

텍스트분할에 의한 색인방법 연구

A Study on Indexing Method using Text Partition

강무영(Mu-Yeong Kang)*, 이상구(Sang-Gu Lee)**

목 차

- | | |
|--------------------------------|-------------------------------------|
| 1 서론 | 4 개선된 색인기법의 적용사례 |
| 2 관련 연구 | 4.1 적용시스템의 소개 |
| 2.1 역파일 색인의 특성 | 4.2 KRISTAL-II 색인 API를 이용한 색인시스템 개발 |
| 2.2 역파일 색인기법 | 4.3 실험환경 및 Test Collection |
| 2.3 기존 역파일 색인기법에 관한 고찰 | 4.4 실험결과 |
| 2.4 기존 역파일 색인기법들의 특성 및 문제점 | 5 결론 및 향후 계획 |
| 3 개선된 텍스트 분할에 의한 색인 | |
| 3.1 개선된 색인기법의 개요 | |
| 3.2 개선된 색인기법에 의한 색인시스템 설계 및 구현 | |

초 록

색인은 데이터베이스에 저장된 문서를 효과적으로 검색하기 위한 정보검색시스템의 필수 기능이다. 컴퓨터의 발달로 전자정보가 점점 많아짐에 따라 데이터베이스에 저장해야 할 문서가 대량화되고 있고, 이러한 대용량 문서를 색인하기 위해서는 많은 시스템 자원과 처리 시간을 필요로 한다. 따라서 본 논문에서는 대용량 문서를 적은 자원으로 짧은 시간에 색인할 수 있는 개선된 텍스트분할에 의한 색인기법을 제안한다. 제안한 색인기법은 실제 검색시스템에 적용하고, 실험을 통해 우수성을 증명한다.

ABSTRACT

Indexing is a prerequisite function for the information retrieval system in order to retrieve the information of the documents effectively which are saved in database. As a digital data increases in accordance with the development of a computer, the numbers of literatures to be saved in database have also been increased in a large volume. To retrieve such documents of large volume, a lot of system resources and processing time will be required.

In this paper, we suggest a advanced indexing method using text partition. This method can retrieve the documents of large volume in short processing time. We applied this suggested indexing method to real information retrieval system, and proved its excellent functions through the demonstration.

키워드 : 텍스트 분할 색인, 대용량문서 색인, 정보검색, 검색시스템

* 연구개발정보센터 선임연구원 kmy@ns.kordic.re.kr

** 한남대학교 컴퓨터공학과 교수 sglee@eve.hannam.ac.kr

■ 논문 접수일 : 1999년 12월 3일

1 서 론

컴퓨터의 눈부신 발전으로 수많은 문서들을 전자적으로 저장하여 검색을 통해 손쉽게 필요한 정보에 접근할 수 있게 되었다. 최근 정부기관 및 출연(연)에서 추진한 국가전자도서관사업, 과학기술정보유통사업, 과학기술통합정보시스템, 학회정보화사업, 지역정보화사업, 교육정보화사업 등을 통해 국내에서 생산되는 수많은 문서들이 전자화되어 데이터베이스에 저장되고 있으며, 국·공립도서관, 대학도서관, 일반사립도서관 등에서도 전자도서관 체제 구축에 박차를 가하고 있다(연구개발정보센터 1998; 전산원 1998; 전산원 1999). 또한 해외에서 생산되고 있는 유명 저널의 초록을 포함한 서지사항은 물론 원문까지도 국내에 도입하여 데이터베이스를 구축하여 인터넷을 통해 서비스하고 있다.

이와 같이 전자정보의 데이터베이스는 점점 대량화되고 있으며, 이러한 정보를 효율적으로 저장하고 수요자들이 요구하는 정보를 검색을 통해 빠르고 정확하게 제공할 수 있도록 연구하는 분야가 정보검색분야이다. 정보화사회에 있어서 매우 중요한 역할을 하고 정보검색분야는 지금까지 많은 연구가 활발히 진행되어 왔다(Moffat and Zobel 1992; Bell et al. 1995; Zobel and Dart 1995). 1950년대 Taube가 고안한 유니텀시스템을 필두로 1960년대에는 오프라인 및 실험용 온라인 정보검색시스템, 1970년대에는 온라인 정보검색시스템, 1980년대에는 전문데이터베이스가 출현하였고, 1990년대에는 CD-ROM, 멀티미디어 정보검색 및 지능형 정보검색시스템 등이 등장하였다(사공철, 서경주 1996). 또한 국내에서도 대량의 정보를 컴퓨터에 저장하여 수요자의 요구에 적합한 정보를 제공해줄 수 있는 정보검

색시스템(Information Retrieval System:IRS)의 개발이 활발히 진행되고 있다. 대표적인 정보검색시스템으로는 연구개발정보센터에서 개발한 KRISTAL과 한국전자통신연구소에서 개발한 MIDAS를 들 수 있다. 이들 정보검색시스템은 순수 국내기술로 개발된 것으로 지속적으로 발전시켜나가고 있다(연구개발정보센터 1995; 연구개발정보센터 1997; 한국전자통신연구소 1995).

정보검색시스템은 일반적으로 비정형 데이터를 대상으로 하기 때문에 내용기반 검색을 필요로 한다. 따라서 정보검색시스템을 기능별로 나누면 비정형데이터를 저장하는 하부저장 분야, 색인어를 추출하여 저장하는 색인 분야, 사용자 질의에 적절한 검색결과를 제공해 주는 검색 분야가 있다(정영미 1993). 특히 색인 분야는 효과적인 질의검색을 위해 필수기능으로서 일반적으로 널리 알려진 색인구조는 역파일(inverted file), 요약파일(signature file), 비트맵파일(bitmap file) 등이 있다. 이 구조들 중에서 역파일구조는 빠른 검색속도와 가중치, 위치정보 등 부가정보 표현이 용이하여 정보검색시스템에 널리 채용되고 있다(Salton 1989).

색인시스템의 성능을 결정하는 요소는 색인하는 데 걸리는 시간과 소요되는 시스템 자원이다. 특히 대량의 자료를 색인하는 데에는 필연적으로 대량의 시스템 자원을 필요로 하며 많은 시간이 소요된다. 따라서 지금까지 연구된 여러 색인방법들은 정보검색시스템의 특성에 따라 적절한 기법들을 적용해 운용하고 있다(Moffat, Zobel and Klein 1995; Zobel, Moffat and Sacks-Davis 1993).

한편, 데이터베이스의 대용량화 추세와 더불어 이 대용량 데이터로부터 사용자가 원하는 정보에 정확하게 접근하기 위한 많은 노력이 시도되고

있다. 예를 들면 사용자 질의를 통해 얻어진 검색 결과에 대해 순위화 기법을 이용하여 사용자가 원하는 정보순으로 정렬하여 보여주는 것이다 (Moffat and Zobel 1994; Persin 1994; 김철, 이승채, 김병기 1996; 고미영, 정영미 1999). 이러한 처리를 위해서는 색인에 가중치를 주거나 위치정보를 표현하는 등 여러 가지 색인정보를 추가해야 한다. 색인정보가 많아질수록 색인에 소요되는 시간과 자원은 늘어나게 된다.

본 논문은 역파일(inverted file)구조를 기반으로 한 색인방법을 연구하고, 대용량 문서를 대해 적절한 자원(resource)을 이용하여 빠른 시간 안에 색인을 할 수 있는 방법을 제안한다. 제안한 색인방법은 개선된 텍스트분할에 의한 색인방법으로서 대용량 문서의 텍스트를 분할하여 메모리 기반에 의한 색인방법을 이용하여 부분 역파일을 생성하고, 이 부분 역파일들을 순차적으로 병합(Merge)하여 색인을 완료한다. 제안한 색인방법은 실제 검색시스템에 적용시켜 실험을 통해 성능평가를 하고, 우수성을 검증한다.

본 논문의 구성은 다음과 같다. 제2장에서는 정보검색시스템에서 널리 사용되는 역파일구조에 의한 색인방법과 지금까지 연구된 대표적인 색인

방법을 알아본다. 제3장은 본 논문에서 제안한 개선된 텍스트분할에 의한 색인기법과 구현방법에 대해 기술하고, 제4장에서 제안한 색인방법을 검색시스템에 적용하여 성능평가를 한다. 마지막으로 제5장에서 연구 수행에 대한 결론과 향후 연구과제를 제시한다.

2 관련 연구

2.1 역파일 색인의 특성

역파일(inverted file)은 불리안 질의(Boolean Query)과 순위결정 질의(Ranked Query)를 위한 가장 실질적인 형태의 색인구조이라 할 수 있다(Salton 1989; Witten, Moffat and Bell 1999). 역파일은 어휘사전에 있는 모든 용어에 대해 각자의 역파일 엔트리(inverted file entry)를 가지고 있다. 역파일 엔트리는 문서에 나타난 모든 용어들에 대한 포인터들의 목록을 저장하고 있으며, 각각의 포인터는 해당 용어가 나타난 문서들의 번호이다. 이것은 가장 자연스러운 색인 방법으로서 책에서 가장 일상적으로 볼 수 있는

〈표 1〉 텍스트 예제; 각각의 줄을 한 문서로 표현

문서번호	텍스트
1	Pease porridge hot, pease porridge cold,
2	Pease porridge in the pot,
3	Nine days old.
4	Some like it hot, some like it cold,
5	Some like it in the pot,
6	Nine days old.

전통적인 방법에 가깝다.

〈표 1〉과 같이 하나의 텍스트를 구성하고 있는 각각의 문서에 여러 개의 색인어를 포함하고 있고, 각각의 색인어가 여러 개의 문서에 나타난다

2.2 역파일 색인기법

〈표 2〉 텍스트 예제에 대한 배열

	cold	days	hot	in	it	like	nine	old	pease	porridge	pot	some	the
1	1		1						2	2			
2				1					1	1	1		1
3		1					1	1					
4	1		1		2	2						2	
5				1	1	1					1	1	1
6		1					1	1					

〈표 3〉 도치된 도수행렬

일련번호	용어	문 서					
		1	2	3	4	5	6
1	cold	1			1		
2	days			1			1
3	hot		1		1		
4	in		1			1	
5	it				2	1	
6	like				2	1	
7	nine			1			1
8	old			1			1
9	pease	2	1				
10	porridge	2	1				
11	pot		1			1	
12	some				2	1	
13	the		1			1	

1. 주어진 N 개의 문서와 n 개의 용어에 대한 배열을 주기억장치에 만들고 초기화 한다.
2. 각각의 문서들에 대해서
 - ① 문서를 읽고 어휘분석을 하여 색인어와 출현빈도를 추출한다.
 - ② 각각의 색인어들에 대해서 배열에 출현빈도를 기록한다.
3. 도수행렬에 있는 각 용어들에 대해서
 - ① 새로운 역파일 엔트리를 시작한다.
 - ② 각 문서에 대한 빈도가 0보다 크면 역파일 엔트리에 추가한다.
 - ③ 필요하면 역파일 엔트리를 압축한다.
 - ④ 역파일 엔트리를 역파일에 추가한다.

〈그림 1〉 역파일 색인과정

고 하면, 이러한 관계는 <표 2>와 같이 각 행에는 한 문서를, 각 열에는 한 단어를 나타내는 배열로 표현할 수 있다. 이렇게 표현한 배열을 <표 3>과 같이 도치(invert)시켜 역파일을 생성하게 된다.

역파일을 생성하기 위한 방법은 그림 1과 같다. 아주 단순한 알고리즘이지만 배열의 크기 때문에 실질적으로 역파일 색인작업은 간단치 않다. 이 때문에 배열을 구성하여 역파일 색인작업을 하기보다는 좀더 경제적인 방법이 고려되어야 한다.

2.3 기존 역파일 색인기법에 관한 고찰

색인방법은 크게 두 가지로 대별할 수 있다. 하나는 전체 텍스트를 한번만 읽어서 색인을 완료하는 방법과 다른 하나는 전체 텍스트를 두 번 이상 읽어서 색인을 하는 것이다. 두 방법 모두 데이터베이스에서 추출한 색인어의 어휘사전을 관리하기 위해 해쉬테이블(hash table)이나 이진탐색트리(binary search tree)와 같은 동적 사전 자료구조(dynamic dictionary data structure)를 사용한다(Witten, Moffat and Bell 1999).

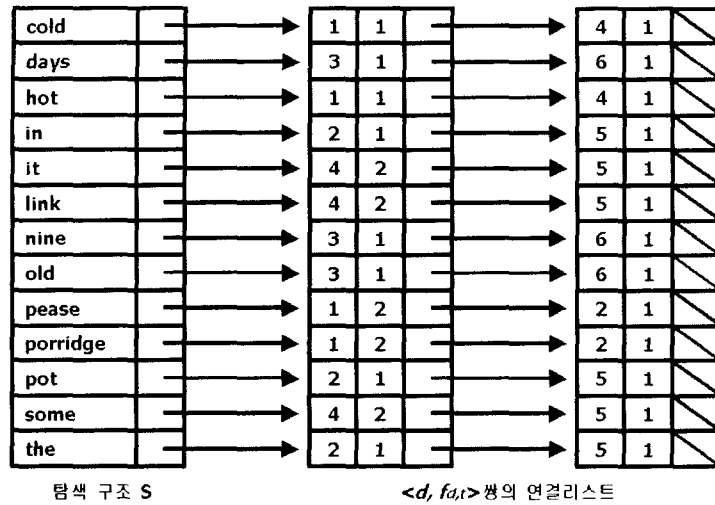
2.3.1 텍스트를 한번 읽어서 색인하는 방법

가장 기본적인 방법으로는 주기억장치에 동적

사전을 구축하고, 각 사전 엔트리(dictionary entry)와 관련된 행번호(line number)와 빈도수를 저장하는 노드(node)를 연결리스트(linked list)로 연결한다. 모든 문서를 처리한 후, 이 사전구조(dictionary structure)를 운행(traverse)하여 색인어별로 역파일 엔트리를 생성한다. 생성된 역파일 엔트리를 역파일에 추가함으로써 색인이 완료된다. 그림 2는 [표 1]의 텍스트 예제를 사전구조와 연결리스트로 표현한 것이다.

이 색인방법은 모든 색인정보를 주기억장치에 저장하므로 처리속도는 빠르지만 엄청난 주기억장치를 필요로 한다. 이와 같이 대량의 주기억장치를 요구하는 색인방법은 현실적으로 대용량 문서의 색인방법으로 부적합하다.

이러한 문제점을 보완하기 위해 개선한 색인방법은 주기억장치에 저장할 연결리스트의 모든 노드를 보조기억장치에 파일로 저장한다. 모든 문서에 대한 색인어 추출이 끝나면 색인어별로 이 파일을 읽어서 역파일 엔트리를 생성한 후 역파일에 추가한다. 이러한 방법으로 대용량문서의 색인이 가능하지만 여기에 또 다른 문제점을 가지고 있다. 즉, 연결리스트를 생성할 경우 순차적으로 파일에 기록하지만, 이 연결리스트로부터 역파일 엔트리를 생성하기 위해서는 모든 노드에



〈그림 2〉 〈표 1〉을 역파일로 표현한 구조

대해 비순차접근(random seek)을 해야한다. 비순차접근은 엄청나게 많은 시간을 필요로 하며 색인시스템의 성능을 떨어뜨리는 원인이 된다. 따라서 보조기억장치를 이용하여 색인할 경우 색인시간을 단축하기 위하여 모든 색인정보에 대해 순차적 저장과 순차적 읽기를 해야한다.

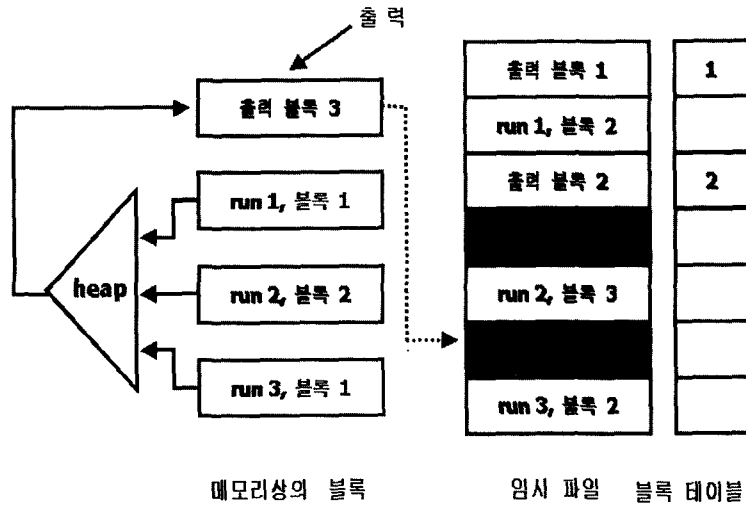
순차처리를 위해서는 모든 문서에 대해 어휘분석하여 색인정보를 추출한 후 임시파일로 저장한다. 임시파일에는 텍스트에 나타나는 용어번호(t) 4 바이트, 문서번호(d) 4 바이트, 문서 내에서의 용어 출현빈도($f_{d,t}$) 2 바이트 등 세가지 항목으로 구성된 레코드로 저장한다. 다음에 임시파일을 t 와 d 의 오름차순으로 정렬하고, 정렬된 리스트를 순차적으로 읽어서 역파일엔트리를 생성하여 역파일에 추가한다.

정렬은 전형적인 외부병합정렬(external merge sort)방법을 사용한다. 정렬의 초기단계는 임시파일을 k 개 단위의 레코드로 블록(block)을 형성하고, 블록단위로 읽어서 Quick-Sort와 같은 메모리정렬기법을 사용하여 정렬한 후, 임시파일

에 다시 저장한다(이하 "run"이라 한다). 이 정렬된 run 들을 병합함으로써 정렬이 완료된다.

이 색인방법은 병합처리과정에서 항상 임시파일에 대해 두 개의 복사본을 갖게 되므로 병합의 최종단계에서는 임시파일의 두 배에 해당하는 보조기억장치가 필요한 것이다. 여기서 추출한 색인정보를 압축하면 보조기억장치의 사용량을 줄일 수 있다. 압축요소로는 색인어 식별자간의 간격 t -gap과 문서번호간의 간격 d -gap을 이용한다. 압축기능을 추가하면 소요시간을 지연시키는 원인이 되지만, 이 추가계산시간은 압축된 자료가 디스크 전송시간(disk transfer time)을 줄여주기 때문에 거의 상쇄된다.

여기서는 색인시간을 크게 줄이지 못하지만 보조기억장치의 소요공간을 많이 줄일 수 있다. 다음으로 색인시간을 단축하기 위해서는 병합단계에서 다중병합(multi-way merge)를 사용하는 것이다. N 개의 색인어에 대해 R -way 병합을 하면 $\lceil \log_R N \rceil$ 단계 후에 병합이 완료되므로 많은 시간을 단축할 수 있다. 다중병합을 수행하기 위



〈그림 3〉 In-place multi-way merge sort

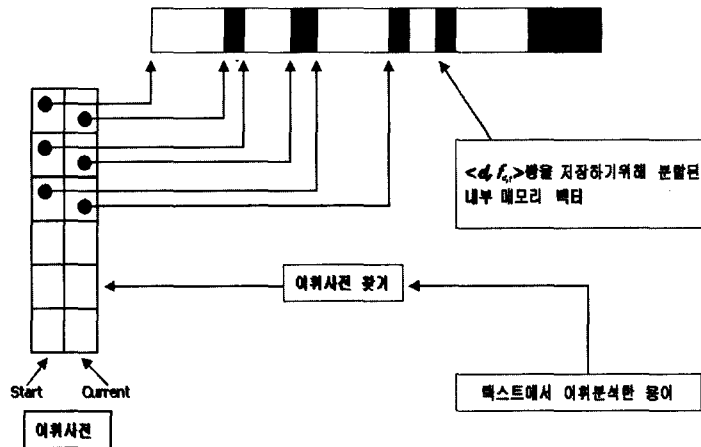
해서는 heap과 같은 우선순위 큐(priority queue)를 이용한다.

마지막으로 보조기억장치의 사용공간을 좀더 줄이기 위해 in-place multiway merge를 한다. 즉, 각각의 정렬된 run을 다시 바이트 크기의 블록 단위로 나누어 heap을 통해서 후보를 보급한다. 병합작업은 가장 작은 $\langle t, d, f_{a_i} \rangle$ 후보를 반복적으로 취함으로써 처리되고 이것은 출력 블록에 채워진다. 출력 블록 또한 정확히 b 바이트가 되도록 한다. 이때 각 출력 블록들은 임시파일의 빈 블록에 다시 저장한다. 이 과정은 각 run의 마지막 블록이 처리될 때까지 계속한다. 그림 3은 세 개의 정렬된 run들의 병합에 대한 중간단계를 보여주고 있다. 블록단위로 정렬한 후, 이 블록들을 출력 순서대로 교환하면 임시파일의 정렬이 완성된다. 이 정렬된 임시파일을 순차대로 읽어서 역파일 엔트리를 생성하고 역파일에 추가하면 색인작업이 완료된다. 이 방법은 앞에 기술한 방법에 비해 보조기억장치의 사용량을 최소화할 수 있다.

2.3.2 텍스트를 두 번 이상 읽어서 색인하는 방법

색인을 할 때 각각의 색인어 t 에 대해서 문서의 빈도수 f_t 를 미리 알 수 있다면 문서번호 d 와 빈도수 f_{a_i} 의 리스트를 저장할 수 있는 정확한 크기의 배열을 할당할 수 있다. 이것은 앞에서 설명한 주기억장치를 이용한 기본적인 색인방법에 비해 두 가지 측면에서 주기억 공간을 절약할 수 있다. 첫째 각각의 리스트 노드에 다음 포인터가 불필요하고, 둘째 문서의 수 N 을 미리 알 수 있으므로 $32 \times f_t$ 비트와 같이 고정된 값이 아니라 $f_t \times \lceil \log N \rceil$ 비트를 역파일 엔트리의 색인어 t 에 대한 구성요소인 문서번호 d 크기로 할당할 수 있다.

이러한 색인정보를 미리 알기 위해서 먼저 전체 텍스트를 읽어서 분석한 후, $\langle d, f_{a_i} \rangle$ 쌍을 저장하기 위한 정보를 수집한다. 수집된 정보를 통해 정확한 크기의 대형 1차원 배열과 각각의 용어에 대해 \langle 문서번호, 출현빈도 \rangle 쌍을 저장할 위치를 추적하기 위한 배열을 할당한다. 이와 같이 주기억장치가 할당되었을 때 이 색인 방법의 두 번



〈그림 4〉 주기억공간 할당에 의한 색인방법

제 단계는 그림 4에 도식화되어 있다. 모든 용어들을 미리 알 수 있기 때문에, 용어들을 손실하지 않고 저장할 수 있는 어휘사전을 minimal perfect hash function으로 설계할 수 있다.

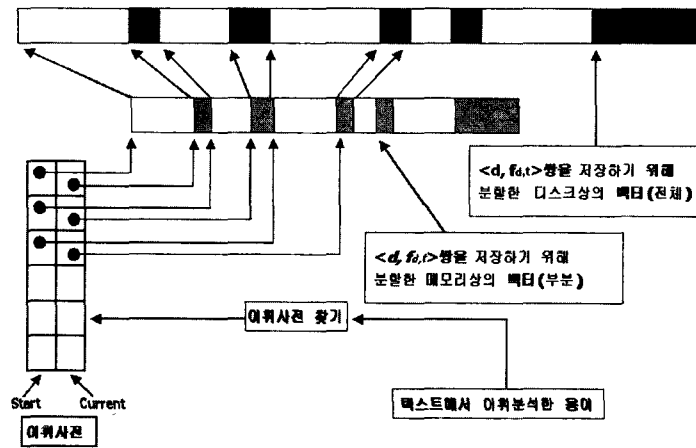
이 색인방법은 연결리스트를 이용한 방법보다는 훨씬 작지만 아직도 모든 색인정보를 주기억장치에 저장하므로 많은 주기억장치의 자원을 필요로 하고 있다.

과다한 주기억장치 공간에 대한 요구를 해결하기 위한 색인방법으로는 역파일 색인을 작은 작업들로 나누고, 이용 가능한 주기억장치의 범위 내에서 내부 메모리 벡터를 만들어 각각의 작업을 진행시킨다. 작은 작업으로 나누는 방법은 두 가지가 있다.

첫 번째 방법은 전체 색인어 사전을 일정 크기로 분할하여 색인하는 것이다. 각각의 분할된 작업단위를 load라 하면, 각 load를 처리할 때마다 전체 텍스트를 읽어서 분할한 범위내의 어휘만 처리한다. 이것은 주기억장치의 사용을 최소화할 수 있지만, 전체 텍스트를 load 수만큼 읽어야 하므로 많은 시간이 소요된다. 이 단점을 보완하기 위해 보조기억장치를 이용하면 처리시간은 크

게 향상시킬 수 있다. 즉, 두 번째 패스에서 〈용어번호 t , 문서번호 d , 빈도수 $f_{d,t}$ 〉로 구성된 레코드를 임시파일에 load 별로 저장해 두었다가, 각 load를 처리할 때 해당 임시파일만 읽어서 처리한다. 따라서 모든 load를 처리하는 데 전체 텍스트는 한번만 읽게 되므로 색인시간을 단축할 수 있다. 하지만 색인정보를 미리 추출하여 임시파일에 저장하므로 그만큼 많은 보조기억장치를 필요로 한다.

두 번째 방법은 사전분할보다는 텍스트 자신을 분할하여 색인하는 것이다. 텍스트의 분할 단위를 chunk라 하면, 첫 번째 chunk의 문서들부터 차례대로 부분 역파일 생성하고, 이 부분 역파일을 전체 역파일이 완성될 때까지 병합해 나간다. 전체 역파일을 저장할 분할된 벡터는 그림 5과 같이 보조기억장치에 두고, 하나의 chunk에 대한 부분 역파일은 메모리에 구축하였다가, 디스크에 있는 전체 역파일에 병합시킨다. 병합방법은 전체 역파일 엔트리를 블록단위로 읽어서 새로운 부분 역파일 엔트리를 추가한 후에 다시 저장한다. 각 색인어에 대한 엔트리의 크기는 미리 보조기억장치에 할당하였으므로 부분 역파일 엔트리



〈그림 5〉 텍스트기반의 분할에 의한 색인방법

와 병합된 전체 역파일 엔트리는 보조기억장치의 읽어들이는 곳으로 다시 저장시킬 수 있다. 이 방법은 보조기억장치를 이용한 사전분할 색인방법보다는 색인시간이 약간 더 걸리지만 보조기억장치의 사용량을 획기적으로 줄일 수 있다.

2.4 기존 역파일 색인기법들의 특성 및 문제점

지금까지 설명한 색인기법들의 특성과 문제점을 알아보자. 먼저 전체 텍스트를 1회만 읽어서 색인하는 경우, 색인속도 향상과 소요자원을 최소화하기 위해 압축기법을 도입하였다. 압축기법의 도입은 색인해야 할 정보가 많아질수록 압축비가 높아지는 것은 보증할 수 없고, 색인시스템을 구현하는 데 많은 노력이 필요하다. 반면에 전체 텍스트를 2회 이상 읽는 경우에는 그만큼 색인시간이 지연된다. 특히 어휘분석 단계에서 형태소분석기를 통해 색인어를 추출하거나, 물리적으로 다른 시스템에 존재하는 텍스트를 색인한다면 네트워크 트래픽을 고려할 때 심각한 색인시간 지연의 원인이 될 수 있다.

따라서 색인작업은 전체 텍스트를 1회만 읽어

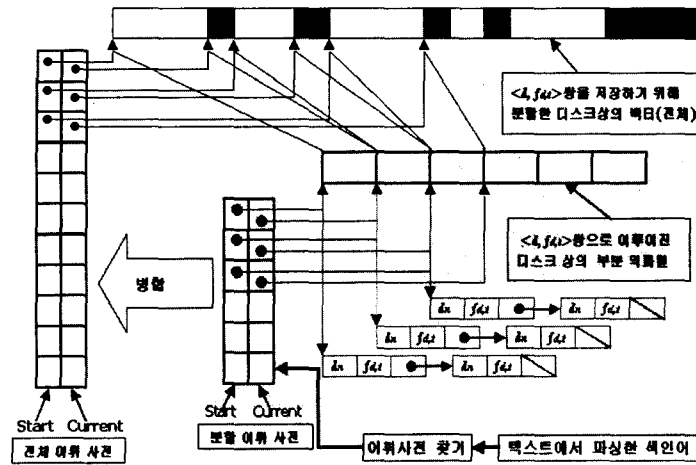
서 압축을 고려하지 않은 상태로 적절한 자원을 이용하여 빠른 시간에 이루어져야 한다.

3 개선된 텍스트 분할에 의한 색인

3.1 개선된 색인기법의 개요

본 논문에서는 제안한 색인방법은 텍스트를 일정한 크기로 분할하여 부분 역파일을 생성한 후, 이를 병합함으로써 색인을 완료한다. 기존 역파일 색인방법에서 설명한 텍스트기반의 분할에 의한 색인방법과 차이점은 전체 텍스트를 한번만 읽어서 처리한다는 것과 텍스트 분할 크기는 문서의 수가 아니고 추출된 색인어의 수라는 것이다. 이를 구체적으로 설명하면 다음과 같다.

색인정보를 연결리스트로 주기억장치에 저장할 경우 최대 수용할 수 있는 크기가 텍스트를 분할하는 단위이다. 연결리스트로 주기억장치에 저장할 색인정보가 문서번호(4 바이트), 출현빈도(2 바이트) 및 다음포인터(4 바이트)로 구성되어 있고, 주기억장치가 40MB이며, 동적사전구조를



〈그림 6〉 개선된 텍스트분할에 의한 색인방법

위해 30MB가 사용된다고 하자. 이때 텍스트를 분할하는 단위는 $(40 \times 10^6 - 30 \times 10^6) / 10 = 1,000,000$ 개의 색인어가 될 것이다.

전체 텍스트를 문서별로 읽어서 어휘분석을 하고, 이 분할단위의 색인어 수만큼 주기억장치에

서 색인작업을 한 후, 부분 역파일을 생성하여 임시파일에 저장한다. 이 때 전체 역파일 구조를 형성하는 데 필요한 정보도 수집한다. 모든 문서에 대해 부분 역파일이 완료되면, 부분 역파일 생성 과정에서 수집한 정보로 전체 역파일 구조를 만

〈표 4〉 색인대상 자료 및 시스템 성능을 표현한 심볼

구 분	심 볼
총 텍스트의 크기	B
텍스트를 구성하고 있는 총 문서 수	N
텍스트를 구성하고 있는 유일한 단어 수	n
텍스트를 구성하고 있는 총 단어 수	F
색인어 포인터 수	f
압축된 역파일의 최종 크기	I
동적사전구조의 크기	L
디스크 접근 시간	t_s
바이트당 디스크 전송시간	t_r
바이트당 역파일 부호화 시간	t_d
10-바이트 크기 레코드의 비교/교환 시간	t_c
한 단어에 대한 어휘분석 및 사전찾기 시간	t_p
주기억장치 용량	M

들어서 여기에 부분 역파일을 병합해 나간다.

모든 부분 역파일을 병합하면 한 텍스트의 색인이 완료되며, 이 색인방법을 도식화 하면 그림 6과 같다. 부분 역파일은 색인어별로 문서번호순으로 이미 정렬되어 있기 때문에 부분 역파일이 만들어진 순서대로 임시파일을 읽어서 전체 역파일에 추가하면 된다.

색인대상이 되는 자료 크기와 시스템의 성능을 (표 4)와 같이 표현했을 때, 제안한 색인기법의 색인시간 T (초)를 일반적으로 표현하면 다음과 같다.

$$T = Bt_r + Ft_p + I(t_r + t_d) + cIt_r / lb + 2It_r$$

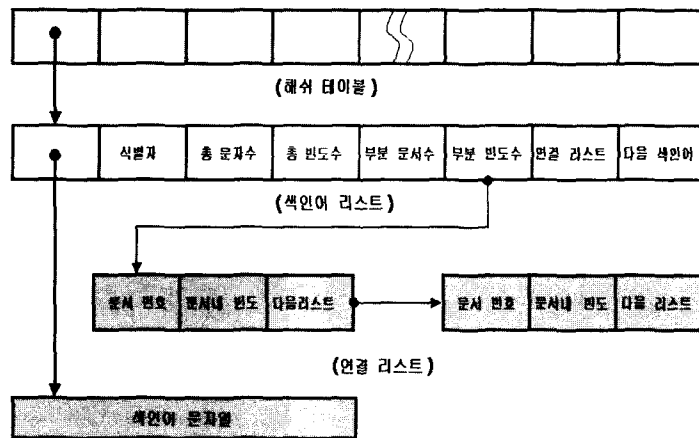
(텍스트 읽기, 어휘분석)
(부분 역파일 기록)
(전체 역파일 병합)

여기서 c 는 부분 역파일(chunk)의 수를 말하며 $c = \lceil I/(M-L) \rceil$ 이다. 또 b 는 chunk당 평균 색인어 수이다.

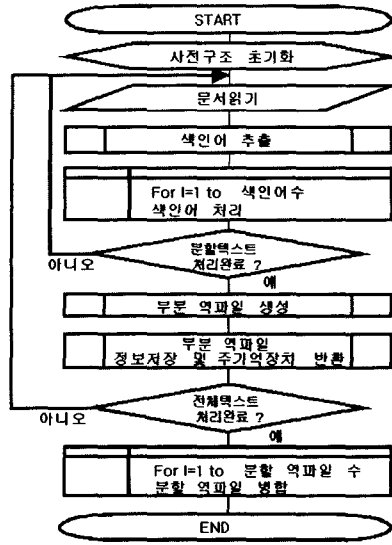
3.2 개선된 색인기법에 의한 색인시스템 설계 및 구현

색인작업을 위해서는 주기억장치와 보조기억장치에 필요한 여러 가지 자료구조를 정의해야 한다. 먼저 메모리에 유지해야 할 자료구조는 사전 구조로서 빠른 접근을 위해 해쉬테이블로 구성한다. 여기에 필요한 정보는 색인어, 색인어 식별자(번호), 색인어를 포함하고 있는 총 문서 수, 색인어별 총 출현 빈도수, 색인어를 포함하고 있는 부분 문서 수, 색인어별 부분 출현 빈도수, 및 문서번호와 문서별 출현 빈도수의 쌍으로 그림 7과 같이 자료구조를 유지한다. 이 자료구조는 주기억장치에 존재하여 분할된 텍스트에 대한 부분 역파일을 생성할 때까지 유지되고, 부분 역파일을 생성한 후에는 연결리스트만 개방하여 재사용이 가능하도록 한다.

이 자료구조를 이용하여 역파일을 생성하는 과정은 그림 8과 같다. 즉, 하나의 문서를 읽고 어휘분석을 하여 문서에 나타난 색인어를 추출한 후, 색인어와 빈도수에 대한 정보를 색인어별 처리과정으로 넘겨준다. 이 색인어별 처리단계에서 색인어는 해쉬함수에 의해 해쉬테이블에 저장될 위치로 변환한다. 해쉬테이블의 해당 위치에서



<그림 7> 해쉬테이블을 이용한 사전구조



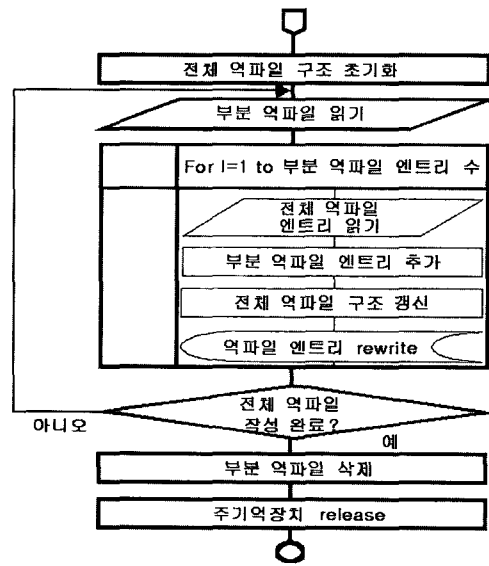
〈그림 8〉 역파일 생성과정

색인어를 비교하여 추출한 색인어가 없을 경우에는 색인어와 색인어 식별자를 색인어 리스트에 추가하고, 색인어에 대한 출현 빈도수와 문서 수를 누적한 후 문서번호와 문서별 출현 빈도수를 연결리스트에 추가한다. 이미 색인어 리스트에 색인어가 존재하는 경우에는 색인어에 대한 출현 빈도수와 문서 수를 누적하고, 문서번호와 문서별 출현 빈도수를 연결리스트에 추가한다.

분할한 텍스트를 모두 처리한 다음, 부분 역파일을 생성한다. 부분 역파일 생성은 해쉬테이블을 문서번호순으로 운행하면서 해당 색인어의 연결리스트를 참조하여 부분 역파일 엔트리를 생성하고, 이것을 부분 역파일에 추가한다. 부분 역파일 엔트리를 생성한 후에는 연결리스트를 개방하여 재사용이 가능하게 한다. 또한 부분 역파일 엔트리에 대한 정보를 별도의 파일로 관리한다. 여기에는 색인어 식별자, 부분 역파일에서의 시작과 끝 위치를 저장한다. 마지막으로 부분 역파일

을 모두 생성한 후 전체 역파일로 병합하기 위해 필요한 정보를 갱신한다. 즉, 색인어별 총 출현 빈도수와 총 문서 수를 누적시킨다.

모든 분할 텍스트의 역파일을 생성한 후, 부분 역파일 수만큼 전체 역파일에 병합한다. 이를 위해서는 먼저 부분 역파일 생성시 만들어진 색인어별 총 문서수와 총 출현빈도수를 참조하여 전체 역파일의 구조를 생성하고, 이 구조에 따라 전체 역파일 엔트리를 수용할 수 있는 빈 파일을 생성해야한다. 이 역파일 구조에 저장할 내용은 각 색인어별로 역파일 엔트리의 총 길이와 전체 역파일에 저장되어야 할 위치이다. 또 부분 역파일이 병합될 때마다 각 역파일 엔트리에 추가된 문서번호와 빈도수 쌍의 수를 누적할 곳이 필요하다. 이 누적정보는 다음 부분 역파일을 처리할 때 해당 색인어 엔트리의 시작 위치이다. 이 부분 역파일의 병합과정은 그림 9에 나타내었다.



〈그림 9〉 부분 역파일 병합과정

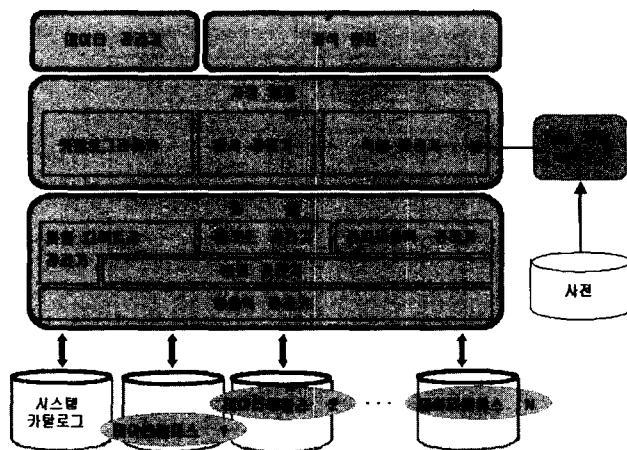
4 개선된 색인기법의 적용사례

4.1 적용시스템의 소개

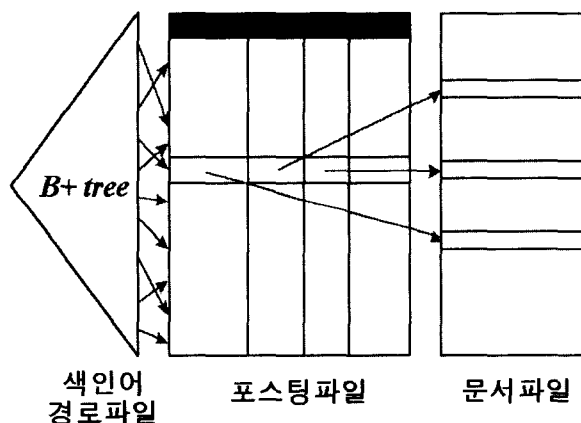
제안한 색인기법을 적용할 대상은 KRISTAL-II(Korea Research Information of Science and Technology Access Line-II)라는 정보검색시스템으로 연구개발정보센터(Korea R&D Information Center)가 과학기술정보의 효율적인 유통을 위하여 '93년부터 개발하여 왔다. 순수 독자적인 국내 기술로 개발된 KRISTAL-II는 한글, 한자 및 영문이 혼용된 문서를 효율적으로 저장·검색할 수 있으며, 과학기술정보유통체제의 기본시스템으로 활용되고 있을 뿐만 아니라, 국가전자도서관사업 등 대형과제의 검색시스템으로 활용되고 있으며, 기타 정부산하 여러 연구소, 대학 등에서 사용함으로써 많은 필드 테스트를 통해 입증받고 있는 시스템이다(연구개발정보센터

1995; 연구개발정보센터 1997; 연구개발정보센터 1998; 전산원 1998). 그림 10은 KRISTAL-II의 구성도를 보여주고 있다.

본 논문에서는 KRISTAL-II의 색인시스템에 제안한 색인기법을 적용하고자 한다. 그림 11은 KRISTAL-II의 색인화일 구조이며, [표 6]은 색인을 위한 KRISTAL-II의 API들이다. 그림에 나타난 바와 같이 색인화일은 색인어 경로화일과 포스팅화일로 구성되어 있다. 색인어 경로화일은 사용자의 질의에 나타난 단어를 효율적으로 탐색할 수 있도록 전체 문서 화일에 출현한 색인어들을 B+ 트리구조로 관리한다. 포스팅화일은 색인어들이 출현한 문서들에 대한 정보를 저장하는 화일로서, 색인어마다 하나의 엔트리를 갖는다. 포스팅화일의 엔트리에는 색인어가 출현한 문서 식별자와 사용자 질의의 단어간 근접도 연산을 지원하기 위해 문서내의 색인어 위치정보를 저장한다. 이 위치정보는 문서내의 단어번호로 표현



〈그림 10〉 KRISTAL-II 구성도



〈그림 11〉 KRISTAL-II의 색인파일 구조

되며, 색인어 추출시스템에 의해 자동적으로 부여된다.

4.2 KRISTAL-II 색인 API를 이용한 색인시스템 개발

[표 6]의 API를 이용하여 개발한 색인시스템은 실험결과를 비교하기 위해 세 가지 색인기법을 적용하여 구현하였다. 즉, 주기억장치를 이용한 색인

방법, 보조기억장치를 이용한 색인방법, 및 본 논문에서 제안한 개선된 텍스트분할 역파일 색인기법을 적용하여 색인시스템을 구현하였다.

구현한 색인시스템은 주 프로그램에서 색인기법을 선택하여 색인할 수 있도록 모듈화하였다. 각 모듈별 기능은 다음과 같다.

- 1) 주기억장치를 이용한 역파일 색인
해쉬테이블을 사용하여 데이터베이스내의 각 색인어에 대한 사전을 구축하고, 색인어에 관련

〈표 6〉 KRISTAL-II 데이터베이스 색인을 위한 관련 API

함수명	기 능
KNL_Initialize()	KRISTAL-kernel 초기화
KNL_Terminate()	KRISTAL-kernel 종료
KNL_FormatVol()	지정된 디바이스에 새로운 볼륨을 format
KNL_MountVol()	물리적 볼륨을 mount하고 볼륨 ID를 return
KNL_DismountVol()	지정한 볼륨을 dismount
KNL_CreateFile()	주어진 볼륨에 새 파일을 Create
KNL_OpenFile()	지정한 파일을 Open하고 open file number를 return
KNL_CloseFile()	지정한 파일을 Close
KNL_CreateIndex()	empty index file을 Create
KNL_OpenIndex()	index file을 open하고 open file number를 return
KNL_CloseIndex()	index file을 close
KNL_BeginLoadEntities()	bulk index loading 시작
KNL_LoadEntity()	index file에 index entity <key, element>를 load
KNL_EndLoadEntities()	bulk index loading 종료

된 정보는 linked list로 연결하였다. 색인을 시작하여 텍스트의 모든 문서를 처리하면, 색인어 정보를 저장하고 있는 linked list를 traverse하여 색인어별로 역파일 엔트리를 생성한다. 이 역파일 엔트리를 역파일에 추가함으로써 색인작업을 종료한다. 함수명은 UsingMemory()이고, 이 함수의 전달인수는 텍스트명과 색인파일명이다.

2) 보조기억장치를 이용한 역파일 색인

색인작업을 시작하면 텍스트의 각 문서로부터 색인어별 관련된 정보, 즉 텍스트에 나타나는 색인어 식별자 t , 문서번호 d , 출현빈도 f_d 로 구성된 $\langle t, d, f_d \rangle$ triple을 추출하여 순차적으로 임시 파일에 저장한다. 모든 문서로부터 색인어 추출이 완료되면 임시파일을 external merge sort하여 용어순으로 정렬한다. 이 정렬된 임시파일을 순차적으로 읽어서 색인어별로 역파일 엔트리를

생성한다. 생성된 역파일 엔트리를 역파일에 모두 추가하면 색인작업이 완료된다. 함수명은 UsingSort Based()이고, UsingMemory() 함수와 마찬가지로 이 함수의 전달인수는 텍스트명과 색인파일명이다.

3) 개선된 텍스트분할 역파일 색인

주기억장치를 이용한 역파일 색인방법과 마찬가지로 해쉬테이블을 사용하여 데이터베이스내의 각 색인어에 대한 사전을 구축하고, 한번에 처리할 텍스트를 분할한다. 분할된 텍스트의 색인정보를 추출하여 linked list로 연결한다. 분할 텍스트의 색인정보 추출이 끝나면 색인어 정보를 저장하고 있는 linked list를 traverse하여 색인어별로 부분 역파일 엔트리를 생성한다. 이 부분 역파일 엔트리를 부분 역파일에 추가한다. 이때 전체 역파일을 생성하는데 필요한 정보를 동시에 수집

〈표 7〉 실험을 위한 H/W

분 류	사 양	비 고
모델명	Center 2000E	
CPU 종류	SuperSPARC 60MHz(sun4d)	
CPU 갯수	4	
Main 메모리	512MB	
Disk 용량(Formatted)	59GB	
OS	Solaris 2.5.1	

한다. 분할 텍스트별로 부분 역파일이 끝나면 수집한 정보를 이용하여 전체 역파일 구조를 생성하고, 부분 역파일을 전체 역파일에 병합시켜 색인작업을 완료한다. 함수명은 UsingAdvanced() 이고, 이 함수의 전달인수는 UsingMemory() 함수와 마찬가지로 텍스트명과 색인파일명이다.

4.3 실험 환경 및 Test Collection

구현한 색인시스템을 실험하기 위해 사용한 시스템은 Center 2000으로서 상세한 사양은 [표 7]과 같다.

또한 색인시스템의 성능 평가를 위해 사용한 test collection은 연구개발정보센터가 보유하고 있는 외국 저널 서지사항 및 초록 635,964건

(1,074,506,564 bytes \approx 1 GByte)을 대상으로 하였다. 본 실험에서는 색인 대상자료의 크기에 따라 색인기법별 처리시간을 측정하는 것이므로, 이 test collection을 [표 8]과 같이 나누어 실험하였다.

4.4 실험 결과

본 실험에서는 [표 7]에 보여준 사양의 Center 2000을 통해 [표 8]의 Test Collection 분류표에 따라 각 색인기법별 색인시간을 측정하였다. [표 9], [표 10] 및 [표 11]은 각각의 색인기법별, test collection별로 각 단계별 소요시간을 나타낸 것이다. 실험을 통해 알 수 있듯이 주기억장치를 이용한 역파일 색인의 경우 색인 속도는 가장 빠

〈표 8〉 test collection의 분류 및 색인정보

document	Unique Term	Total Index Term
1,000건	13,890	188,024
10,000건	51,315	1,721,955
100,000건	188,476	16,061,110
635,964건	503,344	99,397,347

〈표 9〉 주기억장치를 이용한 색인기법 실험결과

Sample Data 진행단계	1,000건	10,000건	100,000건	635,964건
색인어 추출	11	95	946	처리불가
색인어 정렬	1	3	16	
B+tree 생성	15	21	184	
총 소요시간	0.3분	2분	19분	

〈표 10〉 보조기억장치를 이용한 색인기법 실험결과

Sample Data 진행단계	1,000건	10,000건	100,000건	635,964건
색인어 추출	12	137	676	5,359
색인어 병합-정렬	280	1,173	9,616	64,862
(알파벳순 정렬)	1	3	61	284
B+tree 생성	8	31	268	1,427
총 소요시간	5분	23분	3시간	20시간

〈표 11〉 개선된 텍스트분할 색인기법 실험결과

Sample Data 진행단계	1,000건	10,000건	100,000건	635,964건
부분 역파일 생성	12	102	1,040	6,049
부분 역파일 병합	3	13	114	701
B+tree 생성	9	47	206	1,251
총 소요시간	0.4분	3분	23분	2시간 14분

르지만 주기억장치의 한계성 때문에 대용량 문서에 대한 색인 알고리즘으로는 적합치 않다. 그러나 소형시스템에서는 아주 좋은 성능을 발휘할 수 있다. 한편 보조기억장치를 이용한 색인기법은 주기억장치의 한계를 뛰어 넘어 대용량 문서

의 색인이 가능하다. 다만 색인속도에 있어서 주기억장치를 이용한 색인기법에 비해 거의 10배 정도의 차이가 난다. 따라서 보조기억장치를 이용한 색인기법도 대용량 문서의 색인방법으로는 부적합하다.

본 논문에서 제안한 개선된 텍스트분할 색인방법은 속도면에서 주기억장치를 이용한 색인방법보다 약간 떨어지지만, 대용량 문서의 색인이 가능하면서 짧은 시간 내에 색인을 완성할 수 있으므로 아주 적절한 색인기법이라 할 수 있다.

5 결론 및 향후 계획

색인은 정보검색시스템에서 사용자의 질의에 대한 검색결과를 제공하기 위한 매우 중요한 기능이다. 본 논문에서는 기존의 여러 색인방법들에 대해 연구하고, 이들 색인방법의 특성을 비교 분석하여 문제점을 도출하였다. 기존의 색인방법들은 대부분 소요자원의 최소화를 기반으로 한 색인시간 단축을 실현하는데 중점을 두었다. 대부분의 색인방법들이 압축을 통한 시스템자원 절감을 하였고, 압축된 자료의 입출력에서 얻어지는 시간절감의 효과를 나타내었다. 이러한 압축 방법에 의한 색인은 많은 정보를 필요로 하는 정보검색시스템의 색인방법으로서 유연성이 떨어진 다. 또한 원시자료를 두 번 이상 읽고 파싱을 해야 하는 색인기법의 경우에는 더욱 심각한 문제가 발생할 수 있다. 즉, 색인어 추출단계에서 형태소 분석기를 사용하거나, 물리적으로 다른 시스템에

존재하는 텍스트를 색인한다면 네트워크 트래픽을 고려해야 하기 때문에 심각한 색인시간 지연의 원인이 될 수 있다.

본 논문에서 제안한 개선된 텍스트분할 색인기법은 대용량 문서를 적은 자원으로 빠르게 색인하는 기법이다. 색인을 위한 원시자료를 한번만 읽어서 처리하므로 형태소 분석기를 채용하거나 물리적으로 다른 시스템에 존재하는 자료를 네트워크를 통해 읽어들이어서 색인을 하더라도 색인시스템의 성능에 영향을 미치지 않는다.

제안한 색인기법은 실제 검색시스템에 적용시켜 실험을 통해 우수성을 증명하였다. 실험결과에서 알 수 있듯이 주기억장치를 이용한 색인기법은 색인속도는 가장 빠르지만 주기억장치의 범위를 초과하는 대용량문서의 색인에는 부적합하다. 또한 보조기억장치를 이용한 색인기법은 대용량 문서의 색인이 가능하지만 색인속도가 너무 느려서 대용량 문서의 색인방법으로 채용할 수 없다. 제안한 색인기법은 주기억장치를 이용한 색인기법에 비해 색인속도 면에서 뒤지지 않고, 대용량 문서의 색인이 가능하므로 아주 좋은 색인기법이다. 향후 압축기능을 추가하여 처리할 경우 지금보다 훨씬 효과적인 시스템으로 발전해 나아갈 수 있을 것이다.

참 고 문 헌

- 강무영, 이상구. 1999. 교육정보화를 위한 문서저장 및 정보검색에 관한 연구. 『한국정보교육학회 논문지』, 3(1): 1-12.
- 고미영, 정영미. 1999. P-norm 검색의 문헌 순위화 기법에 관한 실험적 연구. 『한국정보관리학회지』, 16(1): 7-30.
- 김철, 이승채, 김병기. 1996. 색인어 퍼지관계와 서열기법을 이용한 정보 검색 방법론. 『한국정보처리학회 논문지』, 3(6): 1160-1169.
- 사공철, 서경주. 1996. 정보검색 발전사. 『한국정보관리학회지』, 13(2): 19-37.
- 연구개발정보센터. 1995. 『정보검색을 위한 효율적인 저장시스템 개발(Ⅰ)』. [대전]: 동연구소. 과학기술부 특정연구과제 보고서.
- 연구개발정보센터. 1997. 『과학기술정보서비스를 위한 정보시스템 개발』. [대전]: 동연구소. 기관고유사업 최종보고서.
- 연구개발정보센터. 1997. 『전자도서관을 위한 KRISTAL-II 성능 개선』. [대전]: 동연구소. 현대정보기술(주)의 KRISTAL-II 사용권 및 Customizing에 대한 최종보고서.
- 연구개발정보센터. 1998. 『과학기술정보유통체제 구축사업(VIII)』. [대전]: 동연구소. 기관고유사업 최종보고서.
- 장재우, 이정기, 김강석, 이진수. 1998. 정보검색 응용을 위한 MIDAS 저장시스템의 확장. 『한국정보과학회 논문지(B)』, 25(4): 623-634.
- 전산원. 1998. 『과학기술통합정보시스템』. 정보통신부 과제 최종보고서.
- 전산원. 1999. 『국가주요전자도서관연계사업』. 정보통신부 과제 최종보고서.
- 정영미. 1993. 『정보검색론』. 개정판. 서울: 구미주역출판부.
- 한국전자통신연구소. 1995. 『MIDAS 정보검색시스템 개발』. [대전]: 동연구소. 최종연구보고서.
- Bell, T. C., Moffat, A. and Nevill-Manning, C. G. and Witten, I. H. and Zobel, J. 1993. "Data compression in full-text retrieval systems." *Journal of the American Society for Information Science*, 44(9): 508-531.
- Bell, T. C., Moffat, A. and Witten, I. H. and Zobel, J. 1995. "The MG retrieval system: compressing for space and speed." *Communications of the ACM*, 38(4): 41-42.
- Moffat, A. 1992. "Economical inversion of large text files." *Computing Systems*, 5(2): 125-139.
- Moffat, A. and Zobel, J. 1992. "Coding for compression in full-text retrieval systems." *Proc. 2'nd IEEE Data Compression Conference*: 72-81.
- Moffat, A. and Zobel, J. 1994. "Compression and fast indexing for multi-gigabyte text databases." *Australian Computer Journal*, 26(1): 1-9.
- Moffat, A., Zobel, J. and Klein, S.T. 1995. "Improved inverted file processing for large text databases." *Proc. 6'th Australian Database Conference*: 162-171.
- Moffat, A., Zobel, J. and Sacks-Davis, R. 1994.

- "Memory efficient ranking." *Information Processing & Management*, 30(6): 733-744.
- Persin, M. 1994. "Document filtering for fast ranking." *Proc. 17' th ACM SIGIR International Conference on Research and Development in Information Retrieval*: 339-348.
- Peter Wayner. 1999. *Compression Algorithms for Real Programmers*. San Francisco, California: Morgan Kaufmann Publishers, Inc.
- Salton, G. 1989. *Automatic Text Processing: The Transformation, Analysis, and Retrieval Information Processing & Management*. Addison Wesley.
- Sara Baase. 1993. *Computer Algorithms: Introduction to Design and Analysis*. Addison-Wesley.
- William B. Frakes. 1992. *Information Retrieval: Data Structure & Algorithms*. Englewood Cliffs, New Jersey: Prentice Hal.
- Witten, I. H., Moffat, A. and Bell, T. C. 1999. *Managing Gigabytes: Compressing and Indexing Documents and Images*. San Francisco, California: Morgan Kaufmann Publishers, Inc.
- Zobel, J., Moffat, A. and Sacks-Davis, R. 1992. "An efficient indexing technique for full-text database systems." *Proc. 18' th Conference on Very Large Databases*: 352-362.
- Zobel, J., Moffat, A. and Sacks-Davis, R. 1993. "Searching large lexicons for partially specified terms using compressed inverted files." *Proc. 19' th Conference on Very Large Databases*: 290-301.
- Zobel, J. and Dart, P. 1995. "Finding approximate matches in large lexicons." *Software-Practice & Experience*, 25(3): 331-345.