

PIX: XML 문서 검색을 위한 색인 분할 기법

(PIX: Partitioned Index for Keyword Search over XML Documents)

이 흥 래 † 이 형 동 † 유 상 원 † 김 형 주 ††
 (Hongrae Lee) (Hyungdong Lee) (Sangwon Yoo) (Hyoung-Joo Kim)

요 약 정보 검색의 대상이 XML 문서가 되면서 구조 정보를 이용하는 것과 같은 단순한 텍스트 기반의 검색에서는 어려웠던 일이 가능해졌다. 그러나 문서 단위로 처리하던 정보를 엘리먼트 단위로 상세하게 해야 하므로 처리의 부담이 가중되어 많은 수의 엘리먼트를 효과적으로 처리할 수 있는 알고리즘이 필요하다. 본 논문에서는 결과가 될 가능성이 있는 엘리먼트들끼리 미리 분할 한 후 저장하여 처리 대상이 되는 엘리먼트들의 수를 줄이는 역색인 방법을 제안한다. 분할은 특정 레벨을 기준으로 하여 이 레벨에서 공통 선조를 가질 수 있는 거의 여부에 따라 수행한다. 그리고 분할 병합을 통하여 분할하지 않은 것과 동일한 결과를 생성할 수 있도록 하였다. 이는 기존의 XML 문서에 대한 키워드 검색의 성능을 향상시키는 결과를 가져왔고 이를 실험적으로 검증하였다.

키워드 : XML, 키워드 검색, 분할 색인, 역색인

Abstract As XML documents have much richer information than plain texts, we can perform very elaborated, fine-grained search which was difficult in past years. However, as the cost of finer grained element level search is very high, the processing overhead has become a new challenge. We propose an inverted index structure called PIX, which reduces the number of elements processed by partitioning elements according to their match potentiality. We choose a base level and partition elements according to whether they have possibility of having a common ancestor higher than the level. We also propose partition merging technique by which we can get same results as unpartitioned case. Our experimental results show that the index partitioning strategy can reduce processing time considerably.

Key words : XML keyword search, partitioned index, inverted index

1. 서 론

XML(Extensible Markup Language)는 W3C에 의해 표준으로 제창된 텍스트 형식이다[1]. 현재 XML은 인터넷을 포함한 다양한 장소에서 데이터 교환의 실질적인 표준으로 자리 잡아서 학계, 산업계 등 거의 모든 분야에서 XML을 이용한 표준이 제정되어 전자책, 신문, 증권, 의료, 학술 등의 각종 데이터가 XML로 저장되고 교환되고 있다. 현재에도 기관이나 조직의 주요 데

이타가 많은 부분 XML 문서로 저장되어 있고 이러한 추세는 점점 가속되고 있다.

XML로 저장되고 교환되는 데이터의 양이 늘어남에 따라 XML 문서에 대한 검색의 필요성이 점점 커지고 있다. XML 문서의 검색에 사용되는 질의어를 구분하는 한 방법으로 구조적 질의어와 비구조적 질의어 두 가지로 나누는 방법이 있다. 구조적 질의어는 RDBMS 상의 SQL과 비슷한 역할을 하는 질의어로서 XML 문서상에서 특정 부분을 정확하게 처리하여 결과로 생성할 수 있는 질의어로 문서의 내용뿐만 아니라 구조에 대한 표현이 가능한 질의어이다. 이러한 형태의 질의어로 XQL[2], XML-QL[3] 등 많은 언어가 제안되었고 XPath[4], XQuery[5] 등이 W3C에서 표준으로 제안되었거나 준비 중이다. 이런 구조적 질의어는 그 표현 능력이 뛰어나고 정확한 결과를 얻을 수 있는 반면에 문법이 복잡하고 문서의 구조에 관한 정확한 정보가 필요

· 본 연구는 BK-21 정보기술 사업단과 정보통신부 ITRC(e-Biz 기술 연구센터)에 의해 지원되었음

† 비 회 원 : 서울대학교 컴퓨터공학부
 hrlee@oopsla.snu.ac.kr
 hdlee@oopsla.snu.ac.kr
 swyoo@oopsla.snu.ac.kr

†† 종신회원 : 서울대학교 컴퓨터공학부 교수
 hjk@oopsla.snu.ac.kr

논문접수 : 2004년 1월 29일
 심사완료 : 2004년 7월 22일

하다는 단점이 있다. 이보다 좀 더 쉽게 정보에 접근할 수 있는 질의어가 비구조적 질의어로 문서의 구조에 대한 정확한 표현이 어려운 반면에 사용하기가 쉬운 장점이 있다. 이러한 비구조적 질의어로 가장 대표적인 방법이 키워드 검색이다. 키워드 검색은 이미 인터넷의 검색 엔진에서 널리 사용되고 있는 방법으로 일반 사용자가 선호하는 검색 방법이다. 이 두 가지 구분은 개략적인 구분일 뿐으로, 두 구분의 중간 적인 접근 방법으로 비정형적인 구조를 표현하거나 간단한 질의에 약간의 구조 정보를 표현할 수 있는 방법들도 존재한다. 본 논문은 대량의 문서가 XML로 존재하나 사용자가 문서들의 정확한 정보를 알기 힘들거나 문서들이 비정형적 구조를 지닌 환경을 가정하여 질의 방법으로써 키워드 검색을 사용하였다.

XML 문서에 더해진 구조적 정보는 검색의 과정에서 여러 가지로 활용될 수 있다. 그 중 가장 중요한 하나는 검색의 결과로 문서 전체가 아니라 일부분을 반환할 수 있다는 것이다. 과거 텍스트 기반의 연구 중 구문 검색 (passage retrieval)에 관한 연구[6-8]들이 있었으나 구조에 대한 정보가 한정적이어서 결과가 부정확하거나 응용이 제한적인 경우가 많았다. 그러나 XML 문서의 경우에는 이 구조가 자유롭고 정확하게 정의될 수 있어서 이를 활용하여 검색 결과의 품질을 높일 수 있다. XML 문서 검색의 한 예를 살펴보자.

예제 1 그림 1은 DBLP XML 문서의 한 부분을 간략화 한 것이다. 사용자가 Jagadish가 저술한 XML에 관한 논문을 찾아보기를 원한다고 가정하자.

일반적인 키워드 검색이라면 위의 문서가 Jagadish와 XML을 모두 포함하고 있기 때문에 질의의 결과로 그림 1의 문서 전체를 반환할 것이다. 그 결과에는 사용자가 원하는 논문에 관한 정보도 있지만 관련 없는 다른 정보도 포함되어 있다. 위의 문서에서 해당 문서에 대한 구조 정보를 활용하여 Jagadish가 저술한 TAX 논문만을 결과로 반환한다면 더 좋은 검색 결과가 될 것이다.

이렇게 일반 문서에 비하여 XML 문서가 검색의 측

면에서 장점을 가지기도 하지만 거기에 따르는 단점도 많다. 그 중 가장 대표적인 것은 처리해야 할 정보가 늘어난다는 점이다. 기존의 문서 검색에 있어서는 문서 단위로 질의를 처리하였으나 대상이 XML문서인 경우에는 엘리먼트 수준의 상세한 단위로 질의를 처리한다 [9-14]. 실제로 사용되고 있는 XML 문서들을 분석한 최근의 연구 결과[15]를 보면 한 문서에서 처리해야 할 엘리먼트의 수가 수백, 수천 개 정도의 단위로 굉장히 많은 것을 알 수 있다. 문서 단위로만 정보를 처리하는 경우에도 문서 집합의 크기가 커질 경우 색인의 크기, 질의 처리의 성능이 문제가 되어서 색인 압축 기법[16, 17], 중단 기법[18]등 많은 연구가 이루어 졌다. 이러한 문제점은 XML 문서 검색의 경우에는 문서 집합의 크기가 커질수록 더욱 심각해진다.

본 논문에서는 이러한 문제점을 해결하기 위하여 XML 검색에서 엘리먼트 단위로 정보를 처리하면서도 큰 규모의 문서 집합에도 적용될 수 있는 효율적인 검색 방법을 연구하였다. 기본적인 접근 방법은 검색의 결과가 될 수 있는 엘리먼트끼리 미리 분할(partition)하여 색인에 저장하고 질의 처리에 이 분할된 정보만을 이용하여 처리되는 엘리먼트의 개수를 대폭 줄이는 것이다. 구체적으로 엘리먼트를 분할하여 색인을 구성하는 방법과 이를 이용하여 질의를 처리하는 알고리즘을 제시한다. 또 분할된 색인을 이용하여 질의를 처리하더라도 분할을 하지 않은 경우와 동일한 결과를 얻을 수 있음을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 XML 키워드 검색과 관련된 연구들을 고찰하고 문제점을 제기하며 3장에서는 검색 모델을 정의한다. 4장에서는 문제점 해결을 위한 분할 색인 방법을 제시하고 5장에서 이를 실제로 구현하여 실험한 결과를 가지고 고찰한 후 6장에서 결론을 내린다.

2. 관련 연구

XML에 관한 표준이 제정되기 시작한 초기부터 XML 질의에 관한 연구[9,10,19]가 활발하게 이루어 졌다. 이 중 키워드를 사용한 XML 검색에 관한 초기 연구로는 [20]가 있지만 질의어에 키워드 검색을 포함하였다는 이상의 의미는 찾아보기 힘들다. [20]는 키워드 검색을 지원하지만 이는 기본적으로 XML-QL에 키워드 검색을 위한 문법을 추가한 형태이다. 순수하게 키워드 검색을 지원하는 초기 연구로는 XKeyword[11]를 들 수 있다. XKeyword는 그래프로 표현된 XML 데이터 상에서 키워드 검색을 수행하는 시스템으로 노드들 간의 연결 관계에 초점을 맞추었다. 노드들이 복잡한 상관관계를 가지며 연결된 경우에도 XKeyword는 중복된 결과 없이

```

<inproceedings key="conf/sigmod/Jagadish90">
  <author>H. V. Jagadish</author>
  <title>Linear Clustering of Objects with Multiple Attributes.</title>
</inproceedings>
<inproceedings key="conf/sigmod/ZhangRL96">
  <author>Tian Zhang</author>
  <author>Raghu Ramakrishnan</author> <author>Miron Livny</author>
  <title>BIRCH: An Efficient Data Clustering Method for Very Large
Databases.</title></inproceedings>
<inproceedings key="conf/dbpl/JagadishLST01">
  <author>H. V. Jagadish</author> <author>Laks V. S. Lakshmanan</author>
  <author>Divesh Srivastava</author> <author>Keith Thompson</author>
  <title>TAX: A Tree Algebra for XML.</title>
</inproceedings>

```

그림 1 DBLP XML 문서의 일부분

효율적으로 데이터를 처리하나 모델링한 XML 문서 자체가 일반적인 데이터베이스를 변환한 형태로 텍스트가 많은 XML 문서의 경우에는 적용하기 힘들다.

XML 문서는 그 유연성 때문에 단순한 데이터를 표현하는 한 방식이 될 수도 있지만 기본적으로 일반 텍스트를 구조화 한 것이기도 해서 데이터베이스 기반 기술과 정보 검색(IR) 기술의 융합이 필요하다. 따라서 본 연구는 이러한 다양한 성격의 XML 문서 검색을 지원하는 것을 한 목표로 하였다. 이와 같이 최근에는 데이터베이스 기술과 정보 검색 기술을 융합한 시스템이 많이 연구되고 있다. XRANK[12], XSearch[13], [14] 모두 이러한 경향을 반영한 최근의 연구 결과이다. 또 XML 검색 결과의 단위는 여러 연구에 따라 다르게 정의되어 왔는데 이들을 크게 분류하여 보면 결과의 단위와 형식이 질의 자체에 표현되는 경우, 주어진 조건을 만족하는 최소 서브트리인 경우, 따로 검색 결과가 지정되는 경우, 별도의 로직에 의해서 검색 결과를 결정하는 경우 등이 있다.

XRANK는 구글의 PageRank[21] 알고리즘을 엘리먼트의 가중치를 구하는데 이용하였다. 또 최소 공통 선조(least common ancestor) 노드를 질의 결과로 정의하고 이를 구하기 위한 DIL 색인구조와 질의 처리 알고리즘을 제안하였다. DIL은 질의 처리 시에는 각 키워드들의 역색인 리스트 전체를 한 번 스캔하며 처리한다. 역색인 리스트를 구성하는 엘리먼트 단위의 정보는 문서 단위로 존재할 때 마다 매우 많음을 고려하면 큰 문서 집합에서 이 역색인의 로드 시간과 처리 시간은 큰 부담이 될 수 있다. DIL은 아무리 길더라도, 전혀 답이 되지 않는 엘리먼트가 대다수라도 모든 역색인 리스트 전체를 처리하여야 답을 생성할 수 있다. 이에 반해 본 연구는 XRANK와 같이 최소 공통 선조를 결과로 정의하나 전체 역색인 리스트에서 일부분만을 로드하여 디스크 I/O 와 CPU 시간을 줄여 시스템의 성능을 향상시켰다. 답이 될 가능성이 있는 노드들끼리 분할하여 저장함으로써 처리하였기 때문이다. XRANK도 이러한 문제점을 인식하여 엘리먼트의 번호 순이 아니라 순위 순으로 정렬된 RDIL도 제안하였다. 그러나 질의 처리 알고리즘은 엘리먼트의 번호 순으로 처리하므로 이를 위하여 각 역색인은 B+-트리 색인을 가지고 있는데 이러한 구조는 Top-K 질의 처리에는 적당하나, 색인 구성이 복잡하고 오히려 역색인 리스트에 덧붙여진 색인 접근을 위하여 추가적으로 발생하는 디스크 I/O가 문제가 될 수 있다. 각 방법의 장점을 섞은 방법으로 HDIL이 있는데 이는 RDIL로 처리하다가 어느 정도 시점이 되면 그냥 DIL로 처리하는 방식을 취하고 있는데, 전환되는 시점에 대한 좀 더 명확한 판단 기준이 필요하다.

XSearch는 XML 키워드 검색에서의 의미론적인 면에 좀 더 초점을 맞춘 연구 결과이다. XSearch는 위에서 기술한 검색 결과 정의 중 마지막의 접근 방법을 취하는 연구로 노드의 라벨을 분석하여 연결 관계(inter-connection relationship)를 정의하여 의미 있는 결과를 추론한다. 반복되는 라벨들은 하나의 객체(entity)로 보고 이를 이용하여 결과를 추론하는데, 다양한 성격의 XML 문서에 이러한 방법이 적용가능한지는 검증이 필요하다. 본 연구는 엘리먼트의 순위를 정하는 방법으로 XSearch에서 사용한 TF·ILF를 사용하였다. 이는 기존의 텍스트 검색에서 벡터 스페이스 모델[22]의 순위화 방법인 TF·IDF를 트리의 영역으로 확장한 것이다. 다른 순위화 알고리즘을 사용할 수도 있는데, [14]은 기존의 벡터 스페이스 모델을 확장한 용어 가중치와 순위화 알고리즘을 제안한 연구 중의 하나로 여러 가지 가중치 알고리즘들을 비교분석하였다.

3. 검색 모델

3.1 XML 문서

XML 문서는 여러 가지로 모델링을 될 수 있지만 논문에서는 순서가 있는, 노드에 라벨된 트리(ordered node labeled tree)로 모델링을 하였다. 문서 노드(document node)가 트리의 루트노드이고 트리의 레벨은 루트노드를 0으로 정의한다. 그리고 레벨이 루트에 가까울수록 '낮다'고 하고 루트에서 멀어질수록 '높다'고 한다. 즉 레벨이 더 낮은 노드는 루트에 더 가까운 노드를 뜻한다. 그리고 엘리먼트와 애트리뷰트의 구별은 두지 않았다. 실제로 30~50% 정도의 노드가 애트리뷰트로 애트리뷰트에 정보를 두는 경우도 많기 때문이다[15]. 논문에서는 엘리먼트와 애트리뷰트 모두 노드라 지칭한다.

그림 2는 예제 1의 XML 문서를 트리로 모델링한 그림이다. 트리의 각 노드에는 노드 번호가 붙어 있는데 이는 각 노드를 유일하게 식별하는 식별자로 실제 구현은 다양한 방법으로 가능하다.

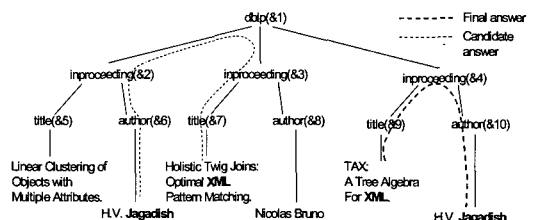


그림 2 예제 1의 문서의 트리 모델과 질의 결과

3.2 질의 모델

본 논문에서 사용한 질의 모델인 키워드 검색에서 사용자로부터의 입력은 단순한 키워드의 집합이다. 이 키워드들은 보통 두 가지 의미 중에 하나로 해석되는데 첫 번째가 AND로 연결된 단어로 해석하는 방법이고 두 번째가 OR로 연결된 단어로 해석하는 방법이다. 첫 번째 해석에서 결과는 모든 키워드 들을 포함해야 답이 되고 두 번째 해석에서는 어느 한 단어만 포함하여도 답이 될 수가 있다. 물론 많은 단어를 포함하면 높은 순위의 결과가 될 것이다. 본 논문에서는 첫 번째 의미로 키워드 검색을 정의하여 결과는 모든 키워드들을 포함하여야 하는 것으로 정의한다.

검색의 결과로는 XML 문서 전체가 아니라 XML 문서의 서브 트리를 반환한다. 논문에서 이 서브 트리와의 서브 트리의 루트 노드는 서로 같은 의미로 사용한다. 결과는 모든 키워드를 포함하는 전체 문서 트리의 서브 트리이고 결과의 어떤 서브 트리도 전체 키워드를 포함하지 않는다고 정의한다. 이는 최소 공통 선조 노드(LCA, least common ancestor)를 결과로 정의하는 것으로 높은 레벨에 있는 결과인수록 좋은 답으로 보는 것이다. 문서에 관한 DTD정보를 가지고 있거나 관리자가 문서의 형식을 잘 알고 있는 경우에는 이것보다 의미 있는 검색 결과의 단위를 결정하기 쉬우나 본 논문은 인터넷처럼 이질적인 문서가 존재하는 환경은 대상으로 하여 XML 문서에 관한 DTD 정보를 가정하지 않았다. 이러한 경우 검색의 결과의 단위를 결정하는 문제는 현재 연구가 진행되고 있는 분야로 아직 결정적으로 받아들여지고 있는 알고리즘은 없고 가장 선별적인 즉 가장 높은 레벨의 결과를 반환하는 접근 방법은 일반적으로 쓰이는 한 방법이다[11,12].

예제 XML 문서에 대한 “Jagadish가 저술한 XML에 관한 논문을 찾아라”는 질의를 키워드 검색으로 표현하면 (“Jagadish”, “XML”)이다. 이 질의에 관한 가장 적절한 결과는 그림 2에서 노드 &4로 표현되는 서브 트리이다. 이 노드는 하위 서브 트리에 “Jagadish”, “XML” 두 단어를 모두 포함하고 있고 이 노드의 후손 노드 중 어느 노드도 이 두 키워드를 모두 포함하고 있지 않다. 노드 &1이 루트인 서브트리도 결과로 볼 수 있으나 하위에 결과가 될 수 있는 다른 노드(&4)를 포함하고 있고 레벨이 &4보다 더 낮으므로 덜 정확한 결과이다.

4. PIX(Partitioned Index for Keyword Search over XML Documents)

4.1 역색인을 이용한 질의 처리 개관

역색인. 역색인(inverted index)은 정보 검색 분야에 질의 처리를 위해 이용되는 가장 일반적인 색인 구

조이다. 역색인은 시스템이나 질의 처리 알고리즘에 따라 다양한 구조를 지니나 기본적인 구조는 특정 단어를 입력하면 해당 단어가 존재하는 위치 정보(포스팅)의 리스트를 얻을 수 있는 구조를 지니고 있다. 이 위치 정보는 단순히 문서 번호에서 시작하여, 문서 번호와 문서 내 위치, 출현 빈도 등 다양한 부가적인 정보를 포함하여 질의 처리를 돕는 구조로 구성된다. XML 문서 검색의 경우 이 위치는 문서 단위가 아니라 엘리먼트 단위의 정보를 가지도록 구성된다. 예를 들면 예제문서의 경우 그림 3과 같이 역색인이 구성된다.

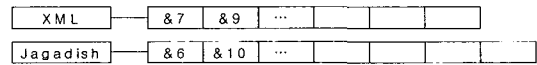


그림 3 예제 1의 문서에 대한 역색인

질의 처리. 이 역색인을 이용하여 키워드 검색을 하는 가장 기초적인 알고리즘은 각 키워드 별 위치 정보들의 조합을 검사하는 방법이다. XML은 모든 노드가 루트에서는 공통 선조를 가지므로 모든 조합이 잠재적인 답이 된다. 그림 2에 대한 역색인(그림 3)을 이용한 질의 처리를 예를 들어보면 다음과 같이 할 수 있다. 먼저 사용자가 입력한 키워드 “Jagadish”와 “XML” 두 단어의 포스팅 리스트를 가져온다. 이 두 단어를 모두 포함하는 쌍은 (&7, &6), (&7, &10), (&9, &6), (&9, &10) 이렇게 4가지이고 이들 조합의 결과는 각각 (&1, &1, &4) 이다. 이 중 가장 선별적인 결과는 &4이므로 답으로 &4를 출력한다. 이러한 질의 처리는 여러 번 리스트를 스캔하며 처리하므로 비효율적으로 실제로는 DIL[12]과 같은 좋은 알고리즘들이 존재한다. 그러나 이들 알고리즘들도 최소한 각 키워드의 포스팅 리스트를 최소 한 번 검색하여야 질의 처리가 가능하다. 기존에 문서 단위로 색인을 하던 시스템에서도 문서 집합의 크기가 커지면서 이 역색인 리스트의 크기가 커져서 많은 압축과 리스트 구성 기법들이 연구[23]되었던 것을 고려하면 처리 단위가 엘리먼트 수준인 XML 검색에서 이는 성능 상의 큰 문제가 된다. 다음 장에 제시하는 색인 분할 기법은 색인을 분할하여 질의 처리 시에 색인 리스트의 일부분만을 로드하여 처리할 수 있도록 하여 이러한 문제점을 해결한다.

4.2 색인 분할

분할의 기본 개념. 앞의 예제에서 &9와 &10은 서로 매치되어 결과로 되고 &9와 &6은 매치되어도 루트 노드를 답으로 생성하여 의미 있는 결과가 되지 못한다. 색인 분할의 기본적인 아이디어는 의미 있는 결과를 생성할 수 있는 가능성이 있는 노드들끼리 분할하여 색인에 저장하여 역색인 리스트에서 처리하는 엘리먼트의

개수를 줄이는 것이다.

이런 매치 가능성은 특정 레벨을 기준으로 생각한다. 즉, 기준이 되는 레벨을 정하여 이 레벨 아래에서 공통 선조를 갖는 노드들은 결과로 보지 않고 이 기준 레벨 보다 큰 레벨에서 공통 선조를 갖는 노드들만을 의미 있는 결과로 본다. 이 기준이 되는 레벨을 기준 레벨(base level)이라 정의한다. 물론 질의의 결과가 되는 노드들은 다양한 레벨에서 존재할 수 있고 기준 레벨은 이들을 나누어서 처리하는 기준의 역할을 한다. 즉 기준 레벨보다 높은 레벨의 답을 먼저 찾고 차차 레벨이 낮은 답들을 찾는 것이다. 이러한 질의 처리 방법은 Top-k 질의 처리에도 적합하다. 즉 기준 레벨 이상의 답을 먼저 찾고 만약 k개미만의 답이 생성되었을 경우에는 기준 레벨을 낮춰가며 답을 찾다가 충분한 개수의 답이 나오는 시점에서 처리를 중단하는 것이다.

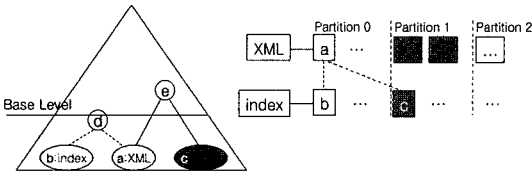


그림 4 공통선조를 갖는 레벨에 의한 분할의 개념

그림 4에서는 노드 a와 b는 기준 레벨 이상에서 공통 선조 노드를 가지므로 의미 있는 답이 될 수 있다. 그리고 노드 a와 c는 기준 레벨 보다 낮은 레벨에서 공통 선조 노드를 가진다. 따라서 이들은 일차적으로 답에서 제외된다. 이런 경우 a와 c를 다른 분할(partition)에 속하도록 하고 a와 b는 같은 분할에 속하도록 엘리먼트들을 나누어 그림 4의 역색인처럼 역색인을 분할하여 저장한다면 서로 다른 분할에 속해 있는 a와 c는 아예 비교를 하지 않을 수 있을 것이다. PIX는 이렇게 어떤 레벨에서 공통선조를 가질 수 있는 지의 가능성을 가지고 역색인을 분할한다. 이후의 절들에서는 PIX의 구체적인 분할 알고리즘과 구성을 기술한다.

분할함수 PF. 기준 레벨에서의 노드들의 매치 가능성¹⁾에 따라 노드들을 분할하여 주는 기준으로 트리 해쉬 함수 PF를 정의한다.

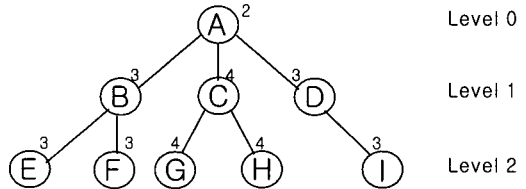
정의 1. (분할함수 PF) 분할함수 PF는 Value × Node → Value의 도메인을 갖는 함수로 첫 번째 인자는 트리의 레벨인 정수이고 두 번째 인자는 트리의 노드이다. 함수를 적용한 결과는 숫자이다. 레벨 i에서 노드 n에 적용한 PF(i, n)를 PF_i(n)으로, 고정된 i에 대하여 PF_i로 표기한다. PF는 다음의 성질을 만족하여야 한다.

1) 두 노드가 기준 레벨에서 공통 선조를 가질 수 있는 가능성을 의미한다.

트리 T의 임의의 노드 $n_a \in T, n_b \in T$ 에 대하여 노드 n_a 와 n_b 에 대하여 T의 i 레벨 이상에 최소 공통 선조 노드가 존재하면 $PF_i(n_a) = PF_i(n_b)$

다음 예제는 간단한 트리에 정의된 한 분할 함수 PF를 보인다.

예제 2



위와 같은 트리에서 함수 f가 $f(B) = f(D) = f(E) = f(F) = f(I) = 3, f(C) = f(G) = f(H) = 4$ 인 매핑을 갖는다고 하자. 그러면 함수 f는 PF₁이다. 임의의 두 노드에 관하여 i = 1 일 때의 PF의 성질을 만족하기 때문이다. 노드 E와 F는 레벨 1에서 최소 공통 선조를 갖고 $f(E) = f(F)$ 이다. 또 노드 G와 노드 I에서 $f(G) \neq f(I)$ 이고 레벨 1 보다 낮은 레벨 0에서 최소 공통 선조를 갖는다. 여기에서 $f(E) = f(I)$ 이지만 E와 I는 레벨 1에서 공통 선조를 갖지 않음에 주의한다. 즉 PF_i 값이 같다고 하여 반드시 레벨 i 이상에서 반드시 최소 공통 선조를 갖지는 않는다. 따라서 i보다 낮은 레벨에 존재하는 노드들에 대하여도 PF_i 값은 정의될 수 있다. 두 노드의 PF_i 값에서 정의된 성질을 만족하기만 하면 되는 것이다.

공통선조관계 \approx_i . PF값을 이용하여 노드들의 잠재적인 매치 가능성을 분류하기 위하여 다음과 같은 관계를 정의한다.

정의 2. (공통선조관계 \approx_i) 공통선조관계는 노드들의 집합에 대하여 특정 레벨i를 기준으로 PF와 함께 정의되며 두 노드에 적용되는 이항관계(binary relation)이다. 트리T의 임의의 노드 $n_a \in T, n_b \in T$ 에 대하여 다음과 같이 정의된다.

$$n_a \approx_i n_b \Leftrightarrow \forall j \leq i, PF_j(n_a) = PF_j(n_b)$$

직관적으로 이 공통 선조 관계는 두 노드가 공통 선조를 가질 가능성이 있음을 의미한다. 만약 두 노드의 최소 공통 선조가 i 레벨에 존재하면 이들은 i 레벨 이하에서 항상 공통 선조를 갖고 이들은 최소 공통 선조에서 루트까지의 패스 상에 존재하는 노드들이다. 그리고 노드 자신을 자기의 선조라고 본다면 다음의 정리가 성립한다.

정리 1. 공통선조관계는 동치관계(equivalence relation)이다.

증명. 선조노드에 대한 가정에서 $n_a \approx_i n_a$ 이므로 주어진 관계는 반사관계(reflexive relation)이다. 또 두 노드

n_a, n_b 간의 선조노드와 n_b, n_a 간의 선조노드는 같으므로 대칭관계(symmetric relation)이다. 그리고 $n_a \approx_i n_b$ 이면 n_a 와 n_b 는 i 레벨에서 공통 선조를 갖는다. 마찬가지로 $n_b \approx_i n_c$ 이면 n_b 와 n_c 는 i 레벨에서 공통 선조를 갖는다. 이 레벨에서의 n_a 와 n_b 의 공통 선조 노드를 n_x, n_b 와 n_c 의 공통 선조 노드를 n_y 라 하면 한 노드에서 루트까지의 경로에서 특정 레벨의 노드는 하나 밖에 존재할 수 없으므로 n_x 와 n_y 는 같은 노드이다. 따라서 $n_a \approx_i n_c$ 가 성립하고 주어진 관계는 추이관계(transitive relation)이다. 따라서 공통선조관계는 동치관계이다.

분할 P. 공통선조관계는 동치관계 이므로 트리의 모든 노드들을 서로 겹치지 않게 분할한다. PF 함수의 성질의 대우는 PF 값이 틀리면 기준 레벨 이상에서 공통 선조를 가지지 않는다는 것이다. 따라서 이렇게 나눈 서로 다른 분할들 사이의 노드들 간에는 비교를 할 필요가 없다. 즉 기준 레벨 이상에 존재하는 공통 선조들을 찾기 위하여 같은 분할들 안의 노드들끼리만 비교를 하면 된다.

정의 3. (공통선조관계에 의한 분할 P) i 레벨의 공통선조관계는 전체 트리 노드들의 집합을 겹치지 않은 부집합들로 분할한다. 같은 분할에 속하는 노드들은 같은 PF값을 갖고 이 값을 분할 값으로 정의하고 P-value라 한다. P-value가 v 인 i 레벨에서의 분할을 $P_{i,v}$ 로 표시한다. 그리고 색인 시의 기준 레벨을 색인 레벨(index level)이라 한다.

PIX의 구조. PIX는 이 공통선조관계 분할 P를 이용하여 노드들을 분할하여 저장하는 역색인이다. 특정 키워드의 역색인 리스트 내에서 노드들은 각 분할 별로 그룹핑되어 저장된다. 즉 미리 정의된 PF 함수와 기준 레벨 i 를 기준으로 하여 같은 PF값을 갖는 노드들끼리 분할하여 색인에 저장한다. 같은 분할 내 노드들은 질의 처리 알고리즘에 맞게 정렬되며 PIX에서 따로 분할 내의 정렬 순서나 포스팅의 구조를 정의하지는 않는다. 그리고 PIX는 역색인 리스트 내의 각 분할들에 효율적으로 접근하기 위하여 리스트의 처음에 분할 색인 정보 P-index를 가진다. 이 색인 정보에는 해당 키워드의 역색인 리스트 내의 전체 분할 수, 각 분할들의 분할 값과 리스트 내의 오프셋 정보, 포함하는 노드의 수에 관한 정보로 구성되어 있다. 이 구조는 그림 5에 도시되었다.

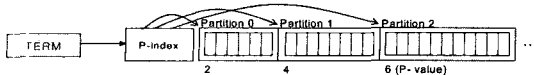


그림 5 PIX의 구조

예제 3. 그림 6은 분할을 한 경우의 역색인을 보여준

다. 그림의 트리는 예제 1의 XML 을 확장하여 트리로 간략하게 모델링한 것이고 A는 분할을 이용하지 않은 일반적인 역색인, B는 분할을 한 역색인을 나타낸다. 검색 단어는 “XML”과 “index”로 A의 일반적인 역색인을 이용할 경우 두 역색인 리스트의 모든 노드들은 최소한 번씩은 처리되어 비교된다. 그러나 분할하여 구성된 역색인 B에서 같은 분할에 속하는 노드들끼리만 질의 처리를 하면 된다. 레벨 1 이상이 답이 된다고 가정할 때 다른 분할에 속하는 (&10, &11)와 (&9)는 루트까지의 경로 중 첫 번째 레벨에 있는 노드가 다르기 때문에 레벨 1에서 답을 가질 수 없는 반면, 같은 분할에 속하는 (&8, &15)과 (&9)는 가능성이 있다. 실제로 &8과 &9는 매치되어 &2를 답으로 생성하지만 &15과 &9는 루트에서 매치되어 답에서 제외된다.

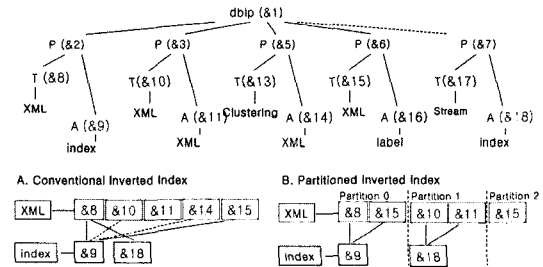


그림 6 분할된 역색인 예시

질의처리 알고리즘 XML 키워드 검색에서 질의처리는 기본적으로 모든 키워드를 포함하는 최소 공통 선조 노드들을 찾는 과정이다. 즉 주어진 키워드들을 포함하는 노드들의 리스트에서 각 키워드를 포함하는 노드를 최소한 하나 이상 포함하는(후손 노드로 두는) 노드 중 가장 높은 레벨에 위치한 노드, 즉 하위의 어떤 노드도 모든 키워드를 포함하지는 않는 노드를 찾는 알고리즘이다. PIX는 이러한 결과노드들을 찾을 수 있는 임의의 알고리즘을 사용할 수 있고, 본 논문은 XRANK[12]의 DIL 알고리즘을 사용하여 주어진 키워드의 노드들의 리스트에서 최소 공통 선조 노드들을 찾는 과정을 구현하였다. PIX에서의 질의 처리의 특징은 이 알고리즘들이 각 분할 내에서만 적용된다는 것이다. 즉 분할값 P-value를 기준으로 같은 P-value를 갖는 분할들 사이에서만 알고리즘을 적용하여 결과를 내면 된다. 다른 P-value를 갖는 분할에 존재하는 엘리먼트들은 서로 매치가 되어도 공통선조가 기준 레벨보다 낮은 레벨에 존재하기 때문에 의미 있는 결과를 생성하지 못하므로 서로 비교할 필요가 없다. 개략적인 질의 처리의 단계는 다음과 같다.

1. 각 단어 별 P-index 정보 검색

2. 단어 별 존재하는 분할들 중 모든 단어에 P-value 가 공통되는 분할들만을 선별
3. 해당 분할들 사이에서 최소 공통 선조 노드를 구하는 알고리즘 적용

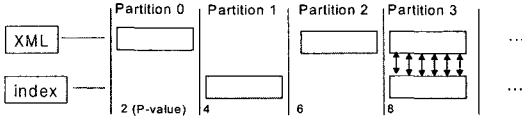


그림 7 분할을 이용한 질의 처리

위의 경우는 XML과 index 두 단어로 검색을 한 경우이다. XML은 P-value가 2, 6, 8인 분할들이 존재하고 index는 P-value가 4, 8인 분할이 존재한다. 이 두 단어의 분할들 중 공통되는 P-value는 8 밖에 없다. 따라서 XML의 해당 분할과 index의 해당 분할에 속하는 엘리먼트들에만 질의 처리 알고리즘을 적용하여 답을 생성하면 된다. XML의 P-value로 2, 6을 같은 분할들과 index의 P-value로 4를 갖는 분할은 다른 단어의 포스팅 리스트 중 어느 노드와 매칭이 되어도 의미 있는 결과를 생성하지 못한다. 즉 기준 레벨 미만의 레벨에서 답을 갖기 때문에 처리할 필요가 없다. 이 PIX 알고리즘은 입력 단어 중 선별적인 단어, 즉 특정 분할에만 존재하는 단어가 있으면 다른 분할들을 전혀 처리하지 않기 때문에 매우 효율적이다.

분할 병합. 앞 절에 기술한 질의 처리 알고리즘을 통해 생성된 답은 기준 레벨 이상의 답만을 포함한다. 이는 기준 레벨이 문서의 의미 있는 결과의 레벨과 일치할 때는 타당성을 가지나 그렇지 않은 경우에는 기준 레벨 보다 낮은 레벨에 존재하는 다른 의미 있는 결과들을 생성하지 못하는 단점을 가진다. 이는 정보 검색에서 결과의 재현율(recall)이 저하됨을 의미한다.

분할 병합은 이렇게 기준 레벨만을 기준으로 생각하여 누락되는 답을 생성할 수 있게 하여준다. 분할 병합은 기준 레벨을 좀 더 낮은 레벨로 이동하는 과정이다. 기준 레벨이 낮아지면 낮아진 레벨까지의 답이 생성되기 때문에 더 많은 결과를 포함한다. 기준 레벨이 낮아지면서 원래의 레벨에서는 서로 다른 P-value를 가졌던 분할들이 낮아진 레벨에서는 같은 P-value를 가지게 된다. 이러한 분할들은 낮아진 레벨에서는 같은 분할에 속하는 분할들로 이러한 같은 P-value를 갖는 분할들을 하나의 분할로 합하는 과정이 분할 병합이다. 분할 병합의 간단한 알고리즘은 다음과 같다.

1. 현재 존재하는 모든 분할들에 대하여
2. 새로운 레벨에서의 P-value를 계산하고 같은 P-value를 갖는 분할들끼리 합병

분할 병합으로 인하여 전체 질의 처리 알고리즘은 낮

은 레벨부터 차례대로 레벨을 높여가면서 답을 찾는 식으로 구성되며 Top-k 질의 처리를 효율적으로 구현할 수 있다. 즉 특정 레벨에서 충분한 개수(k개 이상)의 답이 생성되면 그 시점에서 처리를 중단하는 것이다.

다음 그림 8은 분할 병합의 한 예이다.

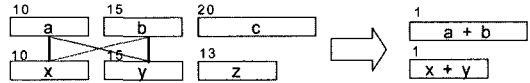


그림 8 분할 합병의 예

위의 색인의 기준 레벨이 4이고 각각 XML과 index의 역색인이라 하자. XML이라는 단어는 $P_{4,10}^{(2)}$, $P_{4,15}$, $P_{4,20}$ 의 세 분할들에 존재하고, index는 $P_{4,10}$, $P_{4,13}$, $P_{4,15}$ 의 세 분할에 존재한다. 레벨4를 기준으로 질의처리를 하였는데 충분한 개수의 결과가 나오지 않았다면 기준 레벨을 더 낮여 레벨4 이상에서 존재하는 결과들도 추가할 필요가 있다. 여기에서 기준 레벨을 한 단계만 낮춰 레벨3에 있는 답들까지 결과에 추가한다고 하자. 그리고 이들 각 분할의 P-value가 레벨4에서 10이었던 분할들은 레벨 3에서 P-value 1을 가지고, 15였던 분할들은 1, 20이었던 분할들은 2를 가진다고 하자. 이러면 분할 $P_{4,10}$ 과 $P_{4,15}$ 는 분할 내부의 노드들끼리는 P-value가 달라 매치를 해도 기준 레벨 4에서는 답을 생성하지 못하지만 기준 레벨3에서는 P-value가 1로 같아 의미 있는 답을 생성할 가능성이 있다. 따라서 $P_{4,10}$ 과 $P_{4,15}$ 의 분할이 합병되어 새로운 분할 $P_{3,1}$ 을 형성하여 XML의 $P_{3,1}$ 분할과 index의 $P_{3,1}$ 분할에 다시 최소 공통 선조 알고리즘을 적용한다. 이 과정에서 $P_{3,1}$ 분할에 속하는 노드들 중에는 기준 레벨4에서 이미 매치가 일어났던 노드들이 있다. 따라서 새로 추가된 $P_{4,10}$ 에 속했던 엘리먼트와 $P_{4,15}$ 에 속했던 노드들 사이에서만 새로 매치를 하여 답을 생성하면 합병으로 인한 중복된 질의처리를 방지할 수 있다.

분할에 의한 질의 처리 시에는 기준 레벨 미만의 답을 제거하지만 이렇게 하면 분할 병합 시에 이미 같은 분할에 속해서 비교를 했던 엘리먼트들을 모두 다시 비교하여야 한다. 이러한 중복된 처리를 막기 위하여 처음 색인 레벨에서 분할 별로 질의처리를 할 때 기준 레벨이 안 되는 결과들을 제거하지 않고 결과를 레벨과 함께 저장하여 합병 시 효율적으로 처리할 수 있도록 한다. 만약에 분할이 합병되어 보다 낮은 레벨에서의 결과가 필요한 경우, 답으로 출력되지 않은 결과들 중 새로운 기준 레벨 이상의 답을 새로운 답으로 추가하도록

2) 레벨 4에서 정의된 P-value가 10인 분할. 정의 3 참조

질의 처리 알고리즘을 수정한다.

PIX의 구현. PIX는 다음에 정의하는 PIXPF 함수와 PIXMERGE 분할병합 방법을 사용하여 구현하였다. 물론 앞 절에 기술한 분할의 성질을 만족시키는 어떤 PF 함수나 합병방법을 사용하여도 된다.

정의 4 PIXPF와 PIXMERGE

$$PIXPF_i(n) = \sum_{k=1}^i (o_k(n) \bmod 2) 2^k$$

i : i 트리 레벨

n : 노드

$o_k(n)$: 노드 n 에서 루트까지의 경로에 존재하는 노드 중 k 레벨에 있는 노드의 순서. 해당하는 경로에 노드가 없을 경우에는 0

PIXMERGE (분할 병합 방법)

i 레벨에서 $i-1$ 레벨로 기준 레벨 이동 시에 두 분할 $P_{i,a}$ 와 $P_{i,b}$ 에서 $a \bmod 2^i = b \bmod 2^i$ 이면 $P_{i,a}$ 와 $P_{i,b}$ 를 병합

직관적으로 PIXPF는 특정 노드에서 루트까지의 경로를 축약해서 나타낸다. 어떤 노드의 i 레벨의 선조의 순서가 그 레벨에서 형제노드(sibling)들 간에 짝수 번째 있으면 0, 홀수 번째 있으면 1을 가지고 있다. 그 숫자를 이진수로 i 번째 비트로 나타낸 것이다. 루트에서 어떤 노드까지의 경로를 생각할 때 첫 번째 레벨에서의 선조가 루트에서 짝수 번째 있고, 그 다음 레벨의 노드는 홀수 번째 있으면 $0 \times 2^0 + 1 \times 2^1 = 2$ 를 P-value로 갖게 된다. 즉 이진수로 나타낸 P-value는 각 i 번째 비트가 i 레벨의 선조가 홀수 번째인지, 짝수 번째 인지를 나타내는 것이다.

정리 4. PIXPF는 분할함수 PF의 성질을 만족한다.

증명. 트리 T 의 임의의 노드 n_a 와 n_b 가 i 레벨 이상에서 최소 공통 선조를 갖는다면 n_a 에서 루트까지의 경로는 최소한 루트에서 i 레벨까지는 n_b 에서 루트까지의 경로와 동일한 경로를 갖는다. 루트에서 i 레벨의 노드까지 공통 선조를 가지므로 n_a 와 n_b 는 $1 \leq k \leq i$ 인 모든 k 에 대하여 o_k 값이 모두 같으므로 $\sum_{k=1}^i (o_k(n_a) \bmod 2) 2^k = \sum_{k=1}^i (o_k(n_b) \bmod 2) 2^k$ 이다. $PIXPF_i(n_a) = PIXPF_i(n_b)$ 이 성립하므로 PIXPF는 PF의 성질을 만족한다.

엘리먼트 식별자로는 XRANK와 같이 DeweyID를 사용하였었다. DeweyID는 각 레벨 별로 자식 노드의 순서를 가지고 있다. 예를 들어 루트에서 두 번째 자식 노드에서 시작하여 트리의 경로 상으로 에서 차례대로 다섯 번째, 세 번째, 열일곱 번째의 자식 노드인 노드는 2.5.3.17로 인코딩된다. DeweyID는 PIXPF를 쉽게 계산

가능하다는 특징을 가지고 있는데 PF함수에 따라 공통 선조만 구할 수 있으면 다른 방식을 사용하여도 무방하다. 결과로서 엘리먼트 식별자들이 나왔을 때 그 식별자들을 가지고 실제 XML 문서에서 해당 부분을 얻는 구조는 XML 저장을 할 수 있는 많은 다른 연구들을 활용할 수 있으므로 별도로 기술하지 않는다. 본 시스템은 XDBOX[24]시스템을 XML 문서의 저장 장치로 활용하였다.

5. 구현 및 실험

PIX는 C++로 구현되었으며 BerkeleyDB[25]을 사용하여 색인인을 구현하였다. 실험은 512MB의 메모리를 가진 PentiumIII 993 MHz의 PC에서 하였다. 실험에 사용한 데이터 셋은 INEX 2003[26]과 Shakespeare[27]이다. 첫 번째 실험은 대표적인 XML 키워드 검색 시스템인 XRANK의 여러 알고리즘과 성능을 비교하였고, 두 번째 실험은 최대 분할의 개수변화에 따른 여러 가지 질의의 수행 속도 변화를 관찰하였다. 세 번째 실험은 검색 단어의 개수를 늘려가면서 분할을 한 경우와 하지 않은 경우의 성능을 비교 평가 하였다. PIXPF는 색인레벨이 i 일 때 최대 2^i 개의 분할이 존재하므로 색인레벨을 높이는 것과 분할의 개수를 늘리는 것은 같은 의미이고 분할의 개수는 최대 분할의 개수와 동일한 의미로 사용하였다.

첫 번째와 두 번째 실험은 INEX 2003 컬렉션을 가지고 수행하였다. INEX 2003 컬렉션은 현재 XML 문서 검색의 주요 테스트 컬렉션으로 1995~2002년도까지의 컴퓨터 관련 저널 XML 문서들과 질의 셋으로 구성되어 있고 500M 이상의 크기이고 한 해의 특정 저널들이 합하여져 하나의 XML 문서를 이루는데 이 각각의 크기는 1~10M 정도의 크기이다. 저널명, 이슈, 목차, 논문, 챗터, 단락 등 일반적인 저널의 구조를 지니고 있고, 최대 레벨은 10 정도로 특이하지 않은 일반적인 형상의 XML 문서로 볼 수 있다. INEX에 정의된 질의 셋은 크게 CO(content only)와 CAS(content and structure) 질의로 나눌 수 있는데 CO질의는 구조를 포함하지 않는 내용에 관한 질의들이고 CAS질의는 내용과 더불어 구조에 관한 제약사항이 같이 있는 질의들이다. PIX에서는 구조 질의를 지원하지 않으므로 CO 표준 질의들을 사용하여 실험을 수행하였다. 이들 질의들 중 7개의 질의에 대한 결과를 분석하였는데 다른 질의의 결과들도 이들과 비슷하였다. 표 1은 이들 질의를 보여준다. Q1~Q4의 3,4개의 단어로 이루어진 간단한 질의이고 Q5~Q7의 출현 빈도가 매우 높거나 열 개 이상의 단어들로 이루어진 수행 시간이 오래 걸리는 복잡한 질의들

표 1 실험에 사용된 INEX 표준 질의어

질의 번호	키워드	질의 번호	키워드
Q1	Hypertext Information Retrieval	Q5	information data visualization technique hierarchy space
Q2	wireless security application	Q6	concurrency control semantic transaction management application performance benefit
Q3	recommender system agent	Q7	machine learning adaptative algorithm probabilistic model neural network support vector machine
Q4	singular value decomposition		

이다.

그림 9는 DIL, RDIL, HDIL과 PIX를 비교 실험한 결과이다. 실험 결과는 수행 속도로 측정하였다. DIL, RDIL, HDIL은 각 키워드의 상관관계에 따라 상관관계가 높은 경우에는 RDIL의 수행 속도가, 상관관계가 적은 경우에는 DIL이 우수하였고, HDIL은 그 중간정도의 성능을 평균적으로 보여주었다. 하지만 대부분의 경우 PIX는 이들보다 몇 배 이상 우수한 성능을 보여주었고 상대적인 수행 속도 향상이 가장 적은 경우(Q5)에도 이들 중 가장 우수한 것과 비슷하였다. 이는 PIX가 역색인 분할 알고리즘으로 인하여 실제 역색인 중 일부분만을 처리하였기 때문에 가능한 것이다. 성능 속도가 크게 차이가 나지 않은 경우에는 PIX의 분할로 인한 필터링 효과가 상대적으로 미비한 경우로 볼 수 있다. 비슷한 현상이 두 번째 실험에도 나타나고 그 이유는 두 번째 실험 결과 분석에 기술하였다.

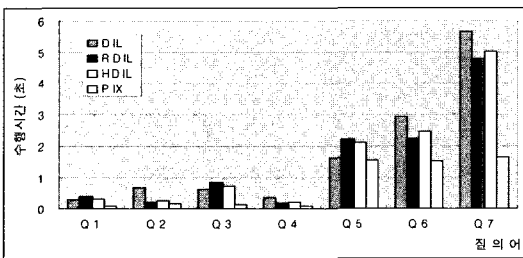


그림 9 여러 질의 처리 알고리즘의 수행 속도 비교

그림 10은 색인 레벨을 변화시켜가며 실험한 결과이다. 각 질의를 분할을 전혀 하지 않은 경우부터 두 번째 레벨에서 분할을 수행한 경우, 네 번째, 여섯 번째, 여덟 번째 레벨에서 분할을 수행한 다섯 가지 경우로 나누어 측정하였다. 실험 결과 분할의 수가 많을수록 성능이 향상되는 결과를 보였지만 여섯 번째 레벨 이하에서 성능 향상은 크지 않았다. 그림 12는 해당 XML 문서들을 분석한 결과로 각 레벨 별 최고 차수(3)를 보여준다. 다섯, 여섯 번째 레벨이 가장 차수가 높고 이 레벨은 성능 향

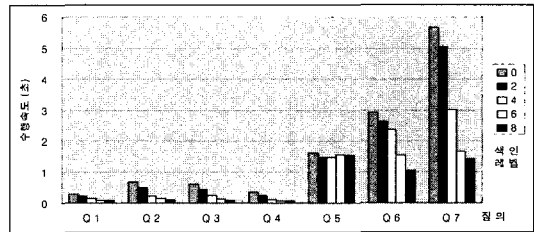


그림 10 분할 개수에 따른 질의 수행 시간

상이 가장 높은 색인레벨과 동일함을 알 수 있다. Q5의 경우에는 모든 경우가 거의 동일한 성능을 보여 좀 더 자세한 분석이 필요하다. Q5 질의에 사용된 키워드들의 실제 구성된 역색인 리스트를 보면 레벨 4에서 색인한 경우 모든 단어를 포함하고 있는 분할의 개수가 15였다. 레벨 4에서 최대 분할 수가 16인 것과 비교하였을 때 거의 모든 분할에 모든 단어가 존재하여 사실상 분할하지 않은 것과 비슷한 것을 알 수 있었다. 이와 대조적으로 Q7의 경우에는 모든 단어가 존재하는 분할의 수가 4로 전체 16개의 분할 중 4개의 분할만을 처리하여 질의 처리 알고리즘이 효과적이었음을 알 수 있었다. 이 실험에서 알 수 있듯이 PIX는 질의 대상 단어가 전 분할에 골고루 분포할 경우 큰 성능의 효과를 보이지 못하였는데 이는 보다 선별적인 PF함수를 고안함으로써 그런 경우를 줄일 수가 있을 것이다. 현재 PIX 구현에 사용된 PIXPF는 단순히 각 레벨에서 존재하는 노드가 짝수 번째 인지 홀수 번째 인지를 가지고 분할 값 P-value를 정하는데, INEX 컬렉션처럼 같은 DTD를 가지고 있는 균등한 성질의 문서들에서는 정보가 존재하는 위치가 균등하여 분할의 효과가 줄어들 수 있다.

그림 11은 Shakespeare 문서 셋에 대하여 검색 단어의 수를 늘려가면서 분할의 개수에 따른 성능 변화를 측정된 결과이다. 분할을 하지 않은 경우나 한 경우나 검색 단어의 개수와 비례하여 수행 속도가 오래 걸렸다.

3) 특정 레벨에 존재하는 노드들 중 자식 노드의 개수가 가장 많은 노드의 자식 노드의 개수

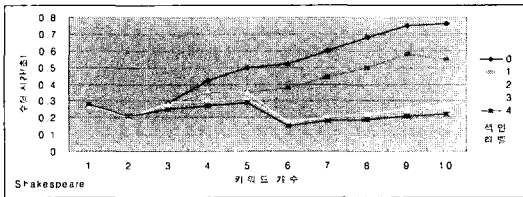


그림 11 키워드 개수에 따른 질의 수행시간

그러나 분할을 하지 않은 경우는 단어의 특성과 상관없이 선형적으로 증가하는 데 반하여 분할을 하지 않은 경우는 단어가 추가되면서 오히려 수행 속도가 줄어들기도 하고 전반적으로 검색 단어의 수에 비해 완만하게 수행 속도가 증가하였다. 이는 단어가 늘어나면서 모든 단어가 존재하는 분할의 수가 줄어드는 경우에는 단어의 수가 늘어나더라도 질의 처리의 대상이 되는 엘리먼트의 개수가 오히려 줄어들 수 있기 때문으로 분석할 수 있다. 그리고 분할에 의한 성능 향상은 INEX 컬렉션과는 다르게 레벨 2 이상에서는 거의 없음을 알 수 있다. 그림 12를 보면 테스트 문서의 경우 레벨 2에서 최대 차수가 최고임을 알 수 있다. 가장 성능향상이 큰 색인 레벨이 최고 차수가 가장 높은 레벨인 사실은 INEX 문서의 실험결과와 일치하는 것으로 문서 색인시 색인레벨을 정하는 한 가이드라인으로 판단할 수 있다.

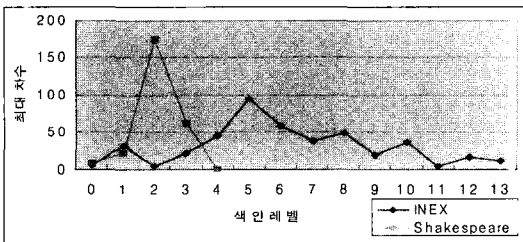


그림 12 테스트 문서 셋의 각 레벨 별 최대 차수 분포

6. 결론 및 향후 연구

정보 검색의 대상이 XML 문서가 되면서 이전에는 불가능했던 부분 검색 등이 가능해 졌다. 그러나 처리해야 할 정보의 양이 많아져서 이는 질의 처리에 큰 부담이 되었다. PIX는 이러한 XML문서에 단어 검색을 하기 위한 역색인 구조로 내부적인 엘리먼트들의 매칭 가능성에 따라 미리 엘리먼트들을 분할하여 질의처리 시간을 줄일 수 있었다. 분할에 의한 성능의 향상은 전체 역색인 리스트 중 처리가 필요한 분할들만을 로드, 디코딩하여 질의처리를 하였기 때문이다. 분할은 색인 시에 색인레벨을 정하여 이루어 졌고 색인 레벨 보다 더 자세한, 높은 레벨에 존재하는 답만을 생성한다. 더 일반

적인 답이 필요한 경우는 분할들을 합병하여 결과를 생성하는 것이 가능하여 분할을 하더라도 분할을 하지 않은 경우와 동일한 답을 생성할 수 있다. 이러한 분할, 합병 알고리즘은 Top-k 질의를 효율적으로 처리할 수 있다.

분할에 의한 성능 향상은 대부분의 질의에 나타났으나 단어들이 굉장히 고르게 분포하는 경우에 성능 향상은 크지 않았다. 이러한 고른 분포를 보이는 단어들에 대하여도 성능 향상을 보일 수 있는 분할 알고리즘의 연구가 필요하다. 이러한 분할 알고리즘은 규칙적인 형상을 가지고 있는 문서들도 분할에 의한 질의 처리 효과를 볼 수 있도록 분할 할 수 있어야 할 것이며 계산의 부담이 크지 않아야 할 것이다. 또 본 논문은 XML 문서에 대한 단어 검색의 성능 향상을 목표로 하여 검색의 품질에 관한 연구는 하지 않고 기존의 연구 결과를 활용 하였다. 여러 가지 가중치 알고리즘과 순위화 알고리즘을 비교 평가하여 검색의 품질을 높이는 문제와 전체 XML 문서에서 어떤 부분을 결과로 반환할 것인지의 문제는 현재에도 진행되고 있는 연구 분야이고 풀어야 할 문제이다.

참고 문헌

- [1] <http://www.w3.org/XML>
- [2] J. Robie, et al. XML query language(XQL). The Query Languages Workshop. W3c, Dec. 1998, <http://www.w3.org/TrandS/QL/QL98/pp/xql.html>.
- [3] A. Deutsch, M. Fernandez, et al. XML-QL: A query language for XML. The Query Languages Workshop. W3c, Dec. 1998.
- [4] XPath: XML Path language, Nov. 1999. <http://www.w3.org/TR/xpath>.
- [5] XQuery: A query language for XML, Feb. 2001. <http://www.w3.org/TR/xeury>.
- [6] J.P. Callan. Passage-Level Evidence in Document Retrieval. In Proc. of SIGIR, pp. 302-310, 1994.
- [7] R. Wilkinson. Effective retrieval of structured documents. In Proc. of SIGIR, pp.311-317, 1994.
- [8] J. Zobel, A. Moffat, et. al. Efficient retrieval of partial documents. Information Processing and Management, Vol. 31, No. 3, pp. 361-377, 1995.
- [9] N. Fuhr, K. Großjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In Proc. of SIGIR, pp.172-180, 2001.
- [10] A. Theobald, G. Weikum. The Index-based XXL Search Engine for Querying XML Data with Relevance Ranking. In Proc. of EDBT, pp. 477-495, 2002.
- [11] V. Hritidis, Y. Papakonstantinou, A. Balmin. Keyword Proximity Search on XML Graph. In Proc. of ICDE, pp. 367-377, 2003.
- [12] L.Guo,etal. XRANK: Ranked Keyword Search over

- XML Documents. In Proc. of SIGMOD, pp. 16-27, 2003.
- [13] S. Cohen, J. Mamou, Y. Kanza, Y. Sagiv. XSEarch: A Semantic Search Engine for XML. In Proc. of VLDB, pp.45-56, 2003.
- [14] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, A. Soffer. Searching XML Documents via XML Fragments. In Proc. of SIGIR, pp.151-158, 2003.
- [15] L. Mignet, D. Barbosa, P. Veltri. The XML Web: a First Study. WWW 2003.
- [16] S. Putz. Using a Relational Database for an Inverted Text Index. XEROX Technical Report '91.
- [17] D. Cutting, J. Pedersen. Optimizations for Dynamic Inverted Index Maintenance. In Proc. of SIGIR, pp. 405-511, 1990.
- [18] V. N. Anh, O. Krester, A. Moffat. Vector-Space Ranking with Effective Early Termination. In Proc. of SIGIR, pp. 35-42, 2001.
- [19] A. Theobald, G. Weikum. Adding Relevance to XML. WebDB 2000.
- [20] D Florescu, et al. Integrating Keyword Search into XML Query Processing. WWW '99.
- [21] S. Brin, L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine. WWW7 '98.
- [22] G. Salton and M.J McGill, "Introduction to Modern Information Retrieval," McGraw-Hill, New York, 1983.
- [23] A. Moffat, J. Zobel. Self-Indexing Inverted Files for Fast Text Retrieval. TODS Vol. 14, No. 4, pp. 349-379, 1996.
- [24] Jongik Kim, Ilhwan Choi, Hyun-Sook Lee and Hyoung-Joo Kim, "XDBox: Impelementation of XML object repository," in Proc. of KISS Spring Conference, April 2003.
- [25] <http://www.sleepycat.com>.
- [26] Initiative for the evaluation of XML retrieval.
- [27] <http://www.ibiblio.org/xml/examples/shakespeare/>.



이 형 동

1997년 홍익대 컴퓨터공학과(학사). 1999년 서울대 컴퓨터공학과(석사). 1999년~현재 서울대 컴퓨터공학부 박사과정. 관심분야는 데이터베이스, 정보검색



유 상 원

2002년 서울대 컴퓨터 공학과(석사). 2002년 3월~현재 서울대학교 전기, 컴퓨터공학부 대학원 박사과정



김 형 주

1982년 서울대학교 컴퓨터공학과 학사
1985년 University of Texas at Austin 전산학 석사. 1988년 University of Texas at Austin 전산학 박사. 1988년~1990년 Georgia Institute of Technology 조교수. 1991년~현재 서울대학교 컴퓨터공학과

교수



이 흥 래

1992년 3월~1997년 2월 서울대학교 원자핵공학과. 2002년 3월~2004년 2월 서울대학교 컴퓨터공학과 대학원 석사과정