순차 패턴 마이닝을 사용한 두 XML 문서간 최대 유사 경로 추출

(Extracting Maximal Similar Paths between Two XML Documents using Sequential Pattern Mining)

이 정 원 † 박 승 수 †
(Jung-Won Lee) (Seung-Soo Park)

요 약 최근 XML 저장 기법, 질의 최적화, 인덱싱 등의 XML 관련 기술이 활발히 연구되고 있다. 이와 관련하여 하나의 DTD나 XML Schema로 정의된 고정 구조를 공유하는 문서 집합이 아니라 다양한 구조를 가진 문서 집합인 경우 다중 문서간의 구조적 유사성이나 차이점 등을 파악할 필요가 있다. 예를 들어 서로 다른 사이트나 문서 관리 시스템에서 도출된 문서들을 합병하거나 분류할 필요가 있을 때, 문서를 처리하기 위해 공유 구조를 발견하는 일은 매우 중요하다. 본 연구에서는 다양한 문서들의 구조를 구성하는 경로들간의 유사성을 파악하기 위해 기존의 순차패턴 마이닝 알고리즘[1]을 변형하여 두 XML 문서간 최대 유사 경로를 추출한다. 몇 가지 실험을 통해 본 논문에서 제안한 변형된 순차패턴 마이닝 알고리즘이 두 문서간의 최대 유사 경로를 찾아 내고 또한 두 문서간의 정확한 공유 경로 및 최대 유사 경로를 정확히 찾을 수 있음을 보인다. 또한 실험 결과 분석을 위해 최대 유사 경로를 기반으로 정의된 유사성 척도가 XML 문서를 정확하게 분류할 있음을 보인다.

키워드 : XML, 마이닝, 구조 발견, 유사성, 순차 패턴

Abstract Some of the current main research areas involving techniques related to XML consist of storing XML documents, optimizing the query, and indexing. As such we may focus on the set of documents that are composed of various structures, but that are not shared with common structure such as the same DTD or XML Schema. In the case, it is essential to analyze structural similarities and differences among many documents. For example, when the documents from the Web or EDMS (Electronic Document Management System) are required to be merged or classified, it is very important to find the common structure for the process of handling documents. In this paper, we transformed sequential pattern mining algorithms[1] to extract maximal similar paths between two XML documents. Experiments with XML documents show that our transformed sequential pattern mining algorithms can exactly find common structures and maximal similar paths between them. For analyzing experimental results, similarity metrics based on maximal similar paths can exactly classify the types of XML documents.

Key words : XML, mining, structure discovery, similarity, sequential patterns

1. 서 론

현재 가장 활발히 진행되고 있는 XML 관련 연구는 자유롭게 정의된 XML 문서로부터 유용한 데이타를 추출하여 관계형 혹은 객체형 데이타베이스 상에서 운용하기 위한 연구에 그 초점을 맞추고 있다. 따라서 데이타 이주에 필요한 스키마 발견 기법, 인덱싱 기법, 저장 기법, 질의 처리 및 최적화, 그리고 처리 결과에 대한 뷰(view) 정의 등에 연구가 집중되어 있다[2].

그러나 이러한 연구들은 구조가 비슷한 문서를 대상으로 하나의 DTD를 공유하거나 XML Schema 정의를 공유하는 것이 대부분이다. 그 이유는 데이타베이스로 이주하기 위한 대상 문서가 이미 하나의 공통적인 특성을 토대로 작성된 그룹 문서들이 대부분이기 때문이다. 예를 들어 B2B를 위한 기업간의 문서 공유를 위해 기존의 데이타베이스 정보를 XML 문서화 하여 이를 교

† 정 회 원 : 이화여자대학교 컴퓨터학과 교수
jungwony@ewha.ac.kr
sspark@ewha.ac.kr
논문접수 : 2004년 2월 28일
심사완료 : 2004년 6월 30일

환한 뒤 다시 데이타화 하는 반복적인 행위는 이미 교환을 위한 프로토콜이 정의된 XML 문서를 기반으로 한다. 또한 일반적인 XML 사례로 많이 들고 있는 기업 내의 인력 및 급여 정보, 출판물 정보 등은 이미 상식적인 수준에서 문서의 형식, 즉 구조가 정의되어 있는 실정이다.

그러나 일반적으로 XML 문서는 DTD나 XML Schema 정의와 같은 명시적인 구조 정의 없이도 작성될 수 있다. 또한 다양한 구조를 가진 문서를 대상으로 데이타 통합을 하고자 할 수도 있다. 따라서 다양한 구조를 가진 문서를 대상으로 하는 경우, 바로 인덱싱이나 저장 기법을 설계할 것이 아니라 문서 구조들간의 공유 정도를 미리 파악하는 일은 매우 중요하다. 더욱이 다양한 사이트나 데이타베이스, 그리고 문서 관리 시스템으로부터 도출된 문서들을 하나의 시스템에 적용하기 위해 병합하거나 저장하기 위해서는 먼저 문서들의 구조적 유사성을 파악해 볼 필요가 있다.

따라서 본 논문에서는 먼저 두 XML 문서간의 공유 엘리먼트를 찾아 내고 이를 기반으로 유사한 경로를 찾아 낸다. 찾아낸 유사 경로 중에서 중복 경로를 제거함으로써 두 문서간의 최대 유사 경로를 추출할 수 있음을 보인다. 이를 위해 방대한 데이타베이스에서 순서가 있는 아이템의 시퀀스를 선형시간에 찾을 수 있는 순차 패턴 마이닝 알고리즘[1]을 변형한다. 본래 순차 패턴 마이닝 알고리즘을 구성하는 정렬(sort), 빈발항목(litemset) 찾기, 변형(transformation), 시퀀스화(sequence), 최대 빈발(maximal) 시퀀스 추출의 총 5 단계를 모두 두 XML 문서간의 공유 정보를 추출해 낼 수 있도록 변형하고 이를 통해 최종적으로 공유 정보를 요약한 최대 유사 경로를 구한다.

2. 관련 연구

2.1 기존 연구의 문제점

XML 문서의 구조를 발견하는 것은 모든 XML 관련 연구에서 필수적이다. XML 문서의 데이타를 관계형 데이타 베이스로 옮기거나 체계적인 구조로 저장하기 위한 구조 발견을 목적으로 하는 연구들로서 다중 문서를 대표할 수 있는 하나의 구조 추출을 목표로 한다[3]. 여기에서는 하나의 구조 정의를 공유하는 문서집합을 대상으로 약간의 구조 정보의 손실을 감수하고서라도 최적화된 구조를 추출하고 있다. 그러나 하나의 구조를 공유하고 있더라도 실제 문서 인스턴스에 나타나는 구조는 크게 차이가 날 수 있다. 또 완전히 다른 엘리먼트에 다른 구조를 사용하는 문서간의 공유 구조를 파악하기는 더욱 어렵다.

발견된 구조에서 매칭되는 패턴을 찾는 문제는 컴파일러의 최적화 과정에서 트리 패턴 매칭 문제로 오랫동안 연구 되어 왔다[4,5]. 그리고 반구조적인 데이타의 스키마를 발견하는 연구에서도 연관 규칙을 사용하여 빈번히 발생하는 중복된 경로를 추출하는 방법이 있다[6]. 그러나 여러 XML 문서들간 사이의 구조를 비교하기 위해서는 두 가지 연구들을 그대로 이용하기 어렵다. 트리 패턴 매칭 알고리즘은 트리의 일부라고 하더라도 부분표현(subexpression)의 대치를 목적으로 하기 때문에 완전 매칭이어야 한다[4]. 즉 a-b-c-d와 같은 경로에서 a-b, b-c, c-d와 같은 부분 매칭은 가능하나 a-b-d와 같은 매칭은 불가능하다. 또한 연관 규칙을 이용한 반구조적인 데이타의 빈발 경로를 찾는 문제는 한 문서 내에서의 구조의 중복된 패턴을 찾는 방법이다[6]. 따라서 서로 다른 문서 구조간의 유사 경로를 추출하기 위해서는 새로운 방법이 필요하다.

한편 프로그램의 중복된 코드를 찾는 문제는 소프트웨어 유지 보수에 드는 비용을 줄이기 위해서 혹은 프로그램의 표절을 찾기 위하여 등의 목적으로 여러 분야에서 연구되어 왔다[7]. 프로그램의 대상은 프로그래밍 언어로 최근 연구들은 구문 트리(abstract syntax tree)를 이용하여 부분 트리의 중복된 코드를 찾는 방법을 선택하고 있다[8]. 그러나 각 언어의 파서에 의존적이고 여러 언어에 동시에 적용되기는 어렵다고 알려져 있다. 또한 부분 트리의 중복성을 검사하기 위한 알고리즘의 시간 복잡도가 트리의 노드의 개수를 N이라고 했을 때, $O(N^2)$~$O(N^3)$에 이른다[4,8] ($O(N^2)$은 [7]이 제시한 해쉬 함수를 이용하여 비교 대상을 줄임으로써 이룰 수 있다.). 이는 프로그램이 아닌 방대한 웹 문서인 XML을 대상으로 한다면 실제적으로 적용되기 어려운 복잡도이다. 또한 본 논문에서 대상으로 하는 XML은 일반 프로그래밍 언어와는 달리 프로그램의 변수 개념도 존재하지 않고 반드시 따라야 하는 특정 구문도 존재하지 않는 특징을 갖는 마크업 언어이다. 따라서 XML만이 가지는 의미를 고려하여 문서의 계층적인 구조 최소화하여 이를 시퀀스화 함으로써 최대 유사한 경로를 찾을 수 있어야 한다.

2.2 순차패턴 마이닝 알고리즘

본래 순차 패턴 마이닝 알고리즘[1]은 데이타베이스에서 사용자-정의 최소 지지도를 만족하는 트랜잭션의 시퀀스들 가운데 최대 시퀀스를 찾는 알고리즘이다. 연관 규칙[9]과는 달리 트랜잭션의 발생 횟수만이 아닌 발생 순서(sequence)를 고려한다는 점에서 구조를 이루는 경로상에서 엘리먼트들의 순차를 고려 할 수 있다. 이 때 트랜잭션 시퀀스는 '우유', '기저귀', '맥주'와 같이 고객이 구입한 물건에 시간 개념을 부과한 것이다. 따라서 방대한 고객정보 데이타베이스에서 각각의 고객이 구입

한 물품들이 시간에 의해 시퀀스를 형성하고 이 정보들을 기반으로 순차 패턴 마이닝 알고리즘을 적용하여 고객들이 구매하는 물품과 그 순서 정보를 알아 낼 수 있다. 이러한 정보는 개인화(personalization)나 추천시스템(recommendation system)과 같이 개인의 구매성향을 파악하는데 이용되고 있다.

다음 예를 통하여 보다 구체적으로 순차 패턴 마이닝 알고리즘의 동작을 살펴 본다. 표 1은 5명의 고객의 시간에 따른 트랜잭션 시퀀스에서 최대 시퀀스(밑줄 친 항목)를 찾아 내는 프로세스를 보여 주고 있다. 이때, 사용자 정의 지지도는 40%이상으로 설정해 놓은 것으로 적어도 2명 이상의 고객 시퀀스에 나타난 항목만이 남게 된다.

여기에서 고객의 트랜잭션 정보는 XML의 경로 표현과 의미적으로 연관 지을 수 있다. 즉, 사용자 정의 최소 지지도를 만족하는 항목이 살아 남듯, 두 문서간의 공유 경로만을 남긴다면 최종적으로 두 문서간 최대 유사 경로를 추출할 수 있게 된다. 그러나 본 논문에서는 사용자 정의 최소 지지도 개념이 40%와 같은 사용자 설정 값이 아니라 두 문서에 반드시 나타나야 한다는 조건으로 변형된다.

2.3 오토마타를 이용한 구조 형식화 및 최소화

반구조적인 데이타의 구조를 형식화 하는 방법에는 여러 가지가 있지만 본 논문에서는, 오토마타 이론을 적용하여 XML의 구조를 형식화 하고 이를 최소화하는 방법[11]을 사용한다. 이 방법은 다음과 같은 과정을 통해 XML의 문서로부터 최소화된 구조를 추출해 낼 수 있다.

- **NFA를 이용한 XML 문서의 형식화** : 기존의 비결정적 유한오토마타(Nondeterministic Finite Automata : 이하 NFA)는 어느 상태에서 하나의 입력 심벌에 의해 전이될 수 있는 상태가 하나 이상 존재할 경우를 자연스럽게 모델링 할 수 있는 도구이다[10]. 따라서 XML 문서의 엘리먼트의 계층 구조 역시 어느 한 상태에서 하나의 엘리먼트를 보고 갈 수 있는 다음 상태가 하나 이상 존재하므로 NFA를 이용하여 쉽게 형식화 할 수 있다. XML을 위한 NFA는 (Q, D, Σ_N, Σ_T, δ, q0, F)로 정의되며 각각은 유한 상태 집합 Q, 유한 데이타 집합 D, 컨텐트를 유도하지 않고 새로운 상태를 생성시키는 유한 엘리먼트 집합으로 중간 노드들 Σ_N, 직접 컨텐트 D를 유도하는 유한 엘리먼트 집합, 즉 단말 노드들 Σ_T, 전이 함수(입력은 XML 엘리먼트이고 출력은 상태번호) δ, 시작 상태로 루트 엘리먼트 q0, 종결 상태로 컨텐트 D를 유도하는 단말 상태 F를 말한다[11].
- **중복된 엘리먼트 제거를 통한 DFA로의 변경** : 일반적으로 NFA는 문서의 구조를 쉽게 표현할 수는 있으나 구조의 단순화를 위해 엘리먼트의 중복성을 제거해야 한다. 따라서 한 상태에서 엘리먼트를 보고 갈 수 있는 다음 상태가 하나로 결정되는 DFA로 변환해야 한다. 즉, NFA는 같은 구조를 인식할 수 있는 결정적 유한 오토마타(Deterministic Finite Automata : 이하 DFA)로 변환할 수 있는 NFA-to-DFA 변경 알고리즘[10]을 가지고 있으므로 XML문서를 NFA로 형식화 한 뒤, NFA-to-DFA 변경 알고리즘을 사용하여 DFA로 변경하여 한 엘리먼트에 의한 전이를 결정적으로 만들 수 있다.
- **DFA 최소화** : 동일(equivalent)한 전이를 일으키는 상태 집합을 상태-최소화 알고리즘(state-minimization algorithm)[10,11]을 통해 합병한다. 비로서 이 단계를 거쳐야만 하나의 XML 문서로부터 추출될 수 있는 구조가 유일성을 보장 받게 된다. 즉 하나의 NFA로부터 여러 DFA를 만들 수 있지만 동치류(equivalence class)들을 합병함으로써 유일한 DFA를 얻게 된다. 자세한 알고리즘은 [11]에 제시되어 있다.

3. 최대 유사 경로 추출을 위한 변형 절차

본 논문은 두 XML 문서 구조간의 최대 유사 경로를 구하기 위하여 변형된 순차 패턴 마이닝 알고리즘을 제안한다. 따라서 먼저, 본래의 순차 패턴 마이닝 알고리

표 1 순차 패턴 마이닝 알고리즘

고객별 트랜잭션 시퀀스	빈발 1 항목 및 지지도		빈발 2 항목 및 지지도		빈발 3 항목 및 지지도		빈발 4 항목 및 지지도	
			<1 2>	2				
			<1 3>	4				
<{1 5} {2} {3} {4}>	<1>	4	<1 4>	3	<1 2 3>	2		
<{1} {3} {4} {3 5}>	<2>	2	<1 5>	3	<1 2 4>	2		
<{1} {2} {3} {4}>	<3>	4	<2 3>	2	<1 3 4>	3	<u><1 2 3 4></u>	2
<{1 3 5}>	<4>	4	<2 4>	2	<u><1 3 5></u>	2		
<{4 5}>	<5>	4	<3 4>	3	<2 3 4>	2		
			<3 5>	2				
			<u><4 5></u>	2				

즘으로부터 변형된 개념을 요약, 비교 설명하고 알고리즘의 입력이 되는 경로 표현을 추출하기 위한 전처리 과정을 설명한다.

3.1 변형 개념

본래 순차 패턴 마이닝 알고리즘은 정렬(sort), 빈발 항목 발견(litemset), 변형(transformation), 빈발 시퀀스 구하기(sequence), 그리고 최대 빈발 시퀀스 구하기(maximal) 단계로 총 5단계로 구성된다. 본래의 순차 패턴 마이닝 알고리즘은 몇 가지의 시퀀스를 방대한 데이타베이스로부터 찾는데 비해 변형 알고리즘은 기준 문서의 구조를 구성하는 경로 하나하나가 입력이 되고 비교 대상 문서의 구조가 검색을 위한 데이타베이스가 된다. 이렇게 변형 순차 패턴 마이닝 알고리즘에서 본래의 알고리즘으로부터 변형된 개념 및 동작을 요약하면 다음 표 2와 같다.

- **1 단계(정렬-경로 추출)** : 먼저 다음 절에서 설명할 전처리 과정의 결과로 경로 표현을 얻는다. 경로 표현을 구성하는 엘리먼트는 자연스럽게 그 엘리먼트의 레벨로 정렬되어 있다. 즉, 본래의 알고리즘에서는 고객의 ID와 고객이 발생시킨 트랜잭션의 시간에 의해 데이타 베이스가 정렬된다. 그러나 변형 알고리즘에서는 구조를 구성하는 경로는 왼쪽부터 정렬되고(left-most) 경로는 다시 엘리먼트의 레벨로 정렬된다.
- **2 단계(빈발 항목 찾기-유사 엘리먼트 식별)** : 두 문서간의 경로 표현을 구성하는 엘리먼트들 중 유사한 엘리먼트를 식별한다. 이는 길이 1의 빈발 항목(large 1-path)으로 선택된다. 본래의 알고리즘에서는 사용자가 정의한 최소 지지도(minimum support)를 만족시키는 고객의 트랜잭션들이 빈발 항목이 된다. 그러나 변형 알고리즘에서는 두 문서에서 유사하다고 판정되는 엘리먼트가 있다면 이 엘리먼트는 빈발 항목으로 선택된다.
- **3 단계(변형-변형)** : 대부분의 시스템들이 빠른 계산을 위해 데이타를 정수로 변환하듯이 본래의 알고리즘은 고객의 트랜잭션 시퀀스를 정수로 변형한다. 변형 알고리즘 역시 추출된 경로 표현이 정수로 대치되는데, 이 때, 앞 단계에서 식별된 유사 엘리먼트는 같은 정수로 변형된다. 따라서 이러한 유사 엘리먼트를 같은 정수로 매핑 한 테이블을 유지하게 된다.
- **4 단계(시퀀스화-공유 경로 발견)** : 본래의 알고리즘에서 시퀀스화 단계가 핵심이다. 길이를 증가시키면서 최소 지지도에 못 미치는 빈발 항목들을 계속 해서 추려 나가게 된다. 변형 알고리즘에서도 길이 1의 유사한 엘리먼트 정보만을 확장하여 다시 길이 2의 빈발 후보 경로(large candidate 2-path)를 만든다. 길이 2의 빈발 후보 경로는 두 문서에 존재하지 않으면 제거(pruning)하여 길이 2의 빈발 경로(large 2-path)로 채택된다. 여기에서 다시 길이 3의 후보 항목(large candidate 3-path)을 만들고 두 문서에 존재하지 않는 경로는 제거한다. 이와 같은 과정을 반복하여 더 이상 빈발 후보 항목이 발생하지 않을 때까지 수행한다.
- **최대 빈발 시퀀스 추출-최대 유사 경로 추출** : 길이 1에서 n까지의 공유 경로들 중 포함관계가 있는 중복 경로를 제거하면 유일한 두 문서간의 최대 유사 경로가 남게 된다. 이 단계만이 유일하게 본래의 알고리즘과 차이가 없다. 단지 입력이 되는 빈발 항목이 트랜잭션의 시퀀스인가 공유 경로인가에만 차이가 있다.

위와 같이 본래의 순차 패턴 마이닝 알고리즘을 변형하여 XML 문서간의 최대 유사 경로를 추출한다.

3.2 전처리

앞 절에서 설명한 변형 순차 패턴 마이닝 알고리즘을 실행하기 위해서는 두 가지 필수적인 전처리를 요구한다. 첫째, XML 문서로부터 경로 표현을 추출해야만 한

표 2 변형된 개념

본래 순차 패턴 알고리즘		변형된 순차 패턴 알고리즘	
정렬 (Sort)	데이터베이스는 고객 ID와 트랜잭션 시간으로 정렬된다. 따라서 본래의 트랜잭션 데이터베이스는 고객 시퀀스로 구성된 데이터베이스로 변경된다.	경로 추출 (Sort)	최소화 된 XML문서 구조에서 엘리먼트 중심 경로 표현을 추출한다. 경로를 구성하는 엘리먼트는 그 레벨로 정렬된다.
빈발항목 찾기 (Litemset)	최소지지도를 만족하는 빈발항목을 찾는다.	유사 엘리먼트 식별 (Identification)	문서간에 유사하게 사용된 XML 엘리먼트를 식별한다.
변형 (Transformation)	고객 시퀀스는 빠른 검색을 위한 형태로 변경된다.	변형 (Transformation)	모든 추출된 경로들이 정수로 교체되는데 이때 유사하게 사용된 빈발 엘리먼트는 같은 숫자로 재명명 된다.
시퀀스화 (Sequence)	빈발항목 1, 2, …, n 시퀀스를 최소 지지도에 입각하여 찾는다.	공유 경로 발견 (Sequence)	길이 1, 2, …, n의 빈발 경로를 공유 여부에 따라 찾는다.
최대 빈발 시퀀스 추출 (Maximal)	발견된 빈발 시퀀스 중 최대 빈발 시퀀스를 발견한다.	최대 유사 경로 추출 (Maximal)	공유 경로로 판단된 경로 중 최대 유사 경로를 추출한다.

다. 경로를 추출하기 위해서는 XML의 구조를 발견하는 일이 선행되어야 한다. 둘째, 유사 엘리먼트 식별을 위해 엘리먼트간 유사성을 판단할 수 있는 장치가 구축되어야만 한다. [11]은 이러한 전처리 과정을 엘리먼트의 의미간 유사성 계산과 XML 구조 발견 및 최소화로 나누어 자세한 알고리즘과 예를 통해 설명하고 있으며 간략히 소개하면 다음과 같다.

먼저 DTD나 XML Schema와 같은 명시적 구조 정의를 이용하지 않고 문서 인스턴스만을 이용하여 구조를 추출한다. 또한 단일 문서 내에서 레벨 정보를 유지한 구조를 추출하지만 형제 노드들간의 순서 정보는 생략한다. 이를 위하여 유한 오토마타를 이용하여 문서 구조를 형식화하고 이를 다시 최소화 된 오토마타로 변형하여 유일하면서도 최소화된 XML 구조를 얻을 수 있다.

XML 문서는 자유롭게 정의되는 엘리먼트로 인하여 XML 문서에 내재된 의미를 자동으로 해석하기 어렵다. 여러 문서에서 서로 다른 단어를 사용하여 정의된 엘리먼트를 식별해 내기 위해 앞에서 준비된 최소화된 구조로부터 XML 엘리먼트를 추출하고 WordNet[12] 시소러스와 사용자-정의 용어 사전을 이용하여 엘리먼트의 유사어(synonyms)와 합성어(compound words), 약어(abbreviations)를 가진 확장-엘리먼트 벡터(extended-element vector)를 생성한다.

본 논문에서 사용한 WordNet의 버전은 1.7.1로서 144,684개의 단어로 구성되어 있다. 그러나 WordNet으로부터 자동으로 얻을 수 없는 신종 약어나 특수 용어들을 사용자가 직접 정의할 수 있는 사용자-정의 용어 사전이 필요하다. 현재 사용자-정의 용어 사전에는 'book'을 주제 영역으로 하는 문서와 실험 대상이 되는 문서들을 중심으로 <paragraph>, <p>, <para>의 관계, <emphasis>, <emph>, <EM>의 관계, <contents>, <tableofcontents>의 관계, <picture>, <image>등과 같은 유사어 관계를 가진 XML 엘리먼트들을 추출하여 약 100개의 단어들의 유사어 관계가 정의되어 있다. 이러한 다중 문서의 엘리먼트들간의 관계에 따라 유사 행렬이 구축되고 이 행렬을 통해 유사 엘리먼트 식별이 가능하다[11].

구조 발견 및 유사 행렬 구축의 두 가지 전처리 작업을 통하여 본 논문에서 제안한 변형 순차 패턴 마이닝 알고리즘을 실행시킬 수 있는 기본적인 자료인 두 문서의 경로 표현을 준비하였다. 다음 장에서는 단계별 알고리즘을 제안하고 예시함으로써 구체적인 동작을 살펴본다.

4. 알고리즘

이제 XML 문서간 최대 유사 경로를 구하기 위해 변형된 순차 패턴 마이닝 알고리즘을 설명한다. 3.1절의 변형 개념에서 비교한 대로 제안한 알고리즘은 총 5 단계로 경로 추출, 유사 엘리먼트 식별, 변형, 공유 경로 발견, 최대 유사 경로 추출 순으로 설명하고 간단한 사례를 제시한다.

4.1 경로 표현 추출

앞의 전처리 과정에서 이미 설명한 과정대로 문서를 NFA-XML로 형식화 하고 중복된 경로를 제거하여 DFA-XML로 변환한 뒤, 상태 최소화를 한다. 최소화된 XML 구조로부터 경로 표현을 추출한다. 경로 표현은 최소화 된 DFA-XML의 깊이 우선 탐색으로 정렬된다. 이때, 경로 표현은 상태 정보를 생략하고 엘리먼트의 전이를 중심으로 추출된다. 다음 그림 1은 SortStructure 알고리즘으로 다수의 문서를 받아 들여 경로

```
Procedure SortStructure (B:base document, Q1..n : query document)
returns · PEB: set of path expressions of B,      // path expressions are sorted by tree levels
          PEQ1..n : set of path expressions of each Q1..n;
begin
          convert B to NFA-XML of B;
          transform NFA-XML of B to DFA-XML of B ;
          minimize DFA-XML of B ;
          PEB = path expressions from the minimized DFA-XML of B;
          for (k=1; k <= n; k++) do
          begin
                    convert Qk to NFA-XML of Qk ;
                    transform NFA-XML of Qk to DFA-XML of Qk ;
                    minimize DFA-XML of Qk ;
                    PEQk = path expressions from the minimized DFA-XML of Qk;
          end
end
```

그림 1 경로 추출 알고리즘 : SortStructure

표현을 정렬하는 과정을 기술한 것이다. 다중 문서로의 확장을 위하여 기준 문서는 B로, 비교 문서는 $Q_{1..n}$로 설정한다.

< 예 1 : 경로 추출 >

표 3은 책을 판매하고자 하는 두 온라인 서점으로부터 얻은 예제 문서로서 두 XML 문서에서 추출된 경로 표현을 보여 준다. 먼저 하나는 기준 문서로 가정하고 다른 하나는 비교 문서로 본다. 비교 문서의 경로 표현은 기준 문서의 경로 표현에 비해 비교적 간단하다. 또한 '⊥' 표시는 단말 노드로 컨텐트를 의미한다.

4.2 유사 엘리먼트 식별 및 변형

이제 경로 표현으로부터 공유 경로의 빠른 검색을 위하여 모두 정수로 변형한다. 이는 먼저 유사 엘리먼트를 식별할 수 있도록 전처리에서 정의된 유사 행렬로부터 정보를 얻어 유사 엘리먼트 매핑 테이블을 구축한다. 매핑 테이블에서 유사 엘리먼트 혹은 동일 엘리먼트는 같은 숫자로 재 명명된다. 이를 길이 1의 빈발 경로(large 1-path)로 간주한다.

이와 같은 수행을 기술한 알고리즘은 다음 그림 2와 같다.

<예 2 : 유사 엘리먼트 식별 및 변형>

표 3의 PE_B와 PE_{Q1}의 경로 표현과 3.2절의 전처리

표 3 경로 표현

기준 문서의 경로 표현 (PE_B)	비교 문서의 경로 (PE_{Q1})
book.title.⊥	
book.bookinfo.page.⊥	
book.bookinfo.paperback.⊥	
book.bookinfo.edition.⊥	bookinfo.title.⊥
book.bookinfo.date.⊥	bookinfo.price.list.⊥
book.bookinfo.publisher.⊥	bookinfo.price.our.⊥
book.bookinfo.ISBN.⊥	bookinfo.author.name.⊥
book.bookinfo.size.⊥	bookinfo.tableofcontents.section.chap.⊥
book.buyinginfo.price.list.⊥	bookinfo.reviews.reviews_no.⊥
book.buyinginfo.price.our.⊥	bookinfo.reviews.avg_rating.⊥
book.buyinginfo.price.save.⊥	
book.contents.section.chap.⊥	
book.reviews.customer.rating.⊥	
book.writer.name.⊥	

```
procedure Identification&Transformation ( PE_B: path expressions of a base document,
                                          PE_Q1..n : path expressions of query documents,
                                          SM : similarity matrix between elements)
returns L_1^{B•Q1..n}; // all large 1-paths between PE_B and PE_Q1..n
begin
        initialize MappingTable;        // build a mapping table
        for (k=1; PE_B ≠ nil or PE_Q1..n ≠ nil; k++) do
        begin
                E = read (one element from PE_B or PE_Q1..n);
                if (E ∉ MappingTable)
                then   if (similar element of E in SM ∉ MappingTable)
                       then insert MappingTable(k , E);
                       else append MappingTable(similar element of E, E)
                                                  // with the same index of similar element of E
        end
        // elements in path expressions are replaced by integers
        map elements of PE_B and PE_Q1..n to integers in MappingTable;
        generate new path expressions in which elements are represented in integers;
        // find all large 1-paths L_1^{B•Q1..n}
        for (k=1; k <= n ; k++) do
                foreach element E in PE_Qk do
                        if (E ∈ PE_B)   // minimum support 100% between PE_B and PE_Qk
                        then insert E into L_1^{B•Qk};
                end
        end
end
```

그림 2 유사 엘리먼트 식별 및 변형 알고리즘 : Identification&Transformation

과정에서 설명한 유사 행렬을 이용하여 유사 엘리먼트 매핑 테이블을 구축한다. 예를 들면 표 3의 기준 문서의 첫번째 경로 book.title.⊥와 두번째 경로인 book. bookinfo.page.⊥의 경우, 제일 먼저 나오는 엘리먼트인 <book>은 1, 그 다음 <title>은 2로 매핑하고 두 번째 경로의 <book>은 이미 1로 매핑되어 있으므로 <bookinfo>를 3, <page>를 4로 매핑하게 된다. 이러한 순서로 기준 문서의 엘리먼트를 모두 정수로 매핑하고 비교 문서의 엘리먼트를 매핑 하게 된다.

한편, 전처리 과정에서 구축한 유사 행렬에는 (author, writer, 7), (products, goods, 7), 또는 (bib, bibliography, 3)과 같이 서로 유사한 엘리먼트를 쌍으로 하여 유사한 정도를 판단하기 위한 척도를 적용한 수치 정보가 들어 있다[11]. (여기에서 수치는 0부터 7까지의 척도로 숫자가 크면 클수록 유사하다). 기준 문서의 매핑이 끝나고 비교 문서의 엘리먼트들을 매핑 하게 될 때, 유사 행렬의 정보를 기준으로 유사 엘리먼트를 식별하게 되어 비교 문서의 <writer>는 새로운 번호로 매겨지지 않고 <author>의 22번으로 매핑 된다.

따라서 기준 문서의 엘리먼트를 순서대로 정수로 변환하고 비교 문서의 엘리먼트를 변환하기 위해 유사 행렬을 검색하여 유사한 엘리먼트는 동일 정수로 변환하게 된다. 즉 <writer>와 <author>, <contents>와 <tableofcontents>, 그리고 <rating>과 <avg_rating> 등은 같은 숫자로 표현되며 모두 변환하면 다음 표 4와 같다.

표 4의 매핑 테이블을 이용하여 표 3의 경로 표현을 변형하면 다음 표 5와 같다. 이제 정수로 표현된 경로를 생성함으로써 빠른 검색 및 비교가 가능한 자료 구조를 생성하게 된 것이다. 정수로 표현된 경로는 앞에서 설명한 순차패턴 마이닝 알고리즘에서 사용된 고객의 트랜잭션 시퀀스와 비교해 볼 때 하나하나의 트랜잭션은 엘

표 4 매핑 테이블

No	Element	No	Element	No	Element
1	book	9	isbn	17	section
2	title	10	size	18	chap
3	bookinfo	11	buyinginfo	19	reviews
4	page	12	price	20	customer
5	paperback	13	list	21	rating, avg_rating
6	edition	14	our	22	writer, author
7	date	15	save	23	name
8	publisher	16	contents, tableofcontents		

표 5 변형된 경로 표현

문서	본래의 경로 표현	변경된 경로 표현
기준 문서 PE_B	book.title.⊥	<{1} {2}>
	book.bookinfo.page.⊥	<{1} {3} {4}>
	book.bookinfo.paperback.⊥	<{1} {3} {5}>
	book.bookinfo.edition.⊥	<{1} {3} {6}>
	book.bookinfo.date.⊥	<{1} {3} {7}>
	book.bookinfo.publisher.⊥	<{1} {3} {8}>
	book.bookinfo.ISBN.⊥	<{1} {3} {9}>
	book.bookinfo.size.⊥	<{1} {3} {10}>
	book.buyinginfo.price.list.⊥	<{1} {11} {12} {13}>
	book.buyinginfo.price.our.⊥	<{1} {11} {12} {14}>
	book.buyinginfo.price.save.⊥	<{1} {11} {12} {15}>
	book.contents.section.chap.⊥	<{1} {16} {17} {18}>
	book.reviews.customer.rating.⊥	<{1} {19} {20} {21}>
	book.writer.name.⊥	<{1} {22} {23} >
비교 문서 PE_{Q1}	bookinfo.title.⊥	<{3} {2}>
	bookinfo.price.list.⊥	<{3} {12} {13}>
	bookinfo.price.our.⊥	<{3} {12} {14}>
	bookinfo.author.name.⊥	<{3} {22} {23}>
	bookinfo.tableofcontents.section.chap.⊥	<{3} {16} {17} {18}>
	bookinfo.reviews.reviews_no. ⊥	<{3} {19} {24}>
	bookinfo.reviews.avg_rating.⊥	<{3} {19} {21}>

리먼트 자체가 되고 트랜잭션의 발생 시간은 엘리먼트의 레벨이 되어 루트로부터 단말 노드에 도달하는 시퀀스로 변형된다.

여기에서 변형된 경로를 구함과 동시에 길이 1의 빈발 경로(large 1-path)로서 <2>, <3>, <12>, <13>, <14>, <16>, <17>, <18>, <19>, <21>, <22>, <23>을 얻을 수 있다. 이는 매핑 테이블에서 유사하다고 판단되어 동일 정수로 변환된 엘리먼트들이다.

4.3 공유 경로 발견 및 최대 유사 경로 추출

이 단계는 앞에서 준비된 매핑 테이블 정보로부터 알 수 있는 길이 1의 빈발 항목 L_1과 두 문서의 정수로 변형된 경로 표현을 입력으로 한다. 두 문서간 공유 경로 발견을 위한 알고리즘은 다중 패스로 구성된다. 첫 패스에서는 길이 1의 빈발 경로(이하 L_1)를 씨앗 값으로 후보-생성(candidate-generation) 알고리즘[1]을 이용하여 길이 2의 후보 경로(C_2)를 구한다. 더 나아가 이러지는 패스들에서는 길이 2에서 n에 이르는 L_2, … L_n의 빈발 경로를 발견하기 위하여 본래 순차 패턴 마이닝 알고리즘의 'AprioriAll' 알고리즘[1]을 이용한다. 더 이상의 후보 경로가 발생하지 않으면 알고리즘을 종결한다. 구해진 L_2, … L_n의 빈발 경로 중 중복되는 경로를 제거하여 최대 유사 경로를 추출할 수 있다. 다음 그림 3은 알고리즘 Sequnence&Maximal을 보여준다.

< 예 3 : 공유 경로 발견 및 최대 유사 경로 추출 >

다음 표 6은 길이 1의 빈발 경로를 토대로 길이를 확장해 나가면서 빈발 경로를 구하는 과정을 보여 주고 있다.

```
procedure Sequence&Maximal (L1^{B•Q1..n}: all large 1-path expressions between a base document B
                                          and each query document,
                            PE_B: path expressions of a base document,
                            PE_Q1..n : path expressions of query documents)
    returns ML^{B•Q1..n}; // all maximal large similar paths between PE_B and PE_Q1..n
begin
    for (i=1; i <= n; i++) do
    begin
        for (k=2; L_{k-1}^{B•Qi} ≠ ∅; k++) do
        begin        // same as apriori-generate function of sequential patterns [10]
            C_k = New candidate-paths generated from L_{k-1}^{B•Qi};
            foreach path expressions in PE_B and PE_Qi do  // pruning
                if (C_k ∈ PE_B and C_k ∈ PE_Qk) // exists on the both
                then L_k^{B•Qi} = C_k ;
            end
            length = k; // longest length of large paths L_k^{B•Qi}
        end
        for (j = length; j >= 1; j--) do      // maximal phase
            foreach j-large paths, L_j^{B•Qi} do
                delete all sub-paths of L_j^{B•Qi} ;
            end
    ML^{B•Qi} = { remained large paths in L^{B•Qi} }
    end
end
```

그림 3 공유 경로 발견 및 최대 유사 경로 추출 알고리즘(Sequence&Maximal)

표 6 빈발 경로 L_2 찾기

길이 1의 빈발 경로 (L_1)	길이 2의 후보 경로 (C_2)	길이 2의 빈발 경로 (L_2)
<2>, <3>, <12>, <13>, <14>, <16>, <17>, <18>, <19>, <21>, <22>, <23>	<2,3>, <2,12>, <2,13>, <2,14>, <2,16>, <2,17>, <2,18>, <2,19>, <2,21>, <2,22>, <2,23>, <3,12>, <3,13>, <3,14>, <3,16>, <3,17>, <3,18>, <3,19>, <3,21>, <3,22>, <3,23>, **<12,13>**, **<12,14>**, <12,16>, <12,17>, <12,18>, <12,19>, <12,21>, <12,22>, <12,23>, <13,14>, <13,16>, <13,17>, <13,18>, <13,19>, <13,21>, <13,22>, <13,23>, <14,16>, <14,17>, <14,18>, <14,19>, <14,21>, <14,22>, <14,23>, **<16,17>**, <16,18>, <16,19>, <16,21>, <16,22>, <16,23>, **<17,18>**, <17,19>, <17,21>, <17,22>, <17,23>, <18,19>, <18,21>, <18,22>, <18,23>, **<19,21>**, <19,22>, <19,23>, <21,22>, <21,23>, **<22,23>**	<12, 13> <12, 14> <16, 17> <16, 18> <17, 18> <19, 21> <22, 23>

먼저 빈발 경로 L_1을 토대로 길이 하나를 확장하여 순서쌍을 구하면 C_2를 구할 수 있다. 물론 확장은 L_1에 있는 항목만을 대상으로 한다. 여기에서 다시 두 문서에 존재하는 경로(진하게 표시된 항목)이면 남기고 그렇지 않으면 제거(pruning)한다. 남은 경로는 L_2가 된다.

표 7 빈발 경로 L_3 찾기

길이 2의 빈발 경로 (L_2)	길이 3의 후보 경로 (C_3)	길이 3의 빈발 경로 (L_3)
<12, 13>, <12, 14>, <16, 17>, <16, 18>, <17, 18>, <19, 21>, <22, 23>	<16, 17, 18 >	<16, 17, 18 >

L_3 찾는 과정에서 확실히 검색 및 비교 시간을 줄일 수 있다는 것을 알 수 있다. AprioriAll 알고리즘을 통하여 길이 2의 빈발 경로부터는 L_2의 마지막 엘리먼트와 첫번째 엘리먼트가 동일한 경로만을 가지고 확장한다. 따라서 표 7에서 L_2를 C_3로 확장되는 엘리먼트는 <16, 17> 그리고 <17, 18>로 유일하며 이를 확장하면 <16, 17, 18> 이 된다. 그리고 이 구조가 두 문서에 존재하므로 L_3가 된다.

다시 빈발 경로 1..n 까지를 나타내면 다음 표 8과 같다. 이중 포함관계가 성립하는 부분 집합을 제거하면 최대 유사 경로가 되며 이를 및 줄로 표현하였다.

표 8 최대 유사 경로 추출

Large 1 paths	Large 2 paths	Large 3 paths
<{2}>, <{3}>, <{12}>, <{13}>, <{14}>, <{16}>, <{17}>, <{18}>, <{19}>, <{21}>, <{22}>, <{23}>	<{12} {13}> <{12} {14}> <{16} {17}> <{16} {18}> <{17} {18}> <{19} {21}> <{22} {23}>	<{16} {17} {18}>

이렇게 추출된 최대 유사 경로를 다시 매핑 테이블에 근거하여 엘리먼트 이름으로 표현하면 다음과 같다. 즉 두 문서에서 공유 하고 있는 최대 유사경로를 나타낸다.

표 9 엘리먼트로 표현한 최대 유사 경로

Large 1 paths	Large 2 paths	Large 3 paths
title bookinfo	price.list price.our reviews.rating (=reviews.avg_rating) writer.name (=author.name)	contents.section.chap (=tableofcontents. section.chap)

길이 1의 유사 경로로부터 길이 3까지 총 7개의 최대 유사 경로가 추출되었다. 추출된 최대 유사 경로를 실제 XML 문서의 경로에 맞추어 보면 예제 문서가 상당히 비슷한 경로를 공유하고 있음을 알 수 있다.

지금까지 두 XML 문서간의 최대 유사 경로를 추출하기 위해 변형된 순차 패턴 마이닝 알고리즘을 제시하였다. 다음 장에서는 실제 웹 문서를 대상으로 실험한 결과를 보여 줌으로써 정확히 문서간의 최대 유사 경로를 추출할 수 있음을 보인다.

5. 실 험

5.1 실험 대상 및 환경

이 장에서는 제안한 변형 순차 패턴 마이닝 알고리즘이 실제 XML 문서간 최대 유사 경로를 잘 추출할 수 있는지를 실험한 결과를 보여준다. 실험은 다음 두 가지로 진행되었다.

- 실험 1 : 하나의 전자 상거래 사이트내의 문서들로 물건을 판매하고자 하는 목적은 같지만 기술하고 있는 정보가 판매 품목에 따라 조금씩 차이가 발생한다. 다시 말해 유사한 구조를 가지는 동일 사이트 내의 문서들을 대상으로 하였다. 실험에 사용된 문서들은 Amazon의 온라인 상점의 상위 카테고리하에 있는 문서들로서 25.6Mb 크기의 661개의 html 파일을 대상으로 한다. 여기에서 수집된 html 파일은 html 파서인 Jtidy[13]를 이용하여 태그가 정돈된 xhtml 파일로 변환하고 다시 일부 정보들을 태그화 하는 번역기를 통해 XML 문서로 변환하여 사용된다.
- 실험 2 : 하나의 방대한 사이트내의 문서이지만 전혀 다른 목적을 가지고 기술된 문서들로 거의 유사 점이 없다. 하지만 사용된 태그의 일부가 같고 정보 제공을 위한 부분적인 목적이 일치하므로 적은 부분에서 유사 경로가 발생한다. 실험에 사용된 문서는 Yahoo!의 금융 관련 사이트에서 18.3Mb 크기의 783개의 파일을 대상으로 한다.

여기에서 완전히 다른 사이트의 완전히 다른 문서는 굳이 실험할 이유가 없다. 이는 공유 구조를 추출할 이유가 없기 때문이다. 또한 완전히 다른 사이트의 유사한 문서는 실험 2로 대치될 수 있다. 그 이유는 눈으로 보아도 전혀 다른 문서를 대상으로 하였기 때문이다. 그러나 실제 숨어 있는 정보들 중 일부가 유사하다는 것을 알 수 있었다.

또한 실험을 위해 Windows NT, 198M RAM 환경에서 본 논문에서 제안한 변형 순차 패턴 마이닝 알고리즘을 구현하였다. 사용 언어는 Java(JDK1.3.1)로서 Silfide XML parser(v0.89), WordNet v1.7.1 패키지, Porter Stemmer Java Class를 이용하였고 시각환경으

로는 JBuilder 4를 사용하였다.

실험 결과의 정확성을 확인하기 위해서는 문서간의 경로 하나 하나를 일일이 조사하는 휴리스틱한 방법이 아닌 자동화된 방법을 사용하기 위해서 간단한 유사성 척도를 정의하였다. 즉, 유사성은 기준 문서의 경로의 수를 기준으로 최대 유사 경로가 차지하는 비율로 다음과 같이 정의한다.

$$\text{유사성} = \frac{\text{두 문서에서 추출된 최대 유사 경로의 수}}{\text{기준 문서 경로의 수} + 1}$$

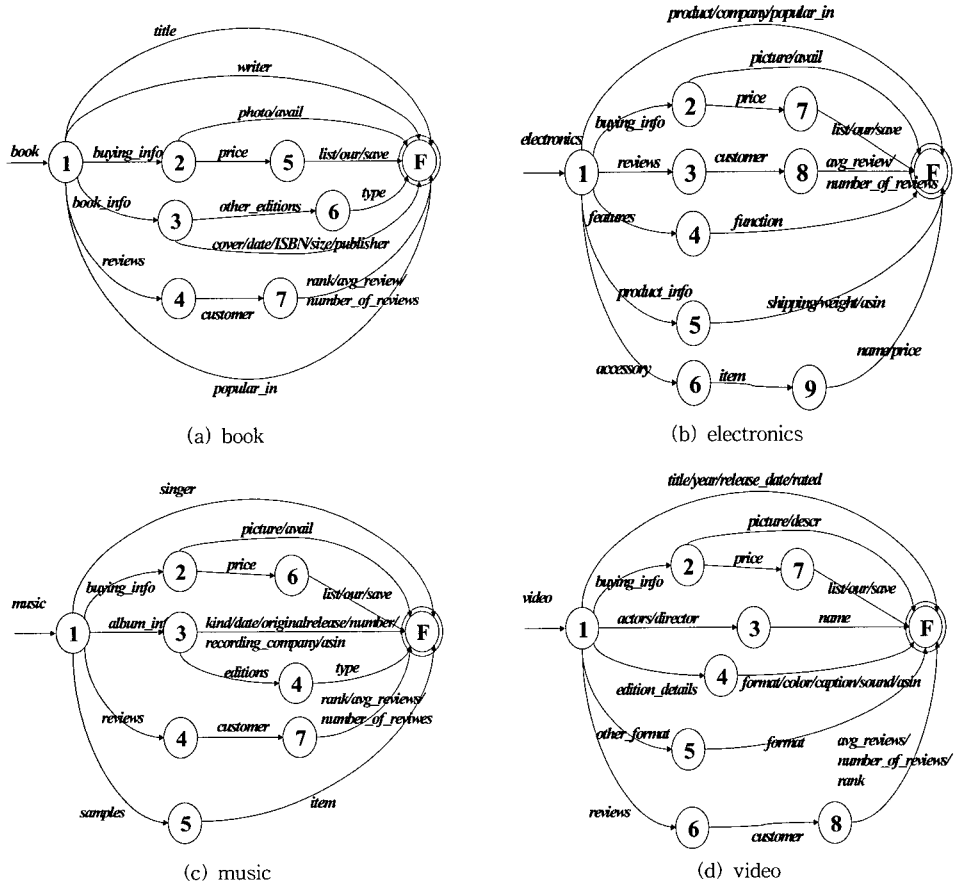
이때 두 문서에서 추출된 최대 유사 경로가 완전한 경로이지 못하고 부분 경로이면 다시 부분적인 확률을 곱한다. 예를 들어 A→B→C의 기준 경로에 최대 유사 경로 A→C는 2/3개로 취급한다. 또한 루트 엘리먼트의 과도한 영향(예를 들면 루트 엘리먼트는 모든 경로의 루트가 되기 때문에 루트 엘리먼트 하나만 다르더라도 전체 유사성을 매우 격감시킬 수 있다)을 막기 위해 루트 엘리먼트는 별도의 경로로 보고 계산한다. 따라서 기준 경로의 수에 1을 더한다. 실험 결과의 정확성은 최대 유사 경로를 기준으로 계산된 유사성에 의해 문서를 분류하고 분류된 문서들이 자신의 카테고리로 분류되었는지를 검사함으로써 알 수 있다.

5.2 실험 1

다음 그림 4는 실험 1에서 사용된 문서들의 범주와 대표 구조를 보여 준다.

각각의 범주는 책, 전자 제품, 음악 CD, 비디오 테잎을 '판매'하고자 하는 목적은 같지만 기술하고 있는 정보가 판매 품목에 따라 다르다. 이제 각각의 문서들의 쌍에 대해 본 논문에서 제안한 변형 순차 패턴 마이닝 알고리즘을 적용하면 다음 표 10과 같이 최대 유사 경로 및 그 개수를 정확하게 추출할 수 있다. 또한 실험을 위해 정의된 간단한 유사성 척도를 이용하여 두 대표 문서간의 유사성을 계산한 결과도 나타내었다.

본 논문에서 제안한 알고리즘을 통하여 문서들간의 최대 유사 경로를 추출할 수 있음을 보였다. 그러나 electronics와 music 문서의 'item'과 같은 엘리먼트는 사용된 의미가 조금 다르다. 하나는 전자 제품과 동반되



(a) book (b) electronics

(c) music (d) video

그림 4 실험에 사용된 문서의 범주별 대표적인 구조

표 10 각 대표 문서간의 최대 유사 경로 (실험 1)

books : electronics (7 개)	**books : music (10)**	**books : video (8)**
buying_info.photo buying_info.avail buying_info.price.list buying_info.price.our buying_info.price.save reviews.customer.avg_review reviews.customer.number_of_reviews	buying_info.photo buying_info.avail buying_info.price.list buying_info.price.our buying_info.price.save date other_editions.type reviews.customer.rank reviews.customer.avg_review reviews.customer.number_of_reviews	title bying_info.photo buying_info.price.list buying_info.price.our buying_info.price.save reviews.customer.rank reviews.customer.avg_review reviews.customer.number_of_reviews
유사성 = 7/18 = 38.89%	유사성 = 59.26%	유사성 = 44.44%
electronics : music (9)	**electronics : video (8)**	**music : video (9)**
buying_info.picture buying_info.avail buying_info.price.list buying_info.price.our buying_info.price.save reviews.customer.avg_review reviews.customer.number_of_reviews asin item	buying_info.picture buying_info.price.list buying_info.price.our buying_info.price.save reviews.customer.avg_review reviews.customer.number_of_reviews asin name	buying_info.picture buying_info.avail buying_info.price.list buying_info.price.our buying_info.price.save reviews.customer.rank reviews.customer.avg_review reviews.customer.number_of_reviews asin
유사성 = 43.51%	유사성 = 36.96%	유사성 = 47.22%

는 작은 소품의 종류와 가격을 열거하기 위한 중간 노드인 반면 다른 하나는 음악 CD를 판촉하기 위한 음악을 샘플링 한 것들을 열거하기 위한 것이다. 이러한 매우 의미적인 차이는 비교 엘리먼트의 상위 노드와 하위 노드를 비교함으로써 같은 이름이지만 서로 다른 의미로 사용된 엘리먼트라는 것을 밝힐 수는 있다. 그러나 계산 복잡도와 지나친 의미의 구별이 오히려 모호함을 가져올 수 있다. 따라서 본 논문에서는 기본적으로 서로 완전히 매칭되는 엘리먼트는 상위, 하위 엘리먼트의 비교 없이 빈발 항목으로 설정한 것이다. 한편 이 문서들은 유사한 엘리먼트의 사용이 극히 미미하고 동일 사이트로부터 추출된 대표 문서이기 때문에 일부 정보, 즉 price→list/our/save, reviews→customer→avg_reviews/number_of_reviews/rank와 같은 구조는 거의 똑 같다.

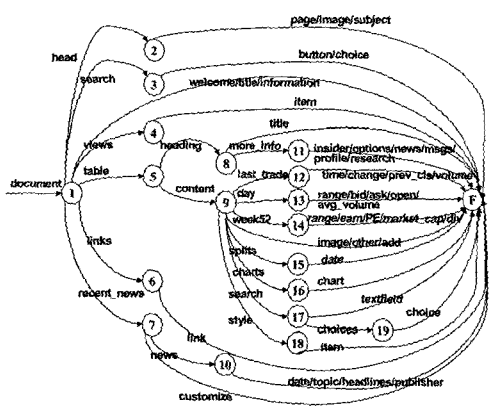
그림 4에서 제시된 각 카테고리 별 문서를 중심으로 661개의 파일을 분류해 본 결과 모두 정확히 자신의 카테고리별로 문서가 분류됨을 알 수 있었다. 즉 표 10에서 제시된 유사성 결과에서 알 수 있듯이 book 카테고리의 문서들을 기준으로 다른 카테고리의 문서들간의 유사성을 계산한 결과 book에 관련된 문서들은 서로 90% 이상의 높은 유사성을 보였고 나머지 문서들은 music, video, electronics 순으로의 60%이하의 유사도를 보였다. 물론 book과 music 카테고리의 문서들은 60%에 달하는 높은 유사성을 보이기는 하였으나 자기 카테고리의 문서와는 30% 이상의 차이를 보임으로써 변별력 있게 분류할 수 있었다.

5.3 실험 2

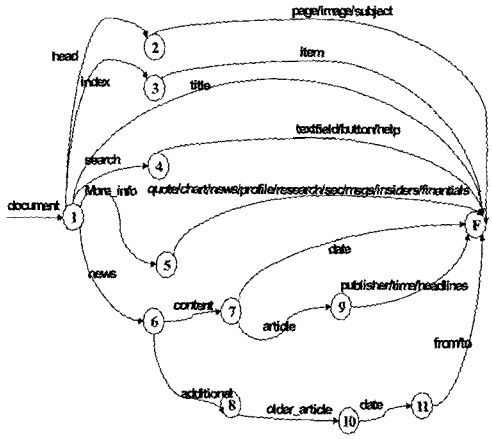
다음 그림 5는 거의 유사점이 없는 문서들로서 Yahoo!의 하위 금융 정보를 담고 있는 사이트로부터 추출된 범주별 대표적인 구조이다.

동일 사이트의 하위 범주의 문서들이기는 하나 각기 다른 모습으로 금융에 관련된 정보를 제공하고 있다. 주식 정보를 차트와 그래프를 이용하여 제공하는 범주(charts), 최근 주식 시장 뉴스 및 금융 정보를 게시판 형태로 보여주는 범주(news), 주식을 가진 회사들의 최근 동향과 주식 정보의 추이 등을 개별적으로 제공하고 있는 범주(profile), 마지막으로 이러한 정보에 관심 있는 개인적인 투자자들의 공유 정보 범주(messages)이다. 이렇게 다른 형태로 페이지를 구성하고는 있으나 그 목적이 금융 정보를 위한 페이지들로서 Yahoo!의 금융 사이트를 구성하는 유사한 목적을 갖는다.

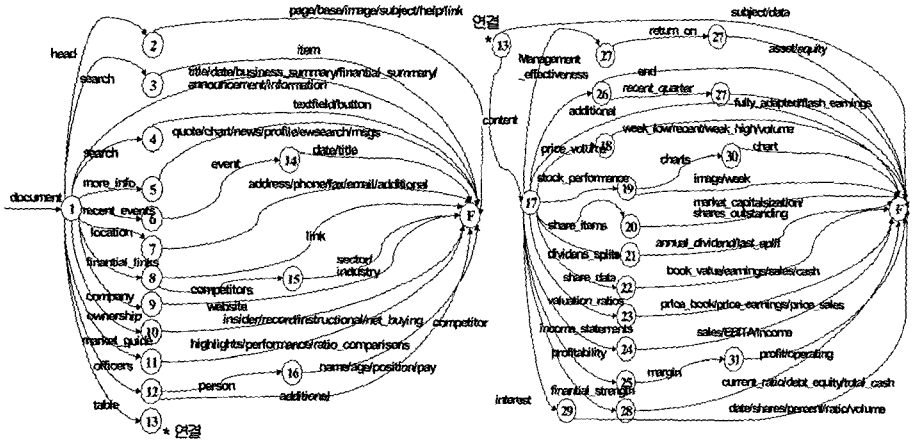
이 문서들은 유사한 구조를 바로 직관적으로 판단하기에는 매우 복잡한 형태이다. 물론 한 사이트에서 정보를 다양한 형태로 제공하고 있기 때문에 그러나 본 논문에서 제시한 알고리즘을 이용하여 이 문서들간의 최대 유사 경로를 추출하면 표 11과 같다. 표 11에서는 chart 문서와 다른 문서간의 결과만 제시하고 다시 news, profile, messages에 대응되는 결과는 생략하였다. 그리고 각각의 빈발 길이에 따른 경로를 순서대로 열거 하였다. 또한 각각의 문서는 모두 "document"라는 루트 엘리먼트를 가지고 출발하기 때문에 본문에 유사한 엘리먼트가 있다면 자동으로 L_2에 속하게 된다.



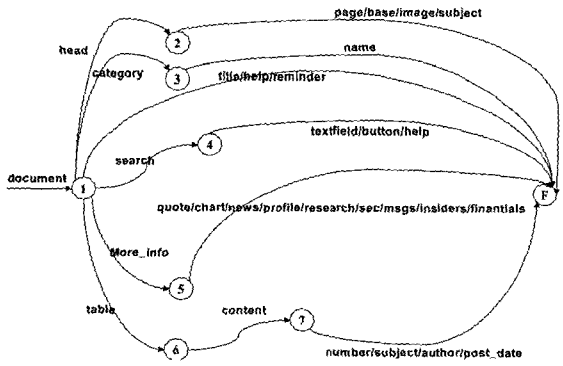
(a) charts



(b) news



(c) profile



(d) messages

그림 5 Yahoo!의 금융(finance) 사이트의 카테고리별 대표적인 문서 구조

그림 5에서 제시된 각 카테고리 별 문서를 중심으로 783개의 파일을 분류해 본 결과 모두 정확히 자신의 카테고리별로 문서가 분류됨을 알 수 있었다. 실험 2에서 사용된 문서들은 매우 복잡한 구조들을 가지며 구조를 구성하는 경로의 수가 profile 카테고리의 문서의 경우 최대 80여 개에 이른다. 유사성 실험은 기준 문서를

표 11 각 대표 문서간의 최대 유사 경로 (실험 2)

charts : news	chart : profile	chart : messages
Large 3 paths : document.head.page document.head.image document.head.subject document.search.button document.more_info.new document.more_info.msg document.more_info.profile document.more_info.research document.more_info.insider document.content.time document.content.date *Large 2 paths :* document.title document.item document.chart document.textfield document.publisher document.headlines	*Large 4 paths :* document.table.content.volume document.table.content.earn document.table.content.date document.table.content.chart *Large 3 paths :* document.head.page document.head.image document.head.subject document.search.button document.more_info.new document.more_info.msg document.more_info.profile document.more_info.research *Large 2 paths :* document.title document.item document.chart document.textfield document.publisher document.headlines document.information document.insider document.link	*Large 4 paths :* document.table.more_info.profile document.table.more_info.research document.table.more_info.insider document.table.more_info.option document.table.content.textfield *Large 3 paths :* document.head.page document.head.image document.head.subject document.search.button *Large 2 paths :* document.title
유사성 = 22.59%	유사성 = 29.81%	유사성 = 19.44%

book으로 설정하고 나머지 문서들과의 유사성을 계산해 내었고 모두 30% 이하의 낮은 유사성을 보였다. 사용된 유사성 척도는 실험의 검증을 위해서 단순히 최대 유사 경로의 수의 비율을 기준으로 설정해 본 것으로 좀 더 보완할 필요가 있다.

5.4 실험 결과 요약

본 논문에서 실험은 유사한 문서들을 대상으로 전자 상거래 판매정보를 기술한 문서들을 다룬 실험 1과 금융정보를 다룬 실험 2로 나누어 실시하였다. 유사점이 전혀 없는 문서를 대상으로 하는 실험은 본 논문에서 찾고자 하는 최대 유사 경로 추출 측면에서 무의미하므로 제외하였다. 실험 1은 판매를 위한 동일한 목적을 가지기 때문에 문서에서 보여지는 구조는 유사한 구조들이 많이 존재한다. 그러나 실험 2는 모든 문서가 금융정보를 담고 있기는 하나 각기 다른 구조로 정보를 제공하고 정보 역시 상이하다. 따라서 표 10과 표 11에서 알 수 있듯이 유사성은 실험 1들의 문서들이 다소 높은 편이다.

실험 결과 각 카테고리별 대표문서와 비교 문서간의 최대 유사 경로는 정확하게 추출됨을 알 수 있었고 다른 대량의 문서 결과의 정확성 결과를 확인하기 위해 계산된 유사성으로 모든 문서들이 100% 정확하게 각 카테고리별로 분류됨을 알 수 있었다.

그러나, 앞에서도 언급한 바와 같이 'item'이나 'content'와 같이 한 문서 안에서도 다양한 의미의 차이를 가져 올 수 있는 엘리먼트는 상하위 엘리먼트와의 비교를 통해 구별될 수 있다. 현재 유사성 척도에는 이러한 경로상의 엘리먼트의 위치와 상하위 관계에 따른 의미적인 차이를 반영할 수 있도록 가중치를 부여 하는 방법을 개발중이다.

6. 결론 및 향후 연구

본 논문에서는 마이닝 분야에서 널리 사용되는 순차 패턴 마이닝 알고리즘을 변형하여 두 XML 문서간의 최대 유사 경로를 추출할 수 있음을 보였다. 실제 알고리즘의 입력이 고객의 물건을 사는 시간적인 행위에서 XML 문서로 크게 변화하면서 순차패턴 마이닝 알고리즘을 그대로 사용할 수는 없었다. 또한 선형시간에 방대한 데이타베이스를 검색하여 최대 시퀀스를 찾는다는 장점을 이용하여 방대한 양의 문서를 처리하는데 이점을 얻을 수 있었다. 본 논문에서 추출된 최대 유사 경로를 기반으로 간단한 유사성 척도를 적용한 결과 모든 XML 문서들이 카테고리별로 100% 정확히 분류될 수 있었다.

현재 본 논문에서 제안한 알고리즘을 이용하여 찾아진 최대 유사 경로를 기반으로 이를 수치화 하여 XML

문서를 자동으로 분류하고자 시도하고 있다. 이는 본 논문에서는 실험 결과 분석을 위해 간단하게 제시된 유사성 척도를 좀 더 보완하여 문서 클러스터링과 분류의 척도로 사용될 수 있도록 할 것이다. 이 외에도 본 논문에서 제시한 최대 유사 경로는 구조가 다른 문서들을 합병하거나 여러 곳에서 얻은 XML 문서를 하나의 시스템에 통합하고자 할 경우에 유용하게 응용될 수 있다.

참 고 문 헌

[1] R. Srikant and R. Agrawal, "Mining Sequential Patterns:Generalizations and Performance Improvements," In *Proc. of the Fifth Int'l Conf. on Extending Database Technology (EDBT)*, March 1996.

[2] J. Shamungasunadarm, Bridging relational Technology and XML, *Dissertation of University of Wisconsin-Medison*, 2001.

[3] Y. Papakonsstantinou, XML and the Automation of Web Information Processing, *Tutorial given at the International Conference on Data Engineering*, 1999.

[4] C. M. Hoffmann and M. J. O'Donnell, "Pattern Matching in Trees," *Journal of ACM 29(1)*, pages 68-95, Jan. 1982.

[5] P. Kilpelainen and H. Mannila, "The Tree Inclusion Problem," In *Proc. the International Joint Conference on the Theory and Practice of Software Development (TAPSOFT' 91), Vol. 1: Colloqium on Trees in Algebra and Programming (CAAP ' 91)*, pages 202-214, 1991.

[6] K. Wang and H. Liu, "Discovering Typical Structures of Documents: a Road Map Approach," *In Proc. of SIGIR*, pages 146-154, 1998.

[7] I. D. Baxter, A. Yahin, L. Moura, M. Sant' Anna, and L. Bier, "Clone Detection using Abstract Syntax Tree," In *Proc. of the ICSM' 98*, Nov. 1998.

[8] 장성순, 서선애, 이광근, "프로그램 유사성 검사기," 제 28회 한국정보과학회 추계학술대회 논문집, pages 334-336, 2001.

[9] R. Agrawal, R. Srikant, "Fast Algorithms for Mining Association Rules," In *Proc. of the 20th Int'l Conference on Very Large Databases*, 1994.

[10] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers : Principles, Techniques, and Tools*, Addison-Wesley, 1986.

[11] J. W. Lee, K. Lee, and W. Kim, "Preparations for Semantics-based XML Mining," In *Proc. of IEEE International Conference on Data Mining (ICDM '01)*, pages 345~352, Nov./Dec. 2001.

[12] C.Fellbaum, *WordNet : An Electronic Lexical Database*, Cambridge:MIT Press. 1998.

[13] Jtidy, http://jtidy.sourceforge.net

이 정 원

1993년 이화여자대학교 전자계산학과 이학사. 1995년 이화여자대학교 전자계산학과 이학석사. 1995년 2월~1997년 5월 LG 종합기술원 연구원. 1997년 9월~2003년 2월 이화여자대학교 컴퓨터학과 공학박사. 2000년 6월~2000년 8월 IBM Almaden (USA) 연구소 인턴쉽. 2003년 3월~2004년 2월 이화여자대학교 공학연구소 박사후연구원. 2004년 3월~8월 이화여자대학교 컴퓨터학과 연구교수. 2004년 9월~현재 이화여자대학교 컴퓨터학과 전임강사. 관심분야는 XML, 데이타마이닝, 프로그래밍 언어

박 승 수

1974년 서울대학교 수학과 학사. 1976년 한국과학기술원 전산학 석사. 1988년 미국 텍사스 대학 전산학 박사. 1988년~1991년 미국 켄사스대학 컴퓨터학과 조교수. 1991년~현재 이화여자대학교 컴퓨터학과 교수. 관심분야는 인공지능, 데이타마이닝, 바이오인포매틱스