NFA 표현을 사용한 문서-중심적 XML의 키워드 기반 필터링 기법*

이경한[O] 박 석
서강대학교 컴퓨터학과
{scenario[O], spark}@sogang.ac.kr

A Keyword-based Filtering Technique of Document-centric XML using NFA Representation

Kyounghan Lee[O] Seog Park
Department of Computer Science , Sogang University

요 약

XPath 명세는 XML 원소 내용을 필터링하기 위한 질의어 작성이 어렵다. 본 논문은 이러한 문제점을 해결하기 위해 SQL의 LIKE 연산자에서 사용되던 특별한 매칭 문자 '%' 를 허용한 확장된 XPath 명세와 그것을 표준 질의어로 사용하는 문서-중심적 XML 필터링 기법인 Pfilter를 제안한다. Pfilter는 값-기반 술어(value-based predicate)에서 피연산자의 공통 앞부분 문자를 공유하여 값-기반 술어의 처리 성능을 향상시킨다. 또한 본 논문은 Pfilter와 대표적인 데이터-중심적 XML 필터링 기법인 Yfilter를 값-기반 술어 처리의 확장성과 효율성에 대해 비교하고 Pfilter의 값-기반 술어 삽입, 삭제, 처리 결과를 제공한다.

1. 서론

XML(Extensible Markup Language)[1]은 원소 내용(element content)에 여러 가지 방법으로 원소(element)를 표기할 수 있다. 표기 형태에 따라 크게 두 가지로 분류할 수 있는데 그것은 데이터-중심적(data-centric) XML과 문서-중심적(document-centric) XML이다. 데이터-중심적 XML이란, XML 태크(tag)로 표기된 매우 구조적인 데이터이다. 그러나 문서-중심적 XML은 느슨하게 구조화된 데이터이다[2]. 문서-중심적 XML문서의 구조는 매우 간단하지만 한 쌍의 XML 태그에는 비교적 긴 원소 내용을 가지고 있다.

지금까지 연구된 XML 필터링 기법은 데이터-중심적 XML에 초점을 두었고 문서-중심적 XML 필터링은 전통적인 XML 질의 처리와 다른 요구사항을 가지고 있기 때문에 새로운 질의 처리 알고리즘이 필요하다.

본 논문은 기존의 XML 필터링 기법을 고찰하고 값-기반 술어 수행 성능을 높인 새로운 기법을 소개한다. 마지막으로 제안된 기법과 Yfilter[4], 그리고 Yfilter의 값-기반 술어 수행 부분을 원래의 Aho-Corasick 사전 매칭 트리[3]로 대체한 실험 대조군 기법과의 유지 비용, 확장성과 효율성을 비교한 후 본 연구의 결론을 맺는다.

2. 기존 연구

현재까지의 XML 필터링 기법 연구는 상위 레벨의 원소가 많이 중첩되는 XML의 특징 때문에 구조 매칭에 중점을 두었다[5, 6, 7]. 이러한 연구는 데이터-중심적 XML에 적용하기에는 매우 효과적이나 주위에서 흔히 접할 수 있는 문서-중심적 XML을 필터링하기에는 부족한 점이 있다.

XPush[8]와 RDBMS를 이용한 XML 필터링 시스템[9]은 원자적 술어(atomic predicate)를 정의하고 이것을 공유한다. 원자적 술어란, 값-기반 술어에서 논리곱(conjunction)의 요소를 이루는 술어를 말한다.

숫자값과 문자열을 모두 값-기반 술어의 피연산자로 사용할 수 있는 Yfilter는 원자적 술어를 공유하여 단축 평가(short-cut evaluation)를 가능케 하는 방법을 일반화시켰다고 볼 수 있다. 그러나 이러한 방법은 시스템이 얼마나 많은 버킷을 가져야 하는지에 대한 특별한 기준을 정하기 어렵고 문서-중심적 XML에 질의하기 위해, 반드시 필요한 특별한 매칭 문자 '%' 를 효과적으로 지원하기 위해서는 또 다른 알고리즘이 필요하다.

값-기반 술어를 효과적으로 처리하기 위해 XSQ[10]는 푸시다

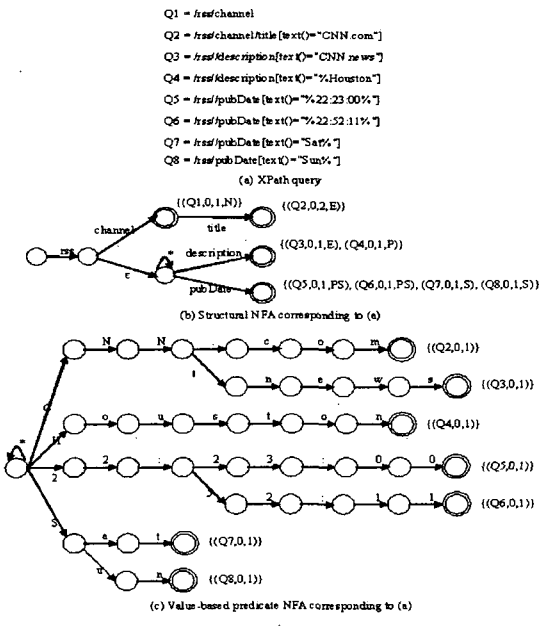
* 본 연구는 한국과학재단 특정기초연구(R01-2006-000-10609-0)지원으로 수행되었음.

운 변환기(pushdown transducer)를 이용하여 원자적 술어를 공유한다. 그러나 XSQ가 생성하는 상태의 개수는 NFA를 이용한 Yfilter 보다 $O(2^n)$배로 증가(단, n은 모든 질의의 테스트 노드에서 '*' 가 나타난 횟수)하므로 많은 메모리를 소모하는 문제점이 발생한다. 따라서 많은 질의를 등록해야 하는 XML 필터링 시스템에 적용되기 어렵다.

3 Pfilter의 값-기반 술어 수행

3.1 결합된 NFA 생성

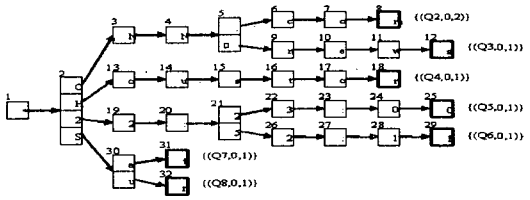
[그림 1]은 8개의 질의를 표현하고 있는 구조 NFA (nondeterministic finite automata)와 값-기반 술어 NFA의 예를 보여주고 있다. 구조 NFA는 Yfilter의 NFA 구성 방법을 그대로 따랐으므로 설명을 생략한다. 그러나 한가지 주의할 점은 Yfilter의 구조 NFA와 달리 각 승인 상태는 (질의번호, 경로번호, 레벨, 매칭 문자 정보) 쌍으로 구성된 부가 정보를 갖는다. 매칭 문자 정보에서 N은 술어가 없음, E는 술어가 존재하나 '%' 가 나타나지 않음, P는 '접두%' 가 나타남, S는 '접미%' 가 나타남, PS는 '접두-접미%' 가 나타남을 의미한다. 그리고 값-기반 술어 NFA의 간선 위에 있는 문자는 천이를 유발하는 입력을 의미한다. 마지막으로 두 NFA에서 음영 처리된 원은 경로 질의의 각 위치 단계 또는 값-기반 술어에서 피연산자의 문자가 공유됨을 의미한다.



[그림 1] XPath 질의의 NFA-기반 표현

3.2 NFA 구조의 구현

본 연구에서는 효과적인 값-기반 술어 실행을 위해 값-기반 술어 NFA를 해쉬 테이블로 구현한다. 그 이유는 해쉬 테이블 기반의 오토마타는 NFA 상태의 삽입/삭제 시간을 낮출 수 있기 때문이다. [그림 2]는 [그림 1] (c)를 해쉬 테이블 기반으로 구현한 모습을 보여주고 있는데 각 해쉬 테이블에 할당된 숫자는 상태번호를 나타내며 굵은 사각형은 승인 상태를 의미한다. 각 승인 상태는 (질의번호, 경로번호, 레벨) 쌍을 가지고 있다.



[그림 2] 값-기반 술어 NFA의 해쉬-기반 구현

지금까지 설명한 값-기반 술어의 표현, 오토마타의 생성 방법과 그 구현을 통해 Pfilter와 ACfilter를 다음과 같이 정의할 수 있다.

정의 1 (Pfilter) Yfilter의 값-기반 술어 처리 부분을 제거한 후 술어 집합을 해쉬-기반의 한 개의 NFA 형태로 구현한 문서-중심적 XML 필터링 기법을 Pfilter라고 한다.

정의 2 (ACfilter) Yfilter의 값-기반 술어 처리 부분을 Aho-Corasick 사전 매칭 트리로 대체한 후 연결 리스트-기반으로 구현한 문서-중심적 XML 필터링 기법을 ACfilter라고 한다.

4. 실험 및 평가

4.1 실험 환경 설정

Pfilter와 ACfilter는 자바를 이용하여 구현했다. 펜티엄-IV 3.2GHz 프로세서와 메모리 1GB가 장착된 윈도우 서버 2003에서 자바 가상 머신 1.5.0을 이용하여 측정했다.

[표 1] 질의와 문서 생성을 위한 작업부하 매개변수

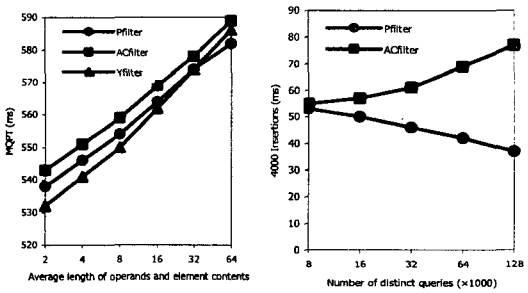
매개변수	범 위	설 명
Q	1000 ~ 500000	중복되지 않는 질의의 개수
ZC	0 ~ 2	피연산자에서 문자의 비대칭도
LO	2 ~ 64	모든 값-기반 술어에서 피연산자의 평균 길이
LC	2 ~ 1000	모든 XML 문서에서 원소 내용의 평균 길이
P	0 ~ 1	모든 값-기반 술어의 피연산자에서 '%' 가 발생할 확률

4.2 실험

Yfilter는 특별한 매칭 문자 '%' 를 포함한 질의를 처리하지 못하기 때문에 첫 번째 실험에서는 값-기반 술어에서 특별한 매칭 문자 '%' 가 나타날 확률을 0으로 고정(P=0)시킨다. 또한 Yfilter는 값-기반 술어의 피연산자 길이와 필터링되는 XML

문서의 원소 내용의 길이가 같을 때(LO=LC)만 값-기반 술어와 XML의 매칭이 발생하기 때문에 본 실험에서 두 실험 매개변수는 항상 같게 한다.

[그림 3] (a)는 값-기반 술어에서 평균 피연산자의 길이와 XML에서 원소 내용의 길이가 변할 때 MQPT를 측정한 그래프이다. 평균 피연산자의 길이가 32보다 크거나 같은 때는 Pfilter가 더 좋은 MQPT를 나타낸다. 32라는 값은 구현 언어와 필터링 시스템이 구동되는 시스템의 사양에 따라 바뀔 수 있는 값이고 중요한 점은 특정 시점에서 MQPT가 역전되는 현상이 발생한다는 것이다.



(a) 피연산자와 원소 내용의 평균 길이가 변할 때 (ZC=1, Q=50000, P=0)

(b) 4000개의 질의를 삽입하는 비용 (ZC=1, LO=8, P=0)

[그림 3] Pfilter, ACfilter, Yfilter의 효율성과 확장성

두 번째 실험은 Pfilter와 ACfilter의 값-기반 술어 오토마타에서 피연산자의 삽입 비용을 알아본다. 공간적 제약 때문에 삭제 비용의 결과는 생략했다. 값-기반 술어를 수행할 때 어느 해쉬 집합에 (질의번호, 경로번호, 레벨) 쌍을 삽입할지에 관련이 있기 때문에 P=0으로 고정한다. 그리고 LO의 증가는 두 시스템에서 전반적인 삽입 시간의 증가를 가져오므로 8로 고정시켰다. 두 번째 실험의 모든 결과는 4000개의 피연산자를 삽입하는데 걸리는 시간을 의미한다.

[그림 3] (b)의 결과처럼 Pfilter는 등록된 질의의 개수가 증가할수록 각 문자가 공유될 확률이 증가하므로 해쉬 테이블에 삽입하는 횟수가 줄어든다. ACfilter는 질의의 개수가 많아질수록 연결 리스트의 길이가 증가하여 탐색시간이 늘어나므로 삽입 시간이 증가한다.

5. 결론

Pfilter는 RSS, 옥션 데이터, DBLP XML, 셰익스피어의 희곡 XML과 같은 문서-중심적 XML을 위한 필터링 기법이다. 이러한 XML의 특징은 일반적인 문서 데이터와 구조화된 데이터의 중간 형태로 특정 원소 내용이 상당히 긴 특징을 가지고 있다. 이러한 XML을 필터링하기 위해 사용자는 XML 구조 매칭 뿐만 아니라 원소 내용이 상대적으로 긴 부분을 키워드를 사용하여 검색하기를 희망한다. 그러나 지금까지의 필터링 기법들은 문서-중심적 XML의 사용자 요구사항을 만족시키지 못했다. 이에 반해 Pfilter는 다량의 값-기반 술어가 필터링 시스템에 등록될 때 각 술어의 피연산자를 이용하여 공통 문자 앞부분 공유된 NFA를 구성함으로써 문서-중심적 XML 필터링에 효과적이다. 이러한 본 연구의 노력은 필터링 시스템에 키워드 기반 검색 기능을 추가시키며 사용자에게 보다 구체적인 필터링이 가능하도록 도와줄 것으로 예상된다.

참고문헌

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. Extensible Markup Language (XML) 1.0 Second Edition W3C Recommendation.

[2] J. Kamps, M. Marx, M. de Rijke, B. Sigurbjörnsson. Best-match Query form Document-centric XML. *In Proc. of the International Workshop on the Web and Databases*, pp. 55-60, 2004.

[3] A. V. Aho and M. J. Corasick. Efficient String Matching: An Aid to Bibliographic Search. *Communications of the ACM*, Vol. 18, Issue 6, pp. 333-340, 1975.

[4] Y. Diao, M. Altinel, M. J. Franklin, H. Zhang, and P. Fischer. Path Sharing and Predicate Evaluation for High-Performance XML Filtering. *The ACM Transactions on Database Systems*, Vol. 28, Issue 4, pp. 467-516, 2003.

[5] T. J. Green, A. Gupta, G. Miklau, M. Onizuka, and D. Suciu. Processing XML Streams with Deterministic Automata and Stream Indexes. *The ACM Transactions on Databases Systems*, Vol. 29, Issue 4, pp. 752-788, 2004.

[6] V. Josifovski, M. Fontoura, and A. Barta. Querying XML Streams. *The International Journal on VLDB*, Vol.14, Issue 2, pp. 197-210, 2005.

[7] J. Kwon, P. Rao, B. Moon, and S. Lee. FiST: Scalable XML Document Filtering by Sequencing Twig Patterns. *In Proc. of VLDB*, pp. 294-315, 2005.

[8] A. K. Gupta and D. Suciu. Stream Processing of XPath Queries with Predicates. *In Proc. of the ACM SIGMOD*, pp. 419-430, 2003.

[9] F. Tian, B. Reinwald, H. Pirahesh, T. Mayr, and J. Myllymaki. Implementing A Scalable XML Publish/Subscribe System Using Relational Database Systems. *In Proc. of the ACM SIGMOD*, pp. 479-490, 2004.

[10] F. Peng, and S. S. Chawathe. XSQ: A Streaming XPath Engine. *The ACM Transactions on Databases Systems*, Vol. 30, Issue 2, pp. 577-623, 2005.