문서 단위 순위화를 통한 XML 문서에 대한 키워드 검색 성능 향상

(Accelerating Keyword Search Processing over XML Documents using Document-level Ranking)

이 형 동 † 김 형 주 ††
(Hyungdong Lee) (Hyoung-Joo Kim)

요 약 XML 문서에 대한 키워드 검색은 사용자로 하여금 XML 문서의 복잡한 구조에 관한 지식 없이 쉽게 정보를 검색할 수 있게 해준다. 또한 사용자의 정보 요구에 대해 해당 정보를 포함하는 문서 전체를 반환하는 기존의 정보 검색 시스템과 달리 문서 내의 해당 정보를 포함하는 문서 조각을 결과로 반환함으로써 보다 빠르게 원하는 정보를 얻을 수 있도록 도와준다. 이러한 특징은 XML 문서 검색 시스템이 XML 문서를 문서 단위가 아닌 세부적인 엘리먼트 단위로 처리함으로써 가능하다. 하지만 이로 인해 대용량 문서들에 대한 질의 처리 부담 역시 가중되었다. 본 논문에서는 엘리먼트 단위 질의 처리의 비용을 줄이기 위해 XML 문서에 대한 문서 단위 순위화 기법을 제안하는데, 이는 결과물의 점수에 영향을 미치는 질의 키워드들의 문서 내에서의 근접도를 경로 노드 집합 정보와 이에 대한 유사도를 통해 구함으로써 엘리먼트 단위 질의 처리 결과를 예측하고 문서 단위 점수를 계산한다. 이러한 문서 중심의 뷰는 대용량 문서에 대한 순위화 혹은 필터링을 가능하게 해주며, 우리는 문서 단위 인덱스를 통해 순위가 높은 문서를 우선적으로 처리함으로써 Top-k 질의에 대해 검색 성능을 높였으며, 실험을 통해 해당 기법의 유효성과 성능 향상을 검증하였다.

키워드 : XML, 키워드 검색, 근접도, 문서 중심 뷰, 정보 필터링

Abstract XML Keyword search enables us to get information easily without knowledge of structure of documents and returns specific and useful partial document results instead of whole documents. Element level query processing makes it possible, but computational complexity, as the number of documents grows, increases significantly overhead costs. In this paper, we present document-level ranking scheme over XML documents which predicts results of element-level processing to reduce processing cost. To do this, we propose the notion of "keyword proximity" - the correlation of keywords in a document that affects the results of element-level query processing using path information of occurrence nodes and their resemblances - for document ranking process. In benefit of document-centric view, it is possible to reduce processing time using ranked document list or filtering of low scored documents. Our experimental evaluation shows that document-level processing technique using ranked document list is effective and improves performance by the early termination for top-k query.

Key words : XML, keyword search, proximity, document-centric view, filtering

1. 서 론

인터넷의 폭발적인 성장과 더불어 널리 사용되는 HTML은 표현 위주의 언어로 정보 관리나 교환 측면에서는 문제점이 발생하였다. 이를 극복하기 위해 XML [1]이 W3C에 의해 표준으로 제안되었고 현재 널리 통용되고 있다. 이는 XML의 풍부한 정보표현 능력 때문인데 학계, 산업계뿐만 아니라 의료, 음악 등 다양한 분야에서 정보 표현 및 교환 수단으로 자리 매김하고 있다.

· 본 연구는 BK-21 정보기술사업단과 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성, 지원사업(IITA-2006-C1090-0603-0031)의 연구결과로 수행되었음
† 학생회원 : 서울대학교 컴퓨터공학과
hdlee@oopsla.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학과 교수
hjk@oopsla.snu.ac.kr
논문접수 : 2005년 8월 17일
심사완료 : 2006년 5월 11일

이러한 XML은 주로 두 가지 측면에서 사용되어지고 있는데, 첫 번째는 데이타의 관점으로 정보교환이나 이종의 데이타베이스 자료의 통합을 위한 공통 형식으로 주로 사용된다. 예를 들면, 기존의 복잡한 EDI 프로토콜을 XML을 이용해서 기술함으로써 보다 이해가 쉽고 처리를 수월하게 해주는 XML-EDI 같은 전자 문서 교환 형식을 들 수 있다. 두 번째는 문서적인 관점에서 바라보는 측면인데, 기존의 문서자료들을 표준화된 XML 형식으로 구조화시켜 저장함으로써 검색 및 관리의 용이성 그리고 손쉽게 다양한 형식으로 출판이 가능한 XML의 장점을 얻기 위함이다. 이러한 관점의 예로는 희곡이나 논문 데이타 등 기존의 일반 문서에 구조적 태그 및 표현을 위한 태그 등을 덧붙인 XML 문서들을 생각해 볼 수 있다. 이러한 문서적인 관점의 자료들은 데이타 관점의 자료보다 비교적 구조가 단순하고, 데이타 관점의 문서가 가지는 값 형식이 단어나 짧은 문장인데 반해 장문이 오는 경우가 많다. 초기의 XML 질의 처리에 관한 연구는 XML을 주로 데이타의 관점으로 보고 기존의 데이타베이스의 질의어와 유사한 형태의 질의어를 이용해서 문서의 정확한 구조 정보를 바탕으로 구조(경로) 질의를 행하고 이에 대한 효율적인 처리에 초점이 맞추어졌다. 그러다가 점차 다른 형태에서의 변환이 아닌 순수 XML 문서로 작성되는 데이타가 증가함에 따라, 문서적 관점의 XML에 대한 처리에 관심이 높아지면서 유사 검색, 순위화 등 문서의 내용과 관련된 연구가 진행되었다.

XML 문서에 대한 질의는 크게 두 가지 형태로 나누어 볼 수 있는데, 구조적 질의와 내용 질의가 있다. 첫 번째는 사용자가 DTD와 같은 정보를 통해 XML 문서의 내부 구조에 대해 잘 알고 있는 경우나 일부 구조에 대한 정보를 가지고 있는 경우다. 이때 사용자는 RDBMS의 SQL과 유사하게 구조에 대한 명세와 내용에 대한 명세를 시스템에 제공하여 결과물을 얻는다. 이러한 질의 방식의 지원을 위해 XQL, XML-QL 등의 질의어가 제안되었고, XPath, XQuery 등이 W3C에서 표준으로 제안되었거나 정비 중이다. 이 방식의 장점은 보다 정확하게 원하는 자료를 추출할 수 있다는 점이며 단점으로는 사용자가 미리 문서의 구조 정보를 자세히 알아야 질의를 할 수 있다는 점이다. 따라서 일반 사용자들에게 보다 쉽게 정보를 전달해주기 위해 내용에 대한 간단한 형태의 키워드 질의어를 던져 원하는 정보를 추출해주는 문서 검색 시스템이 정보 검색 분야나 인터넷 검색 엔진 등에서 널리 사용되어지고 있다.

이러한 XML 문서 검색 시스템이 기존의 일반 문서 검색 시스템과 비교할 때 가지는 가장 큰 차이점은 기존의 문서 검색시스템이 질의어와 관련 있는 문서를 찾아 해당 문서 전체를 결과로 반환하는 반면 XML 문서 검색 시스템은 질의어와 관련있는 보다 작은 단위인 문서 내의 특정 부분을 결과값으로 반환한다는 점이다.

```
<article>
<sec><st></st>
    <p>The increasing availability of microcomputers and their
apparent multipurpose utility...</p>
    <p>...</p>
</sec>
<sec>
    <sub-sec><st>IBM 7010 System</st>
        <p>IBM began another experimental project...</p>
    </sub-sec>
    <sub-sec><st>Prototyping the IBM 1500 System</st>
        <p>Instructional systems such as Plato, which were
designed...</p>
        <p>The Stanford system was based initially...school
children being taught mathematics...</p>
    </sub-sec>
</sec>
<bib>
...
</bib>
</article>
```

그림 1 예제 XML 문서와 질의 결과

예를 들어 그림 1과 같은 XML 문서가 있다고 가정하자. 이 문서는 컴퓨터의 역사에 관한 문서의 일부분으로 여러 컴퓨터들의 변천사와 특징들을 소개하고 있다. 문서를 살펴보면 본문의 내용이 절을 의미하는 <sec> 엘리먼트와 그 하위에 <sub-sec> 엘리먼트로 구조화되어 있음을 알 수 있다. 만일 임의의 사용자가 여러 컴퓨터 시스템들에 관한 내용 중 수학 교육과 관련된 컴퓨터 시스템만을 찾고자 한다면 "instruction"[1]과 "mathematics"라는 키워드 질의어를 사용하여 검색을 수행하게 된다. 이 경우 XML 문서 검색 시스템은 질의 키워드와 관련된 문서를 찾은 후 해당 문서 전체를 반환하는 대신 해당 문서 내에서 수학 교육에 관한 내용을 다루고 있는 문서 조각을 반환한다. 그림에서 보면 해당 내용을 담고 있는 점선으로 표시된 <sub-sec> 엘리먼트가 반환된다. 이로써 사용자는 컴퓨터의 역사를 다루는 문서 전체의 내용을 다 보지 않고 보다 빠르게 원하는 정보를 얻을 수 있게 된다. 이러한 XML 문서 검색 시스템에서의 특징은 XML 문서 검색 시스템이 기존의 문서 검색 시스템과는 달리 XML 문서를 엘리먼트 단위로 처리하기 때문에 가능하다.

XML 문서 검색 시스템에서의 엘리먼트 단위 질의 처리는 일반 정보 검색분야에서 문서 단위로 역색인(inverted index)을 사용해서 처리하는 것과 유사하게 엘리먼트 단위의 역색인을 구축해서 처리한다. 엘리먼트

1) "instructional"과 "instruction"간의 비교에서 일어나는 스테밍(stemming) 문제는 여기에서 고려하지 않는다.

단위의 역색인은 문서 내에 존재하는 모든 색인어(term) 집합과 각 색인어가 출현하는 엘리먼트 포스팅 리스트로 구성되며, 사용자의 키워드 형식의 질의어에 대해 해당 키워드가 출현하는 엘리먼트들을 쉽게 접근할 수 있게 해준다. XML 문서 검색 시스템에서의 질의 처리기는 기존의 구조가 없는 문서와 달리, 구조를 포함하는 XML 문서 특징으로 인해 보다 복잡한 질의 처리 과정을 거친다. 엘리먼트 간의 구조적인 관계 및 이에 따라 변화하는 가중치를 고려해야하며 질의 키워드를 포함하는 하위 엘리먼트들을 결합하여 적절한 단위의 상위 엘리먼트를 결과로 생성해야 한다. 기존의 XRANK [2]와 같은 시스템에서는 해당 질의어의 색인어가 출현하는 엘리먼트 포스팅 리스트 전체를 한번만 스캔함으로써 이러한 과정을 처리할 수 있는 효율적인 알고리즘을 제시하고 있다. 하지만 실제로 사용되고 있는 XML 문서들을 분석한 최근의 연구 결과[3,4]를 보면 한 문서에서 처리해야 할 엘리먼트의 수가 수백, 수천 개 정도로 굉장히 많다는 사실을 알 수 있다. 따라서 대규모의 문서 컬렉션을 다루는 경우에는 엘리먼트 단위의 질의 처리 비용이 매우 커지게 된다.

한편 엘리먼트 단위의 질의 처리를 의미론적으로 생각해보면 다음과 같은 문제점이 있을 수 있다. 만약 문서 단위를 고려하지 않고 세부적인 엘리먼트들의 순위만을 고려해서 부분 결과물을 생성한다면 해당 결과물이 전체적인 문맥에서는 무의미한 결과물일 수 있다. 그림 2를 살펴보면, 두 개의 문서 A와 B가 존재하는데, 문서 A는 많은 질의 키워드가 고르게 분포되어 있고, B의 경우는 적은 수의 질의 키워드가 일부 구역에서만 나타난다. 직관적으로 보면 문서 A가 해당 질의어에 대한 주제를 다루는 문서일 가능성이 높고, 문서 B의 경우는 다른 주제에 관련된 문서에서 해당 질의어가 잠시 언급되거나 다의어 질의어인 경우 전혀 다른 문맥에서 사용되는 경우 일 수도 있다. 하지만 엘리먼트 단위 질의 처리에서는 문서 A와 문서 B가 동일한 조건에서 처리되어지는데, 일반적으로 상위 k개의 결과만을 요구하는 Top-k 질의에 대해서는 이러한 점이 불합리하다. 따라서 문서 단위에서의 순위화가 가능한 경우 높은 순위의 문서들을 우선적으로 처리함으로써 질의 처리 시간을 줄일 수 있다.

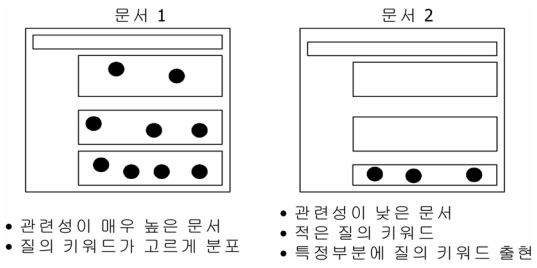


그림 2 질의 키워드 분포가 상이한 두 문서

본 논문에서는 앞서 언급한 엘리먼트 단위 질의 처리에서 발생하는 문제점들을 해결하기 위해 엘리먼트 단위 질의 처리를 기본으로 하면서 상위에 *문서 중심의 뷰(document-centric view)*를 제안함으로써 효율적인 질의 처리를 도모하고자 한다. 이는 엘리먼트 단위 질의 처리에서 결과물의 단위(granularity) 결정과 점수 결정에 영향을 미치는 요인인 키워드들에 대한 근접도(proximity)를 정의하고 이를 계산하여 색인어의 빈도수 등의 통계정보와 함께 결과물의 점수를 미리 예측한다. 키워드간의 근접도는 해당 키워드들이 출현하는 노드들의 조상-자손 관계 및 수평적 거리 등을 고려하며 질의 처리 마지막 단계에 결정이 되는 결과물의 단위를 질의 처리 시간 초기에 예측하기 위해 사용된다.

이러한 문서 중심의 뷰는 문서 순위화를 통해 질의 시간을 단축시키거나 전역 순위 정보를 부분 결과물의 지역 순위 정보에 반영함으로써 검색의 질을 향상 시키는 것도 가능하게 해준다. 예를 들면 인터넷 환경과 같은 대규모의 문서 시스템에 대해서는 모든 문서에 대해 엘리먼트 단위 질의처리를 수행할 수 없고, 질의 결과 역시 모든 결과물을 원하는 것이 아니라 Top-k 상위 결과만을 원하므로 최소 점수(threshold)를 지정하고 그 이하의 문서를 필터링해서 질의 처리를 하는 것이 가능하다. 본 논문에서는 엘리먼트 단위 질의 처리 이전에 문서 단위 순위화를 통해 생성되는 문서 순위 리스트를 이용해 문서 검색에서 일반적으로 사용되는 Top-k 질의에 대한 질의 처리 속도를 향상시키며, 이를 지원하는 문서 단위 인덱스 구조를 제안한다.

논문의 구성은 다음과 같다. 다음 장에서는 XML 문서에 대한 키워드 검색에 관한 데이타베이스 분야와 정보 검색 분야에서의 최근 연구 등을 소개한다. 3장에서는 기존의 엘리먼트 단위 질의 처리에 대한 소개와 XML 문서 순위화 작업에서 발생하는 문제점들을 기술한다. 4장에서는 이를 극복하기 위해 본 논문이 제안하는 문서 순위화 기법 및 이에 기반한 문서 단위 질의 처리 과정을 설명한다. 5장에서는 실험을 통해 본 논문에서 제안한 기법을 검증하며 마지막으로 결론 및 향후 연구를 통해 마무리한다.

2. 관련연구

XML 문서가 표준화되고 널리 사용되면서 질의 처리에 대한 관심도 높아졌다. 초기에는 주로 새로운 XML 질의 언어를 기반으로 하는 정확한 구조 질의에 초점이 모아졌으며, 보다 용이한 정보 검색을 위해 키워드 방식

의 질의어를 이용한 XML 문서 검색 시스템에 관한 연구 역시 진행되었다. 초기에는 주로 RDBMS에 저장된 XML 데이타에 대한 키워드 검색에 관한 연구[5]가 이루어졌으며 기존의 데이타베이스에 전문검색(full-text retrieval) 기능을 추가하기 위한 확장의 형태로 진행되었다. 이후 본격적인 XML 문서에 대한 키워드 검색에 관한 연구가 진행되면서 점차 정보검색 분야의 기술과 많이 융합되었다. 앞서 언급한 XRANK는 웹 페이지 단위에 적용되었던 Google의 PageRank[6] 기법을 XML 문서의 엘리먼트 단위에 적용시켜 엘리먼트 단위의 순위화 기법을 제안하였다. 그리고 조상-자손 관계를 쉽게 알 수 있는 듀이 아이디(Dewey ID)[7]를 이용하여 엘리먼트를 표현하고 이에 대한 역색인 구조를 이용하여 질의 키워드에 대한 최소 공통 조상 노드를 효율적으로 구하는 질의 처리 기법도 제시하였다. 본 논문에서도 기본적인 엘리먼트 단위의 질의 처리를 위해 해당 기법을 활용한다. XSEARCH[8]는 레이블, 키워드 형태의 부분적인 구조 질의어를 지원하며 검색 결과물을 결정 문제에 대해 의미론적(semantic)으로 접근하였으며 질의 처리를 위해 정보검색에서 널리 사용되어지는 벡터 공간 모델을 XML 문서에 적용하였다. [9]에서는 XML 데이타베이스에 전문검색 기능을 지원하기 위해 기존의 데이타베이스 질의어를 확장하였으며, 효율적인 키워드 검색 질의 처리를 위한 TermJoin과 같은 알고리즘을 제안하였다. 한편 정보검색 분야에서도 XML 문서 검색에 관한 연구가 진행되었는데, [10]에서는 질의어로 XML 문서 조각(fragment)을 사용하였으며, 구조적인 XML 문서 환경에 적합한 색인어 가중치 기법을 구하기 위해 기존의 tf-idf(term frequency, inverse document frequency)[11] 방식을 여러 형태로 변형하고 실험하였다.

XML 문서 검색 시스템은 문서 전체가 아닌 XML 문서의 일부분을 반환한다는 측면에서 정보검색 분야에서 기존에 이루어졌던 문단 검색(passage retrieval)[12]과 공통점을 찾을 수 있다. 문단 검색은 하나의 문서를 의미 있는 여러 문단으로 나누어 문단별로 색인하고 검색하여 문서전체를 반환하던 기존의 정보검색과 달리 사용자의 요구에 가장 적합한 문단을 반환한다. 이러한 정보 검색 분야에서의 문단 검색은 처리 방법 보다는 여러 가지 문서 분할 방법에 따른 효과 및 검색 정확도 그리고 순위화 기법의 변화에 초점을 두고 있다. 이러한 내용을 다룬 [13]에서는 문서를 여러 문단 단위로 나누고 결과를 문서 단위 혹은 문단 단위로 반환하는 경우로 나누어 여러 종류의 색인어 가중치 및 순위화 기법을 적용하여 각각에 대한 실험을 통해 검색 정확도를 측정하여 비교하였다. 이 논문에서는 문단 단위 검색에서는 기존의 문서 단위 검색과는 다른 문단별 색인 및 순위화 기법이 필요하다는 것과, 문서 단위 처리 기법을 그대로 문단 단위 검색에 적용하는 경우 검색 정확도가 낮아진다는 것을 보였다.

정보 검색 분야에서 문서 검색 시스템에서의 질의 처리 성능을 향상시키기 위한 시도 역시 계속 되어왔는데, 근본적인 생각은 색인을 생성할 때 문서 순위화를 위한 여러 가지 척도를 제안[14-16]하고 해당 척도에 의해 문서를 내림차순으로 정렬하고 저장하는 것이다. 이후 사용자의 Top-k 질의에 대해 문서 전체에 대해 질의 처리 과정을 수행하지 않고 순위가 높은 문서들을 우선적으로 처리하거나 일정 점수가 되지 못하는 문서들을 지나침으로써 질의 처리 시간을 줄인다. 앞서 언급한 XRANK에서는 기본적인 엘리먼트 단위 질의 처리 알고리즘 외에 RDIL(Ranked Dewey Inverted List) 알고리즘을 제안하였는데, 이는 순위별로 정렬된 엘리먼트 리스트를 통해 Top-k 질의에 대한 성능을 높였으나 듀이 순서를 포기함으로써 공통 조상을 찾기 위해 별도의 인덱스를 유지해야하는 부담이 있다.

3. 검색 모델 및 엘리먼트 단위 질의 처리 소개

이번 장에서는 문서 단위 순위화 기법을 제안하기에 앞서 본 논문에서 사용되는 XML 데이타 모델과 기본적인 질의 처리인 엘리먼트 단위 질의 처리 방식에 대해 소개한다.

3.1 데이타 모델(Data Model)

본 논문에서는 XML 문서를 일반적으로 사용되는 노드에 라벨이 있는 트리(node-labeled tree) 모델을 사용한다. 해당 모델은 그림 3과 같이 구성되는데, 루트 노드로 <article> 엘리먼트가 있고, 자식 노드로 각 절을 나타내는 <sec> 엘리먼트 노드들이 존재한다. 이와 같이 XML의 각 엘리먼트들은 하부에 다시 엘리먼트(sub-element)나 애트리뷰트(attribute)를 가질 수 있다. 일반적으로 엘리먼트와 애트리뷰트는 편의를 위해 구별하지 않고 모두 엘리먼트로 처리해도 무방하며 트리의 노드(node)로 표현된다. 각 노드는 식별을 위하여 고유한 아이디를 가지며 아이디를 부여하는 방법은 구현 방법에 따라 여러 가지가 존재하는데, 본 논문에서는 듀이 아이디[2]를 사용한다. 트리의 레벨은 루트로부터 시작되며, 루트가 레벨이 가장 높고 단말(leaf) 노드가 가장 낮다고 정의한다.

3.2 질의 씨맨틱(Query Semantics)

이제 XML 문서에 대한 키워드 검색의 씨맨틱을 정의해보자. 질의 씨맨틱은 사용자의 키워드 질의어를 어

2) 루트로부터 시블링(sibling)간의 순서를 나타내는 숫자들을 나열하는 방식으로 각 레벨은 점으로 구분한다.

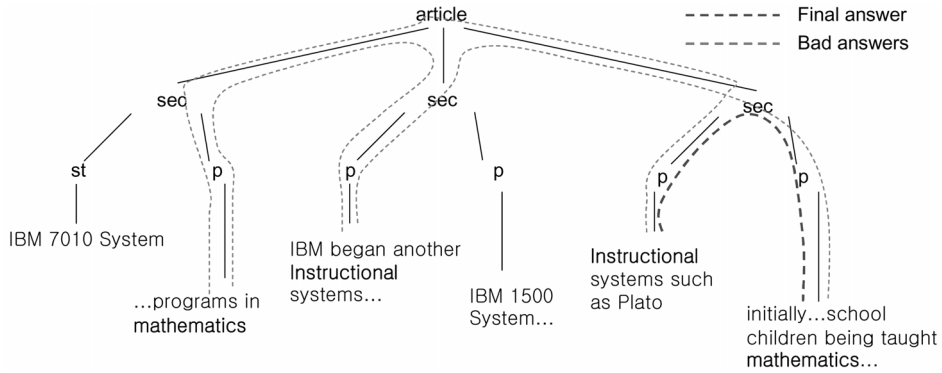


그림 3 XML 문서 모델 및 질의 씨맨틱

떤 의미로 이를 해석할 것인지에 대한 문제와 그에 합당한 결과물이 어떤 것인지를 결정한다. 본 논문에서의 질의어 형식은 구조에 대한 정보는 포함되지 않고 내용에 대한 키워드만을 포함하는 키워드들의 집합이다. 일반적으로 키워드 질의를 해석함에 있어 두 가지 씨맨틱이 있을 수 있는데, 첫 번째는 주어진 모든 키워드를 포함하는(conjunctive) 엘리먼트를 해로 생각하는 경우이며, 다른 하나는 적어도 하나의 키워드를 포함하는(disjunctive) 엘리먼트를 해로 생각하는 경우이다. 전자의 씨맨틱이 사용자의 요구에 보다 부합하는 결과이며 일부 키워드에 대한 질의 처리는 이를 이용해서 쉽게 확장될 수 있다. 따라서 본 논문에서는 키워드 해석을 모든 질의 키워드에 대해 만족하는 결과로 정의한다.

정의. 후보 결과

질의어로 n개의 키워드 집합 $K=\{k_1,...,k_n\}$가 주어지고 키워드 k_i의 출현 노드 집합을 L_i, n_i를 L_i의 원소라고 할 때, 후보 결과는 n개의 L_i 집합 각각의 임의 노드 n_i에 대한 최소 공통 조상 $LCA(n_1,...,n_n)$로 정의한다.

XML의 구조적인 특징으로 인해 모든 질의 키워드 출현 노드에 대한 최소 공통 조상 노드를 결과로 반환하는 것이 일반적이다. 예를 들어 그림 3의 XML 문서를 보면 두 키워드 “instruction”과 “mathematics”에 대한 후보 결과들을 보여준다. 4개의 단말 노드들이 해당 키워드를 포함하고 있으며, 점선으로 표시된 단말 노드 조합에 따라 모든 키워드를 포함하는 후보 결과는 총 4개로 3개의 <article>와 하나의 <sec> 노드가 된다.

하지만 모든 후보 결과를 계산하는 것은 매우 비효율적인데, 질의 키워드의 증가와 이에 따른 키워드 출현 노드의 증가로 인해 그 조합이 매우 많아지기 때문이다. 또한 위의 예와 같이 루트인 <article> 노드를 반환하는 것은 문서 전체를 반환하는 것으로 기존의 문서 검색 시스템과 동일하게 된다. 따라서 후보 결과로 얻어진 결과물에 대해 최종적인 결과 씨맨틱을 정의해야하며, 일반적으로 레벨이 낮은 구체적인 결과를 레벨이 높은 일반적인 결과보다 바람직한 결과로 선택한다. 위의 예에서는 후보 결과 중 레벨이 낮은 <sec> 노드가 결과로 선택된다.

3.3 엘리먼트 단위 질의 처리 개요

XML 문서에 대한 질의 처리는 엘리먼트 단위로 수행된다. 이를 위해 일반적인 정보 검색 시스템에서 에서 널리 사용되어지는 역색인(inverted index) 구조를 이용하는데, 기존의 역색인 구조와 차이점은 키워드에 대한 포스팅이 문서가 아닌 해당 키워드가 출현하는 엘리먼트 노드 리스트라는 점이다. 시스템은 XML 문서에 대한 파싱(parsing)[3] 작업을 통해 모든 엘리먼트들과 해당 엘리먼트에 출현하는 색인어(term)들을 추출한다. 그 후 색인어 별로 해당 색인어가 출현하는 노드들의 포스팅(posting) 리스트를 연계시키는 역색인을 구축해 놓는다. 사용자의 질의어가 주어지면 시스템은 역색인을 이용해 질의어 내의 키워드들이 출현하는 모든 노드들의 포스팅 리스트를 불러온다. 이후의 처리 과정은 포스팅들을 조합하여 모든 질의어가 포함되는 최소 공통 조상 노드를 찾는 작업이 필요하다.

XRANK의 DIL(Dewey Inverted List)과 같은 알고리즘이 최소 공통 조상 노드에 대한 효율적인 계산을 할 수 있게 해주며 본 논문에서도 DIL 알고리즘을 기본적인 엘리먼트 단위 질의 처리에 사용한다. DIL 알고리즘은 엘리먼트 단위의 역색인을 사용하며, 듀이 아이디[17] 순서로 정렬된 엘리먼트 포스팅 리스트를 스캔하면서 최소 공통 조상을 구한다. 이때 조상-자손 관계를 쉽게 알 수 있는 듀이 아이디의 특징을 이용하며 전체 엘리먼트 포스팅 리스트를 한번만 스캔해서 모든 결과를 얻을 수 있는 장점이 있다.

3) XML 문서에 대한 파싱 작업은 SAX(Simple API for XML)나 DOM(Document Object Model)과 같은 파싱을 통해 엘리먼트 라벨과 내용을 추출한다.

3.4 엘리먼트 단위 순위화(Ranking)

기존의 문서검색 시스템이 문서 단위로 점수가 유지된 것과 달리 XML 문서 검색에서는 일반적으로 엘리먼트 단위로 점수가 매겨지며 이들의 결합으로 결과물의 순위가 결정된다. XML 문서의 각 노드는 순위화를 위한 점수를 포함하는데, 일반적으로 출현하는 색인어에 대한 통계 정보를 이용한다. 정보검색에서 널리 사용되어지는 tf-idf[18] 색인어 가중치 기법이나 이를 응용해 기본의 문서 단위를 XML의 단말 노드 단위에 적용시킨 XSEARCH의 tf-ilf와 같은 가중치 기법도 있으며, XRANK의 ElemRank와 같은 경우들 모두 엘리먼트 단위의 점수를 갖는다.

XML 순위화에서 고려해야할 다른 점은 XML의 구조적 특징에서 기인한다. XML 문서는 계층적인 구조를 가지며, 하나의 엘리먼트가 다른 엘리먼드들을 내포하므로 하위 노드가 해당 키워드를 포함하는 경우 해당 노드의 상위 모든 조상 노드들 역시 해당 키워드를 간접적으로 포함하게 된다. 따라서 상위 레벨의 노드들에 대한 추가적인 처리가 필요하다. 이를 위해 질의 처리 과정에서 하위 노드의 점수는 상위 레벨의 노드로 전파되고 합쳐지면서 점차 그 값을 감소(augmentation)[19]시키는 기법이 일반적으로 적용되어진다. 예를 들어 그림 3에서 <p> 노드는 “instruction”이라는 색인어를 직접 포함하고 그 부모 노드인 <sec>은 해당 색인어를 간접적으로 포함하므로 질의 처리 과정에서 <sec> 노드의 “instruction” 색인어에 대한 점수는 “<p>”에서의 점수를 감소시켜준다. 이런 식으로 질의 처리 과정에서 상위 레벨의 노드로 점수가 전파되어가면서 최종적으로 결과물 단위(granularity)가 결정되며 그 시점에 해당 결과물에 대한 점수도 계산된다. 다음 공식은 최종적으로 결정된 노드(n_{result})의 사용자 질의 Q에 대한 점수를 나타낸다. 최종 노드에 대한 점수는 각 키워드가 출현하는 노드(n_{node})들에 대한 점수 합으로 결정된다. w는 각 키워드 점수의 가중치를 나타내며, 각 키워드에 대한 점수는 해당 키워드를 직접적으로 포함하는 리프 노드 점수에서 상위 레벨로 올라감에 따라 경로의 길이에 따라 감소되는 일정 비율(d)에 따라 점수를 감소시킨다.

$$R(n_{result}, Q) = \sum_{1 \le i \le n} S(n_{node}, k_i) \times w,\ Q = \{k_1, k_2, \mathrm{K}, k_n\},\ S(r_{node}, k_i) = S(l_{node}, k_i) \times d$$

4. XML 문서 순위화(Document Ranking)

이번 장에서는 이전 장에서 언급한 XML 문서 검색의 특징 및 기본적인 엘리먼트 단위 질의 처리 기법을 바탕으로 본 논문의 제안하는 XML 문서에 대한 순위화 기법에 관하여 기술한다. 먼저 XML 문서 순위화에 있어 발생하는 문제점에 대해 설명하고, 중요 인자로 작용하는 키워드 근접도에 대해 다루고 이를 이용한 문서 순위화 기법을 설명한다.

4.1 문서 단위 순위화(Document-level Ranking)

XML 문서 순위화 과정에서 기존의 문서 검색 시스템에서의 순위화와 다르게 고려해야할 사항은 동적으로 결정되는 결과물의 단위(granularity)와 엘리먼트 단위의 순위다. 기존의 문서 검색 시스템에서는 색인 및 결과가 모두 문서 단위로 이루어지지만 XML 문서 검색 시스템에서 색인은 엘리먼트 단위로 이루어지며 결과물의 단위 결정은 엘리먼트 단위 질의 처리 과정 마지막에 이루어진다. 또한 한 문서 내에 존재하는 결과물의 개수도 미리 알 수 없다.

결과물의 점수 결정 역시 기존의 문서 단위 순위화와는 차이점이 있다. 기존의 문서 검색 시스템에서 문서의 점수는 일반적으로 문서 내에 존재하는 색인어(term)의 통계정보를 이용하는데 보다 정확한 반영을 위해 색인어의 빈도수에 가중치를 주게 되며 이러한 기법에는 tf-idf 방식이 대표적이다. 이는 문서 내에 존재하는 색인어의 개수와 해당 색인어의 전체 문서 컬렉션에서의 가중치를 고려해서 해당 문서의 중요도를 결정한다. 하지만 XML 문서 검색에서는 해당 문서가 구조화 되어 있으므로, 하위 엘리먼트의 점수에 대한 상위 엘리먼트로의 전파 및 공통 조상 노드 결정 등 추가적인 작업이 필요하다.

따라서 기존의 방식을 그대로 적용하는 것은 문제가 있다. 앞서 언급한 [10]의 연구 결과에 의하면 문서 전체가 아닌 일부분을 반환하는 문단 검색에서 문서 내부 구조를 무시하고 기존의 문서 단위 반환 방식과 동일하게 하나의 단일 개체로 취급해서 전체 문서 내의 색인어 통계 정보를 이용한 순위화는 검색의 질을 떨어뜨린다는 것을 보이고 있다. 이는 문서의 순위가 높다고 해서 해당 문서의 부분 결과물들의 순위도 높은 것이 아니기 때문이다. 따라서 XML 문서에 대한 순위화는 부분결과를 반환한다는 점과 각 부분결과가 지역 정보에 의해 점수가 정해진다는 점 등을 고려한 문서 순위화 기법이 필요하다.

4.2 결과 단위 결정 문제

XML 문서에 대한 엘리먼트 단위 질의 처리 과정에서는 질의어 내의 모든 색인어들이 출현하는 노드 리스트를 스캔하면서 각 색인어가 출현하는 단말 노드들의 조상 노드들을 탐색하여 모든 색인어를 포함하는 최소 공통 노드를 결과로 반환한다. 이때 결과 노드가 결정되는 시점은 최소 공통 노드가 결정되는 엘리먼트 단위 질의 처리의 마지막 단계가 된다. 그러므로 문서에 대한 점수에 영향을 미치는 문서 내의 결과물의 개수나 각 결과물의 점수 등의 정보를 색인 구축 시기나 질의 처

리 초기에 아는 것은 불가능하다. 결국 문서의 점수를 계산하는 가장 단순한 방법은 엘리먼트 단위의 질의 처리 과정 전부를 수행하여 각 문서들이 가지는 결과물들을 구하고 각 점수들을 적절한 가중치에 의해 합하는 것이다. 하지만 이 방법은 엘리먼트 단위 질의 처리 과정 전체가 수행되어야 하므로 비현실적이고 엘리먼트 단위 질의 처리 이전에 문서를 순위화하려는 우리의 목적과 부합하지 않는다. 그러므로 우리의 접근 방법은 엘리먼트 단위 질의 처리 과정에 미치는 영향 요인을 분석하여 질의 처리 초기에 문서 단위 순위를 예측하는 것이다.

엘리먼트 단위 질의 처리 과정에서 얻어지는 결과물의 순위에 영향을 미치는 요인에 대해 알아보면 다음과 같다.

- 키워드 출현 노드의 점수 : 결과물로 반환되는 부분 트리 내에는 주어진 질의 키워드들을 포함하는 여러 노드들이 존재한다. 따라서 노드의 개수 및 해당 노드에 속하는 색인어들의 빈도수가 영향을 미친다. 노드들이 다수 포함[4]되어 있으면 많은 결과물을 생성해 낼 가능성이 높을 뿐만 아니라 해당 결과물의 점수 역시 높아지게 된다. 각 노드의 점수는 해당 노드에 존재하는 질의 키워드 색인어의 빈도수가 높을수록 커지게 된다.
- 문서 내 키워드들 간의 근접도(proximity) : 키워드 간의 근접도는 한 문서 내에서 해당 키워드들이 출현하는 노드들의 거리 및 노드들의 분포가 특정 구역에 집중적으로 나타나는지 아니면 고르게 퍼져있는지 여부를 나타낸다. 근접도가 높은 노드들이 집중적으로 나타나면 해당 노드들이 낮은 레벨에서 많은 결과를 만들어낼 확률이 높아진다. 반면 키워드가 출현하는 노드들이 고르게 분산되어 있고 근접도도 낮으면 생성되는 결과물의 개수도 적어지고 공통 조상 역시 루트 노드에 근접하게 되어 결과물의 점수도 낮아지게 된다. 그림 4를 보면 두 키워드 k1과 k2가 출현하는 노드들을 보여준다. 각 노드들의 점수는 동일하다고 가정하자. 문서 d1의 경우 키워드 k1과 키워드 k2의 출현 노드들이 바로 부모 노드에서 만나므로 거리도 가깝고 노드들이 집중적으로 나타나며 그 결과 공통 조상이 낮은 레벨에서 결정된다. 하지만 문서 d2의 경우는 동일한 출현 빈도에도 불구하고 거리도 멀고 분산되어 나타나는 분포 때문에 높은 레벨에서 결과물이 결정되며 해당 결과물의 점수 역시 낮아지게 된다.

위의 두 가지 요인 중 첫 번째 색인어의 통계적 정보는 문서 색인 시 얻을 수 있는 정적인 정보다. 하지만

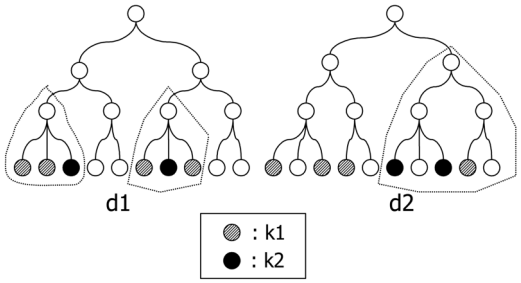


그림 4 키워드 분포에 따른 질의 결과

두 번째 키워드 간의 근접도 정보는 다르다. 사용자의 질의는 임의적(ad-hoc)이므로 질의어가 주어지기 전에는 어떠한 키워드들에 대한 근접도를 요구할지 알 수가 없다. 물론 사전에 모든 키워드 조합에 대해 근접도 값을 계산해 놓는 방법이 있으나 공간적으로나 시간적으로 비현실적이다. 따라서 키워드 근접도 예측을 위해 미리 계산되어진 자료를 저장하고 이를 바탕으로 질의 처리 초기 단계에 주어진 질의 키워드들의 근접도를 예측하는 과정이 필요하며, 다음 절에서 이에 대해 설명한다.

4.3 키워드 근접도(keyword proximity)

이번 절에서는 키워드 근접도에 대한 특성 분석과 함께 이에 대한 정의를 다룬다. 엘리먼트 단위의 질의 처리는 사용자에 의해 주어진 n개의 질의 키워드에 대해 각 키워드가 출현하는 노드들을 동일 키워드 별로 그룹지어진 릴레이션(r)들의 집합{r_1, r_2, ..., r_n}에 대해 처리된다. 이러한 노드들은 듀이 순서에 의해 정렬되어 유지되며 질의 처리기는 순서대로 역색인 리스트를 스캔하면서 처리하게 되는데 모든 질의 키워드를 포함하는 최소 공통 조상이 결정되면 그때까지의 노드들에 대해 점수를 계산하고 결과로 반환한다. 그러므로 노드들이 문서 트리 내에 서로 근접[5]해서 위치해야 낮은 레벨에서 결과가 결정되며 상위 노드로 결합될 때 일어나는 점수의 감소가 적어 보다 높은 점수를 가지게 된다. 또한 모든 질의 키워드들이 집중적으로 분포해야 낮은 레벨에서 보다 많은 해가 결정될 가능성이 높아진다.

키워드 근접도는 해당 키워드가 출현하는 노드들의 위치 및 전체 질의 키워드들의 분포에 영향을 받게 되는데 이들 각각의 특징을 자세히 살펴보도록 하자. 첫 번째 노드의 위치 정보는 해당 노드들의 공통 조상 노드 레벨 결정에 영향을 미친다. XML 문서는 트리 구조이므로 문서 내의 모든 노드들은 결국 루트에서 만나게 되며, 위치가 서로 근접할 경우 보다 낮은 레벨에서 만나게 된다. 공통 조상의 레벨이 높아질수록 점수의 감소가 이루어지므로 트리 내의 임의의 두 노드가 서로 낮

4) 한 결과물에서 동일 키워드에 대한 여러 출현 노드들의 점수는 하나로 합해지거나 가장 큰 값을 선택하는 방식 등이 가능하다.

5) 트리의 넓이 개념에서의 근접도를 의미한다.

은 레벨에서 만나게 되면 근접도가 높다고 하며 높은 레벨에서 만나게 되면 근접도가 낮다고 할 수 있다. 예를 들어, XML 트리의 루트 레벨이 아닌 특정 레벨 H_b를 가정하자. 그러면 레벨이 H_b인 노드 각각을 루트(r_i)로 하는 부분 트리들을 구할 수 있다. 그러면 각 부분 트리에 속하는 노드들은 공통 조상 노드는 H_b보다 낮게 된다. 따라서 원하는 최소 근접도에 따라 적당한 레벨 H_b를 정하면 해당 노드들이 동일 부분 트리에 속하는지 여부에 따라 근접도의 최소 한계가 정해진다. 부언하면 같은 부분 트리에 속하는 노드는 최소 근접도 보다 크거나 같게 되고 속하지 않으면 H_b 보다 높은 레벨에서 만나게 되므로 최소 근접도 값보다 작게 된다. 두 번째 노드들의 분포 정보는 각 질의 키워드가 출현하는 노드들의 밀집도를 의미하며, 생성되는 결과물의 레벨에 영향을 미친다. 질의 씨맨틱에 의하면 모든 질의 키워드를 하위에 포함하는 최소 공통 조상 노드를 결과물로 생성하므로 r1, r2 각각의 빈도수가 높아도 r1과 r2가 집중적으로 나타나지 않으면 높은 레벨에서 해를 생성하게 되므로 해당 결과물의 전체적인 점수는 낮아지게 된다. 따라서 서로 다른 키워드들에 대한 출현 노드들이 응집되어 있으면 근접도가 높다고 할 수 있다.

4.4 경로 노드 집합(Path Node Set, PNS)

앞 절에서 언급했던 특징들을 반영하는 키워드들 간의 근접도를 구하기 위해 각 키워드가 출현하는 노드들의 특징 정보를 유지한다.

정의. 경로 노드 집합

문서 D에 대한 사용자의 질의 키워드 k의 경로 노드 집합을 다음과 같이 정의한다.

$$PNS_{k,D} = (A(k,p_1), A(k,p_2), \mathrm{K}, A(k,p_n))$$

$$A(k',p') = \{n | n \in ancestors(o), o \in occ(k',p'), p' \in partition(D)\}$$

$ancestors(o)$: 노드 o의 조상 노드들

$occ(k',p')$: 분할 p′에 속하며 키워드 k′를 포함하는 노드들

$partition(D)$: 문서 D의 분할들

경로 노드 집합은 전체 문서 트리를 서브 트리로 분할한 후, 한 서브 트리에 속하는 임의의 키워드가 출현하는 모든 노드들에 대한 조상 노드들의 합집합으로 정의된다. 다시 말하면 경로 노드 집합은 한 키워드가 존재하는 노드들의 루트로부터의 모든 경로를 나타낸다.

해당 정보는 각 키워드마다 문서 내의 분할 별로 유지되며 이후 다른 키워드와의 근접도 계산에 이용되며 이에 대해서는 다음 절에서 다룬다. 문서 분할(partition)은 의미적으로는 기존 문서의 문단과 유사한 역할을 하며 문서의 범위(scope)를 줄임으로써 보다 정확한 근접도를 예측하기 위함이다. 실제 분할 작업은 루트에 근접한 상위 레벨 값(H_b)으로 정의되는데, 예를 들어 H_b를 1로 지정하는 경우(루트가 0) 이는 루트의 자식들을 루트로 가지는 부분 트리들로 분할된다. 해당 값은 색인 시 원하는 근접도 정도에 따라 결정된다. 높은 근접도를 원하는 경우 낮은 레벨 값을 지정하게 되며 기준 레벨이 낮아지면 분할의 개수가 늘어나게 된다. XML 문서의 스키마(schema) 정보나 엘리먼트 분포에 관한 통계 정보를 이용하면 보다 의미 있는 분할이 가능하다. 예를 들어, 논문의 각 장을 분할하거나 신문의 각 섹션 등 의미적인 단위로 분할하거나 색인어의 개수가 균등하게 분할하는 것을 생각해 볼 수 있다. 이때 검색 대상이 되는 문서들은 분할의 일관성을 위해 스키마가 동일한(homogeneous) 문서들로 가정한다.

4.5 경로 노드 집합 유사도

집합 유사도의 개념은 집합 간의 원소들을 비교해 집합이 얼마나 비슷한지 여부를 판가름하는 척도이며, 웹 페이지의 중복 검사 및 데이타 마이닝 등 다양한 응용에 사용된다[20]. 예를 들어 웹 페이지 간의 중복 여부를 판가름하기 위해서 페이지 내의 텍스트 조각을 각 페이지 집합의 원소로 보고 두 집합의 유사도를 계산한다. 유사도가 높으면 서로 유사한 텍스트 조각들을 포함하고 있는 경우가 되며 일정 점수 이상의 유사도인 경우 중복 페이지로 판단할 수 있다. 본 논문에서는 키워드들 간의 근접도 문제를 키워드들이 출현하는 노드들의 루트로부터의 해당 노드까지의 경로 집합 간의 유사도 문제로 보고, 문서 트리의 루트로부터 키워드 출현 노드까지의 경로에 나타나는 경로 노드 집합들 간의 유사성을 검사한다. 이 기법의 근본적인 생각은 후보 공통 조상 노드 간의 유사도가 높으면 경로의 중복이 많아지며 그러면 보다 낮은 레벨에서 만나게 되며 보다 많은 해를 생성할 수 있다는 점에서 착안한 것이다. 각 키워드에 대해 얻어진 경로 노드 집합을 이용해서 각 키워드들의 근접도를 다음과 같이 정의한다.

정의. 경로 노드 집합 유사도(Set resemblance of PNSs)

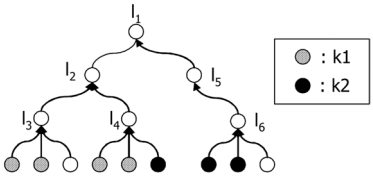
키워드들 간의 근접도를 집합간의 유사도 개념을 적용하여 각 키워드에 대한 경로 노드 집합 간의 유사도로 정의한다. 질의 키워드 Q={k_1, k_2, ..., k_n}의 분할 (p)에 대한 PNS 집합 유사도는 다음과 같다.

$$\rho = \frac{|PNS(k_1,p) \cap PNS(k_2,p) \cap \Lambda \cap PNS(k_n,p)|}{|PNS(k_1,p) \cup PNS(k_2,p) \cup \Lambda \cup PNS(k_n,p)|}$$

각 키워드가 출현하는 노드들의 루트까지의 경로가 많이 중복될수록, 즉 후보 공통 조상 노드가 많을수록 집합들 간의 교집합이 커지면서 집합 유사도는 높아지며, 이는 낮은 레벨에서 공통 조상이 될 확률이 높음을 의미한다. 또한 주목할 사항은 집합 유사도가 단순히 집합들의 중복 비율만을 표현하는 것이 아니므로 비대칭

적으로 원소가 많은 집합과 적은 집합인 경우 중복 비율은 높을 수 있으나 유사도는 낮게 된다. 이러한 경우 일부 결과물의 점수는 높을 수 있으나 생성되는 결과의 개수가 적어진다. 그림 4를 다시 살펴보면 문서 d2의 경우 PNS 유사도가 낮으며 출현 노드들이 서로 근접관계에 있지 않아 루트에서 결과가 생성된다. 문서 d1의 경우는 PNS 유사도가 높음을 쉽게 알 수 있으며, 문서 d1의 경우 루트의 자식으로 분할을 구성하면 보다 정확한 유사도 측정이 가능함을 알 수 있다.

두 키워드의 출현 노드가 바로 부모 노드에서 만나게 되는 경우 가장 높은 값으로 1을 가지며, 분할의 루트에서 만나게 되는 경우 루트 노드만을 공유하므로 가장 작은 값을 가지게 된다. 그림 5를 보면 문서 내의 임의의 분할(p)에서 키워드 k_1과 k_2에 대한 노드 근접도 집합 및 그들 간의 유사도를 보여주고 있다. 두 집합의 교집합인 {l_1, l_2, l_4}는 두 키워드 k_1과 k_2가 출현하는 노드들이 공통 조상 노드가 될 가능성이 있는 후보 공통 조상 노드들임을 알 수 있다.



$PNS(k_1,p) = \{l_1, l_2, l_3, l_4\}$,
$PNS(k_2,p) = \{l_1, l_2, l_4, l_5, l_6\}$

$$\frac{|PNS(k_1,p) \cap PNS(k_2,p)|}{|PNS(k_1,p) \cup PNS(k_2,p)|} = \frac{3}{6}$$

그림 5 노드 근접도 집합 유사도 예제

4.6 경로 노드 집합의 요약 자료

키워드 간의 근접도를 구하기 위한 경로 노드 집합은 색인어별로 각 색인어가 출현하는 모든 노드들에 대해 분할 별로 계산되고 유지된다. 그러므로 노드의 개수가 많은 문서들인 경우 해당 정보를 유지하는 것은 공간적으로나 유사도 계산 비용도 부담이 된다. 공간적인 측면은 특히 빈도가 높고 고르게 분포가 나타나는 키워드에 대한 경로 노드 집합인 경우 단말 노드를 제외한 대부분의 내부 노드들을 다 저장하게 될 수 있으며 키워드의 빈도수에 따라 가변 크기로 결정되므로 관리의 어려움도 있다. 두 번째 시간적인 측면은 집합 간의 유사도를 구하기 위해 질의 처리 시간에 질의 키워드 개수(n)에 따라 n원 교집합 및 합집합 연산을 수행해야한다. 따라서 경로 노드 집합을 그대로 유지하는 것은 비현실적이며, 이러한 문제를 해결하기위해 경로 노드 집합 전체를 유지하는 대신 집합 해싱(set hashing)을 이용한 시그너처(signature)만을 유지한다. 이는 색인 작성 시간에 계산하여 유지하며, 질의 시간에 해당 정보를 이용하여 유사도를 예측한다.

최소 독립 순열(min-wise independent permutations)의 몬테 카를로(Monte Carlo) 기법[21]은 집합 간의 유사도 예측을 위해 보편적으로 사용되며, 집합들의 시그너처 생성을 위해 일련의 해쉬함수(hash function)들을 이용한다. 이 기법은 PNS 집합의 원소의 개수와 상관없이 고정 길이의 시그너처만을 유지하며, 적은 비용으로 집합간의 유사도를 예측할 수 있다. PNS 집합에 대한 집합 해쉬 요약(set hash signature)은 다음과 같이 정의된다[22].

전체집합 U={1,...,n}이고, p가 집합 U에 대한 순열일 때 min{p(PNS)}를 다음과 같이 정의한다.

$$\min\{p(PNS)\} = \min\{p(x) \mid x \in PNS\}$$

이 때 U에 대한 l 개의 순열 p_1, ..., p_l을 선택하면, 집합 PNS에 대한 시그너처 벡터 sig_{cns}는 다음과 같이 정의된다.

$$sig_{CNS} = (\min\{p_1(PNS)\}, \min\{p_2(PNS)\}, \mathrm{K}, \min\{p_l(PNS)\})$$
$$\min\{p(PNS)\} = \min\{p(x) \mid x \in PNS\}$$

실제 구현에서는 전체집합에 대한 중복되지 않는 임의의 순열을 구하기 어려우므로 일련의 선형 해쉬 함수 집합을 유지한 후 이를 이용해서 값을 생성한다. PNS 집합의 각 원소는 해쉬 함수들을 이용해서 각각 해쉬 값을 생성하게 되며, 각 해쉬 함수가 생성한 값들 중 가장 작은 값을 시그너처의 원소로 사용한다. 따라서 해쉬 함수의 개수만큼 시그너처의 원소가 생성된다. 본 논문에서는 임의의 해쉬 함수를 사용하기 위해 universal hashing을 사용하며 시그너처의 개수 등 실제 구현과 관련된 인자에 대해서는 7장에서 다룬다.

4.7 문서 순위화

한 문서에 대한 점수는 해당 문서 내에 존재하는 분할 단위로 계산되어지며 질의 키워드에 대해 분할 내의 색인어 통계정보와 앞서 기술한 경로 노드 집합의 시그너처를 이용한 키워드 근접도 점수를 이용해서 계산되어진다. 분할은 사용자 질의의 모든 키워드를 포함하는 완전 분할인 경우와 일부만 포함하는 부분 분할, 그리고 키워드가 존재하지 않는 분할로 나누어진다. 이 중 점수 계산은 키워드를 모두 포함하는 완전 분할과 일부를 포함하는 부분 분할에 대해 행해진다.

$$\mathrm{Score}_{\mathrm{doc}} = (\sum_{p_i \in P_0} w_i(\rho_{p_i} \times S_{p_i}) + \sum_{p_i \in P_2} w_i(\rho_{p_i} \times S_{p_i} \times decay)) / num_P$$

$$S_{p_i} = \sum_{n \in N} w_n \times tf_{p_i,t_i} \times ilf_{t_i},$$

$$tf_{p_i,t_i} = \frac{occ_{p_i,t_i}}{\max\{occ_{p_i,k} \mid k \in words(p_i)\}},\ ilf_{t_i} = \log\left(1 + \frac{|N|}{|\{n' \in N \mid t_i \in words(n')\}|}\right),$$

$$decay = d^{(H - H_0 + 1)}$$

여기에서 ρ_{p_i}는 분할 p_i가 포함하는 질의 키워드들 간의 근접도를 나타내며, 질의 키워드들의 해당 분할에 대한 시그너처 집합들의 유사도로 구해진다. Sp_i는 분할 내의 존재하는 질의 키워드들에 대한 색인어 통계정보를 이용해서 구해지는 점수다. 본 논문에서는 사용한 점수 방식은 tf-idf를 변형한 방식으로, tf_{p_i,t_i}는 분할 내에 존재하는 색인어에 대한 빈도수 정보이며 해당 색인어의 가중치를 위해 ilf_{t_i} 값을 사용한다. 여기서 N은 문서 내의 모든 노드를 나타내며, n′은 해당 색인어가 존재하는 단말 노드를 나타낸다. 모든 키워드를 포함하는 완전 분할(P_f)은 키워드 근접도에 비율에 따라 점수가 결정되며, 일부 키워드만 포함하는 부분 분할($P_\exists$)은 자체적으로 해를 생성하지 못하고 완전 분할과 결합되어 해가 될 가능성을 지닌 분할로, 해를 생성한다 해도 분할 기준 레벨보다 높은 레벨에서나 완전 분할과 결합되어 해를 생성하므로 점수를 감소시켜준다. d는 감소를 위해 1보다 작은 상수이며 H는 트리의 높이를 H_b는 기준 분할 레벨을 의미한다. 각 분할에 대한 점수는 가중치에 의해 합해지는데, 가중치는 분할별 빈도수 등을 이용해서 지정될 수 있으며, 기본적으로 1의 값을 갖는다.

5. 문서 단위 질의 처리

이번 장에서는 앞서 얻어진 순위화된 문서들을 이용해서 실제 질의 처리를 하는 과정과 문서 단위의 색인 구조에 대해 설명한다. 본 시스템의 인덱스는 그림 6에서와 같이 일반적인 역색인 구조를 문서 단위로 나누어 문서 단위 정보(“DocInfo”)를 저장하는데, 이에는 문서 아이디 및 해당 문서의 경로노드집합 시그너처 정보 등이 포함된다. 또한 각 키워드에 대한 포스팅 역시 문서별로 나뉘어 저장된다. 이러한 문서 단위 색인을 통해 문서 순위화 및 문서 필터링 등 문서 레벨에서의 작업이 가능해진다.

다음은 순위화된 XML 문서를 이용한 사용자 질의 처리 과정을 보여준다.

(1) 질의 키워드 리스트의 모든 키워드를 포함하는 문서 리스트를 구하고, 해당 문서들에 대한 문서 단위 정보 리스트를 읽어온다(그림 6 참조).

(2) 문서 내의 각 분할에 대해 키워드 근접도를 계산하고 5.7절의 공식에 의해 각 문서에 대한 점수를 계산하고 순위화를 수행한다.

(3) 정렬된 문서 리스트를 이용해서 순위가 높은 문서부터 엘리먼트 단위 질의 처리를 수행하면서 결과 힙(heap)에 순위별로 저장한다. 원하는 결과물의 수가 만족되면 처리를 멈추고 해당 결과물을 반환한다.

단계1에서는 주어진 질의 키워드에 대해 문서 단위 인덱스를 통해 해당 키워드들을 포함하는 문서들에 대한 정보 리스트를 디스크에서 읽어온다. 해당 정보에는 문서에 대한 색인어 통계정보 및 시그너처 등이 존재하며 엘리먼트 단위 포스팅과는 별로로 저장된다.

단계2에서는 키워드 근접도 계산을 위해서 질의 키워드마다 저장된 시그너처를 읽어 들인 후 집합 유사도를 계산한다. 효율적인 계산을 위해 시그너처는 색인 생성 시 미리 오름차순으로 정렬시켜 저장하고, 유사도 계산은 질의 시간에 각 키워드마다 가지는 정렬된 시그너처 원소 리스트를 결합(sort-merge)하여 합집합 리스트를 생성하며 그 과정에서 발생되는 중복 원소의 개수를 계산해서 유사도를 계산한다. 한 문서에서 k개의 키워드가 주어지고 p개의 분할이 존재할 때 시그너처의 길이가 s인 경우 소요되는 시간은 O(pks)가 된다.

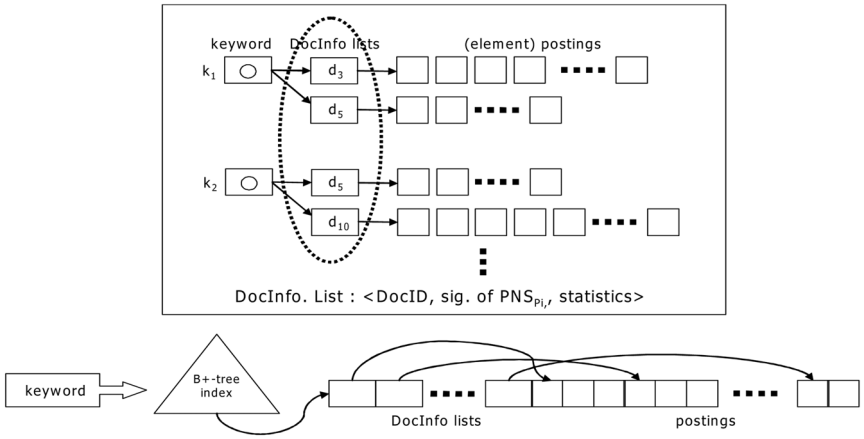


그림 6 문서 단위 인덱스 구조와 물리적 매핑

단계3에서는 단계2에서 얻어진 문서 순위에 의해 높은 순위의 문서부터 엘리먼트 단위 포스팅을 디스크에서 읽어와 엘리먼트 단위 질의 처리 과정을 수행한다.

6. 구현 및 실험

6.1 구현 및 실험 데이타

XML 문서의 각 노드는 아이디를 가지며, 본 구현에서는 기본적으로 듀이 아이디를 사용한다. 이는 듀이 아이디가 조상 노드들에 대한 모든 경로를 내포하고 있어, DIL 엘리먼트 단위 질의 처리 알고리즘에서 효율적으로 최소 공통 부모를 구할 수 있기 때문이다. 키워드 근접도를 구하기 위한 경로 노드 집합은 XML 문서에 대한 SAX 파싱을 할 때 단말 노드에 키워드가 출현할 때마다 키워드 별로 조상 노드 집합이 유지되고, 하나의 분할에 대해 키워드 별로 조상 노드 리스트의 합집합을 계산하여 경로 노드 집합으로 구성한다. 경로 노드 집합은 문서 트리의 각 분할을 전위 순회(preorder traversal)하면서 얻어지는 전위 순위 숫자를 기록한다. 전위 순위 숫자를 이용하는 것은 듀이 아이디보다 공간 효율이 좋기 때문이다. 분할을 위한 기준 레벨은 1로 지정하였고 루트의 각 자식들이 각 분할의 루트 노드가 된다. 구해진 경로 노드 집합을 이용해 각 집합에 대한 시그너처를 계산하는데, 시그너처의 크기는 50으로 하였고, 해쉬 공간은 2의 16승으로 지정했고 각 문서에 관한 정보와 함께 문서 단위 색인을 통해 디스크에 저장되어진다.

본 논문의 시스템은 C++로 구현되었으며 Berkeley-DB[23]을 사용하여 역색인을 구현하였다. 실험 환경은 512MB의 메모리를 가진 PentiumIII 993 MHz의 PC에서 하였다. 실험에 사용한 실험 데이타는 XML 문서 검색을 위한 INEX 2003[24]의 실험 데이타를 이용하였다. 해당 데이타는 현재 XML 문서 검색의 주요 테스트 컬렉션으로 1995~2001년도 까지의 컴퓨터 관련 저널이 XML 문서들로 구성되어 있고 494M의 규모가 큰 데이타이다. INEX에 정의된 질의 집합은 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. 본 시스템에서는 구조 질의를 지원하지 않으므로 CO 표준 질의들을 사용하여 실험을 수행하였다.

6.2 실험 결과

첫 번째 실험은 본 논문에서 제시하는 문서 단위 질의 처리 기법에 대한 유효성 검사를 위해 기존의 엘리먼트 단위 질의 처리의 결과물과 비교 실험을 수행하였다. 그림 7은 5가지 질의별로 엘리먼트 단위 질의 처리 결과물에 대해, 처리 문서수를 달리하면서 얻어지는 문서 단위 질의 처리의 결과물의 적중률(hit-rate, HR)[25]을 비교한 것이다. 적중률은 추천시스템(recommender system)[26] 등에서 추천결과물에 대한 품질을 측정하는데 사용되는 척도로 본 논문에서는 이를 응용해서 사용하였다. Top-k 질의 결과물에 대한 적중률을 얻기 위해 순위화된 문서 리스트에서 상위 문서 d개를 선택해서 질의 처리를 했을 때 얻어지는 결과물 중 상위 d개의 항목들을 선택하고, 이들 항목들이 기존의 엘리먼트 단위 질의 처리 결과물 중 상위 d개의 항목들과 얼마나 일치하는 지 여부를 구하는데, 다음과 같이 정의된다.

$$HR(d) = \text{적중된 항목의 개수} / d$$

사용된 5가지의 질의어는 INEX의 내용 질의 중 비교적 키워드의 개수가 많고 생성되는 결과물의 개수가 많은 질의들을 택하였고 각 질의어와 그 결과에 관한 정보는 표 1과 같다. 그림 7의 (a)는 처리문서수를 증가시켜가면서 얻어지는 적중률을 나타내며, 각 문서에 대한 점수를 계산할 때 해당 문서가 포함하는 분할 중 상위 3개의 분할을 택하였다. 상위 분할을 택하는 것이 모든 분할을 택하는 경우보다 많지는 않지만 보다 좋은 성능

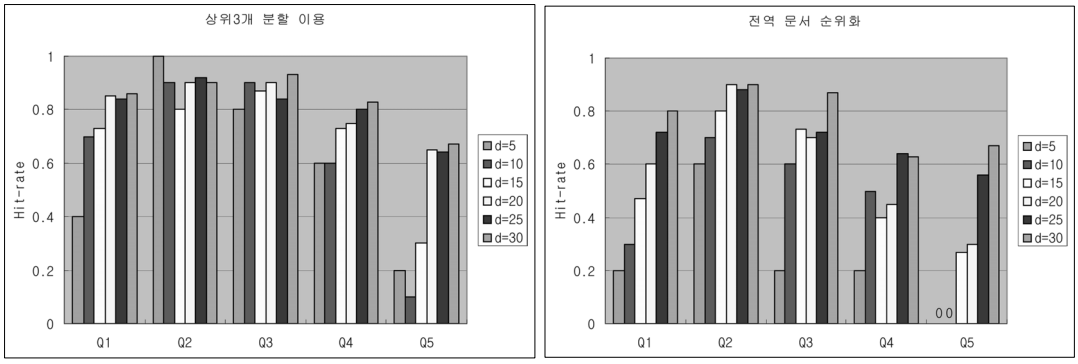


그림 7 5가지 질의(Q1-5)에 대한 적중률 비교

을 보였다. 그래프를 보면 적은 문서에 대한 처리만으로도 높은 적중률을 보여주고 있음을 알 수 있다. 질의어 Q5인 경우 가장 나쁜 성능을 보여주고 있다. Q5는 많은 문서와 관련되어 있으며, 여러 문서에 고르게 해가 존재하기 때문이다. 실제로 “data”, “structure”, “index”와 같이 빈도가 높고 고르게 분포하는 색인어 질의인 경우 적중률은 낮아지게 된다. 두 번째 그래프 (b)는 각 문서를 단일 객체로 보고 각 문서에 대한 질의 색인어 통계정보를 이용해서 문서 순위화를 수행하고 적중률을 측정한 결과다. (a)와 비교해 보면 적은 문서에서의 적중률이 매우 낮음을 알 수 있다.

표 1 질의 관련 정보

	Q1	Q2	Q3	Q4	Q5
관련 문서수	119	117	93	117	120
결과수	796	637	508	477	1653
키워드수	7	7	5	4	3

두 번째 실험 결과인 그림 8은 앞선 실험에서 얻어진 유효 적중률에 대한 엘리먼트 단위 질의 처리와 문서 단위 질의 처리 시간과의 비교를 보여준다. 적중률이 각각 0.5, 0.8 이상이 될 때 최소한 처리해야하는 문서에 대한 처리 시간과 전체 문서에 대한 처리 시간을 나타내고 있다. 각 경우에 적용된 문서의 개수는 표 2와 같다

표 2 유효 적중률에 대한 최소 문서 개수

HR	Q1	Q2	Q3	Q4	Q5
0.5	10	5	6	4	20
0.8	30	9	10	25	40

그림을 보면 상위 문서만을 처리함으로써 질의 처리 시간을 줄일 수 있으며, 질의어 내의 키워드가 많은 질의인 경우 그에 따라 질의 처리 시간이 증가하게 되며 그런 경우 문서 단위 질의 처리 기법이 보다 많은 이득을 얻을 수 있음을 알 수 있다.

7. 결론 및 향후 연구

XML 문서에 대한 키워드 검색은 XML 문서 내부의 구조적인 특징으로 인해 문서 단위가 아닌 엘리먼트 단위의 질의 처리를 수행하여 구체적인 결과를 생성할 수 있지만, 이로 인해 질의 처리 부담이 크게 증가하였다. 본 논문에서는 이와 같은 엘리먼트 단위 질의 처리의 비용을 줄이기 위한 노력으로 기존의 엘리먼트 단위 질의 처리를 기반으로 하는 XML 문서에 대한 문서 중심의 뷰를 제공하고자 했다. 이를 위해 부분 결과물의 순위화에 영향을 미치는 요인을 분석하고 그 중 문서 내 키워드 간의 근접도를 예측하기 위해 키워드 출현 노드들의 경로 정보를 이용하였고, 이를 바탕으로 문서 단위 순위화 기법을 제안하였다. 또한 순위화 된 문서 리스트를 이용 순위가 높은 문서를 우선적으로 처리함으로써 질의 처리 속도를 향상시켰으며 실험을 통해 해당 기법의 유효성 및 성능을 보였다.

XML 문서에 대한 문서 단위 순위화 기법은 지역 정보만을 이용해 순위화 된 부분 결과물에 전역 정보가 추가됨으로써 검색의 질을 높일 수 있을 것으로 기대되며, 각 정보가 미치는 영향에 관해서는 추가적인 연구가 필요하다. 또한 문서 단위 순위화는 질의 키워드가 출현하는 노드의 분포에 많은 영향을 받으므로 고르게 분포되는 사용자 질의 키워드에 대한 처리 방법 역시 연구되어야 한다. 아울러 문서 단위 질의 처리 결과물에 대해 사용자의 판단에 의한 의미론적인 평가도 이루어져야 할 것이다.

참 고 문 헌

[1] http://www.w3.org/XML/
[2] L. Guo, et al. "XRANK: Ranked Keyword Search over XML Documents," SIGMOD, 2003.
[3] L. Mignet, D. Barbosa, P. Veltri. "The XML Web: a First Study," WWW 2003.
[4] D Florescu, et al., "Integrating Keyword Search into XML Query Processing," WWW, 1999.

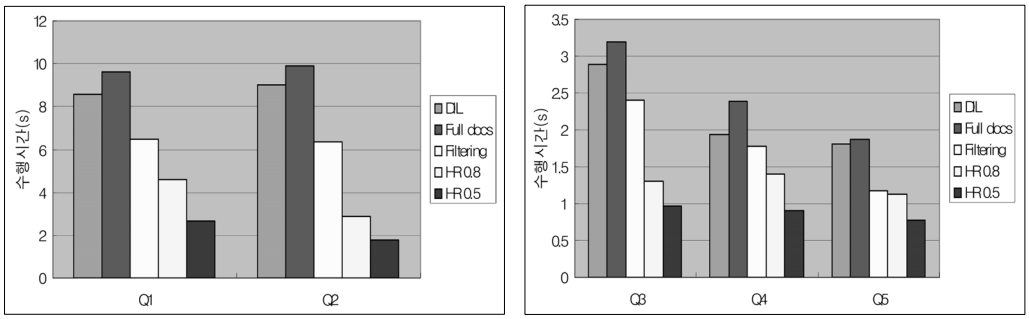


그림 8 5가지 질의(Q1-5)에 대한 수행시간 비교

[5] V. Hritidis, Y. Papakonstantinou, A. Balmin. "Keyword Proximity Search on XML Graph," ICDE, 2003.
[6] S. Brin, L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," WWW7, 1998.
[7] T. Igor, D. V. Stratis, B. Kevin, S. Jayavel, S. Eugene and Z. Chun: "Storing and querying ordered XML using a relational database system," ACM SIGMOD, 2002.
[8] S. Cohen, J. Mamou, Y. Kanza, Y. Sagiv. "XSEarch: A Semantic Search Engine for XML," VLDB, 2003.
[9] Shurug Al-Khalifa, Cong Yu, and H. V. Jagadish. "Querying structured text in an XML database," SIGMOD, 2003.
[10] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, A. Soffer. "Searching XML Documents via XML Fragments," SIGIR, 2003.
[11] R. A. Baeza-Yates and B. A. Ribeiro-Neto, Modern Information Retrieval, ACM Press / Addison-Wesley, 1999.
[12] Gerard Salton, James Allan, Chris Buckley, "Approaches to Passage Retrieval in Full Text Information Systems," SIGIR, 1993.
[13] Ross Wilkinson, "Effective Retrieval of Structured Documents," SIGIR, 1994.
[14] Donna Harman, Gerald Candela, "Retrieving Records from a Gigabyte of Text on a Mini-Computer Using Statistical Ranking," JASIS 41(8), 1990.
[15] Michael Persin, Justin Zobel, Ron Sacks-Davis, "Filtered Document Retrieval with Frequency-Sorted Indexes," JASIS 47(10), 1996.
[16] Ahn Ngoc Vo, Owen de Kretser, Alistair Moffat, "Vector-Space Ranking with Effective Early Termination," SIGIR, 2001.
[17] Igor Tatarinov, Stratis Viglas, Kevin S. Beyer, Jayavel Shanmugasundaram, Eugene J. Shekita, Chun Zhang, "Storing and querying ordered XML using a relational database system," SIGMOD, 2002.
[18] Gerard Salton, "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer," Addison-Wesley, 1989.
[19] Norbert Fuhr, Kai Großjohann, "XIRQL: A Query Language for Information Retrieval in XML Documents," SIGIR, 2001.
[20] Edith Cohen, "Size-Estimation Framework with Applications to Transitive Closure and Reachability," J. Comput. Syst. Sci. 55(3) 1997.
[21] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, Michael Mitzenmacher," Min-Wise Independent Permutations," STOC 1998.
[22] Zhiyuan Chen, Flip Korn, Nick Koudas, S. Muthukrishnan "Selectivity Estimation for Boolean Queries," PODS 2000.
[23] http://www.sleepycat.com
[24] Initiative for the evaluation of XML retrieval
[25] Mukund Deshpande and George Karypis, "Item-based top-N recommendation algorithms," ACM Trans. Inf. Syst. 22(1), 2004.
[26] Resnick, P. and Varian, H.R., "Recommender systems," CACM 40(3), 1997.

이 형 동
1997년 홍익대 컴퓨터공학과(학사). 1999년 서울대 컴퓨터공학과(석사). 2006년 서울대 컴퓨터공학과(박사). 2006년~현재 삼성전자 연구원. 관심분야는 데이타베이스, 정보검색

김 형 주
정보과학회논문지 : 데이타베이스
제 33 권 제 1 호 참조