XML 문서의 자동변환

이준승, 신동훈, 이경호
연세대학교 컴퓨터과학과

An Automated Transformation of XML Documents

Jun-Seung Lee, Dong-Hun Shin, Kyong-Ho Lee
Dept. of Computer Science, Yonsei University

요 약

XML 문서가 인터넷을 비롯한 다양한 분야에서 정보 교환을 위한 표준으로 널리 사용되면서 XML 문서의 변환에 대한 필요성이 증가하고 있다. 본 논문에서는 XML 문서의 자동 변환 방법을 제안한다. 제안된 방법은 스키마 매칭과 XSLT 스크립트 생성의 두 단계로 구성된다. 특히, 제안된 방법은 정교한 수준의 스키마 매칭을 위해서 동의어 사전, 축약어 사전, 도메인 온톨로지 등의 부가정보를 활용한다. 또한 기존 연구와 비교하여 보다 빠른 변환 속도를 지원하는 XSLT 스크립트를 생성한다.

1. 서론

XML(eXtensible Markup Language) 문서는 논리적 구조정보를 표현할 수 있으며 플랫폼에 독립적이라는 장점 때문에 다양한 분야에서 정보의 공유 및 교환을 위한 표준으로 널리 사용되고 있다.

XML은 사용자가 원하는대로 마크업 요소를 정의하여 사용할 수 있다는 특징 때문에, 동일한 분야에서조차 다양한 스키마를 정의 및 사용할 수 있다. 서로 다른 스키마를 이용하여 작성된 XML 문서를 교환하기 위해서는 두 스키마 사이의 의미적 관계에 따라 정보를 변환하여야 한다. 예를 들어, 전자상거래에서 기업마다 서로 다른 구매요청서 스키마를 이용할 경우, 두 기업의 구매요청서를 서로 교환하기 위해서는 스키마 사이에 의미적 연관관계를 추출한 후 이를 기반으로 XML 문서를 변환하여야 한다. 즉, 서로 다른 스키마에 따라 작성된 XML 문서를 교환 및 공유하기 위해서는 XML 문서간의 변환과정이 수행되어야 한다. 특히 XML 스키마의 종류와 수가 급증함에 따라 XML 문서간의 자동변환이 중요한 이슈로 떠오르고 있다.

일반적으로 XML 문서의 변환 과정은 그림 1과 같이 스키마 매칭과 XSLT (eXtensible Stylesheet Language Transformations)에 기반한 변환 스크립트 생성의 두 단계로 구성된다. 스키마 매칭 단계에서는 소스 스키마와 타겟 스키마를 입력으로 받아 의미적인 매칭관계를 생성하고, 변환 스크립트 생성 단계에서는 전 단계에서 계산된 매칭관계를 이용하여 XML 문서를 변환시킬 수 있는 스크립트를 생성한다.

제안된 방법은 크게 스키마 매칭과 XSLT 스크립트 생성의 두 단계로 구성된다. 스키마 매칭은 단말노드간 후보매칭 계산과 경로유사도를 이용한 일대일(one-to-one) 매칭 추출 과정으로 구분된다. 특히, 정확도를 높히기 위하여 동의어 사전, 축약어 사전, 그리고 도메인 온톨로지에 기반한다.

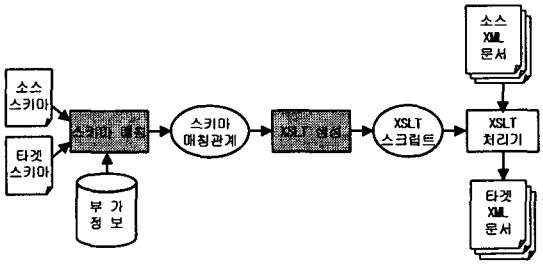

그림 1. XML 문서의 자동변환 시스템

XSLT 생성은 변환 수행속도가 빠른 XSLT 스크립트를 생성하기 위해서 기존 방법보다 적은 수의 템플릿으로 구성하였다.

• 이 논문은 2003년도 한국학술진흥재단의 지원에 의하여 연구되었음. (KRF-2003-003-D00429)

본 논문에서는 알고리즘의 평가를 위해 실제 전자상거래에서 사용되고 있는 스키마를 대상으로 실험하였다. 스키마 매칭의 경우, 평균 97%의 정확률과 81%의 재현률로 높은 정확률을 보였다. 또한 생성된 XSLT 스크립트로 수행속도를 측정한 결과 기존의 다른 연구와 비교하여 우수한 성능을 나타냄을 확인할 수 있었다.

2. 관련연구

본 절에서는 스키마 매칭[1]과 XSLT 스크립트 생성에 관한 기존의 연구결과를 간략히 기술한다.

Xtra[2]는 XML 문서의 자동변환을 위하여 최소 비용의 변환 연산에 해당하는 대응관계를 추출한다. 하지만, 변환 연산에만 치중하여 실제 그 노드가 속하고 있는 구조에 대한 정보를 고려하지 않고, 변환연산에 제한이 많아 실제 스키마에서 일어날 수 있는 삽입 연산 등은 찾지 못한다.

이외에도 학습기법에 기반한 LSD[3]는 소스 스키마 뿐만 아니라 XML 문서에 대한 정보까지 이용하여 매칭을 시도하고 있다. 그럼에도 학습을 위해 필요한 사용자의 수동매칭에 대한 노력이나, 학습을 위한 충분하고 포괄적인 학습데이터에 대한 문제가 해결되야 한다. 이 밖에도 여러가지 방법을 모듈화하여 다양하게 적용하고, 사용자의 피드백을 통해 정확성을 좀 더 향상 시킬 수 있는 시스템에 대한 연구도 진행되고 있다[4].

한편, 기존에 XML 문서간의 변환을 위한 XSLT 스크립트 생성방법이 제안되었다. Kuikka 등 [5]과 Leinonen [6]은 사용자로부터 단말 노드에 대한 레이블 대응관계를 입력받아 중간 노드 대응관계를 생성하고 이를 바탕으로 XSLT 스크립트를 자동으로 생성한다. Tang과 Tompa [7]는 새로운 변환 기술 언어를 제안하고, 제안된 언어를 사용하여 기술한 변환 정보를 XSLT 스크립트로 변환한다.

3. 스키마 매칭

본 논문에서 제안하고 있는 스키마 매칭 알고리즘은 문서의 변환을 위한 스키마 매칭으로 일대일의 단말 노드 매칭관계를 계산한다. 따라서 대부분의 정보를 포함하고 있는 단말노드를 중심으로 후보매칭을 계산하고, 단말노드의 문맥(context) 정보를 포함하고 있는 경로유사도를 비교하여 가장 적절한 매칭을 선택한다. 그림 2는 스키마 매칭 알고리즘의 전체 과정을 도식화한 것이다.

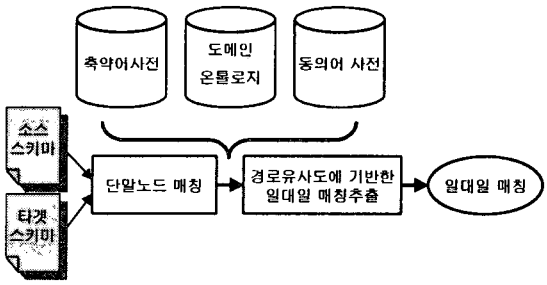

그림 2. 스키마 매칭 알고리즘

3.1 단말노드 매칭

트리로 구성된 두 스키마의 모든 단말노드을 비교하여 유사도가 임계값 이상인 매칭관계를 찾는다. 두 단말노드 N_s와 N_t의 유사도는 식(1)과 같이 노드의 이름을 비교하는 언어적인 유사도와 XML 스키마 명세에서 정의한 데이터 타입에 대한 유사도의 합으로 정의한다.

$$\text{단말노드유사도}(N_s, N_t) = w_l * \text{언어적유사도}(N_s, N_t) + w_t * \text{데이터타입유사도}(N_s, N_t) \quad (1)$$

◆ **언어적 유사도**

주어진 두 노드 이름의 유사도로 먼저 입력된 노드의 이름을 대문자나 특수기호를 기준으로 토큰화하고 문자열 비교 및 부가정보를 이용하여 토큰 사이의 유사도를 계산한다. 언어적 유사도는 식 (2)와 같이 각 토큰 사이의 유사도 평균으로 계산할 수 있다.

$$\text{언어적유사도}(Ns, Nt) = \frac{\sum \text{유사도}(Ts_i, Tt_j)}{|Ts| + |Tt|} \quad (2)$$

Ts_i: 소스 노드의 토큰, $1 \le i \le n$
Tt_j: 타겟 노드의 토큰, $1 \le j \le m$

유사도 계산을 위한 부가정보로는 축약어 사전, 동의어 사전[8], 도메인 온톨로지를 활용한다.

◆ **데이터타입 유사도**

XML 스키마 명세서에서 제공하고 있는 데이터 타입 간의 유사도로서 매칭시 단말노드가 포함하고 있을 정보 손실 정도에 따라 4종류로 나누어 유사도를 부여한다.

3.2 경로유사도에 기반한 일대일 매칭 추출

단말노드 매칭 과정을 통해 얻어진 매칭관계는 다수의 소스노드와 타겟노드가 연결된 다대다 관계일 수 있다. 제안된 방법은 경로 유사도에 기반하여 다대다 관계로부터 일대일 매칭을 추출한다. 본 논문에서는 노드의 부모노드로부터 뿌리노드까지의 노드의 순차적인 집합을 해당 노드의 경로라고 정의한다. 특히 서

로 대응하는 경로 Ps와 Pt사이의 유사도를 경로유사도(Ps, Pt)라고 하며 이에 대한 정의는 식(3)과 같다. 먼저 서로 대응되는 두 경로에 포함되어 있는 중간노드를 비교하며 매칭되는 중간노드를 찾는다. 경로유사도는 두 경로에 포함된 매칭관계를 갖는 중간노드의 비율로 나타낸다.

$$\text{경로유사도}(Ps, Pt) = \frac{Ps\text{와 } Pt\text{사이에 대응관계를 갖는 중간노드의 수}}{|Ps| + |Pt|} \quad (3)$$

Ps:소스 경로, Pt:타겟 경로

경로유사도를 계산하기 위해선 중간노드 사이에 매칭관계를 찾아야 한다. 중간노드 매칭 역시 해당 노드간 유사도를 계산하여 임계값 이상의 값을 보이는 관계를 매칭되었다고 한다. 중간노드 사이의 유사도는 식 (4)와 같이 언어적인 유사도와 구조적인 유사도를 이용하여 구할 수 있다.

$$\text{중간노드유사도}(N_s, N_t) = w_l * \text{언어적유사도}(N_s, N_t) + w_s * \text{구조적유사도}(N_s, N_t) \quad (4)$$

언어적 유사도는 단말노드 매칭과정과 동일한 방법으로 계산한다. 구조적 유사도는 중간노드의 하위 트리에 포함되어 있는 단말노드 중 매칭관계를 가지고 있는 전체 노드에 대한 두 중간노드의 하위트리 안에서 매칭되고 있는 단말노드의 비율로 나타낸다. 즉, 중간노드가 포함하고 있는 서브트리의 단말노드가 유사할 수록 해당 중간노드의 구조적 유사도가 증가한다.

$$\text{구조적유사도}(Ns, Nt) = \frac{|\text{단말노드매칭}(LNs, LNt)|}{|LNs| + |LNt|} \quad (5)$$

LNs : Ns트리(Ns를 루트로 갖는 서브트리)의 대응관계를 갖는 단말노드의 집합

LNt : Nt트리(Nt를 루트로 갖는 서브트리)의 대응관계를 갖는 단말노드의 집합

계산된 경로유사도를 비교하여 일대일 매칭을 계산화는 과정은 소스 스키마를 중심으로 일대다(one-to-many) 관계 중 적절한 일대일 관계를 추출하고, 타겟스키마를 중심으로 다대일 관계 중 적절한 일대일 매칭을 선택하는 과정으로 이루어진다.

즉, 한 소스노드가 여러 타겟노드와 대응관계를 갖는 경우, 경로유사도가 가장 높은 타겟노드를 찾는다. 일대다 관계의 모든 소스노드에 대하여 이 과정을 반복하여 적용한다.

일대다 매칭관계를 제거한다고 해서 일대일 매칭만 남는 것은 아니다. 하나의 타겟노드에 대해 여러 소스노드가 매칭되는 다대일(many-to-one) 매칭관계가 존재할 수 있다. 따라서 이전 단계에서 추출된 매칭 중에서 다대일 매칭을 검색하고 이를 제거하는 과정을 적용한다. 즉, 임의의 타겟노드와 매칭관계를 갖는 소스노드가 다수 존재한다면 해당 매칭 중에서 경로유사도의 값이 가장 큰 매칭을 선택한다. 한편, 경로유사도를 비교하여 가장 유사한 매칭을 찾는 과정에서 동일한 값의 경로유사도를 갖는 매칭이 다수 존재할 수 있다. 이러한 경우, 가장 적절한 매칭의 선택을 위해서 단말노드 유사도가 높은 것을 선택한다. 이같은 과정을 통해 스키마 사이에 최종적인 일대일 매칭이 계산된다.

4. XSLT 스크립트 생성

XSLT 생성 알고리즘은 반복되는 중간노드의 대응관계 생성과 XSLT 스크립트 생성의 두 단계로 구성된다. 첫 번째 단계에서는 대응관계를 갖는 단말 노드의 경로를 비교하여 중간노드 대응관계를 생성한다. 두 번째 단계에서는 중간 노드 대응관계를 기반으로 타겟 스키마에 하향식 깊이 우선탐색 과정을 적용하여 XSLT 스크립트를 생성한다.

4.1 중간 노드 대응관계 생성

제안된 방법은 스키마 매칭의 결과인 단말노드 사이의 일대일 대응관계를 입력으로 한다. 각각의 단말 노드 대응관계에 대해 상향식 방법으로 경로를 비교하여 중간 노드 대응관계를 생성한다. 특히 반복되는 중간 노드들만을 대상으로 대응관계를 생성하며 조상 순서를 위배하는 대응관계는 생성하지 않는다. 중간노드 간의 대응관계 생성을 위해서 Kuikka 등의 방법에서 정의한 TAN(Terminating Associated Nonterminals)[5, 6] 집합 개념을 사용한다.

제안된 방법은 소스와 타겟 스키마에서 빈도수가 1 이상인 중간노드 간의 대응관계만을 고려한다. 단말노드까지의 경로 상에 모든 중간노드의 빈도수가 1이면, 문서에서 단말노드는 한 번만 나타난다. 따라서 대응관계를 갖는 두 단말노드의 경로 상에 반복되는 중간노드가 없는 경우, 중간노드 대응관계를 생성할 필요가 없다. 주어진 단말 노드 간의 대응관계를 사용하여 단말 노드간의 값을 복사하는 것만으로 변환이 가능하다.

또한 한쪽의 경로에만 반복되는 중간노드가 있는 경우에도 중간 노드 대응관계를 생성할 필요가 없다. 한쪽 단말 노드가 문서에서 한번밖에 나타나지 않기 때문에, 빈도수 차이로 인해 하나의 정보밖에 변환될 수 없다. 이 경우도 중간노드 대응관계 생성 없이 단말노드간의 값 복사만으로 변환이 가능하다. 반복되는

노드간에 생성되는 대응관계는 XSLT 스크립트에서 for-each 문을 포함하여 반복되는 구조를 지원한다.

임의의 두 노드에 대해, TAN 집합이 서로 일대일 대응관계를 갖는 경우, 두 노드는 의미상 같은 노드로 간주될 수 있다. 따라서 제안된 방법은 TAN 집합이 서로 일대일 대응관계를 갖는 경우, 두 노드 간에 중간 노드 대응관계를 생성한다.

TAN 집합이 일대일 대응관계를 갖지 않는 경우에도 중간 노드 간에 대응관계가 생성될 수 있다. 이러한 경우, 문맥적 의미를 고려하여 대응관계를 생성한다. 임의의 소스 중간 노드 n과 타겟 중간 노드 n'에 대해 TAN 집합이 서로 일대일 대응관계를 갖지 않는 경우에도, TAN(n)의 노드 중에 대응관계를 갖지 않는 모든 노드가 타겟 문서에서 반복되어 나타날 수 있고, TAN(n')의 노드 중에 대응관계를 갖지 않는 모드 노드가 소스 문서에서 반복되어 나타날 수 있는 경우, 중간 노드 간에 대응관계를 생성한다.

4.2 XSLT 스크립트 생성

반복되는 중간노드간의 대응관계가 생성되면 XSLT 스크립트 생성은 비교적 쉽게 이루어진다. 제안된 XSLT 스크립트 생성 알고리즘은 그림 3과 같다.

입력 : 소스 스티마 트리, 타겟 스키마 트리,
단말노드 대응관계 집합, 중간 노드 대응관계 집합
출력 : XSLT 스크립트

1. 소스 스키마 트리의 루트 노드에 대한 템플릿을 연다.
2. 타겟 스키마 트리를 깊이 우선 탐색 하면서 각각의 노드들에 대한 태그를 열거나 닫는다.
   - 대응관계를 갖는 중간 노드들은 대응되는 소스 중간 노드들에 대한 for-each 태그를 먼저 생성한다.
   - 대응관계를 두 개 이상 갖는 중간 노드는 대응되는 소스 중간노드들 중에서 가장 적게 반복되는 노드에 대한 for-each 태그를 먼저 생성한다.
   - 대응관계를 갖는 단말노드인 경우 3을 수행한다.
   - 모든 노드에 대한 탐색이 끝났다면 4를 수행한다.
3. 소스 단말노드와 타겟 단말노드에 모두 '*'가 붙어있으면 먼저 소스 단말 노드에 대한 for-each 태그를 생성한다.
   - 타겟 단말노드에 대한 태그를 연다.
   - 소스 단말노드에 대한 value-of 태그를 생성한다.
   - 타겟 단말노드에 대한 태그를 닫는다. 2를 수행한다.
4. 소스 스키마 트리의 루트 노드에 대한 템플릿을 닫는다.

그림 3. XSLT 스크립트 생성 알고리즘

제안된 방법은 타겟스키마의 루트 노드에 대한 템플릿을 생성하고, 타겟스키마 트리를 깊이 우선 방식으로 탐색하면서 해당 노드에 대한 태그를 생성한다. 대응관계를 갖는 중간 노드들은 대응되는 소스 중간 노드에 대하여 for-each 태그를 먼저 생성한다.

단말노드 대응관계 집합에 속해있는 각각의 대응관계에 대해 경로를 비교하여 중간 노드 대응관계를 생성하기 때문에, 타겟스키마 트리에서 2개 이상의 대응관계를 갖는 중간 노드가 존재할 수 있다. 하나의 노드와 2개 이상의 대응관계를 갖기도 하고, 서로 다른 노드들과 2개 이상의 대응관계를 갖기도 한다.

하나의 노드와 2개 이상의 대응관계를 갖는 경우는 단순하게 대응되는 소스 중간 노드에 대한 for-each 태그를 사용하여 반복되는 소스 중간 노드를 타겟 중간 노드로 변환한다. 타겟 중간 노드가 2개 이상의 소스 중간 노드와 대응관계를 갖는 경우, 대응되는 소스 중간 노드들 중에서 문서에서 가장 적게 반복되는 중간 노드를 선택하여 for-each 태그를 생성한다.

5. 실험 결과

제안된 스키마 매칭 알고리즘의 평가를 위해서 5개의 구매요청서 스키마를 대상으로 실험하였다. 5개의 스키마를 가지고 각각의 조합으로 10번의 실험을 실시하여, 전문가가 수동으로 매칭한 결과와 시스템을 통해 찾아진 매칭결과를 가지고 정확률과 재현률을 계산하였다.

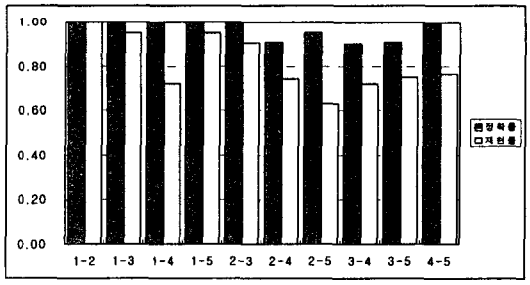

그림 4. 스키마 매칭의 정확률과 재현률

그림 4는 각 실험 데이터에 따른 정확률과 재현률을 그래프로 나타낸 것이다. 평균적으로 97%의 정확률과 81%의 재현율을 보였다. 특히 대부분의 실험에서 높은 정확률을 보임으로 찾아낸 결과는 거의 정확하다고 신뢰할 수 있다.

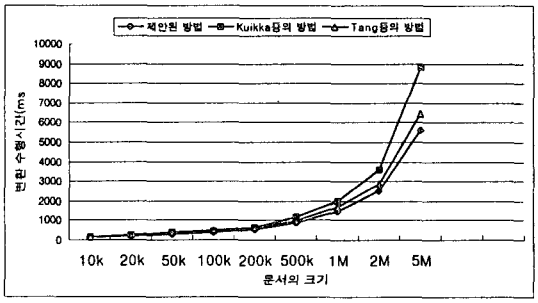

그림 5. XSLT 스크립트의 변환 수행속도 비교결과

한편, 생성된 XSLT 스크립트의 성능을 평가하기 위해서 Kuikka 등의 방법과 Tang 등의 방법의 스크립트와 비교하였다. 제안된 방법은 템플릿 수를 줄임으로써, 생성되는 XSLT 스크립트의 수행 속도를 향상시키는데 목적을 두고 있다. 따라서 전술한 방법들에 의해 생성된 XSLT 스크립트의 변환 수행시간이 문서의 크기에 따라 어떻게 변화하는지를 측정하였다.

실험용 스키마 각각에 대하여 9개의 서로 다른 크기의 XML 문서를 임의로 작성하였고, 각 문서에 대해 각각의 방법에 의해 생성된 XSLT 스크립트를 10번 적용하여 수행시간을 측정한 후 평균을 계산하였다. 실험 결과는 그림 5과 같이 제안된 방법으로 생성한 XSLT 스크립트가 XML 문서를 가장 빠르게 변환하였다.

6. 결론 및 향후연구

본 논문에서는 XML 문서의 자동변환을 위한 스키마 매칭과 XSLT 스크립트 생성을 위한 알고리즘을 제안하였다. 스키마 매칭의 결과인 두 스키마 사이의 일대일 매칭이 XSLT 스크립트 생성을 위한 입력으로 주어진다.

스키마 매칭 알고리즘은 일대일 매칭을 계산하기 위해 단말노드간 후보매칭 집합을 찾고, 경로유사도를 비교하여 최종 매칭을 찾는 두 단계로 구성되며, 정교한 매칭을 위해 다양한 부가정보를 사용한다. XLST 생성 방법은 최대한 템플릿의 수를 줄이는 방법을 제안하여 수행속도를 향상시켰다.

실험결과, 제안된 알고리즘은 스키마 매칭 과정에서 높은 정확률을 보였으며 제안된 방법에 의해 생성된 변환 스크립트도 기존 연구에 비해 수행속도가 빠름을 확인할 수 있었다.

향후 일대일 매칭관계뿐 아니라 병합이나 분할과 같은 복합매칭을 계산하고 변환시킬 수 있는 방법에 대해 연구할 것이다. 또한 매칭의 정확률 향상을 위해 사용자 피드백을 활용하여 동적으로 갱신되는 온톨로지를 활용한 스키마 매칭 알고리즘을 연구할 것이다.

[참고문헌]

[1] Erhard Rahm and Philip A. Bernstein, "A Survey of Approaches to Automatic Schema Matching," VLDB Journal, Vol. 10, No. 4, pp. 334-350, 2001.

[2] Hong Su, Harumi Kuno, and Elke A. Rundensteiner, "Automating the Transformation of XML Documents," Proc. 3rd Int'l Workshop on WIDM, pp. 68-75, 2001.

[3] AnHai Doan, Pedro Domingos, and Alon Halevy, "Learning to Match Schemas of Data Sources: A Multistrategy Approach," Machine Learning, Vol. 50, No. 3, pp. 279-301, 2003.

[4] Hong-Hai Do and Erhard Rahm, "COMA - A System for Flexible Combination of Schema Matching Approaches," Proc. 27th Int'l Conf. VLDB, pp. 610-621, Hong Kong, Aug. 2002.

[5] Eila Kuikka, Paula Leinonen, and Martti Penttonen, "Towards Automating of Document Structure Transformations," Proc. ACM Symposium on Document Engineering, pp. 103-110, 2002.

[6] Paula Leinonen, "Automating XML Document Structure Transformations," Proc. ACM Symposium on Document Engineering, pp. 26-28, 2003.

[7] Xuerong Tang and Frank Wm. Tompa, "Specifying Transformations for Structured Documents," Proc. 4th Int'l Workshop on WebDB, pp. 67-72, 2001.

[8] George A. Miller, "WordNet: A Lexical Database for English," Communications of the ACM, Vol. 38, No. 11, pp. 39-41, 1995.