제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

불리언 질의 최적화에 관한 연구

주원균, 이민호, 강무영
한국과학기술정보연구원 정보시스템연구실
e-mail : joo@kisti.re.kr

A Study on Boolean Query Optimization in Information Retrieval

Won-Kyun Joo, Min-Ho Lee, Moo-Young Kang
Dept. of Information System, KISTI

요 약

본 논문에서는 불리언 모델을 지원하는 정보검색 시스템에서 사용자로부터 입력 받은 불리언 질의를 효율적으로 연산하기 위한 3 가지 방법을 제안한다. 첫째, 불리언 대수를 사용하여 형태적으로 불필요한 노드를 제거한다. 둘째 색인어 출현 빈도 정보를 사용함으로써, 빈도 0 을 가지는 노드와 이를 포함하는 노드의 연산 제외 여부를 결정하고, 연산 수행 시 시간이 적게 걸리는 순으로 피연산자와 연산자의 순서를 재 배열 한다. 셋째, 불리언 질의 내에 복합 명사가 포함되어 있을 경우 구성 명사와 연산자의 조합을 이용한 질의 확장을 실시한다. 처음 두 가지 방법은 검색 속도의 향상을, 세 번째 방법은 정확도의 향상을 목표로 한다.

1. 서론

불리언 모델은 정보검색에서 사용하는 가장 기본적인 모델로서 정보 검색의 발전과 함께 가장 많이 사용된 방법이다. 정보검색이라는 연구 분야가 자리를 잡은 이후 처리해야 할 데이터의 양이 기하급수적으로 증가하였고[2], 초기의 AND, OR, NOT 에 한정되던 연산자가 WITHIN, NEAR 등의 근접도 연산자를 지원하도록 확장되었다. 또한 사용자의 요구사항(질의)은 보다 더 길고 복잡해지고 있다.[1]

복합명사는 한국어에서 가장 빈번하게 나타나는 색인어의 한 형태로서, 영어권 중심의 정보검색모델로는 다루기가 어려운 언어 현상의 하나이다. 복합명사는 2 개 이상의 단일어들의 조합으로 이루어져 있고, 그 형태 또한 여러 가지로 나타나기 때문에 색인과 검색의 큰 문제로 여겨져 왔다.[3]

이에 불리언 질의 모델에서 질의 처리 속도를 보다 향상시킬 수 있는 방법과, 복합 명사 단어를 효과적으로 처리함으로써 정확도를 향상시킬 있는 방법이 필요하게 되었다.

2. 불리언 질의 최적화 알고리즘

본 논문에서 제안하는 불리언 질의 최적화 알고리즘은 질의 레벨에서 1)불리언 질의 처리 속도와 2)정확도의 향상을 목적으로 한다. 불리언 질의 처리 속도의 향상을 위해서 중복되거나 불필요한 연산을 제거하는데, 이 과정에서 불리언 대수(Boolean Algebra)와 색인어 출현 빈도에 의한 처리 방법을 사용 한다. 정확도(precision)의 향상은 각 질의 단어를 처리과정에서 복합 명사 처리에 관한 방법을 도입함으로써 가능한데, 이것은 한국어적인 특성을 반영하는 것으로써 영어권의 처리에는 불필요한 것이지만 한국어와 같은 도메인에서는 유용하게 사용될 수 있다.

제시한 질의 최적화 방법은 크게 세 가지로 구분할 수 있는데, 첫째, 불리언 대수를 사용하여 형태적으로 불필요한 노드를 제거한다. 대상이 되는 노드는 중복된 연산 노드, 동일 연산 내에서의 피 연산자의 중복, 질의 처리에 영향을 미치지 못하는 연산 등을 지칭한다. 둘째, 색인어 출현 빈도를 사용하여 불필요한 노드와 전체 연산을 수행하는데 있어 시간이 적게 걸리는 순으로 연산을 수행하도록 피 연산자와 연산자의 순서를 재배열한다. 셋째, 불리언 질의가 복합 명사를 구성 단어로 포함하는 경우, 복합 명사 부분을 새로운 질의 형태로 확장함으로써 검색 정확도를 향상시킨다.

질의 최적화는 표 1 에 설명되어 있는 알고리즘을 따른다.

표 1 질의 최적화 알고리즘

1) 질의 문자열에서 바이너리 질의 트리 생성
2) 바이너리 질의 트리를 N-ary 질의 트리로 변환
3) 질의 트리에 대해 최적화 과정 적용
i) 불리언 대수 처리 과정 적용
ii) 색인어 출현 빈도 정보 적용
iii) 복합명사 처리 과정 적용
4) 질의 트리를 이용하여 검색 연산 수행

다음 장에서는 알고리즘의 중요 부분에 대해 설명한다.

3. 질의 트리 생성

질의 최적화 과정의 대부분은 트리를 구성한 후, 트리의 각 노드를 합치거나 제거하고 확장하거나 재배열하는 과정에 의존하기 때문에, 어떤 종류의 트리를 사용하는지에 따라 그 성능이 달라질 수 있다. 제안한 알고리즘에서는 바이너리 트리를 확장한 N-ary 트리를 사용하는데, 이러한 트리를 사용하는 이유는 전체 알고리즘에서 언급한 3 가지 질의 최적화 방법을 적용하는 과정에서 융통성을 제공해 주기 때문이다.

트리 생성은 2 단계로 이루어져 있는데, 먼저 사용자가 입력한 질의를 파싱하여 바이너리 트리를 구성한 후, 바이너리 트리의 각 노드들에 대해 N-ary 트리 노드로 변환하는 노드 융합을 실시한다.

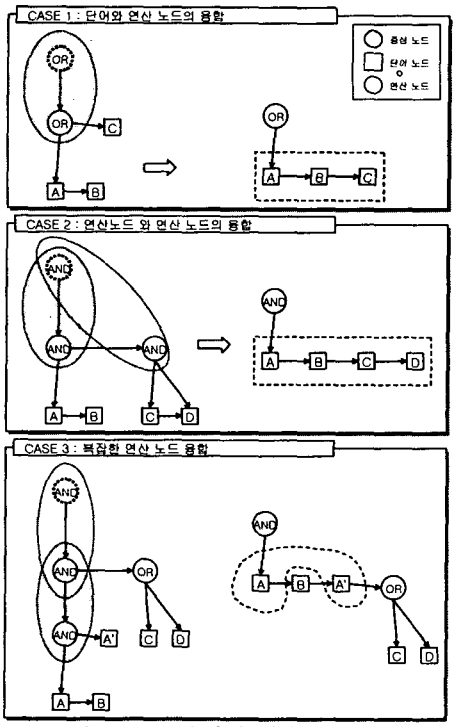


그림 1 노드 융합이 허용되는 경우

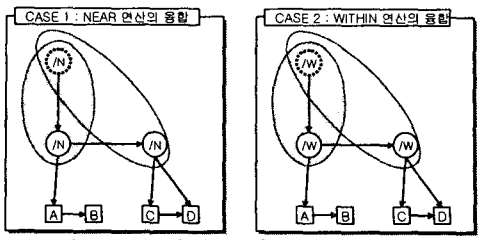


그림 2 노드 융합이 허용되지 않는 경우

노드 융합을 하기 위한 조건은 1) 중심노드를 기준으로 하위 노드가 연산노드를 포함하고, 2) 중심 노드가 하위 노드와 동일 연산을 가지며, 3) 연산자가 피연산자의 순서나 개수의 제약을 받지 않을 경우로 한정된다.

AND, OR 연산은 3 가지 조건을 모두 만족하므로 그림 1 과 같이 노드 융합이 가능한데 반해, 근접도 연산(NEAER, WITHIN)과 NOT 연산은 조건 3 에 위배되어 융합이 허용되지 않아 바이너리 형태를 그대로 유지 하게 된다. 노드 융합이 허용되지 않는 경우에 대한 것은 그림 2 에 설명되어 있다.

이러한 노드 융합 절차를 거치면 불리언 대수와 빈도에 의한 처리를 간편하게 수행 할 수 있게 된다.

4. 불리언 대수에 의한 처리

불리언 질의 처리시 불리언 대수를 사용함으로써, 형태적/의미적으로 불필요한 노드를 제거하여 질의 트리를 간소화 시킬 수 있다. 사용하는 불리언 대수는 다음과 같이 4 가지의 규칙으로 구성되는데, a) 와 d) 는 형태적으로 같은 두 노드를 간소화 시키고, b)는 기본법칙을 적용할 수 있도록 두 노드간의 순서를 교환하며, c)는 동일한 형태로 만들어 주는 역할을 한다.

a)기본 법칙
A▣A = A
A+A = A
b)교환법칙(commutative)
A▣B = B▣A
A+B = B+A
c)분배법칙(distributive)
A▣(B+C) = A▣B + A▣C
d)결합 법칙(associative)
(A▣B)▣C = A▣(B▣C)
(A+B)+C = A+(B+C)
A+(A▣B) = A
A▣(A+B) = A

불리언 대수는 바이너리 트리에 적용되는 것이 일반적인데, N-ary 트리의 노드들을 바이너리로 나눔으로서 N-ary 트리에도 적용할 수 있다. 이미 구성된 N-ary 형태의 질의 트리는 물리적으로 연산 단어로 이루어진 말단 노드와 연산자로 이루어진 중간 노드로 구성된다. 처리의 편리성을 고려하여 말단 노드와 중간 노드의 조합에 따라 가상적인 개념의 단순 노드와 복

합 노드로 분류하였는데, 단 하나의 말단 노드를 가지는 노드를 단순 노드로 말단 노드와 중간 노드의 조합으로 구성된 노드를 복합 노드라 정의하였다.

본 논문에서 사용하는 불리언 대수의 적용 방법과 범위는 그림 3 에 설명되어 있는데, 동그라미는 연산자를 네모는 가상적인 개념의 노드(단순 노드 혹은 복합노드)를 표시하고, 화살표를 기준으로 왼쪽은 불리언 대수 적용 전의 트리 구조를 오른쪽은 적용 후의 트리 구조를 표시한다. 그림 4 에서 단순 노드와 복합 노드는 각각 실선 네모와 점선 네모로 표기되어 있다.

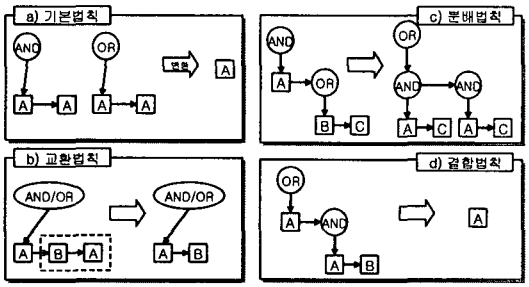
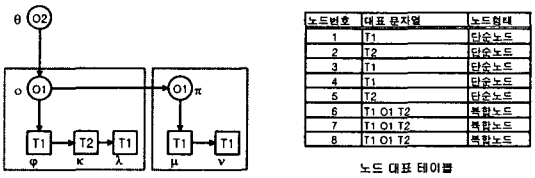


그림 3 불리언 대수에 의한 질의 트리 처리

불리언 대수 처리는 단순 노드와 복합 노드 모두에 적용될 수 있는데, 복합 노드는 단순 노드와는 달리 노드 집합으로 구성되어 있기 때문에 노드를 비교하는 과정에서 시간 복잡도를 야기한다. 이 문제를 해결하기 위해 노드 대표 테이블을 이용한 처리 방법을 제안한다.



노드번호	대표 문자열	노드형태
1	T1	단순노드
2	T2	단순노드
3	T1	단순노드
4	T1	단순노드
5	T2	단순노드
6	T1 O1 T2	복합노드
7	T1 O1 T2	복합노드
8	T1 O1 T2	복합노드

노드 대표 테이블

그림 4 노드대표테이블을 이용한 질의트리 간소화

그림 4 에서 왼쪽은 질의 트리를 오른쪽은 노드 대표 테이블을 나타내는데, 이해를 돕기 위해 질의 트리의 각 노드에 번호를 부여하여 노드 대표 테이블의 노드 번호에 매핑되도록 하였다.

노드 대표 테이블은 각 노드에 대해 자신과 하위 노드들을 대표할 수 있는 대표 코드들을 가지는 테이블인데, 이 테이블을 사용하면 복합 노드들이 동일한 트리 형태의 노드인지를 쉽게 구분할 수 있다. 예를 들면 노드 6 과 노드 7 이 동일 노드 인지를 비교할 때, 노드 대표 테이블을 이용하지 않는다면, 노드 6 과 7 의 하위 노드에 대해 일일이 트리 방문을 해야 하기 때문에 많은 시간을 소비하게 된다. 그러나 대표 테이블을 이용하면 단 한번의 참조로 쉽게 구분해 낼 수 있다. 그림 상에는 대표 코드 값이 모두 기재 되어 있는데, 실제로 이것들은 각 노드 처리시 동적으로 생성된다.

이러한 노드 대표 테이블을 이용한 불리언 대수 처리 방법은 질의 트리에 대해 아래-위(Bottom-up)순으로 적용된다.

5. 색인어 출현 빈도에 의한 처리

대부분의 검색 시스템들은 역파일 기법을 이용한 색인 방식을 채택하여 역파일에 각 단어에 대한 색인어 출현 빈도를 유지 하고 있다. 이러한 정보들은 랭킹 기반의 검색에서 정확도를 높이는 목적으로 사용하는 것이 일반적이지만, 랭킹을 지원하지 않는 불리언 질의 처리에서 질의 처리 시간을 향상시키기 위해 사용될 수 있다. 첫째, 불리언 연산 수행 시 색인어 빈도 0 을 갖는 피 연산자가 존재한다면 해당 연산을 수정함으로써 포스팅 정보를 얻기 위해 소비되는 시간을 절약할 수 있다. 둘째, 색인어 빈도가 높은 연산보다는 빈도가 낮은 연산을 먼저 수행함으로써 검색 계산에 걸리는 시간을 개선할 수 있다.

정보검색에서 사용하는 색인어 출현 빈도에는 문서 빈도(document frequency)와 단어 빈도(term frequency)가 있는데, 전자는 특정 단어가 몇 개의 문서에 출현하는가를 후자는 특정단어가 한 문서 내에 몇 번 출현하는지를 나타낸다. 비근접도 연산(AND, OR, NOT)은 단어 간의 거리를 고려하지 않기 때문에 문서 빈도에 따라 연산 시간이 결정되고, 근접도 연산(NEAR, WITHIN)은 문서 빈도와 단어 빈도 모두에 영향을 받는다.

이러한 빈도를 이용하여 각 노드를 계산하는데 걸리는 시간을 의미하는 노드 시간 가중치(Node-Time-Weight)를 정의한다. 노드 시간 가중치는 문서 빈도와 단어 빈도에 의존하는 상수 형태의 값으로서, 노드 시간 가중치가 높으면 계산에 상대적으로 많은 시간이 걸림을 의미한다.

노드 시간 가중치를 구하는 방법은 표 2 의 식에 설명한다.

표 2 노드 시간 가중치를 구하는 식

피 연산자 : DF_T
AND 연산자 : min (DF_L, … ,DF_R)
OR 연산자 : sum (DF_L, … ,DF_R)
NOT 연산자 : DF_L
NEAR 연산자 : min (DF_L, … ,DF_R) * TF_{AVR}
WITHIN 연산자 : min (DF_L, … ,DF_R) * TF_{AVR}

표 2 에서 DF_T 는 해당 단어의 문서 빈도를, DF_L 은 가장 왼쪽 노드의 문서 빈도를, DF_R 은 가장 오른쪽 노드의 문서 빈도를, TF_{AVR} 는 임의 단어의 한 문서 내에서의 평균 단어 출현 빈도를 나타내는 것으로서 식 1 과 같이 구할 수 있다.

$$TF_{AVR} = \frac{\sum_{n=1}^{N} \frac{C_n}{U_n}}{N} \quad \cdots\cdots\cdots \text{(식 1)}$$

식 1 에서 N 은 전체 문서 수를, Cn 은 특정 문서 내의 전체 단어수, Un 은 특정 문서내의 유일한 단어 수를 나타낸다.

그림 5 는 다음 입력질의에 대해 노드 시간 가중치를 구하는 방법을 보이는 예로서, 평균 단어 출현 빈도는 2 라 가정한다. 노드 시간 가중치가 구해진 질의 트리는 가중치에 따라 노드가 제거 되거나 노드의 순서가 바뀌게 된다. 재배열 할 때, 각 노드들은 연산 노드를 기준으로 하여 하위 노드들의 가중치 값의 오름차순으로 정열된다.

입력질의 : (information /N 1 system) AND development AND (kristal OR architecture)

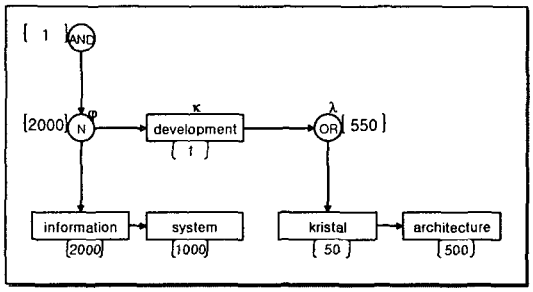


그림 5 색인어 출현 빈도에 의한 질의 트리 구성

현재 그림과 같은 값을 가질 때, "N, development OR" 노드는 "development OR N"순으로 재구성 되어 "1, 550, 2000"의 빈도 순으로 정열 된다. 만약 "system" 이나 "development" 노드의 가중치가 0 이라면, 트리는 NULL 상태로 되어 검색을 위한 별도의 계산은 불필요하게 된다

6. 복합 명사의 처리

기존의 방법들은 복합명사를 일반 단어와 같이 처리하거나, 단일 명사로 분리하였다 하여도 단지 벡터 모델에서만 적용시켰을 뿐인데 비해, 본 논문에서는 불리언 질의어에 출현한 복합 명사를 처리하기 위해서 복합 명사에서 단일 명사들을 분리한 후, 이 단일 명사들을 연산자와 조합하는 복합 명사 확장 방법을 사용하였다.

사용자가 "TITLE:정보검색 시스템"을 입력하였을 때, 질의어는 표 3(그림 6)과 같이 확장된다.

표 3 "정보검색 시스템"에 대한 질의 확장

Query: 정보검색 시스템 Jeongbogeomsaek Siseutem Meaning: information_retrieval system
Extended Query : (정보/W1 검색/W1 시스템) Meaning: (information /W1 retrieval)

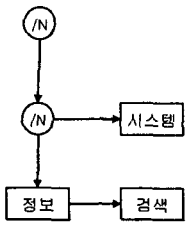


그림 6 생성된 확장 질의 트리

일반적으로 복합 명사 확장에서 띄어쓰기만을 고려한다면 생성된 질의는 "정보검색 /W1 시스템"의 형태를 가지게 되고, 원문에 "정보검색"이 "정보" 와 "검색"으로 띄어 쓰기가 되어 표기된 내용은 검색 할 수 없게 된다.

그러나 본 논문에서 제안 방법을 사용한다면, 복합 명사 "정보검색"은 색인 시스템에 의해 구성 명사 "정보" 와 "검색"으로 분리되고, 단어 간의 순서를 유지하는 근접 연산자 NEAR 를 사용하여 표 3 과 같이 표현된다.

이러한 방법을 사용하기 위해서 색인 저장 시스템과 질의 처리기는 구성 명사를 분리하여 복합 명사를 처리하는 색인 시스템을 사용한다

7. 결론 및 향후 연구방향

정보 검색에서 불리언 질의 최적화를 위한 3 가지 방법을 제시하였다. 이러한 알고리즘을 직접 구현하여 속도와 정확도의 향상이 있음을 확인하였으나, 시간상의 이유로 구체적이고 객관적인 실험 결과는 보이지 못했다. 다음 논문에서 실험결과를 첨부하겠다.

참고문헌

[1]채승기, 남영광, 이준호, 박현주, "효율적인 부울 질의 연산에 관한 연구" 정보관리학회지, 1996.06,v.13,n.1,pp173-185

[2] Witten, I. H., Slistari, M., and Bell, T. C. Managing Gigabytes – Compressing and Indexing Documents and Images, Van Nostrand Reinghold, New York, 1994.

[3] Park, Y. C., Choi, K. S., "*A Korean Compound Noun Retrieval Model Using Statistical NounPattern Categorization*," ICCPOL-97, pp. 361-365, 1997.

[4] Cho, M. J., Yun, B.H.,Rim, H.C.,"A Korean Document Retrieval Model considering Compound Nouns and Derived Nouns," Proc. Of the 22nd KISS conference, pp. 499-502, 1997

[5] Park, Y. C.,Choi, K. S., " A Korean Compound Noun Retrieval Model Using Statistical Noun-Pattern Categorization,"

[6] 박대원, "용언 색인을 적용한 한국어 정보 검색시스템의 검색효율 향상" 석사학위논문 2000 년 8 월