문서의 동적 삽입 삭제를 위한 STEER 역파일 구조

김남일, 김문석, 박영찬, 주종철
한국전자통신연구원, 컴퓨터 소프트웨어기술 연구소, 자연어처리부

STEER Inverted File Structure for Dynamic Document Insertion/Deletion

Namil Kim, Moon-Seok Kim, Young-Chan Park, Zong-Cheol Zhoo
Natural Language Processing Dept Computer Software Technology Lab. ETRI

요 약

역파일 구조(inverted file structure)는 검색 속도가 빠르기 때문에 정보검색 시스템의 색인정보 하부 저장 구조로 널리 이용되지만 문서의 동적 삭제는 어려운 형태이다 본 논문에서는 기존 역파일 구조에 문서마다 색인어의 포스팅 레코드를 기록한 목록을 유지함으로써 문서의 동적 삭제가 용이하고, 위치정보를 포스팅 레코드에서 분리하여 위치 검색이 효율적인 역파일 구조를 설계한다 설계된 역파일 구조는 STEER(Structured Entity Element Retrieval) 정보검색 시스템에서 구현되었다

1. 개요

현대 사회는 정보 사회라 불릴 만큼 개인이 다루어야 하는 정보의 양은 방대하다. 날마다 수많은 문서가 생성되고 개인에게 전달된다. 하지만 정보의 양이 너무 많기 때문에 정작 필요한 정보를 찾는 일은 점점 더 어려워지고 있다. 정보검색 시스템은 이러한 상황을 극복하고 필요한 정보를 효과적으로 찾아주는 도구이다 그러나 정보의 폭발적 증가로 인하여 기존의 정보검색 시스템이 가지고 있던 한계가 드러나고 있다. 인터넷에 있는 문서의 수는 국내에만 천만 건에 육박하고 있고, 회사나 정부기관 등에서 관리하는 문서도 곧 이러한 수준에 이를 것이다. 현재 국내 웹 검색 시스템은 백만 건 단위를 색인, 검색하고 있다 반면 인트라넷 검색 엔진들은 이에 훨씬 못 미치는 상황이다. 이러한 상황에 대처하기 위하여 정보검색 시스템의 성능을 개선하는 문제가 시급하다.

정보검색 시스템의 성능은 검색 속도와 저장 용량으로 측정해 왔다 원하는 정보를 빠르게 찾아주고, 많은 문서를 적은 공간에 저장할 수 있다면 좋은 정보검색 시스템으로 평가해 왔다 그러나 정보검색 시스템이 대량의 문서를 다루면서 검색 속도와 저장 용량이외에 새롭게 고려해야 할 사항이 나타났다. 소량의 문서를 검색할 때는 문서집합 전체를 색인하여 색인구조를 만드는데 시간이 별로 걸리지 않는다.

따라서 문서집합에 변경이 있을 때, 즉 새로운 문서를 삽입하거나 기존 문서를 삭제할 경우, 오프라인으로 문서집합 전체를 색인하여 색인구조를 다시 만들어도 큰 무리가 없다. 동적으로 문서를 삽입, 삭제할 경우에도 문서 수가 적으므로 비효율적인 구조로도 불편함이 적었다.

하지만 문서의 수가 많은 경우 전체를 다시 색인하는 경우 시간이 너무 오래 걸린다. 극단적으로 백만 건의 문서를 색인하여 저장했을 때, 1건의 새로운 문서를 저장하기 위하여 백만 한 건의 문서를 다시 색인해야 한다. 동적으로 문서를 삽입, 삭제할 경우 비효율적인 구조라면 역시 문제가 된다.

본 논문에서는 이러한 문제점들을 해결하고 문서의 동적 삽입, 삭제를 고려한 STEER의 색인구조를 설계한다. 2장에서 기존 방법들과 문제점을 알아보고, 3장에서 STEER의 색인구조를 설계한다 마지막으로 4장에서 결론을 맺는다.

2. 기존 연구 및 문제점

2.1. 역파일 구조 및 문제점

정보검색 시스템이 저장해야 하는 정보는 색인어와, 색인어가 문서 안에서 어느 위치에서 얼마나 자주 나타났는지를 나타내는 위치정보와 문서내 빈도(TF, term frequency), 얼마나 많은 문서에서 나타났는지를 나타내는 문서 빈도(DF, document frequency)등이 있다 이러한 정보를 저장하는데 가장 많이 이용되는 것이 그림 1과 같은 역파일 구조

(inverted file structure)이다[3, 4] 포스팅 파일에는 색인어의 위치정보, 문서내 빈도, 문서 빈도가 저장되고, 단어 색인은 B+ 트리 등의 인덱스구조로 포스팅 파일내의 특정 레코드를 가리킨다.

역파일 구조에서는 여러 문서에 해당하는 정보가 포스팅 파일에서 하나의 레코드를 구성하기 때문에, 색인어 당 한 개의 레코드 읽기로 색인어의 정보를 모두 얻을 수 있어서 검색 속도가 빠르다. 새로운 문서를 삽입할 경우는 기존 레코드의 끝 부분에 덧붙이면 되므로 비교적 수월하다. 반면 기존 문서를 지울 경우 먼저 지울 문서에 속한 색인어의 포스팅 레코드를 모두 찾고, 찾은 색인어의 포스팅 레코드를 부분 삭제하는 두 단계를 거쳐야 한다. 지울 문서에 속한 색인어의 레코드를 찾기 위하여 포스팅 파일을 전부 탐색하여 지울 문서의 DID(document ID)와 비교해야 한다. 이 작업은 색인된 문서의 수에 비례하여 시간이 걸린다. 찾아낸 레코드를 부분 삭제하려면 레코드의 중간 부분을 지우고 뒷부분의 내용을 앞으로 당겨 써야 한다.

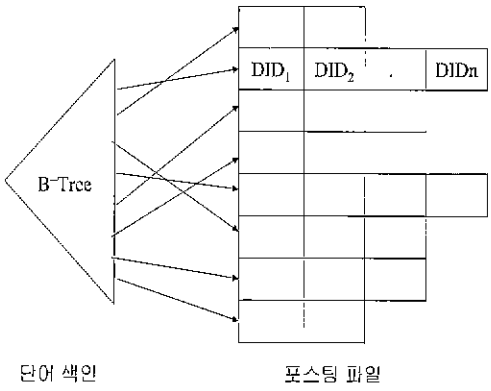

그림 1 역파일 구조

2.2. 동적 삭제를 위한 수정된 역파일 구조

역파일 구조에서 동적으로 문서를 삭제하기 위하여 여러 고려가 이루어졌다. 그 중 포스팅 레코드의 부분 삭제 문제를 해결하기 위하여, 색인어가 나타난 문서마다 별도의 포스팅 레코드를 만들기도 한다(그림 2).

하나의 색인어에 대하여 문서마다 분리된 포스팅 레코드를 만들면, 문서를 지울 때 포스팅 레코드의 일부를 지우는 것이 아니라 전체 레코드를 지우므로 빠르게 문서를 지울 수 있다. 그러나 포스팅 레코드의 수가 지나치게 많아지는 단점이 있다. 아무리 작은 레코드라도 일정한 최소길이를 차지하므로 레코드 수가 많아지면 낭비되는 공간도 늘어난다. 포스팅 정보를 읽을 때에도 블록 단위로 읽기 때문에, 정보가 기록되지 않은 빈 공간도 함께 읽혀져 디스크를 읽는 회수가 증가하여 검색 속도가 느려질 수 있다. 한편 대량의 문서를 한꺼번에 색인할 경우, 생성하는 포스팅 레코드의 수가 많기 때문에 그림 1의 구조보다 시간이 많이 걸린다.

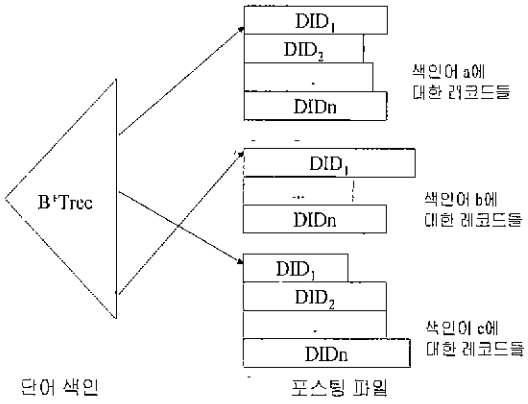

그림 2 수정된 역파일 구조

2.3. MIDAS-III의 역파일 구조

그림 1이나 그림 2의 구조는 포스팅 레코드 안에 색인어의 위치정보를 함께 저장한다. 그런데 위치정보는 두 단어가 동일한 문장에 나타나는가 등의 근접 질의에만 이용되고 불리언 질의에는 사용하지 않는다. 따라서 문서내 빈도, 문서 빈도를 포스팅 파일에 저장하고 위치정보를 분리하여 별개의 위치정보 파일로 나누면 불리언 질의를 처리할 때는 포스팅 파일의 정보만 읽으면 되므로 검색속도가 빨라진다. MIDAS-III[1, 2]가 사용한 역파일 구조를 그림 3에 나타내었다.

MIDAS-III는 문서마다 위치정보 레코드를 만들고, 위치정보 레코드에 대한 포인터를 포스팅 파일에 저장하여 근접 질의를 처리할 경우에만 위치정보를 읽는다. 이와 같은 위치정보 파일 구조는 앞서 포스팅 레코드를 문서마다 분리한 방법의 단점을 그대로 가진다. 즉 위치정보 레코드 수가 지나치게 늘어나기 때문에, 저장 공간의 낭비, 대량 문서 색인시 속도 저하 등의 단점을 지닌다.

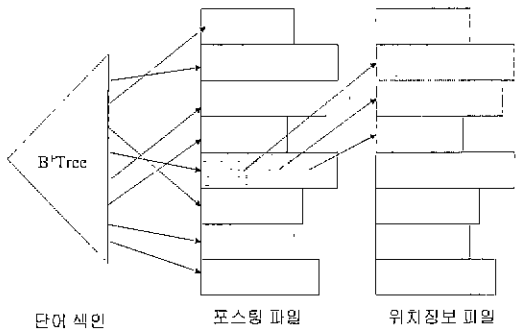

그림 3 MIDAS-III의 역파일 구조

2.4. 지연된 문서 삽입 삭제

문서를 삽입하고 삭제할 때 동적으로 역파일 구조를 갱신하는 것이 어렵기 때문에, 임시 영역에 기록해 두고 오프라인으로 한꺼번에 수정하는 방법이 있다.

이 방법을 사용하면 저장 구조의 동시성 제어와 같은 문제를 피할 수 있다. 그러나 새로운 문서가 삽입되었을 경우, 기존 문서를 검색하고 추가된 문서도 다시 한 번 검색해야 하는 부담이 있다. 두 번 검색했다 하더라도 기존 문서 검색 결과와 추가된 문서의 검색 결과가 제대로 통합되지 않으면 추가된 문서의 랭킹이 잘못되어 검색 성능이 떨어질 수 있다. 문서가 삭제된 경우에도 검색결과에서 삭제된 문서를 걸러내는 작업을 해야 하는 단점이 있다.

3. STEER의 역파일 구조

STEER 정보검색 시스템의 색인구조는 하부 저장 관리기의 레코드 단위 잠금(lock) 기능을 이용하여 동적으로 문서를 삽입, 삭제하도록 설계되었다. 따라서 삽입, 삭제가 역파일 구조에 즉시 반영되므로 앞서 설명한 이중 검색의 단점을 피할 수 있다. STEER의 역파일 구조를 그림 4에 나타내었다.

위치정보를 이용한 검색에서 속도를 향상시키기 위하여 포스팅 레코드로부터 위치정보를 분리하여 위치정보 파일을 별도로 만들었다 하지만 MIDAS-III처럼 위치정보 레코드를 문서마다 분리하지 않고, 하나의 색인어에 대한 위치정보는 하나의 레코드로 나타내어 위치정보로 인한 저장공간의 낭비를 막았다.

2장에서 살펴본 바와 같이 역파일 구조에서의 동적 문서 삭제는 시간이 많이 걸린다 하지만 색인어 레코드를 문서마다 만들면 그 부작용도 크므로 동적 문서 삭제에서 레코드의 부분 삭제 시간을 줄이기는 힘들다. 대신에 STEER는 지울 문서에 속한 포스팅 레코드를 찾는 시간을 줄이기 위하여 문서에 속한 색인어 포스팅 레코드들을 가리키는 파일을 만든다 (그림 2). 따라서 문서를 지울 때, 포스팅 레코드 지정 파일을 읽어서 지울 포스팅 레코드를 알 수 있다. 그러나 문서 삽입시 포스팅 레코드 지정 파일을 생성하기 때문에 문서 삽입 속도는 기존 방법보다 느리다. 그래서 STEER는 문서 삽입에 대하여 두 가지 접근법을 취하고 있다.

소량의 문서 삽입은 포스팅 레코드 지정 파일에 쓰는 작업이 추가된 것 외에는 비교적 수월하므로 기존 방법을 그대로 이용하여 동적으로 삽입한다. 하지만 대량의 문서가 삽입될 경우나 대량의 초기 데이터가 대량인 경우 문서를 한 건, 한 건씩 동적으로 추가하는 방식은 시간이 너무 많이 걸린다. 따라서 대량의 문서가 추가될 경우, 포스팅 레코드 전체를 한꺼번에 만드는 벌크로더(bulk-loader)나 벌크애더(bulk-adder)를 이용하여 색인 구축 효율을 높인다. 또한 삭제할 문서가 전체 문서에 비하여 많은 경우에도 문서를 동적으로 한 건씩 삭제하는 대신, 삭제되지 않을 문서를 벌크로더를 이용하여 다시 색인하는 편이 시간도 적게 걸린다.

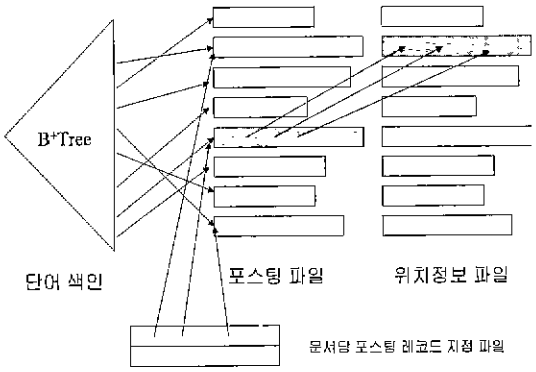

그림 4 STEER의 역파일 구조

4. 결론

본 논문에서는 정보검색 시스템의 역파일 구조에서 동적 문서 삽입 삭제시 발생하는 문제점에 대하여 살펴보고 이를 해결한 역파일 구조를 제안하였다 제안된 역파일 구조는 문서마다 포스팅 레코드 지정 파일을 유지하여 동적 문서 삭제시 속도를 높이고, 대량의 문서 삽입시 벌크로더나 벌크애더를 사용한다. 이 역파일 구조는 STEER 검색 시스템에 구현되어 대량의 문서를 효율적으로 처리하고 있다.

참고문헌

[1] 김강석, 장재우, 이진주, 트랜잭션 지원을 위한 정보검색 인덱스의 설계 및 구현, *97정보처리학회 97 춘계 학술발표논문집*, 1997.9.12(04) pp 140-145

[2] 이정기, 안성현, 김강석, 김연중, 장재우, 허대영, MIDAS-III에 기반한 정보검색 하부구조의 설계, *95 정보과학회 가을 학술 발표 논문집(A)* 1995.10 22(2), pp 259-262

[3] 이준호, 안정수, 정보 검색 시스템 KRISTAL-II 개발, *한국정보과학회지* 97년 2월

[4] W. B. Frakes and R Baeza-Yates, *Information Retrieval Data Structure and Algorithms*, Prentice-Hall, 1992