효율적인 XML 검색을 위한 재질의 시스템 설계

정유나[0] 황인준
아주대학교 정보통신 전문대학교
(serazad, ehwang)@madang.ajou.ac.kr

Design of Query Reformulation System for Efficient Retrieval of XML Documents

Yuna Jung[0] Eenjun Hwang
Graduate School of Information & Communication, Ajou University

요 약

XML이 확산되면서 이를 저장하고 검색하는 XML DB와 검색 엔진들이 만들어졌다. 그러나, 이들 대부분의 시스템에서 초기 질의만으로 문서를 검색하고, 그 대상도 질의 조건에 완전히 정합되는 문서로만 제한하고 있다. 그러나, 사용자가 데이터에 대한 정확한 정보가 없는 경우에는 자신의 요구를 제대로 표현하기가 힘들고 또, 한번의 질의로 사용자 요구에 정확하게 부합되는 문서를 검색하기도 매우 어렵다. 따라서, 본 논문에서는 질의 조건에 부분적으로 정합되는 문서도 검색하고, 사용자 피드백을 받아서 초기 질의를 사용자 요구에 좀 더 근접한 문서들을 검색할 수 있도록 수정하여 재질의를 하는 시스템을 설계하였다.

1. 서론

XML(eXtensible Markup Language)[1]은 W3C에서 웹 문서의 표준으로 제안한 마크업 언어이다. 1998년 2월에 W3C에서 정식으로 채택된 이후, 폭발적으로 확산되어 많은 응용 분야에서 성공적으로 활용되고 있다. 이처럼 XML이 널리 활용됨에 따라, 이를 효율적으로 저장하고 검색하는 것에도 연구의 초점이 모아졌다. 그 결과 여러 방법이 제안되었고 이를 기반으로 여러 개의 XML DBMS와 검색 엔진들이 만들어졌다.

그러나, 대부분의 시스템에서 초기 질의로만 문서를 검색하고, 질의를 확장하거나 수정하는 것은 고려하지 않고 있다. 물론, 사용자가 XML 문서 집합에 있는 모든 데이터에 대해서 정확히 알고 있다면, 한번의 질의로 사용자가 원하는 정보를 얻을 수도 있을 것이다. 하지만, XML 문서의 양이 방대하고 다른 사용자가 작성한 문서들도 포함되어 있다면 단 한번의 질의로 사용자가 원하는 정보를 정확하게 검색하기는 매우 어려운 일일 것이다. XML이 웹 상에서 주고 받는 문서의 표준인 것을 생각해 보면, XML 문서 집합이 여러 사용자가 작성한 문서가 뒤섞여 있고 또 그 양도 방대할 것임을 쉽게 생각해 볼 수 있다. 이런 경우, 초기 질의를 수정하여 사용자가 원하는 것에 더욱 가까운 정보를 검색하는 질의로 재작성 하는 것은 검색의 효율성과 정확성을 위해서 반드시 필요한 과정이다.

게다가, 현재 XML DB[2]와 검색 엔진의 대부분은 모든 질의 조건에 정확하게 정합 되는 문서만을 검색하고 있다. 이는 경우에 따라서, 검색되는 문서집합을 너무 엄격하게 제한하는 면이 있다. 질의 조건에 부분적으로 정합되는 문서도 검색될 수 있게 하면 좀 더 다양하게 검색된 문서 집합을 제공받을 수 있을 것이다.

사실, 유사도를 측정하거나 질의를 수정하여 재질의를 하는 것은 정보검색 분야에서 오랫동안 연구되어 왔던 주제로서, 이미 여러 기법들이 체계적으로 정리되어 있는 상태이다[3]. 하지만, I-R에서 다뤄왔던 질의와 재질의 방법들은 모두 구조를 가지지 않는 편평한 문서에 대해서 이루어지는 것들이었다. 그러나, XML은 구조를 가지는 문서이므로, 기존의 방식과는 다른 방법이 적용되어야 할 것이다.

본 논문에서는, XML을 효율적으로 검색하기 위해서 사용자의 피드백을 받아 재질의를 하는 시스템을 제안해 보았다.

2. 연구 동기

XML의 응용분야는 실로 다양하다. 멀티미디어 분야 역시 XML의 응용 분야 중 하나로, 멀티미디어 데이터에 대한 정보를 XML로 표현함으로써 기존에는 각각 따로 이루어졌던 내용기반 검색과 주석기반 검색을 한 단계에서 같이 할 수 있게 되었다.[4]

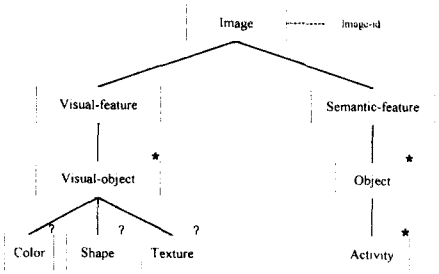

그림 1. Image data를 위한 XML 구조

XML 문서 안에 내용정보와 주석정보를 함께 표현하고 검색하기 때문에 두 가지 검색이 동시에 이루어진다. 그리고, 구조 없이 텍스트로만 주석을 달아 놓았을 때, 사용자에 따라 같은 장면에 대한 주석에 포함되는 정보의 종류가 달라서 검색을 어렵게 했던 것과 다르게, XML 구조로 주석에 포함되어야 할 정보의 종류를 정하면 주석형태의 일관성을 어느 정도 꾀할 수 있었다.

그러나, 다른 사용자가 저장시켜 놓은 그림과 주석의

내용을 알 수 없으므로, 그에 대한 사용자 요구는 막연할 수 밖에 없다. 따라서, 모호한 질의로 검색된 멀티 미디어 정보가 사용자가 원하는 데이터와 정확히 일치하기도 어려워진다. 게다가, 사용자들이 자신이 원하는 멀티 미디어 데이터를 검색하기 위한 질의를 명확하게 구사한다는 보장도 할 수 없다. 이런 이유 때문에, 초기 질의만으로 자신이 원하는 데이터에 접근하기는 매우 힘들다. 그러므로, 사용자의 피드백으로부터 정보를 추출하여 재질의를 함으로써 사용자 요구에 좀 더 근접한 멀티미디어 데이터를 검색하는 것이 반드시 필요하다.

3. 효율적인 XML 검색 시스템의 제안

기존 XML 데이터베이스나 XML 검색엔진의 검색 효율을 높이기 위해서, 사용자의 피드백을 이용한 재질의 시스템을 설계해 보았다.

3.1 시스템의 구조

본 논문에서 제안하는 시스템은 기존의 XML DB나 검색엔진은 그대로 사용하고, 사용자와의 사이에 재질의를 하는 모듈만 추가하는 구조로 되어있어, 기존 시스템을 사용하고 있는 경우에도 별 무리 없이 적용할 수 있다는 장점이 있다.

시스템의 전체 구조는 아래에서 보는 것과 같이 XML DB, Query Convertor, Ranker와 Query Reformulator로 구성되어 있다. XML DB는 기존 시스템이고, Query Convertor와 Ranker, Query Reformulator가 새로 추가된다. 각 모듈의 작동원리는 다음 장에서 좀 더 상세히 다루기로 한다.

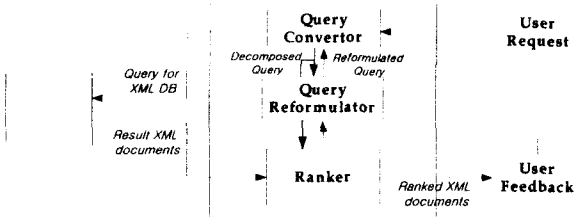

그림 2. 시스템의 전체 구조

3.2 시스템 흐름도

사용자 질의가 들어오면, Query Convertor에서 이를 해당 XML DB에 맞는 질의문으로 변환시킨다. 그 다음, Ranker가 결과를 건네 받아서 검색된 문서들을 질의조건을 만족시키는 정도에 따라 순위를 매긴다. 사용자는 그 결과를 보고, 검색을 끝낼 것인지 재질의를 할 것인지 결정하게 되는데, 사용자가 원하는 문서가 검색되면 검색을 끝내고, 그렇지 않으면 검색된 문서집합을 연관 문서와 비연관 문서로 나누어서 피드백을 한다. Query Reformulator는 사용자가 연관/비연관으로 나눈 문서들로부터 새로운 정보를 추려내서 초기질의를 수정한다. 이렇게 다시 작성된 질의문은 다시 Query Convertor를 거쳐 XML DB에 던지는 질의문으로 변환된다. 위의 피드백 과정은 사용자가 검색 결과에서 원하는 문서를 찾을 때까지 계속된다.

4. XML 검색 시스템

여기서는 3장에서 제안한 검색 시스템의 주요한 동작 기법들을 설명한다.

4.1 사용자 요구를 질의문으로 변환하는 기법

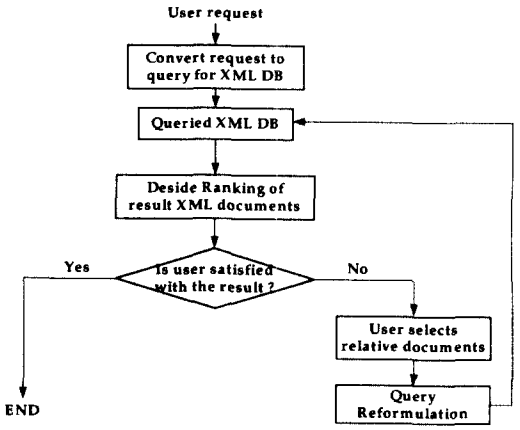

그림 3. 시스템의 흐름도

Query Convertor 모듈은 사용자 요구에 맞는 XML 문서를 찾아내기 위해서 XML repository에 던지는 질의를 생성한다. XML 문서에 대한 사용자의 검색 요구는 XML 문서의 어떤 태그 값이 특정한 값을 갖는 조건들의 집합으로 생각해 볼 수 있다. 예를 들어, "노란 옷을 입은 소녀가 파란 우산을 쓰고 빗속을 걷는 그림을 찾아라" 라는 요구는 아래와 같이 몇 가지 조건으로 표현될 수 있다. 이는 Object-name이라는 반복되는 엘리먼트에 4개의 값으로 질의가 이루어지는 것을 4개의 조건에 나눈 것이다.

조건1. Image//Visual-object = "소녀"
& Image/Semantic-feature/Object = "소녀"
& Image//Activity = "걷다"
조건2. Image//Visual-object = "옷" & Image//Color = "노란색"
조건3. Image//Visual-object ="우산"& Image//Color = "파란색"
조건4. Image//Visual-object = "비"

위의 조건들은 대부분의 XML DB에 대한 질의 형태인 Xquery [5] 질의문이나 Xpath[6] 질의문 또는, XML DB에서 사용되는 고유의 질의 언어로 쉽게 변환할 수 있다.

위의 조건을 Xpath 질의문으로 변환하면 아래와 같다.

/Image[(Visual-feature/Visual-object="소녀" and semantic-feature/Object="소녀" and Semantic-feature/Object/Activity="걷다") or Visual-feature/Visual-object="옷" and Visual-feature/Visual-object/Color="노란색") or (Visual-feature/Visual-object="우산" and Visual-feature/Visual-object/Color="파란색") or (Visual-feature/Visual-object ="비")]

4.2 문서 순위화 기법

Ranker 모듈은 XML DB에 질의를 던져서 결과로 받은 문서들을 순위화 한다. 이들의 순위화는 질의와의 연관성에 의해서 이루어진다. 먼저, 사용자 질의와 XML 문서를 index term의 형태로 생각해 보자. Index term은 DTD상에서 Abstract *element*를 제외한 다른 모든 *element*와 *attribute*들로 한다. 그러면, 하나의 DTD를 공유하는 XML 문서들의 index term은 모두 같은 형태가 된다. 이 때, DTD 상에서 (element)* 형태로 표현되는 엘리먼트에 여러 값으로 조건을 제시하는 경우는 초기 질의를 한 엘리먼트에 대해 단일 값만을 질의하는 몇 개의 조건으로 분해한 것에 따라 질의를 몇 개의 소질의로 분해해서 표현한다.

Index term = $\{k_1, k_2, ..., k_t\}$

k_{i,q_j} : 질의 q_j의 i번째 term 이다

q : XML 문서 집합에 대한 원래 질의

$\vec{q}_1$, $\vec{q}_2$, ... $\vec{q}_m$:q를 분해해서 만든 m개의 단일 질의

$\vec{q}_1 = \{k_{1,q_1}, k_{2,q_1}, .., k_{t,q_1}\}$, $\vec{q}_2 = \{k_{1,q_2}, k_{2,q_2}, .., k_{t,q_2}\}$, ... ,

$\vec{q}_m = \{k_{1,q_m}, k_{2,q_m}, ..., k_{t,q_m}\}$

$Q = q_1 \wedge q_2 \wedge \ldots \wedge q_m$ 이어야 한다.

앞에서 예로 보인 질의를 위와 같이 표현하면 다음과 같다.

Index term = {Image-id, Visual-object, Color, Shape, Texture, Object, Activity}

$\vec{q}_1$ = { Null, 소녀, Null, Null, Null, 소녀, 걷다}

$\vec{q}_2$ = { Null, 옷, 노란색, Null, Null, Null, Null }

$\vec{q}_3$ = { Null, 우산, 파란색, Null, Null, Null, Null }

$\vec{q}_4$ = { Null, 비, Null, Null, Null, Null, Null }

문서도 이와 같은 방법으로 index term 형태로 표현할 수 있다. $\vec{d}_j$에 대한 소질의 $\vec{q}_i$의 가중치는 아래와 같다.

$$w_{\vec{d}_j,\vec{q}_i} = (\vec{d}_j, \vec{q}_i) \begin{cases} 1 : \forall i,\ \vec{d}_j \text{의 } i\text{번째 term} = \vec{q}_i \text{의 } i\text{번째 term} \\ 0 : \forall i,\ \vec{d}_j \text{의 } i\text{번째 term} \neq \vec{q}_i \text{의 } i\text{번째 term} \end{cases}$$

이 때, 질의에 대한 문서의 유사도는 다음과 같다.

$$Sim_j = \sum_{i=1}^{n}\sum_{j=1}^{m} w_{d_i,q_j}$$

이 수식은 문서가 만족시키는 질의 조건의 개수에 따라서 문서의 순위를 결정한다. 즉, 질의의 조건을 모두 만족시키는 문서가 최상위가 되고, 만족시키는 질의 조건의 개수가 작을수록 하위 문서가 되는 것이다. 유사도가 같은 문서들은 임의로 순위를 정한다. 예를 들어, 위의 질의로 아래와 같은 5개의 문서가 검색되었다면 그들의 순위는 다음과 같을 것이다.

1.빨간 옷을 입은 소녀가 파란 우산을 쓰고 사탕을 먹으면서 빗 속을 걷는 그림 (유사도 : 3)

2.노란 비옷을 입고 노란 장화를 신은 소녀가 파란 우산을 쓰고 빗 속을 걷는 그림 (유사도 : 3)

3.노란 비옷을 입고 노란 장화를 신은 소녀가 하늘색 우산을 쓰고 빗 속을 걷는 그림 (유사도 : 2)

4.노란 원피스를 입은 소녀가 하늘색 우산을 쓰고 사탕을 먹으며 빗 속을 걷는 그림 (유사도 : 2)

5.노란 비옷을 입은 소년이 초록색 우산을 쓰고 사탕을 먹으며 빗 속을 뛰어가는 그림 (유사도 : 1)

4.3 재질의 기법

Query Reformulator 모듈은 질의 결과에 대한 사용자의 피드백[7]을 받아서 이를 반영한 확장된 질의를 생성한다. 이 때, 사용자에게 피드백을 받아서 검색된 문서들을 연관/비연관 문서로 나눈 다음, 연관문서에 포함된 index term은 질의에 포함하고 비연관 문서에 포함된 것은 질의에서 제외시켜서 질의의 index term을 조정한다. 재질의를 생성하는 과정은 다음과 같은 일을 수행한다. 검색된 문서 중에서 연관성이 있는 문서들에 공통으로 포함되는 term이 있으면 그것을 반드시 포함하고, 연관성이 없는 문서들에 공통으로 포함되는 term은 포함하지 않는 문서들을 검색하도록 질의를 수정한다. 따라서, 초기질의 Q에 대해 새로 만들어지는 질의는 다음과 같다. 이 때, 하나의 값만 가질 수 있는 엘리먼트에 대해 초기질의 조건과 추가되는 조건이 서로 충돌할 때는 새로 추가되는 조건이 우선되어 초기질의의 조건을 대신한다.

$$Q_{new} = Q' \wedge (R_1 \wedge R_2 \wedge \ldots \wedge R_\alpha) \wedge (\neg N_1 \wedge \neg N_2 \wedge \ldots \wedge \neg N_\beta)$$

Q' :Q에서 새로 추가되는 조건과 충돌하는 것을 제거한 질의

$R_1, R_2, ..., R_\alpha$: 검색된 문서 중 연관성이 있는 문서들에 포함되는 정보들 중에서 사용자가 질의에 추가하기를 원하는 조건

$N_1, N_2, ..., N_\beta$:검색된 문서 중 연관성이 없는 문서들에 포함되는 정보들 중에서 사용자가 문서에 포함되지 않기를 원하는 조건

위에서 처럼 5개의 문서가 검색되었을 때, 사용자가 관련있는 문서로 2번과 3번 문서를 선택하였다면, 원래 질의에 '노란 장화'와 '노란 비옷'이 추가되고, '소녀가 사탕을 먹는다' 조건을 만족시키지 않는 문서를 찾도록 수정된다. 즉, 재질의문은 "노란 옷이나 노란 비옷을 입고, 노란 장화를 신은 소녀가 파란 우산을 쓰고, 사탕을 먹지 않으며 빗 속을 걷는 그림을 찾아라" 가 된다.

Q_{new} 를 변환시킨 Xpath 질의문에는 조건들의 OR로 표현되는데, 이는 다시 AND의 OR 형태로 표현이 가능하다.

$$Q_{new} = (Q_1 \wedge \ldots \wedge Q_m \wedge R_1 \wedge R_2 \wedge \ldots \wedge R_\alpha \wedge N_1 \wedge \ldots \wedge N_\beta) \vee$$

$$(Q_1 \wedge \ldots \wedge Q_m \wedge R_1 \wedge R_2 \wedge \ldots \wedge R_\alpha \wedge N_1 \wedge \ldots N_{\beta-1}) \vee \ldots \vee Q \vee R_1 \vee R_2 \vee \ldots \ldots \vee N_\beta$$

시스템의 부하를 줄이기 위해서, 새로운 질의를 조건의 AND로 표현되는 것을 하나의 부 질의로 하여 질의를 나누어서 수행한다. 가장 많은 조건의 AND로 연결되는 부질의부터 차례대로 질의를 해서 검색된 문서의 수가 특정 수준을 만족시키는 수준까지만 질의한다. 이 때, 검색은 유사도의 순서대로 이루어지게 된다.

5. 결론 및 향후 계획

본 논문에서는 기존의 XML DB와 XML 검색엔진과 다르게, 정보 검색의 재질의 기법을 도입하여 검색의 효율을 높일 수 있는 XML 검색 시스템을 설계해 보았다. 앞으로는, 이 설계를 기반으로 실제 시스템을 구현하여 검색의 재현율과 정확률을 측정하여 위의 설계가 XML 검색의 효율을 높여준다는 것을 증명해 볼 계획이다.

참고문헌

[1] T.Bray et al.,"Extensible Markup Language (XML) 1.0," http://www.w3.org/TR/1998/REC-xml-19980210, 1998.

[2] Vitorio Viarengo,"eXcelon XML Data Server Technical Overview," Object Exchange 98, Object Design User Conference, 1998.

[3] Ricardo Baeza-Yates, Berthier Ribeiro-Neto,"Modern Information Retreival", Addison- Wesley, 1999.

[4] 박선영, 용환승, "XML을 이용한 내용기반 이미지 데이터베이스의 설계 및 검색 시스템 구현," 정보과학회 논문지 제 27권, 2000.

[5] http://www.w3.org/TR/2001/WD-xquery-20010607/, Xquery 1.0: An XML Query Language, W3C Working Draft 07 June 2001.

[6] http://www.w3.org/TR/1999/REC-xpath-19991116/, XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999.

[7] G.Salton, et al., "Advanced Feedback methods in Information Retrieval," Journal of American society for information science, 1985.