended Common Log Format(ECLF)[4] 형태의 로

† 비 회 원 : 공주대학교 컴퓨터공학과
cyhmad@kongju.ac.kr
†† 종신회원 : 공주대학교 정보통신공학부 교수
sylee@kongju.ac.kr
논문접수 : 2002년 6월 12일
심사완료 : 2003년 6월 13일

그 파일을 이용하여 마이닝 작업을 수행한다. 물론 CLF 혹은 ECLF 로그로부터 획득한 데이타를 웹 마이닝 시스템의 입력으로 사용하기까지는 데이타 정제(data cleaning), 사용자 구분(user identification), 세션 구분(session identification), 세션 보정(session completion) 등, 여러 가지 전처리 단계를 거쳐야 한다. 하지만, 표준 웹 로그 형식의 웹 로그만으로는 사용자를 완전히 구분할 수 없다. 따라서 정확한 결과를 얻기 위해 사용자와 세션을 구분할 수 있는 모듈을 웹 서버에서 제공하거나, 각각의 페이지에 적당한 실행 코드를 삽입해야 한다. 또 사용자 정보, 세션 정보 등 마이닝에 필요한 모든 정보를 웹 로그에 저장하고 있다고 해도 웹 로그 내의 필요 없는 정보를 제거하는 등의 이유로 전처리 과정은 필요하다[5]. 전처리 과정 중에 중요한 단계가 사용자 구분과 세션 구분인데, 사실 로컬캐시 문제, 방화벽 문제, IP(ISP)문제, 사용자 프라이버시 문제, 쿠키 문제 등 여러 가지 문제점들에 직면하게 된다.

로컬캐시 문제는 웹 마이닝 시스템의 입력으로 사용되는 사용자 세션을 구분하는데 가장 장애가 되는 요소로서, 한 번 접근한 페이지를 그 세션에 다시 접근하는 경우 그 페이지에 대한 기록이 로그 파일에 남지 않아 정확한 데이타를 얻을 수 없다. 이러한 로컬캐시 문제를 해결하기 위해 기존의 해결 방안으로는 쿠키를 이용하는 방법, 캐시 버스팅을 이용하는 방법, 폼을 이용하는 방법, 페이지에 에이전트나 애플릿을 삽입하는 방법 등 여러 가지 방법들이 쓰이고 있지만 본 연구에서는 참조 로그와 에이전트 로그, 그리고 액세스 로그 등 서버측 클릭스트림 데이타만을 이용하여 로컬 캐시 문제를 해결하고, 사용자 세션을 구분하고 보정하기 위한 휴리스틱 방법을 제안한다.

2. 관련연구

웹 서버 로그로부터의 데이타 마이닝 기술 적용 개념은 Chen[6], Mannila[7], Yan[8]에서 처음 제기되었다.

Chen 등은 사용자 세션을 트랜잭션으로 분류하기 위하여 최대 선점 참고를 이용하는 개념을 소개했다. 최대 선점 참고는 사용자가 특정 사용자 세션동안 이전에 보았던 페이지를 요청하는 곳에서 취소하기 이전에 사용자에 의한 마지막 요청 페이지이다. 예를 들어 사용자 세션이 A-B-A-C-D-C 페이지 순서로 요청이 구성되어 있다면 세션에 대한 최대 선점 참고는 B와 D일 것이다.

Pitkow[9]에 의하여 수행된 연구에서는 웹 서버 로그로부터 사용자와 사용자 세션을 구분하는 문제의 어려움을 보여준다. 확실한 사용 데이타 수집에 가장 큰 두 가지 장애는 로컬 캐싱과 프록시 서버이다. 성능향상과 네트워크 트래픽의 최소화를 위하여 요청된 페이지들을 대부분의 웹 브라우저가 캐시한다. 결과적으로, 사용자가 "뒤로" 버튼을 누를 때 캐시된 페이지는 보여지지만 웹 서버는 페이지 접근을 반복하지 않는다. 프록시 서버는 캐시의 중간 레벨을 지원하고 구분된 사이트 이용으로 더 많은 문제를 일으킨다. 웹 서버 로그에서 프록시 서버로부터의 모든 요청은 요청이 잠재적으로 더욱 많은 사용자를 제시한다 할지라도 같은 인식자를 갖는다. 프록시 서버 레벨 캐싱으로 인하여 서버로부터의 단일 요청은 확장된 시간의 영역을 통하여 다중 사용자에게 보여질 수 있다.

쿠키를 보내는 대신에, 자바 에이전트를 보내어 클라이언트측 브라우저에서 실행되어 웹 서버로 정확한 사용 정보를 보내주는 방법은 사용자 프라이버시 문제에 직면하게 된다[10].

표 1 사용자 구분에 사용되는 방법들의 비교

Method	Description	Privacy Concern	Advantages	Disadvantage
IP Address & Agent	각각의 유일한 IP와 에이전트 쌍을 독립된 사용자로 추정함	낮음	항상 사용 가능 추가적인 기술이 필요하지 않음	유일한 IP임을 보증할 수 없음 유동 IP일 경우 식별하기 어려움
Embedded Session ID	모든 링크에 동적으로 생성된 페이지 ID를 삽입함	낮음/ 보통	항상 사용 가능 IP 주소에 독립적임	반복된 접근에 대한 개념이 없음 완전히 동적인 사이트가 요구됨
Registration	사용자가 사이트에 직접 등록함	보통	브라우저에 국한되지 않고 각 개인을 인식할 수 있음	모든 사용자가 등록을 하는 것을 원하지는 않음
Cookie	쿠키를 사용자의 컴퓨터에 저장함	보통/ 높음	반복된 접근을 인식할 수 있음	사용자들의 부정적인 견해에 의해 거부될 수 있음
Software Agent	에이전트 프로그램이 브라우저에 삽입되어 사용데이타를 웹 서버에 보내 줌	높음	모든 사이트에 대하여 정확한 사용데이타를 얻을 수 있음	사용자에 의해 거부될 수 있음
Modified Browser	브라우저가 사용데이타를 기록함	매우 높음	모든 사이트에 대하여 정확한 사용데이타를 얻을 수 있음	사용자가 브라우저에 요구사항을 명시해 주어야 함

표 1에서는 사용자와 세션의 구분에 사용되어지는 방법들을 비교하였다[11].

3. 사용자와 세션의 구분

본 연구에서 사용된 사용자 구분과 세션 구분을 위한 절차는 먼저 각 로그 데이타를 시간 순으로 정렬한 뒤 정렬된 로그 데이타를 IP와 에이전트, 그리고 시간 별로 구분한다. 그리고 구분된 로그 데이타를 타임아웃과 참조로그를 이용하여 각 세션으로 구분한다.

3.1 사용자 구분

독립적인 세션을 얻기 위해서 웹 서버에 접근하는 사용자들을 각각의 독립된(unique) 사용자들로 구분해야 하는데, 이 작업은 로컬 캐시, 방화벽, 프록시 서버의 존재로 매우 복잡해진다. 웹 마이닝은 이 문제에 대해서 다루는 가장 쉬운 길인 사용자 협력에 크게 의존한다. 로그/사이트 의존적 방법이라 할지라도, 휴리스틱 방법은 독립된 사용자를 구분하는데 유용한 방법이다. IP 주소가 같다 할지라도 에이전트 로그가 브라우저나 OS에서 바뀐다하더라도, 다른 사용자를 표시하는 IP 주소에 대해서 다른 에이전트 형식을 만들어주는 것이다.

본 연구에서 제안한 방법은, 같은 IP 주소에서 접근한 사용자라 할지라도, 페이지 참조들의 연관성을 처리하여 다른 사용자로 구분하게 된다.

그림 1의 웹 사이트를 고려해 보자. 샘플 정보는 표 2에서 보여진 접속, 참조, 에이전트 로그들로부터 수집된다. 모든 로그 앤트리들은 같은 IP 주소와 저장되지 않은 사용자 ID를 갖는다. 그러나 다섯 번째, 여섯 번째, 여덟 번째, 그리고 열 번째 앤트리는 다른 에이전트를 이용하여 접근되고 있기 때문에 적어도 두 사용자 세션

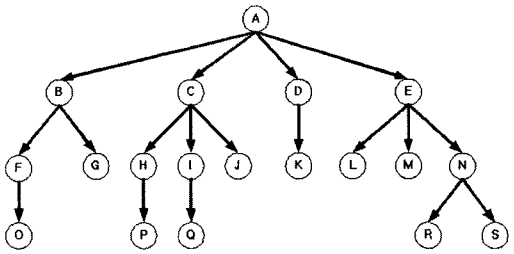

그림 1 웹 사이트의 예 - Hyper Links로 표현된 페이지 사이의 접근경로

을 나타내는 로그로 가정한다.

사용자 구분에 대한 다른 휴리스틱 방법은 참조 로그와 각 사용자 브라우징 경로를 구성하기 위한 사이트 토폴로지를 결합한 액세스 로그를 이용한다. 만약 또다시 사용자가 페이지를 방문한다면 휴리스틱 방법은 같은 IP 주소로 다른 사용자가 접근한 것으로 인식한다.

사용자를 구분하기 위해

1. 샘플 로그를 같은 IP Address를 갖는 로그로 각각 구분한다. 표 2는 샘플 로그를 같은 IP Address를 갖는 로그로 구분한 결과이다.

2. 같은 IP Address를 갖는 로그에 대해 같은 Agent를 갖는 로그로 구분하게 되면 A-E-H-N-P-R-M-A-D와 A-E-B-G이다.

3. 같은 IP Address와 Agent를 갖는 로그를 타임아웃을 적용시켜 각각의 사용자로 구분한다.

사용자를 구분한 결과는 A-E-H-N-P-R-M, A-D, A-E-B-G가 된다.

3.2 세션 구분

세션은 한 사용자가 주어진 사이트에 요청을 처음 시

표 2 Access, Referrer, Agent Logs로부터 얻어진 정보의 예

No	IP Address	Userid	Time	Method/URL/Protocol	Status	Size	Referred	Agent
1	210.106.81.115	-	[25/Jul/2001:09:05:45+0900]	"Get A.html HTTP/1.0"	200	3290	-	Mozilla/4.2(Win98,I)
2	210.106.81.115	-	[25/Jul/2001:09:06:37+0900]	"Get E.html HTTP/1.0"	200	2050	A.html	Mozilla/4.2(Win98,I)
3	210.106.81.115	-	[25/Jul/2001:09:06:43+0900]	"Get H.html HTTP/1.0"	200	4130	-	Mozilla/4.2(Win98,I)
4	210.106.81.115	-	[25/Jul/2001:09:07:12+0900]	"Get N.html HTTP/1.0"	200	5096	E.html	Mozilla/4.2(Win98,I)
5	210.106.81.115	-	[25/Jul/2001:09:07:59+0900]	"Get A.html HTTP/1.0"	200	3290	-	Mozilla(IE5.0,WinNT)
6	210.106.81.115	-	[25/Jul/2001:09:08:35+0900]	"Get E.html HTTP/1.0"	200	2050	A.html	Mozilla(IE5.0,WinNT)
7	210.106.81.115	-	[25/Jul/2001:09:08:47+0900]	"Get P.html HTTP/1.0"	200	8140	H.html	Mozilla/4.2(Win98,I)
8	210.106.81.115	-	[25/Jul/2001:09:12:21+0900]	"Get B.html HTTP/1.0"	200	1820	A.html	Mozilla(IE5.0,WinNT)
9	210.106.81.115	-	[25/Jul/2001:09:12:57+0900]	"Get R.html HTTP/1.0"	200	2270	N.html	Mozilla/4.2(Win98,I)
10	210.106.81.115	-	[25/Jul/2001:09:13:16+0900]	"Get G.html HTTP/1.0"	200	9430	B.html	Mozilla(IE5.0,WinNT)
11	210.106.81.115	-	[25/Jul/2001:09:14:27+0900]	"Get M.html HTTP/1.0"	200	7220	E.html	Mozilla/4.2(Win98,I)
12	210.106.81.115	-	[25/Jul/2001:11:26:22+0900]	"Get A.html HTTP/1.0"	200	3290	-	Mozilla/4.2(Win98,I)
13	210.106.81.115	-	[25/Jul/2001:11:28:23+0900]	"Get D.html HTTP/1.0"	200	1680	A.html	Mozilla/4.2(Win98,I)

도해서 사이트를 떠날 때까지로 구분한다. 따라서 실제 사용자의 웹 사용 패턴을 발견하기 위한 세션 구분은 데이타 마이닝 입력 데이타로서 매우 중요한 의미를 가진다.

세션 구분의 목표는 개인 세션으로 각각의 사용자 페이지 접근을 구분하는 것이다. 그러나 실제 웹 로그만으로 세션을 구별하기에는 어려움이 있다. 사용자들은 로그아웃 버튼을 누르지 않고, 웹 브라우저를 닫는 경우가 많다. 역시 표준 웹 로그 형식을 따르는 로그만이 있는 경우, 사용자 구별이 어렵다면, 세션 구분도 어려워진다. 어쩔 수 없이 IP에 따라 사용자를 구별한다.

세션의 정의에 따라 같은 IP에서 다른 시간에 접속하는 경우, 이 세션은 다른 세션으로 보아야 한다. 이 경우 역시 사용자가 사이트를 떠나는 시점을 파악해야 한다. 이를 위한 가장 간단한 방법은 타임아웃을 사용하는 것이다. 본 연구에서는 30분을 디폴트 타임아웃으로 이용한다.

표 2를 보면, 세 번째 엔트리-페이지 H는 A와 E 페이지에 직접적으로 접근하지 못할 뿐만 아니라, 다른 이전 로그 엔트리의 어느 곳에서도 온 것이 아니다. 이것은 사용자 구분된 첫 번째 결과에서 두 사용자가 같은 IP 주소로 들어왔다는 것을 보여준다. 그러므로 첫 번째 사용자 구분을 각각의 사용자 세션으로 구분해야 한다. 사용자 세션 구분에 대해서는 휴리스틱 방법만이 존재한다는 것은 매우 중요하다. 같은 타입의 기계에서 같은 브라우저를 이용하는 같은 IP 주소의 두 사용자는 만약 그들이 페이지들의 같은 집합을 보고 있다면 사용자 구분을 쉽게 할 수 없다. 반대로, 다른 브라우저에서 실행하는 단일 사용자나, 사이트 링크 구조를 이용하지 않고 URL을 직접 입력하는 것은 다중 사용자 상황에서 놓칠 수 있는 부분이다.

사용자 세션을 구분하기 위해 관리자가 샘플로그에 대해서 미리 각각의 페이지로 접근하기 위한 모든 패스를 구하여 참조패스 집합(R)를 생성한 다음, 참조패스와 비교하여 사용자 구분된 결과를 새로운 사용자 세션으로 구분한다. 다음은 생성된 참조패스 집합이다.

R = {A-B, A-C, A-D, A-E, B-F, B-G, C-H, C-I, C-J, D-K, E-L, E-M, E-N, F-O, H-P, I-Q, N-R, N-S}

사용자 구분(U_i)에서 각 로그엔트리(l_j)에 대한 참조(r_j)를 참조패스 집합(R)에서 검색하여 참조패스(R_i)를 생성시킨다.

다음은 사용자 세션을 구분하는데 사용된 방법이다.

U_i = {$l_1, \cdots, l_n$}(사용자 구분된 결과)

S_i : 사용자 세션

P_i : 사용자 구분에 대한 참조패스

l_j : log entry

r_j : referrer

같은 IP Address와 Agent를 갖는 각각의 U_i에 대해

1. i, j = 1
2. Create P_i, S_i
3. Add r_j to P_i, l_j to S_i
4. $j = j + 1$
5. for each l_j do
6. for each P_i do
7. if $r_j \in P_i$ then
8. Add r_j to P_i, Add l_j to S_i
9. goto 15

표 3 샘플로그의 세션 구분 결과

No	IP Address/Agent	Time	Request URL	Referring URL
1	210.106.81.115/ Mozilla/4.2(Win98.I)	[25/Jul/2001:09:05:45+0900]	A.html	-
		[25/Jul/2001:09:06:37+0900]	E.html	A.html
		[25/Jul/2001:09:07:12+0900]	N.html	E.html
		[25/Jul/2001:09:12:57+0900]	R.html	N.html
		[25/Jul/2001:09:14:27+0900]	M.html	E.html
2	210.106.81.115/ Mozilla/4.2(Win98.I)	[25/Jul/2001:09:06:43+0900]	H.html	-
		[25/Jul/2001:09:08:47+0900]	P.html	H.html
3	210.106.81.115/ Mozilla/(IE5.0,WinNT)	[25/Jul/2001:09:07:59+0900]	A.html	-
		[25/Jul/2001:09:08:35+0900]	E.html	A.html
		[25/Jul/2001:09:12:21+0900]	B.html	A.html
		[25/Jul/2001:09:13:16+0900]	G.html	B.html
4	210.106.81.115/ Mozilla/4.2(Win98.I)	[25/Jul/2001:11:26:22+0900]	A.html	-
		[25/Jul/2001:11:28:23+0900]	D.html	A.html

```
10.            else
11.            i = i + 1
12.        end;
13.        Create Si, Pi
14.        Add rj to Pi, lj to Si
15.        j = j + 1
16.    end;
```

세션 구분의 결과는 A-E-N-R-M, H-P, A-E-B-G, A-D로 구성된 네 개의 사용자 세션이다. 표 3에서는 샘플 로그의 세션 구분 결과를 보여주고 있다.

4. 세션보정

사용자 세션을 구분하는데 있어 또 하나의 중요한 문제점은 중요한 액세스 정보가 로컬캐시로 인해 액세스 로그에 기록되지 않을 수도 있다는 점이다. 세션 보정이란 Usage Data의 전처리의 마지막 중요한 단계로 캐시된 페이지 참조를 추론하여 로컬캐시로 인해 손상된 참조패스를 완성하는 단계이다. 어떤 요청 페이지가 참조 로그에 없는 경우일 때 이것은 대부분의 브라우저에서 사용하는 "Back" 버튼을 누른 경우이다. 이미 한 번 가져온 페이지는 브라우저의 캐시에 저장되었다가 다시 요청했을 때는 캐시에 있는 내용이 사용자에게 다시 보여지기 때문에 서버의 액세스 로그에는 기록되지 않는다. 참조 로그가 명확하지 않을 때는 사이트 구조를 이용하여 누락된 페이지들을 사용자 세션 파일에 추가시켜야 한다.

4.1 세션보정 과정

세션을 보정하기 위한 휴리스틱 방법은 다음 순서에 따라 수행된다.

1. 관리자가 미리 참조 패스(R)를 생성한다.
2. 같은 IP Address와 Agent를 갖는 항목에 대해, 페이지 뷰의 세션 집합에 있는 각 세션을 참조 패스와 비교하여 참조 패스에 있는 세션이 나오면 스택(st)에 기록한다.
3. 참조 패스에 없는 세션이 나올 때까지 계속 수행한다.
4. 참조 패스에 없는 세션이 나오면 Back 버튼을 누른 것으로 간주 바로 전 패스를 스택에 기록하는 것을 반복한다.
5. 더 이상 참조 패스에 없는 세션이 나오지 않으면 하나의 세션보정을 완료한다.
6. 다음 세션을 보정한다.

다음은 세션을 보정하기 위한 방법이다.

S : 페이지 뷰의 세션 집합
r_v : 페이지 참조 파일
st : stack
R : 참조 패스 집합
V : 페이지 뷰

```
1. for each V ∈ S do
2.     if hi ≠ null then
3.         st.push(vi-1)
4.     else
5.         st.pop()
6. set n = S.size() + 1
7. set notfound = true
8. while ((Vs = st.pop()) ≠ null) AND rv ∈ R
9.     add vs to Stemp
10.    if rv ∈ Vs then
11.        set notfound = false
12.        insert Stemp into S at n
13. if (notfound) then
14.    for j = n - 2 down to 1 do
15.        if rv ∈ Vj then
16.            insert Vj into S at n
17.            set rv = hj
18. end;
```

4.2 세션보정 결과

다음은 위의 휴리스틱 방법에 의해 A-E-N-R-M 에서 A-E-N-R-N-E-M 으로 세션보정한 결과이다.

```
1. A                S←A     S = {A}
2. E   A-E ∈ R      S←E     S = {A, E}
3. N   E-N ∈ R      S←N     S = {A, E, N}
4. R   N-R ∈ R      S←R     S = {A, E, N, R}
5. M   R-M ∉ R      S←N     S = {A, E, N, R, N}
6.     N-M ∉ R      S←E     S = {A, E, N, R, N, E}
7.     E-M ∈ R      S←M     S = {A, E, N, R, N, E, M}
```

먼저 첫 페이지인 A가 들어오면 페이지 뷰의 세션 집합인 S에 포함시킨다. 다음 E가 들어오면 A-E로의 참조 패스가 R에 있는지 검색하고 R에 참조 패스가 있기 때문에 S에 포함시키고 다음 페이지 뷰를 가져온다. N, R까지는 같은 방법으로 S에 포함시킨다. 그 다음 M이 들어왔을 때 참조 패스집합 R에 R-M이 없기 때문에 스택의 바로 전 페이지 뷰인 N을 S에 포함시키고 N의 참조 패스인 N-M과 비교한다. 역시 없기 때문에 그 전 페이지 뷰인 E와 비교한다. 비교 결과 E-M이 있기 때문에 S에 E를 포함시키고 마지막 페이지 뷰인 M을 포함 시킨다. 첫 번째 세션의 보정이 완성되었다.

동일한 방법으로 나머지 세션에 대해서도 휴리스틱 방법을 수행한다.

다음은 A-E-B-G 에서 A-E-A-B-G 으로 세션보정한 결과이다.

```
1. A                    S←A     S = {A}
2. E   A-E ∈ P          S←E     S = {A, E}
```

3. B E-B ∉ P S←N S = {A, E, A}
4. A-B ∈ P S←R S = {A, E, A, B}
5. G B-G ∈ P S←N S = {A, E, A, B, G}

다음 표 4는 세션구분된 사용자 접근 경로에 대한 세션보정의 최종결과이다.

결과에서 볼 수 있듯이 사용자에게 어떠한 요구도 하지 않고 단지 웹 로그만을 분석하여 사용자와 사용자 세션을 구분할 수 있게 되었다.

표 4 세션구분된 데이타의 세션보정 결과

Session No	IP Address/Agent	Request URL	Referring URL
1	210.106.81.115/ Mozilla/4.2(Win98.I)	A.html	-
		E.html	A.html
		N.html	E.html
		R.html	N.html
		N.html	E.html
		E.html	A.html
		M.html	E.html
2	210.106.81.115/ Mozilla/4.2(Win98.I)	H.html	-
		P.html	H.html
3	210.106.81.115/ Mozilla/(IE5.0,WinNT)	A.html	-
		E.html	A.html
		B.html	A.html
		A.html	-
		G.html	B.html
4	210.106.81.115/ Mozilla/4.2(Win98.I)	A.html	-
		D.html	A.html

4.3 실험 및 분석

휴리스틱 방법에 대한 결과의 정확도를 측정하기 위해 실제 사이트(공주대학교 인공지능 연구실 홈페이지)의 로그 파일을 대상으로 실험하였다. 실제로 접속한 페이지에 대한 참조 페이지를 직접 기록한 후 접속한 로그 파일을 수집하여 필터링한 후 휴리스틱 방법을 적용하여 테스트를 하고, 직접 기록한 세션과 비교하여 정확도를 측정하였다. 사이트의 로그 파일은 하루동안 접속한 페이지를 대상으로 하였다.

첫 번째 실험은 샘플 페이지를 대상으로 하여 실험하였고, 두 번째는 연구실 내에서 접속한 로그만을 대상으로 연구실 내 컴퓨터에서 접속한 세션을 접속자들이 직접 기록하고, 직접 만들어진 100개의 세션과 휴리스틱 방법을 적용하여 만들어진 세션을 비교하였다.

세 번째는 모든 로그에 대하여 실험을 하였다.

실험결과 첫 번째 샘플페이지를 대상으로 한 실험에서는 100%의 정확도를 보였고, 직접 기록한 페이지의 참조만을 대상으로 한 두 번째 실험에서는 91%의 정확도를 기록하였고, 모든 로그를 대상으로 한 세 번째 실험에서는 82%의 정확도를 기록하였다.

분석 결과 두 번째 실험에서는 프레임으로 나누어진 페이지의 문제와, PHP를 통한 참조 페이지를 정확히 기록하지 못하였기 때문에 9%의 오차가 발생하였고, 세 번째 실험에서의 오차는 ISP를 통하여 접속한 로그들에 대하여 참조 페이지를 정확히 작성 할 수 없었기 때문에 18%의 오차가 발생한 것으로 분석되었다.

추후 PHP나 CGI, 스크립트 파일에 의한 참조에 관한 해결 방안과 ISP를 통한 접근에 대한 보정방법에 대한 연구로 더 정확한 결과를 얻을 수 있을 것이다.

5. 결론 및 향후 연구과제

로컬캐시 문제는 웹 마이닝 시스템의 입력으로 사용되는 사용자 세션을 구분하는데 가장 장애가 되는 요소로서, 한 번 접근한 페이지를 그 세션에 다시 접근하는 경우 그 페이지에 대한 기록이 로그 파일에 남지 않아 정확한 데이타를 얻을 수 없다. 본 연구에서는 참조 로그와 에이전트 로그, 그리고 액세스 로그 등 서버측 클릭스트림 데이타만을 이용하여 로컬캐시 문제를 해결하고, 사용자 세션을 구분하고 세션을 보정하는 휴리스틱 방법을 제안했다. 계속해서 패스의 구조를 미리 만들지 않고 자동으로 참조패스의 구조를 만들어 가면서 세션을 보정하는 방법에 관한 연구가 필요하다.

CGI나 PHP, 스크립트 파일 등은 현 방법으로는 분석을 할 수 없다. 분석을 하기 위해서는 패킷 스니핑 방식을 사용해야 한다. 패킷 스니핑 방식은 서버가 아니라 네트웍 스위치나 허브에 설치하여 그곳을 통과하는 모든 패킷을 분석하는 방식으로서 실시간 고객 분석이 가능하고 서버에 전혀 부하를 주지 않는다. 로그추출 모듈이 문제를 일으킬 경우 로그를 잃게 되는 단점이 있지만 서버에는 아무런 영향을 미치지 않으므로 웹 서비스에 지장을 주지는 않는다. JSP 페이지나 CGI 페이지의 경우에도 모듈 추가를 비롯한 운영 및 관리비용이 추가되지 않는다는 점이 장점이다. 단점이라면 암호화된 페이지의 분석을 위해서 별도의 플러그인이 필요한 경우가 생길 수 있다.

본 논문에서는 웹 로그 추출 방법에 로그 파일 분석 방식을 사용하였고 이후 연구에서는 새로운 방식으로 스크립트 파일이나 CGI, JSP, PHP 파일 등을 분석하는 것에 대한 연구를 할 것이다.

참 고 문 헌

[1] 김재형, 노효원, 김남호, 정정화, "인터넷 비즈니스 기반의 고객관계관리(CRM)를 위한 웹 로그 분석에 관한 연구", 한국정보처리학회 춘계 학술발표논문집 제7권 제1호, 2000.

[2] R. Cooley, B. Mobasher, J. Srivastava. Web mining : Information and pattern discovery on the World Wide Web. In: *International Conference on*

Tools with Artificial Intelligence, Newport Beach, CA, pp.558-567, 1997.

[3] 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.

[4] http://www.w3.org/Daemon/User/Config/Logging.h-tml.

[5] Srivastava, J. Cooley, R., Deshpande, M. & Tan P.N. Web Usage Mining: Discovery and Application of Usage Patterns from Web Data. SIGKDD Explanations, 1. 2000.

[6] M. S. Chen, J. S. Park, P. S. Yu. Data Mining for path traversal patterns in a Web environment. In: *Proc. 16 th International Conference on Distribut-ed Computing Systems*, pp.385-392, 1996.

[7] H. Mannila, H. Toivonen, Discovering Generalized episodes using minimal ocurrences. In: *Proc. Seco-nd International Conference on Knowledge Disco-very and Data Mining*, Portland, Oregon, pp.146-151, 1996.

[8] T. Yan, M. Jacobson, H. Gracia-Molina, U. Dayal. From user access patterns to dynamic hypertext linking. In : *Fifth International World Wide Web Conference*, Paris, France, 1996.

[9] J. Pitkow. In search of reliable usage data on the WWW. In : *Sixth International World Wide Web Conference,* Santa Clara, CA, pp.451-463, 1997.

[10] P. P. Bonissone and K. S. Decker. Selecting uncertainty calculi and granularity: An experiment in trading-off precision and complexity. *Uncertainty in Artificial Intelligence*, pp. 2217-2247, 1986.

[11] R. Cooley, B. Mobasher, J. Srivastava. Data Prepa-ration for mining World Wide Web Brow sing Pat-terns, In: *Knowledge and Information Systems 1*, Springer Verlag, pp. 1-26. 1999.

최 영 환

1999년 공주대학교 전자계산학과(이학사) 2001년 공주대학교 대학원 전자계산학과(이학석사). 2001년~현재 공주대학교 대학원 컴퓨터공학과 박사과정. 2001년~현재 한국표준연구원 학연 박사과정. 관심분야는 인공지능, 에이전트, 웹 마이닝, eCRM, Personalization

이 상 용

1984년 중앙대학교 전자계산학과(공학사) 1988년 일본동경대학대학원 총합이공학연구과(공학석사). 1988년~1989년 일본 NEC 중앙연구소 연구원. 1993년 중앙대학교 일반대학원 전자계산학과(공학박사) 1993년~현재 공주대학교 정보통신공학부 교수. 1996년~1997년 University of Central Florida 방문교수. 관심분야는 인공지능, 에이전트, 컴퓨터게임, 바이오인포매틱스