Syslog 데이터의 의미론적 검색을 위한 XML 기반의 모델링

이 석 준[†] · 신 동 천[††] · 박 세 권[†††]

요 약

이벤트 로깅은 시스템 및 네트워크 관리에 있어 그 역할이 증대되고 있으며, syslog는 해당 분야에 있어 사실상의 표준으로 사용되고 있다. 그러나 대부분의 로그 분석은 반구조적 특징을 보이는 로그 형식으로 인하여 빈번히 출현하는 패턴에만 집중하고 있다. XML은 syslog 데이터를 구조화하는 데 있어 유용한 방식을 제공하고 정보 탐색을 용이하게 해 준다. 하지만 이전의 XML 형식들 및 어플리케이션들은 로그 데이터를 위한 순위 기반 검색이나 유사도 측정 등과 같은 의미론적 접근에 적합하지 않다. 본 논문에서는 XML 기반의 순위 키워드 검색 기법을 기초로, 새로운 로그 데이터 모델링을 통해 syslog 데이터를 위한 XML 트리 구조를 제안한다. 그리고 기존의 XML 구조보다 의미론적 검색에 적합함을 보인다.

키워드 : 이벤트 로깅, 의미론 검색, XML

XML-based Modeling for Semantic Retrieval of Syslog Data

Lee Seok Joon[†] · Shin Dong Cheon[††] · Park Sei Kwon[†††]

ABSTRACT

Event logging plays increasingly an important role in system and network management, and syslog is a de-facto standard for logging system events. However, due to the semi-structured features of Common Log Format data, most studies on log analysis focus on the frequent patterns. The eXtensible Markup Language can provide a nice representation scheme for structure and search of formatted data found in syslog messages. However, previous XML-formatted schemes and applications for system logging are not suitable for semantic approach such as ranking based search or similarity measurement for log data. In this paper, based on ranked keyword search techniques over XML document, we propose an XML tree structure through a new data modeling approach for syslog data. Finally, we show suitability of proposed structure for semantic retrieval.

Key Words : Event Logging, Semantic Retrieval, XML

1. 서 론

이벤트 로깅 메커니즘은 시스템의 유효성과 신뢰성을 증가시킬 수 있는 수단을 제공한다. 로그 데이터는 로깅 작업에 의해 획득된 다량의 정보를 담기 때문에, 로그 데이터로부터 정보를 찾아내는 것은 시스템 관리 작업에서 매우 중요한 부분이다. 오늘날 가장 대중적으로 사용되는 로깅 시스템인 syslog는 매우 다양한 범위를 다루며, 어떤 메시지들을 전달하고 기록할지를 사용자 정의1[1]의 범주로 보기 때문에, 생성되는 데이터에서 필요한 정보를 찾는 것은 간단한 문제가 아니다. 하지만 다양한 정보탐색 관련 기법들이 존재함에도 불구하고 로그 데이터에 대한 접근들은 결코 다양하다고 볼 수 없다. 예를 들어 grep 이나 정규표현식을 지원하는 perl 스크립트들은 단순 키워드 매칭을 통한 필터링 혹은 패턴 반복 횟수의 집계를 기반으로 한다. 마찬가지로, 패턴을 인지하여 classification 및 categorization을 수행하는 클러스터링[11, 15]은 좀더 체계적이지만 기본적인 개념에서는 별다른 차이점이 없는 방법론이다. 로그 데이터에 대한 현존하는 접근들은 이 범주를 크게 벗어나지 않고 있다.

로그 분석을 위한 클러스터링의 기준이 엔티티간의 유사도일때, 기존 연구들의 유사계수 대상은 사용자 정의 단계에서의 주체인 프로세스 식별자가 아니라 MSG 파트[13]의 문자열 패턴이다. MSG 파트의 문자열을 대상으로 하는 클러스터링은 MTBF, MTBR, MTTR, 그리고 시스템 가용성을 이용한 신뢰도 측정 등의 작업에 적합하다. 하지만 이러한 접근은 시스템 로깅에 대한 적절한 통계의 개념일 뿐이

† 정 회 원 : 중앙대학교 대학원 정보시스템학과
†† 종신회원 : 중앙대학교 정보시스템학과 교수
††† 정 회 원 : 중앙대학교 정보시스템학과 교수
논문접수 : 2005년 6월 21일, 심사완료 : 2006년 1월 11일

1) syslog.conf(5) man page

며 효율적인 모니터링이나 예측, 혹은 의미전달의 기반으로 보기는 어렵다. 이는 로그 분석의 목적을 가진 기존 어플리케이션들[3, 9]의 한계로 볼 수 있다. 기록된 로그 데이터에서 엔티티나 이벤트 갯수의 파악과 키워드에 대한 문자열의 단순 매칭을 알아내는 작업은 중요하다. 하지만 엔티티와 이벤트 자체가 절대적 혹은 상대적으로 얼마나 중요한 데이터인가의 문제 또한 중요한 문제이다. 왜냐하면 이들은 데이터에서 정보를 추출하는 또 다른 기준이 될 수 있기 때문이다[17, 8]. 로그 데이터 기록의 주체인 프로세스의 구분 및 이에 기반한 로그 데이터 엔티티간의 관계 복원은 이 문제 해결을 위한 근본적인 사항이나, 이를 고려하는 접근법은 존재하지 않는다.

로깅 시스템에 국한하지 않고 일반적인 범주에서 보면, 다량의 데이터에서 상대적으로 더 중요한 부분을 추출하는 방안들은 여러 분야에서 다양하게 제시되어 왔다. PageRank [10]를 이용한 Google 검색엔진은 매우 성공적인 예이며, 의미전달을 위한 순위결정 방안을 XML 문서에 적용한 XRank [8]와 XSEarch[17] 역시 흥미로운 제안임에 틀림 없다. XML 문서는 태그를 사용하는 Self-Describing Data이며, 로그데이터를 위한 포맷으로 제안된 몇몇 사례가 있다 [6, 9, 18, 19]. 그러나 이들은 단지 결과물의 표현을 위해 XML 형식을 사용하며, 의미론적 검색과는 명확히 구분된다. 예를 들어 [18, 19]의 XML 로그 문서 형식은 의미론적 접근의 시각에서 데이터 모델링을 위해[16] XML 트리 형태로 표현할 때 매우 단순한 구조를 보여준다.

의미론적 검색에 있어 노드간 관련도 파악은 매우 중요한 항목이며, 각 관련도의 산출 근거는 XML 트리를 구성하는 노드간의 관계이다. 이는 곧 노드간의 관계가 단조로우면 단조로울수록, 노드들간 연관성의 정도를 알게 해 주는 기준들이 감소함을 의미한다. 직관적으로 이 구조는 효율적인 키워드 검색 작업에 적합하지 못하다는 것을 알 수 있다.

본 논문의 목적은 반구조적인 로그 데이터를 구조화하여 syslog protocol이 전달하는 메세지의 의미를 찾는 것이다. 즉 기존의 로그 데이터 처리가 syslog의 어떤 의미를 분실하였는지, 또한 의미를 어떻게 복원할 것인지에 대한 분석 및 방법론을 제시하는 것이다. 이를 위해 본 논문에서는 syslog에 대한 분석을 통하여 로그 데이터 모델링을 한 후, 의미론적 검색에 적합한 트리구조를 제안하고 평가한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구의 기반이 되는 이론적 배경을 소개하며 3장에서는 로그 데이터의 XML 변환에 대해 기술한다. 4장에서는 제안한 구조가 의미론적 접근에 적합한지 평가하며 5장에서는 결론을 기술한다.

2. 이론적 배경

의미론적 검색이란 데이터 구조에 의미를 두고 정보를 찾는다는 개념이다. XML 검색 방법론들은 다양하고 유용한 기법들을 제공하고 있으며, 특히 의미론적 검색을 주제로 하는 두 가지 주목할만한 모델이 존재한다.

• XRank(XML 문서를 위한 순위 기반의 키워드 검색 기법) : XRank는 PageRank 모델을 기반으로 하며 엘리먼트 순위 결정의 주 요인으로 ElemRank 모델을 제시하였다. 이 모델은 현 노드에 대한 부모와 자식, 그리고 이웃 노드에서의 링크를 통한 접근 확률 및 임의의 접근 확률을 포함하여 다음과 같은 수식으로 정의된다.

$$e(v) = \frac{1-d_1-d_2-d_3}{N \times N_{de}(v)} + d_1 \sum_{(u,v)\in HE} \frac{e(u)}{N_h(u)} + d_2 \sum_{(u,v)\in CE} \frac{e(u)}{N_c(u)} + d_3 \sum_{(u,v)\in CE^{-1}} e(u)$$

N_e는 XML 엘리먼트의 총 개수이며 $N_c(u)$는 u의 하부 엘리먼트 개수, $N_h(u)$는 문서 u와 외부로 연결된 hyperlink의 개수, $N_{de}(u)$는 엘리먼트 v를 포함한 XML 문서의 엘리먼트 개수이다. CE와 HE는 각각 hyperlink edge와 containment edge를 말하며 CE^{-1}이 reverse containment edge 집합일 때 $E = HE \cup CE \cap CE^{-1}$이 된다. $d_1 \sim d_3$는 현 노드에 연결된 노드들로부터의 접근 확률을 의미하며 $0 < d < 1$의 값을 가진다.

위 수식은 XML 문서의 각 노드들을 구분하며 이들과의 연결을 근거로 현재의 검색 대상인 v 노드를 평가한다. 구분에 사용되는 기준을 정의하는 작업은 가중치 부여를 위한 중요한 요소로 ElemRank와 PageRank의 상이성은 여기서 비롯된다. XRank는 XML 문서를 그래프로 표현하며 그래프는 노드와 containment edge, hyperlink edge들로 구성된다. 그리고 이들을 기반으로 XML 엘리먼트들 각각의 순위를 계산한다. HTML을 대상으로 하는 PageRank와의 차이점은 HTML의 랭킹 메소드가 hyperlink에만 주목하는 반면 XML을 대상으로 할 때는 containment link의 존재도 고려된다는 점이다. 각 엘리먼트들의 근접도를 판단할 때 전자는 키워드 사이의 거리만을 고려하는 반면, 후자인 XML 키워드 검색에서는 질의 결과물인 XML 트리의 높이와 너비의 두 가지 차원으로 접근한다.

• XSEarch(XML을 위한 의미론적 검색 엔진) : XSEarch의 순위결정 모델은 XML 문서 각 요소의 관계와 이를 표현한 XML 트리에 기반한다. XML 문서에서 관계를 도출할 때, 트리 내에서 태그는 노드에, 데이터는 리프에 매칭된다. 데이터의 Relationship Tree는 리프들과 그들의 조상 노드들을 찾아 하나로 묶어줌으로 생성되며 이러한 연결은 종종 interconnected와 strongly interconnected로 구분되기도 한다. strongly interconnected는 Relationship Tree를 이루는 노드 $n_1,...,n_k$에 동일한 라벨명을 가진 두개 이상의 노드가 존재하지 않는 연결을 말한다. interconnected는 strongly interconnected를 포함하며, 그렇지 않을 경우 연결의 주체를 이루는 두개의 노드가 Relationship Tree를 이루는 노드 $n_1,...,n_k$ 중 동일한 라벨명을 가진 유일한 노드들인 연결을 말한다. 요약하면, Relationship Tree 및 interconnected 등은

XML 문서 내 노드들을 조건에 따라 묶은 개념이다. 이 항목은 보다 직관적인 설명이 필요하므로, 본 논문은 3장에서 로그 데이터의 노드들이 구조화를 거쳐 적절한 관계를 부여받는 과정을 예를 들어 설명한다. 모든 관련 용어들은 [7, 16]의 정의를 따른다.

XSEarch에서 순위 결정에 관여하는 항목[2]들은 노드들 간의 관계뿐만이 아니며 다음 두 가지가 더 존재한다.

－TFILF(Term Frequency, Inverse Leaf Frequency) : 문서들을 위한 벡터 모델인 TFIDF를 수정한 이 수치는 다음과 같이 계산되고, 결과값은 인덱스에 저장된다. n_1이 리프 노드이고 k는 키워드, $occ(k,n_1)$은 키워드 k가 리프 노드 n_1에 나타난 횟수라고 가정한다면, 키워드 k에 대한 TF 즉 리프 내 빈도수는

$$tf(k,n_l)=\frac{occ(k,n_l)}{\max\{occ(k',n_l)\mid k'\in words(n_l)\}}$$

이며, 키워드가 포함된 리프들의 역 빈도수는 다음과 같다.

$$ilf(k)=\log\left(1+\frac{|N|}{|\{n'\in N\mid k\in words(n')\}|}\right)$$

N은 전체 모든 리프 노드들의 집합이다. $tfilf(k,n_l)$ 값은 $tf(k,n_l)\times ilf(k)$로 정의되며 이 값이 0 이라면, k는 n_l에 포함되지 않음을 뜻한다. 리프 내 키워드들의 빈도와 리프들의 역 빈도수 파악은 전통적인 정보탐색 기법과 맥락을 같이하지만, 그 대상은 문서가 아닌 리프로 정의된다.

－질의와 결과물의 유사성 : 질의에 대한 결과물의 만족도는 벡터 스페이스 모델[12]로 산출된다. 질의 Q와 답변 N의 유사성 $SIM(Q,N)$은 N내 노드 벡터와 Q에 매칭되는 검색어 벡터 사이의 코사인 거리를 말한다. 이 항목은 4장에서 사례와 함께 설명한다.

3. 로그 데이터의 XML 변환

syslog는 다양한 분야에서 사용되는 중요한 데몬 소프트웨어이다. 그러나 syslog에 대한 접근은 앞서 언급한 대로 다양하지 못하며, 특히 데이터의 가치를 순위화하는 작업 및 이를 위한 데이터의 의미를 찾는 작업은 사용자의 몫으로 남아있다. 본 장에서는 로그 데이터를 위한 새로운 XML 트리 구조를 제안한다.

3.1 syslog 데이터의 형식과 매핑

syslog가 받아들이는 메시지는 디바이스와 데몬 프로세스의 위험 신호, 그들의 삶과 죽음에 필요하고 관련된 것들, 그리고 많은 다른 것들을 포함한다[2]. 이 움직임은 운영체제 내부에서, 그리고 네트워크로 연결된 호스트 혹은 디바이스－예를 들어 프린터, 라우터, 허브, 스위치, 그리고 디스크가 없는 워크스테이션 등－사이에서 발생한다. 사용자들의 움직임 또한 중요한 메시지로서, 로깅 시스템은 사용자 및 객체의 사용 기록 및 패턴을 남김으로서 해당 시스템의 통제 작업을 지원한다.

〈표 1〉 주요 Facility 및 Severity

구분	내용
Facility 0	kernel messages
Facility 1	user-level messages
Facility 3	system deamons
Facility 4	security / authorization messages
Facility 5	messages generated internally by syslogd
Severity 1	Alert : action must be taken immediately
Severity 3	Error : error conditions
Severity 5	Notice : normal but significant condition

메시지를 출력하거나 포워딩하는 것은 유연한 룰들에 의해 제어되며 전체 포맷은 PRI, HEADER, MSG의 세 파트로 구분된다. PRI 파트는 0 에서 165 까지의 메시지 priority 값(facility number×8 + severity value)을 담고 있으며, 일부 facility와 severity의 값은 <표 1>과 같다. HEADER는 시각 정보(TIMESTAMP)와 호스트명 혹은 디바이스의 아이피(HOSTNAME)를 포함한다. MSG 파트는 프로세스의 생성된 정보를 담는 텍스트 메시지 부분이다. 이 부분은 다시 Tag와 Content로 구성되며 이들은 프로그램명이나 프로세스가 생성한 메시지, 그리고 보다 상세한 메시지 내용에 대응한다.

〈표 2〉 complex node의 형태

$<l,n>\Rightarrow$ "connection"
$<l,ns>\Rightarrow$ "connection cluster"
$<l_1,ns_1>,...,<l_k,ns_k>\Rightarrow$ "connection cluster set"

본 논문에서 데이터 정의는 시멘틱 네트워크 이론에 기반하며[7], 두 가지 레벨로 나누어 전개된다. 첫번째 레벨인 의미적 레벨에서 atomic(basic) 노드 와 complex 노드는 syslog를 구성하는 엔티티들을 표현한다. 전자는 emanating-edge를 가지지 않는 리프 노드를 말하며, unordered collection인 set, bag, ordered collection인 list, 그리고 union 값을 가질 수 있다. 후자는 그래프 상에서 하나 이상의 라벨이 붙은 emanating edge를 가지며, 내부에 관계와 튜

2) [17]에서는 유사도와 전체 노드들 및 최적값이 아직 미정인 임의의 변수들을 결합하여 최종적인 순위 결정 항목을 제시한다. 하지만 이는 전체 XSEarch 시스템에 의존적이고 해당 문헌의 평가 부분에만 필요한 항목이다. 본 논문에서는 불필요한 부분이기 때문에 제외하였다.

플, 그리고 basic 노드 들을 담을 수 있다. 이들의 상태는 다시 표 <표 2>와 같은 세 가지 형태로 분류 가능하며, 좌측은 라벨을, 우측은 노드 혹은 노드의 집합을 의미한다. 집합 N_{ode}에 속한 노드 n은 내부에 튜플들을 가질 수 있으며, N_b가 set of basic, N_c가 complex 노드 일때, $N_b \cap N_c = \phi$, $N_b \cup N_c = N_{ode}$가 된다.

노드들을 연결하는 directed edge들은 각 엔티티 간의 관계를 나타낸다. aggregation, generalization, association, 그리고 of-property 관계를 모두 edge로 표현하며 이들의 구분은 연결 형태로 판단한다. 일련의 라벨들은 의미 관계의 다른 타입을 나타낸다. edge$\in E_{dge}$의 구성요소를 E_{label}, $E_{source}N_{ode}, E_{target}N_{ode}$로 놓을 때, 다음 사항을 도출 할 수 있다[7, 16].

- $E_{source}N_{ode}$는 E_{label}을 통해 $E_{target}N_{ode}$에 연결된다.
- $(E_{label} = p)$는 $(E_{source}N_{ode} \in N_c)$와 $(E_{target} \in N_b)$를 포함한다.
- $(E_{source}N_{ode} \in N_c)$와 $(E_{target}N_{ode} \in N)$는 $(E_{label} = \{g, a, s\})$에 포함된다.
- 링크 타입이 "of-property"라면 source 노드는 complex node(N_c)이며 target 노드는 basic node(N_b)이다. 링크 타입이 "generalization", "aggregation", "association" 이면 source 노드는 complex 노드이다.

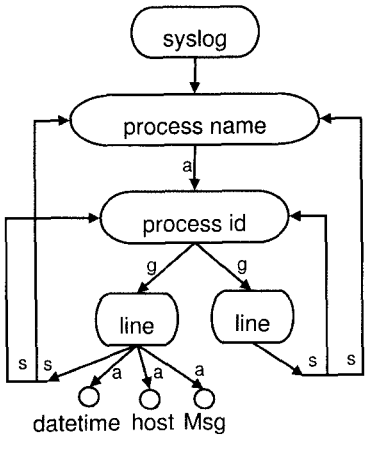
두번째 레벨은 스키마 레벨이다. 엘리먼트와 속성들은 XML 스키마에서 누락없이 simple 혹은 complex 타입으로 정의되어야 한다. XML 스키마는 XML DTD를 대체하는 개념으로 문서의 의미와 사용 그리고 관계의 일관성을 정의하며, 로그의 스키마는 syslog 개념적인 그래프를 실체화 할 수 있는 중요한 수단이 된다. 본 논문에서는 세부적인 XML 스키마 생성은 생략하고 Well-formed XML 문서로 전개하기로 한다.

3.2 프로세스 기반의 로그 트리

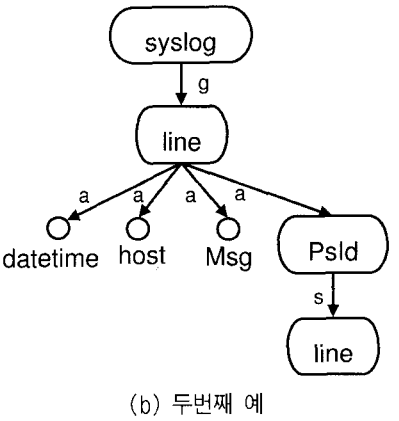
앞서 언급한 바와 같이, syslogd의 로깅작업은 사용자 정의가 매우 중요한 부분을 차지한다. 본 논문에서는 로그 메시지에 대한 사용자의 관점을 다음과 같이 가정하였다.

- syslogd의 메시지 기록은 프로세스의 이벤트 발생 단위로 이루어진다. 활성화된 프로세스는 복수의 자식 프로세스를 포함할 수도 있지만, 단일 프로세스일 수도 있다. 순차적으로 발생한 두 개의 이벤트가 각기 다른 두 개의 프로세스에 의해 일어났다면, 이 경우 서로 association 관계를 가진다고 보기 어렵다.
- 동일 프로세스 혹은 동일한 프로세스 아이디를 가진 자식 프로세스에 의해 생성된 복수의 이벤트라면, 같은 목적을 가진 행위에 대한 기록으로 볼 수 있다. 이것은 association 관계이다. syslogd는 메시지를 파일로 보낼 때, association에 대한 계층 구분 없이 모두 동등한 라인으로 기록한다. 그러나 각 라인에는 프로세스 아이디가 남기 때문에 해당 이벤트의 최상위 부모 프로세스 아이디와 관련지어 유추할 수 있다.

(그림 1) (a)는 프로세스 이름과 아이디를 이벤트 및 메시지 전달의 주체로 하여, 일차적으로 로그 라인을 간단한 시멘틱 네트워크로 표현한 것이다. line이 프로세스 아이디와 generalization 관계를 가지고 프로세스 아이디와 프로세스 이름이 aggregation관계에 놓여 있는 것은, syslog 데이터의 의미 파악에 있어 프로세스 식별이 가장 중요하다는 관점의 표현이다. 또한 line이 하부 HEADER 및 MSG 항목들과 aggregation 관계에 있음은 앞서 살펴본 syslog 메시지 형식에서 알 수 있다. (그림 1) (a)와 같은 XML 변환은 중요한 문제점을 내포하고 있다. 즉, line이 부모 노드인 process id 및 조상 노드인 process name과 association 관계에 있다는 것이다. 이 형태는 순환이 존재하며, XML 검색 구현 과정의 용이성을 위해서는 완전한 트리 구조로 바꾸어야 할 필요성이 있다[7, 16]. 순환과는 별개로 또 다른



(a) 첫번째 예



(b) 두번째 예

(그림 1) 시멘틱 네트워크

중요한 문제점이 존재한다. 원본 로그 라인들간의 평등한 계층이 그대로 유지된다면 모든 리프 노드의 깊이는 5로 고정되고 line은 leaf node-1 깊이에만 위치하게 되는 반면에 너비는 라인 수에 정비례하여 증가한다. 직관적으로, 이러한 형태의 트리는 Relationship Tree를 이용한 의미론적 검색[17, 8]은 물론 그 이전에 일반적인 XML 검색 방법론들에도 적합하다고 볼 수 없다.

이를 해결하기 위한 두번째 시도는 (그림 1) (b)와 같다. (그림 1) (b)는 시멘틱 네트워크의 주요 이슈중 하나인 cycle convertion[7]을 참고하여 순환이 존재치 않는 DAG (directed acyclic graph), 즉 트리 형태를 보이고 있다. 그래프의 순환 참조를 피하기 위해 사용되는 것은 새로운 리프 노드의 추가이며 (그림 1) (b)에서는 프로세스 이름과 프로세스 아이디를 합친 PsId 라는 complex 노드가 추가되었다. 이후 해당 프로세스 이름을 참조하는 로그 라인은 PsId의 자식으로, 해당 프로세스 아이디를 참조하는 로그 라인은 PsId의 후손으로 정리할 수 있다. 본 논문에서는 이 트리 구조를 "프로세스 기반의 로그 트리"로 지칭한다.

이 트리 구조는 다음과 같은 특성을 갖는다. 최상위 루트 노드인 syslog의 자식들은 모두 line을 라벨명으로 하며 line의 분기 계수는 6으로 고정된다. 한 line 내 자식 노드는 중복 없는 고정 라벨명을 지닌다. 또한 다음과 같은 상황일 때 line의 자식 노드인 PsId는 또다른 line 노드를 자식 노드로 두게 된다. 그렇지 않다면, line의 자식 노드들은 모두 리프 노드가 된다.

- 임의의 프로세스가 새로운 자식 프로세스를 fork[3]하고 이들이 각기 별도의 메시지를 전달했을 경우
- 단일 프로세스가 복수의 메시지를 전달할 경우

2장에서는 다량의 데이터에 의미를 부여하는 방법으로 순위 할당을 사용하는 방법론들을 언급하였다. 이들의 순위 결정은 기존의 문자열 패턴 일변도의 메커니즘이 아니라 노드들 간의 연결 상태 및 거리와 계층, 즉 트리에서 발견되는 구조적 특징에 기반한다. 제안하는 트리 구조에서 노드들간의 거리와 계층은 (그림 1) (a)의 트리 구조에 비해 상대적으로 다양한 모습을 보여주며, 이러한 요인은 기존 형식들보다 다양한 순위를 얻을 수 있는 기반이 된다.

3.3 노드간 관련도

<표 3>은 4장의 평가에 사용된 로그의 일부분이다. 로그 데이터의 (a)-(h) 라인들은 각기 "rpc.mount"와 "postgres"

〈표 3〉 로그 데이터의 예

(a) Sep 18 14:43:09 stt1 rpc.mountd: authenticated unmount request from dog:871 for /var/log (/var/log)

(b) Sep 18 14:43:09 stt1 rpc.mountd: authenticated unmount request from dog:873 for /root (/root)

(c) Sep 18 14:43:09 stt1 rpc.mountd: authenticated unmount request from dog:875 for /home (/home)

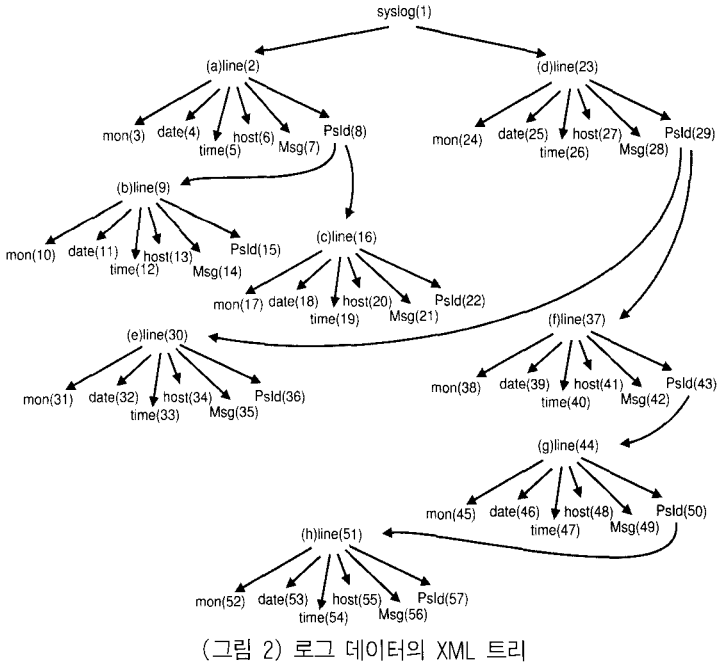
(d) Sep 18 15:02:02 stt1 postgres[20872]: [1] LOG: connection received: host=[dog]

(e) Sep 18 15:02:02 stt1 postgres[20874]: [1] LOG: connection received: host=[dog]

(f) Sep 18 16:38:47 stt1 postgres[21005]: [1] LOG: connection received: host=[dog]

(g) Sep 18 16:38:47 stt1 postgres[21005]: [2] LOG: connection authorized: user=meyrin database=zaft

(h) Sep 18 16:38:47 stt1 postgres[21005]: [3] LOG: query: begin; select getdatabaseencoding(); commit



(그림 2) 로그 데이터의 XML 트리

3) fork(2) man pages

라는 프로세스 이름을 공유하고 있기 때문에 (그림 2)와 같은 구성이 이루어진다. (그림 2)에 다음과 같은 질의[5]가 주어진다면, (a)부터 (f)까지 모든 라인이 리턴된다.

/syslog/line[mon='Sep' and contains(Msg, 'dog')]

또한, 아래 질의에 대한 결과값은 (d)부터 (h)까지의 모든 라인이 될 것이다.

/syslog/line[contains(Msg, 'connection')]

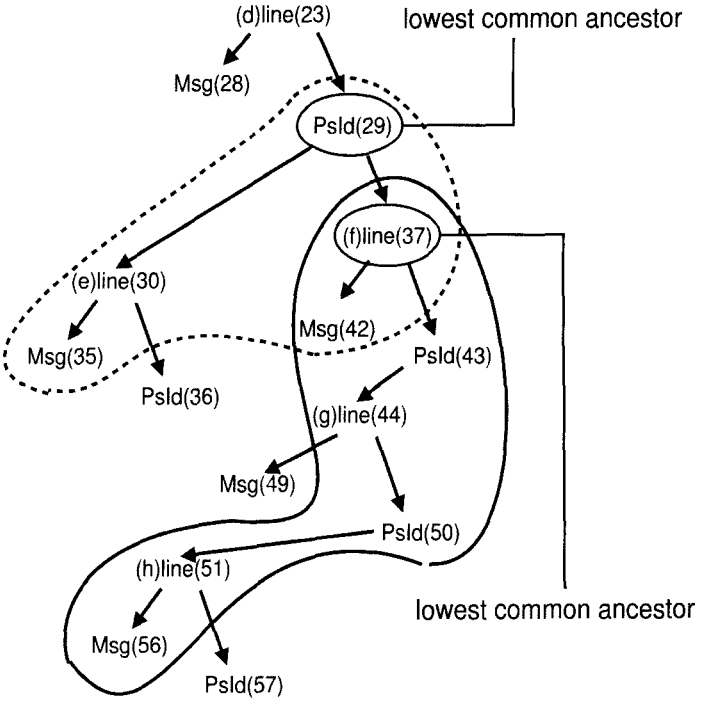
하지만 의미론적 관점에서, 위의 질의로 얻는 결과값들 상호간의 관련도는 동일한 것이 아니다. 노드 간 관련도를 결정지을 때는 다음과 같은 요인들을 고려해야 하며[17], "프로세스 기반의 로그 트리" 구조는 평면적이었던 로그 라인간의 관계를 재구성하여 이를 가능케 한다.

- Relationship Tree의 크기 : Relationship Tree의 크기는 작을수록 해당 구성요소들의 관련도는 높아진다. 첫 번째 질의 결과물 중 (b)와 (c)의 관계, 그리고 (b)와 (f)의 관계를 각각 Relation Tree로 표현한다면 두 경우의 Relation Tree는 7개의 노드와 10개의 노드를 포함하기 때문에 크기에서 차이를 보인다. 상대적으로 작은 Relation Tree 크기를 갖는 전자의 경우가 더욱 관련성 있다고 볼 수 있다[17].
- 연결 상태 : 예제에서 line(16)이 lowest common ancestor인 time(19)와 Msg(21)은 strongly interconnected이다. 또한 PsId(8)을 중심으로 연결된 line(9)와 line (16)은 interconnected이다.
- 조상-후손 관계 : 두번째 질의에 대한 결과물로 (그림 3)의 점선으로 표시된 Relation Tree를 A, 곡선으로 표시된 Relation Tree를 B로 놓는다면 각각의 Relation Tree 크기는 A가 5, B는 7이 된다. 이 사실만 놓고 보면 Msg(35)와 Msg(42) 사이가 Msg(42)와 Msg(56) 사이보다 더 관련성 있다고 할 수 있다. 또한 line(30)과 line(37)은 interconnected 상태이다. 하지만 line(37)은 line(51)의 조상이며, line(30)와는 이웃이 된다. Relation Tree 내 조상-후손 관계는 연관성 부여에 있어 Relation Tree 크기나 interconnected 상태보다 우선시되는 요인이며 (f)는 (h)와 좀더 연관성이 있다.

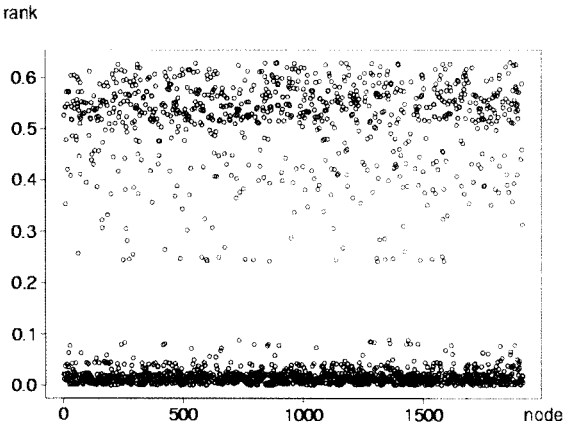
4. 평 가

본 장에서는 제안한 구조가 의미론적 검색 접근에 적합함을 보이기 위해 기존 XML 로그 구조[18, 19]와 제안한 "프로세스 기반의 로그 트리"를 대상으로, 특정 키워드에 대한 각 노드들의 순위 및 유사도를 측정하였다. 또한 이들이 의미론적 검색에서 어떤 차이를 보이는지 확인하였다.

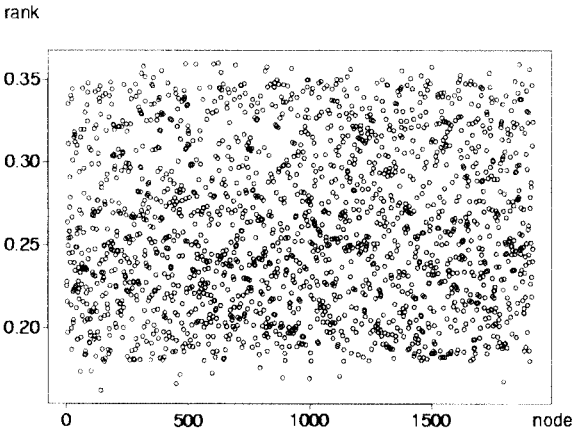
2장에서 소개된 순위 결정 기법들에는 단일 XML 문서라는 파라메터가 존재한다. 이것은 모델이 적용되는 임의의 "전체 데이터"를 의미한다. 하지만 연속된 로그 데이터를 문서 단위로 나누는 것은 사용자 정의의 범주이므로, 본 평가의 모델들은 다음과 같은 값을 부여한다. 문제의 파라메터를 doc라고 정의할 때, (그림 4) (b)에 사용된 데이터는 서비스 중인 DBMS의 로그로서 해당 데몬의 로그 데이터를 doc로 정의한다. 또한 (그림 5) (a), (그림 5) (b)는 FTP 서



(그림 3) Relationship Tree



(a) 기존 XML 로그 구조



(b) 프로세스 기반의 로그 트리 구조

(그림 4) 로그 데이터의 우선 순위 분포

버 로그 데이터의 일부분이며, 임의의 기준 키워드와 대상 키워드, 그리고 그들 사이의 노드들을 doc 값으로 한다. 평가에 사용된 로그 데이터는 이더넷 기반 네트워크로 연결된 3대의 Linux 시스템에서 전달되었으며 전체 CLF 형식의 크기는 40,160kb, 변환을 거쳐 467,397 라인을 포함한 82,286kb의 XML 로그 데이터가 사용되었다.

4.1 각 노드의 순위

전체 노드들의 집합 N_e에서 노드 e의 순위 결정을 위한 모델은 ElemRank를 기반으로 하지만, 그래프가 아닌 트리 구조에 적용해야 하기 때문에 random jump는 없는 것으로 간주된다.

$$e(v) = d_1 \sum_{(u,v)\in HE} \frac{e(u)}{N_h(u)} + d_2 \sum_{(u,v)\in CE} \frac{e(u)}{N_c(u)} + d_3 \sum_{(u,v)\in CE^{-1}} e(u)$$

[8]에서 d_1, d_2, d_3는 임의의 값이 부여되는 파라메터이며 각각 0.35, 0.25, 0.25로 셋팅되었지만, 본 논문에서는 다음과 같다. tsize가 키워드 혹은 임의의 기준에 의해 지정될 때, 이의 범위를 갖는 Relationship Tree의 노드들과 그들의 부모/형제/자식 노드들을 $f(e)$로 정의해 보자. $f(v)$와 동일한 값을 가지고, 동시에 $f(v)$ 자신 혹은 $c(f(v))$나 $c^{-1}(f(v))$가 아닌 $f(e)$를 $N_h(v)$로 표현할 때, $d_1 = N_h(v) / N_h(v) + N_c(v) + N_{c-1}(v)$로 정의한다. 또한 d_2는 v의 자식 노드의 비율이며 $d_2 = N_h(v) / N_h(v) + N_c(v) + N_{c-1}(v)$로 정의되고, d_3는 e의 부모 노드와의 연결 비율이며 $d_2 = N_{c-1}(v) / N_h(v) + N_c(v) + N_{c-1}(v)$로 표현된다.

(그림 4) (b)에서 "프로세스 기반의 로그 트리"로 변환된 노드들은 다양한 값을 가지고 있음을 알 수 있다. 그러나 (그림 4) (a)와 같이 기존 XML 로그 형식을 사용한다면, 대부분의 경우에서 극단적인 순위값을 얻게 된다. "이는 의미있는 순위 산출"과 배치되는 것이며 기존 구조의 문제점을 말해주고 있다.

4.2 유사도

특정 키워드에 대한 유사도는 벡터 스페이스 모델의 적용으로 계산된다. 예를 들어 (그림 2)의 PsId(36)이 키워드로 주어졌을 때, 이에 대한 PsId(15)의 유사도는 다음과 같다.

〈표 4〉 $f(e) \leftarrow R(PsId36 : PsId15)$에서 term 출현 횟수

connection	dog	postgres	rpc.mountd	stt1	unmounted	15:02:02
2	4	2	2	4	2	2

〈표 5〉 $f(e) \leftarrow R(PsId15)$에서 term 출현 횟수

connection	dog	postgres	rpc.mountd	stt1	unmounted	15:02:02
0	1	0	1	1	1	0

PsId(15)의 부모와 자식, 그리고 이웃을 통틀어 가족 $f(e) \leftarrow R(PsId15)$로 놓고, $f(e) \leftarrow R(PsId36)$과 $f(e) \leftarrow R(PsId15)$를 연결하는 Relationship Tree를 $f(e) \leftarrow R(PsId36 : PsId15)$라고 표현할 때, 이에 속한 모든 노드 값(term)들은 <표 4>, <표 5>와 같이 수집된다.[4] 이들간의 코사인 거리를 측정하면 다음과 같다.

$$(2,4,2,2,4,2,2) \times (0,1,0,1,1,1,0) = 4 + 2 + 4 + 2 = 12$$

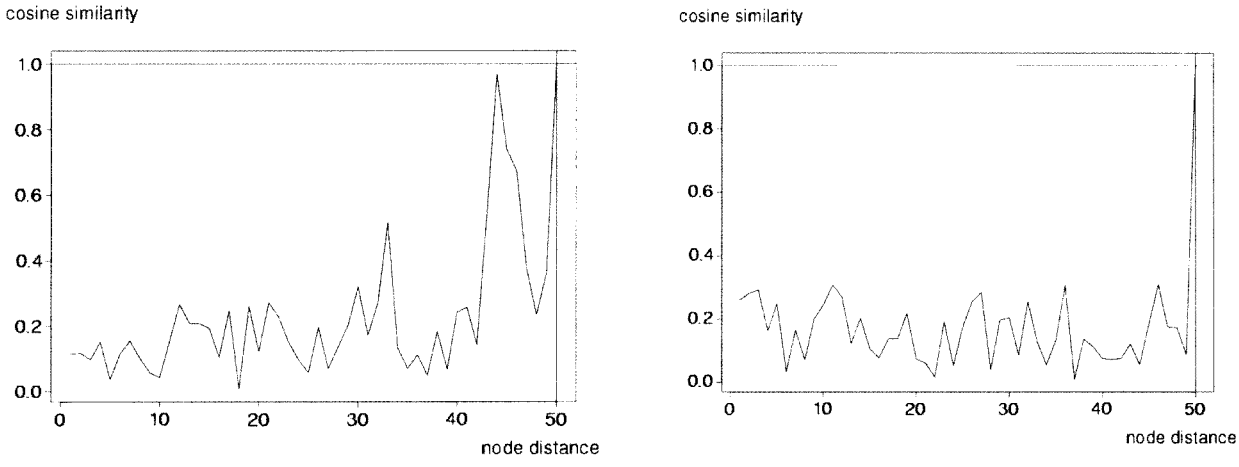
$$|(2,4,2,2,4,2,2)| = \sqrt{(2^2 + 4^2 + 2^2 + 2^2 + 4^2 + 2^2 + 2^2)} = 7.21$$

$$|(0,1,0,1,1,1,0)| = \sqrt{(1^2 + 1^2 + 1^2 + 1^2)} = 2$$

$$\text{cosine similarity} = \frac{12}{21 \times 2} = 0.28$$

(그림 5) (a)와 (그림 5) (b)에서 나타난 유사도는 차이가 있다. 전자에서 키워드에 대한 노드들의 유사도는 임의의 대상 키워드에 근접할수록 순위값이 큰 폭으로 증가한다. 반면 후자의 경우 노드들의 유사도는 대상 키워드에 근접하는 것

4) 본 예에서는 간략한 예시를 위해 일부만 임의로 선정하였다.



(a) 기존 XML 로그 구조의 전형적인 패턴 (b) 프로세스 기반의 로그 트리 구조에서 발생 가능한 패턴 예

(그림 5) 키워드에 대한 각 노드별 유사도

과 큰 연관이 없다. 전자의 결과는 기존 XML 로그 구조에서 항상 유사하게 발생하는 패턴이지만, 제안한 트리구조에서는 전/후자의 패턴 및 그 이외의 비정형적인 패턴이 다수 발생하였다. 이는 제안한 구조가 일률적인 결과값을 산출하는 한계에서 자유로움을 의미하며, 특정 키워드에 대한 유사도가 노드 내 문자열 패턴뿐만 아니라 주변 노드와 트리상의 위치에 따라서도 다양한 값으로 산출될 수 있다는 것을 보여준다.

4.3 의미론적 검색 시나리오

로그를 분석하는 관점과 데이터의 성격은 매우 다양하기 때문에 "프로세스 기반의 로그 트리"가 사용자에게 얼마나 유용한지는 일률적으로 결론짓기 어려우나, 본 논문에서는 간단한 사례를 통해 제안의 유용성을 확인한다.

실험에 사용한 로그에는 2004년 9월 13일경 특정 B 클래스 네트워크의 몇몇 아이피에서[5]실험용 호스트에 연속적인 ssh 접속을 시도하고 실패한 기록이 존재한다. 관리자는 최초의 로그인 실패 기록을 기준으로 다음과 같은 질의를 제시하였다.

$$Q_{tag+kw}(+host:165.194,+PsId:sshd)$$

질의는 XML 트리의 각 노드들에게 고유의 무게를 할당하고, 질의 기준이 되는 노드와의 유사도를 측정하는 것으로 진행된다. 이어서 질의 조건과 일치하는 태그 및 키워드를 가진 노드를 선택하고, 이들이 획득한 점수에 따라 질의 결과값을 확인한다.

<표 6>은 기준 노드와 관련 있는 노드들 및 이들이 의미론적 검색을 통해 획득한 순위를 나타낸다. 질의에 시간은 포함되어 있지 않지만, 가장 높은 순위를 차지한 노드들은 기준 노드의 발생 시점과 관련도가 높음을 직관적으로 알 수 있다. 따라서, 다량의 질의 결과값 중에서 순위를 참조하여 사용자가 원하는 만큼의 결과를 쉽게 얻어 활용할 수 있음을 알 수 있다. 반면에, <표 7>의 결과는 기존의 XML 데이터를 대상으로 한 의미론적 검색의 결과를 보여주고 있다. (그림 4) (a)에서 확인된 바와 같이 결과값의 순위는 세부적이지 못하여 정보로서의 가치가 떨어짐을 알 수 있다.

〈표 6〉 프로세스 기반 XML 로그의 검색 결과

순위	시간	아이피	메시지
1	Sep13 02:47:23	156.142	Authentication failure for root
2	Sep13 02:47:21	156.142	(pam_unix) authentication failure
3	Sep13 02:47:26	156.142	Authentication failure for root
4	Sep13 02:47:24	156.142	(pam_unix) authentication failure
5	Sep13 02:47:25	181.105	Authentication failure for root
6	Sep13 02:47:22	181.105	(pam_unix) authentication failure
7	Sep13 02:47:30	156.142	Authentication failure for root
8	Sep13 02:47:28	156.142	(pam_unix) authentication failure

〈표 7〉 기존 XML 로그의 검색 결과

순위	시간	아이피	메시지
1	Aug27 09:13:32	200.199	Accepted keyboard-interactive
1	Aug27 11:38:06	200.199	session opened for user root
...	...	...	...
1	Sep13 02:47:23	156.142	Authentication failure for root
1	Sep13 02:47:21	156.142	(pam_unix) authentication failure
...	...	...	...
1	Oct25 15:39:52	200.180	check pass; user unknown
1	Oct13 15:39:53	200.180	authentication failure

5. 결 론

반구조적인 특성을 갖는 로그 데이터에 대한 기존의 접근은

5) 보안 및 부수적인 이유 때문에 모든 아이피는 임의의 숫자로 대체하였다.

다른 일반적인 데이터들에 비해 상대적으로 유연하지 못하며 데이터에서 얻게 되는 정보의 양과 질에 문제점이 존재한다. 다시 말해, 데이터에서 정보를 얻기 위해서는 사용자가 적극적으로 질의를 계획하고 리턴된 결과물을 다시 분석해야 한다.

본 논문에서는 반구조적인 로그 데이터를 구조적인 데이터로 변환하였고, 기존에 제안된 XML 로그 데이터 형식들의 문제점과 해결 방법을 제시하였다. 즉, HADP(Human-Active, DBMS-Passive) 모델로 수행되던 기존의 작업이 DAHP(DBMS-Active, Human-Passive) 모델로 변경될 수 있는 기반을 제공한다. 잠재적으로, 본 연구는 정적이고 단순한 사용자 질의 및 이를 가능케 해 주는 DBMS의 능동적인 역할 확대[1]를 지원한다. 이를 위해 본 논문에서는 시멘틱 네트워크와 트리 구조를 사용하였다. 그러나 시멘틱 네트워크 혹은 XML에서 트리가 그래프보다 우월한 근거는 없다. 트리 구조는 단지 현 시점의 편의성을 위한 것으로 트리를 사용한 대상화는 그래프의 그것에 비해 제약이 존재한다. 예로, ElemRank 모델에서 hyperlink로 간주한 것은 실제 XML 표준의 링크인 IDREF가 아니었음을 들 수 있다.

본 논문의 결과는 syslog 데이터 및 이와 유사한 성격을 갖는 반구조적 데이터 처리에 방향성을 제시할 수 있는 것으로 판단한다. 또한, 의미론적 정보 검색을 위해 XML 분야의 연구 결과들을 활용했다는 점에서 또 다른 의의를 찾아볼 수 있다. 본 논문이 포함하고 있지 않은 스키마 레벨의 모델링은 제안한 트리 구조의 제약사항과 맥을 같이 하는 문제이다. 로그 데이터에 대한 보다 나은 모델링이 가능하다면 프로세스 만이 중심이 될 필요는 없을 것이며 이 문제는 중요한 추후 연구 과제가 된다.

참 고 문 헌

[1] B. Babcock and S. Babu and M. Datar and R. Motwani and J. Widom, "Models and Issues in Data Stream Systems", *ACM Symposium on Principles of Database Systems*, pp. 1-16, June, 2002.

[2] C. Lonvick, "The BSD syslog Protocol", RFC3164, 2001.

[3] J. Abela and T. Debeaupuis, "Universal Format for Logger Messages", Hervé Schauer Consultants, http://www.hsc.fr/, 1999.

[4] H. Mannila and H. Toivonen and A. I. Verkamo, "Discovery of Frequent Episodes in Event Sequences", *Data Mining and Knowledge Discovery*, (1, 3), pp.259-289, 1997.

[5] J. Clark and S. DeRose, "XML Path Language", W3C Recommendation, 1999

[6] J. Punin and M. Krishnamoorthy and M. Zaki, "LOGML-Log Markup Language for Web Usage Mining", *Lecture Notes In Computer Science*, Vol.2356, pp.88-112, 2001.

[7] L. Feng and E. Chang and T. Dillon, "A Semantic Network-Based Design Methodology for XML Documents", *ACM Transactions on Information Systems*, (20, 4), pp.390-421, Oct., 2002.

[8] L. Guo and F. Shao and C. Botev and J. Shanmugasundaram, "XRANK: Ranked Keyword Search over XML Documents", In *Proc. 2003 ACM SIGMOD International Conference on Management of Data*, pp.16-27, June, 2003.

[9] Lire Documentation, http://logreport.org/lire/, 2004.

[10] L. Page and S. Brin and R. Motwani and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web", Stanford Digital Library Technologies Project, 1998.

[11] P. Berkhin, "Survey of Clustering Data Mining Techniques", Accrue Software, http://www.accrue.com, 2002.

[12] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison-Wesley Longman Publishing Company, 1999.

[13] R. Gerhards, "The syslog Protocol", syslog Working Group, http://www.ietf.org, 2005

[14] R. Vaarandi, "A Breadth-First Algorithm for Mining Frequent Patterns from Event Logs", *INTELLCOMM*, Vol. 3283, pp.293-308, Nov., 2004.

[15] R. Vaarandi, "A Data Clustering Algorithm for Mining Patterns From Event Logs", *Proc. of the 2003 IEEE Workshop on IP Operations and Management*, pp.119-126, 2003.

[16] S. Cohen and Y. Kanza and Y Sagiv, "Generating Relations from XML Documents", *ICDT*, Vol. 2572, pp.285-299, Jan., 2003.

[17] S. Cohen and J. Mamou and Y. Kanza, and Y. Sagiv, "XSEarch : A Semantic Search Engine for XML", *Proc. of 29th International Conference on Very Large Data Bases*, pp.45-56, Sep., 2003.

[18] XML Interface to Syslog Messages, http://www.cisco.com, 2004.

[19] XML-Logs : Analyse your logs using XML encoding, Hervé Schauer Consultants, http://www.hsc.fr/ressources/outils/xml-logs/index.html.en, 2004.

이 석 준

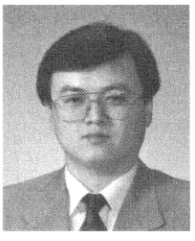
e-mail : seokjoon@gmail.com

2002년 중앙대학교 정보시스템학과(학사)

2005년 중앙대학교 대학원 정보시스템학과 (석사)

관심분야 : Spatial & Time Series Data Handling, Soccer Intelligence

신 동 천



e-mail : dcshin@cau.ac.kr
1985년 서울대학교 컴퓨터공학과(학사)
1987년 한국과학기술원 전산학과(석사)
1991년 한국과학기술원 전산학과(박사)
1991년~1993년 한국전산원 선임연구원
1993년~현재 중앙대학교 정보시스템학과 교수
2004년~2005년 Indiana University Bloomington, School of Informatics 방문 교수
관심분야 : 이동 컴퓨팅, 데이터 웨어하우스, 데이터베이스

박 세 권

e-mail : psk3193@cau.ac.kr
1978년 서울대학교 공과대학 산업공학과 (학사)
1981년 서울대학교 대학원 산업공학과 (석사)
1985년 Texas A&M 대학원 산업공학과 (박사)
1985년~1987년 한국전자통신연구소 선임연구원
1987년~1990년 농촌경제연구원 수석연구원
1990년~현재 중앙대학교 정보시스템학과 교수
관심분야 : System Engineering, Web Engineering