색인어 추출기 비교 및 분석

최임천*, 박순철*
* 전북대학교 컴퓨터 공학
e-mail cis1124@chonbuk.ac.kr

Indexing System comparison and analysis

Lim Cheon Choi*, Soon Cheol Park*
* Chonbuk National University

요 약

정보화 시대에 범람하는 정보들 중 원하는 정보를 빠르고 정확하게 검색할 수 있도록 도와주는 정보검색 시스템의 중요성이 대두 되고 있다. 정보 검색 시스템의 한 축을 담당하는 색인어 추출기는 보통 형태소 분석을 통하여 작성이 되지만 색인어 추출만을 위하여선 불필요한 작업들이 있는 것이 사실이다. 그래서 이 논문에서는 미리 정의된 색인어 리스트를 가진 사전을 이용한 색인어 추출 시스템을 제안하고 그에 맞는 데이터 구조들을 분석하여 성능 비교를 하였다.

1. 서론

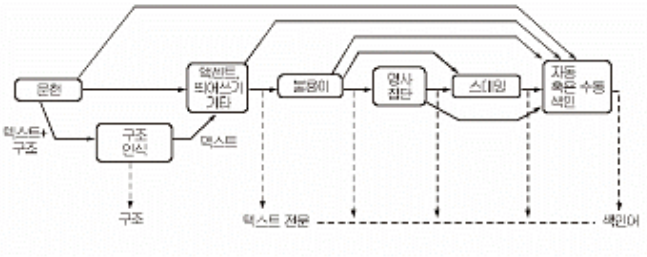
인터넷은 정보의 바다다. 지금 우리는 문서화된 많은 정보들이 인터넷을 떠도는 시대에 살고 있다. 이런 시기에 사람들이 원하는 정보를 빠르고 정확하게 찾을 수 있도록 도와주는 정보 검색 시스템이 필요한 것은 당연한 일이다. 정보 검색 시스템은 색인어 추출, 저장 시스템, 질의어 처리 시스템, 검색 시스템으로 구분되는데 검색 시스템의 효율을 높이기 위해서는 특히 색인어 추출 시스템이 중요하다. [1,2,3] 색인어 추출 시스템은 주어진 문서에서 내용을 나타내는 중요한 키워드를 찾아내는 시스템을 말하는데 정보 검색 시스템의 정확도, 속도 등의 성능에 많은 영향을 끼친다. 이러한 색인어 추출은 일반적으로 형태소 분석기[1,2]을 이용하여 구현되지만, 스테머[1,2], N-Gram [4,5]등을 이용하여 구현되기도 한다. 하지만 일반적으로 중요한 색인어로 구분되는 단어들이 명사임을 감안하면 어절의 모든 부분을 분석하여 색인어를 추출하는 것은 비효율적일 수 있다. 그래서 본 논문은 미리 정의된 명사형 사전에서의 단순 비교를 통한 부분, 혹은 완전 일치여부로 색인어를 추출하는 시스템을 제안하고 그에 맞는 데이터 구조가 어떤 것인지 알아보았다.

이 논문은 다음과 같이 구성된다. 우선 2 장에서 색인어 추출 시스템의 기본적인 구조와 특징에 대해서, 3 장에서는 제안하는 시스템의 구조와 특징에 대해서, 4 장에서는 그 결과를 분석하여 보고, 5 장에서 향후 계획과 추후 연구에 대해서 알아보기로 한다.

2. 색인어 추출 시스템

색인어 추출 시스템은 주어진 문서를 분석하여 그 문서의 중요한 키워드를 찾아내는 역할을 한다. 이는 정보 검색 시스템의 가장 중요한 부분 중의 하나인데 추출된 색인어를 바탕으로 검색을 하기 때문에 정보 검색 시스템의 정확성, 속도 같은 성능에 많은 영향을 주게 된다.

색인어 추출기의 일반적인 구조도는 (그림 1)과 같다.



(그림 1) 색인어 추출기 구조도

(그림 1)에서 보이는 것처럼 색인어 추출기는 먼저 문서를 입력받아 문서구조를 인식한다. 특수 문자, 띄어쓰기 등을 이용하여 어절을 구분하고 불용어 처리를 한다. 불용어란 색인어로 크게 의미가 없다고 여겨지는 단어로 관사 전치사 조사 접속사 등이 있다. 명사 집단과의 비교, 스테밍 작업을 통하여 색인어가 추출된다. 색인어를 추출하는 방법은 형태소분석기를 이용하는 방법, 스테밍 기법, N-Gram 기법 등이있다. 이러한 기법들 중 가장 많이 사용되는 것은 형태소 분석기을 이용하는 방법이다. 형태소란 의미 있는 어절의 가장 작은 단위로서 형태소 분석은 모든 의미 있는 단어를 분석하는 작업을 뜻한다. 실제로 색인어로 추출되는 단어들이 명사임을 감안할 때 모든 단어를 분석하여 색인어 추출을 한다는 것은 매우 비효율적일 수 있다. [1,2]

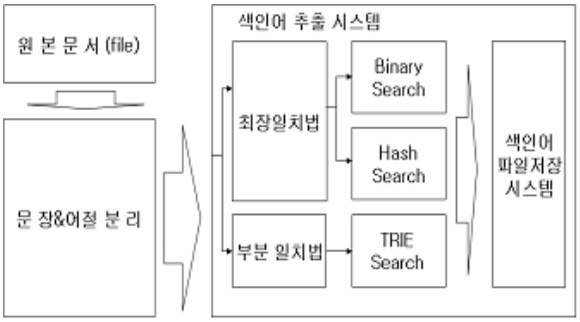
스테밍은 영어권에서 많이 사용되는 방법으로 단순한 스테밍(예, porter stemmer)기법을 이용하여 색인어를 추출하는 방법이다. 언어의 특성상 한국어는 변화형, 복합명사가 많기 때문에 스테밍은 부적절하다.

[1,2] N-Gram 은 단순히 입력 문장을 N 개의 문자씩 끊어서 색인어로 추출하는 시스템이다. 단순한 만큼 빠르고 성능 또한 우수하다. 그러나 의미 없는 상당수의 단어들이 동시에 추출되며, 너무 많은 수의 단어를 색인어로 추출해야한다는 문제점이 발생한다. [4,5]

3. 제안 시스템

이 논문에서는 제안한 색인어 추출 방법은 미리 구축한 명사형 사전을 이용하는 것이다. 즉 문서에 있는 문장을 단순한 어절로 분리하여 분리된 어절과 사전 내에 있는 단어와 비교하여, 부분 혹은 완전일치 여부를 검토한 후, 색인어를 추출하는 시스템이다.

제안한 시스템의 전체적인 구조는 (그림 2)와 같다. 시스템은 원본문서(file)에서 문서를 받아 들여 문장과 어절 분리한다. 나누어진 어절은 각각의 자료 구조가 자신에게 적합한 방법으로 어절을 분석하고 각각의 자료 구조는 자신의 검색 시스템을 이용하여 색인어를 추출하게 된다.



(그림 2) 제안 시스템 색인어 추출 시스템 구조도

3.1 문장과 어절의 분리

시스템에서 가장 먼저 하는 일은 원본 문서를 문장과 어절 단위로 적절하게 분리해주는 일이다. 문장과 어절의 분리가 부적절한 경우, 시스템상의 문제가 일어날 수도 있고 색인어를 추출함에 있어서 비효율적인 과정을 거칠 수 있다. 본 논문에서는 문서에서 문장을 추출해주는 부분과 문장에서 어절을 분리해주는 부분으로 알고리즘을 나누어 구성하였다. (그림 3)의 문서에서 문장을 추출하는 알고리즘이고 (그림 4)는 문장에서 어절을 추출하는 알고리즘이다.

```
infile : 입력 문서 파일
Temp : 임시 저장 공간
C : 임시 저장 공간 (char)
While ( (C = get_char(infile)) != End of File ) {
   if( C == 문장 특수 문자 ) Temp를 문장으로 선택
   else Temp += C;
}
```

(그림 3) 문장 추출 알고리즘

본 시스템에서는 문장 특수 문자(물음표, 느낌표, 쉼표...등)를 문장을 구분짓는 구분자로 사용하였고 어절을 끊어내는 어절 특수 문자로는 공백(space)만을 사용하였다.

```
Sen : 입력 문장
Temp : 임시 저장 공간
C : 임시 저장 공간 (char)
While ( (C = get_char(Sen)) != End of Sen) {
   if( C == 어절 특수 문자 ) Temp를 문장으로 선택
   else Temp += C;
}
```

(그림 4) 어절 추출 알고리즘

3.2 색인어 추출 알고리즘

본 논문에서 제안하는 색인어 시스템에서는 이미 구축되어 있는 명사사전의 데이터를 저장하고 검색하는데 세 가지 검색 알고리즘을 사용하였다. 세 가지 검색 알고리즘은 이분 검색, 해쉬 검색, TRIE 검색이다.

3.2.1 명사사전을 이용한 이분 검색 기법[4,6]

이분 검색은 이미 정렬된 배열에 대하여 log(n)의 평균 검색 길이를 갖는 매우 빠르고 단단한 검색 알고리즘이다. 이분 검색은 자료를 정렬이라는 방법으로 조직한다. 그래서 검색은 간단하지만 자료를 삽입하거나 삭제할 때는 자료의 구조를 깨뜨리지 않아야 하는 어려움이 있다.

사전 파일이 정렬되어 있다는 것이 전제되기 때문에 삽입 알고리즘은 단순한 파일 읽기와 읽어온 데이터의 저장으로 구현된다. 삽입 알고리즘은 [그림 5] 로 나타냈다.

```
S[] : 이진 탐색을 위한 저장 공간(string)
Temp : 임시 저장 공간(string)
File : 명사형 사전 저장 파일
count : 0 (int)
while ( (Temp = Read_line(file)) != EOF) {
  terms_list[count++] = Temp;
}
```

(그림 5)이분 검색 삽입 알고리즘

```
S[] : 명사형 사전이 적재되어 있는 배열
low : 0,  high : S[]의 크기, key : 찾고자 하는 값
Index B_Search (index low, index high, key) {
   index mid;
   if( low > high) return 0;
   else {
      mid = (low + high) / 2;
      if( key == S[mid]) return mid;
      else if( x < S[mid])
         return B_Search(low, mid - 1, key);
      else
         return B_Search(mid + 1, high, key);
   }
}
```

(그림 6) 이분 검색 알고리즘

검색된 키와 배열의 중간의 값을 비교한다. 같다면 검색 완료 중간의 값보다 키가 크다면 이분하여 작은 쪽의 배열에서 중간의 값보다 키가 작다면 이분하여 큰쪽의 배열에서 다시 검색을 반복한다. 구간의 크기가 0 이면 찾지 못한 것이자. (그림 6)은 검색의 알고리즘을 나타낸다.

3.2.2 명사사전을 이용한 해쉬 검색[4,6]

해쉬 검색은 해쉬 함수라는 것을 이용해 키를 만들고 그 키로 값을 찾을 수 있는 검색 알고리즘이다. 기존의 알고리즘들이 자료수 N 이 커짐에 따라 평균 검색 시간도 길어지는 데 비하여 해쉬 검색은 자료수 N 에 관계 없이 상수의 검색 길이를 가지는 매우 빠른 검색 방법이다. (그림 7)은 해쉬에서의 삽입알고리즘을 (그림 8)은 해쉬에서의 검색 알고리즘을 나타낸다.

```
H[] : HASH Table
Input : 입력 값
Hash_key : HASH 함수
Key : Hash_key(input);

while ( H[key] != null) key ++;

H[key]= Input;
```

(그림 7) 해쉬 삽입 알고리즘

```
H[] : HASH Table
Input : 입력 값
Hash_key : HASH 함수
Key : Hash_key(input);

while ( H[key] != null) {
   if( H[key] == Input) return key;
   else   key ++;
}

return -1; 찾는 값이 존재하지 않음
```

(그림 8) 해쉬 검색 알고리즘

3.2.3 이분 검색 & 해쉬 검색의 최장일치법

입력된 어절의 가장 끝 부분부터 완전일치로 검색을 한다. 검색되어 나온 결과가 없다면 한 글자를 잘라서 검색을 하게 된다. (그림 9)은 최장일치법 알고리즘을 사용한 예이다

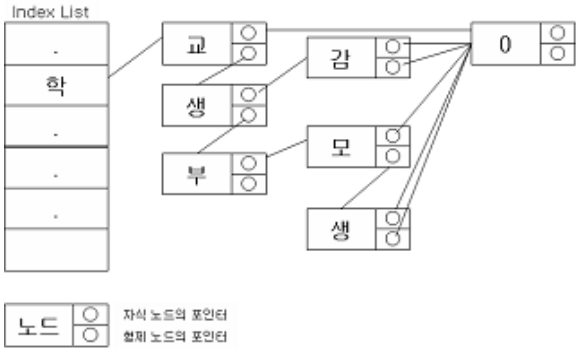
```
ex) 입력 : 전북대학교에서

1. 전북대학교에서 >> 검색 결과 X
2. 전북대학교에  >> 검색 결과 X
3. 전북대학교    >> 검색 결과 Y >> 색인어 추출
```

(그림 10) 최장 일치법알고리즘의 사용 예

3.3.3 명사사전을 이용한 TRIE 기법[4,6,7]

하나의 key 값의 일부분만을 사용하여 분기하는 tree 구조이다. 보통 하나의 노드는 m 개의 포인터로만 구성된 1 차원 배열이며 배열의 각 원소의 주소는 그 주소에 대응하는 숫자를 나타낸다. TRIE 의 높이는 key 값의 길이에 의해 결정된다. link 가 없는 노드의 key 값은 0 으로 나타내며 포인터가 모두 0 인 node (종속 tree 가 없는 node)는 나타내지 않음으로써 높이를 낮추고 기억공간을 절약할 수 있다. 탐색시 B+-tree 와 같이 leaf 노드까지 가야만 key 값을 찾을 수 있다. 하지만 찾는 key 값이 없을 경우 어느 level 에서든지 끝낼 수 있다. 따라서 key 값이 없는 경우의 탐색 효율이 좋다. 본 시스템의 트라이는 [그림 10]와 같은 데이터 저장 구조를 가지고 있다.



(그림 10) TRIE 의 데이터 저장 구조

인덱스를 저장하는 배열과 그 뒤의 요소들을 저장하는 노드들로 구성되고 그 관계는 Linked List 를 통하여 저장되게 된다. 이러한 구조에 의해서 본 시스템의 TRIE 삽입 알고리즘은 (그림 11)과 같다.

```
입력값 : Input
Point
Index[] : index를 저장하는 배열
C : 임시 저장 공간(char)

while ( (C = get_cahr(Input)) != End of Input) {
  if ( C == input의 첫문자) {
     if( !C:find(Index[])) index[]에 삽입
     Point = index[]의 노드
  }
  if( C:find(Point의 연결리스트) {
     Point = 연결리스트의 노드
  }else {
     C 노드 추가
     Point = C 노드
  }
}
```

(그림 11) TRIE 삽입 알고리즘

```
입력값 : Input
Point
Index[] : index를 저장하는 배열
C : 임시 저장 공간(char)

while ( (C = get_cahr(Input)) != End of Input) {
  if ( C == input의 첫문자) {
     if( !C:find(Index[])) return -1 : 찾는 값 없음
     Point = index[]의 노드
  }
  if( C:find(Point의 연결리스트) {
     Point = 연결리스트의 노드
  }else return -1 : 찾는 값 없음
}
```

(그림 12) TRIE 검색 알고리즘

(그림 12)으로 보여진 TRIE 의 검색알고리즘은 검색

하고자 하는 키값을 한 문자씩 자르고 인덱스에서부터 일치하는 노드가 있는지 탐색하는 형식으로 구현된다.

4. 실험 및 결과 분석

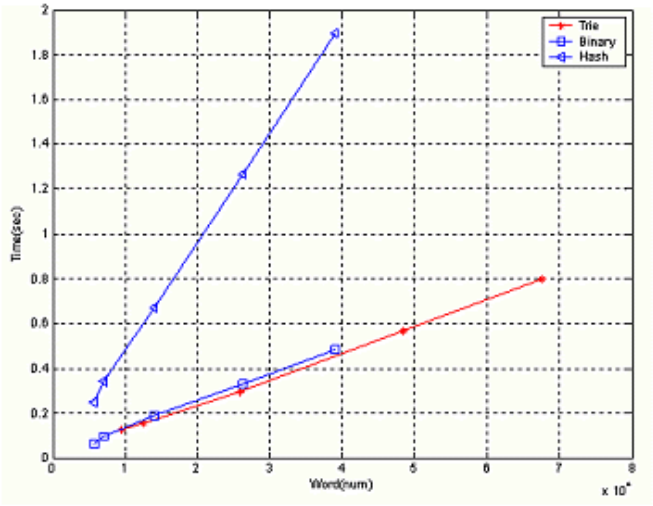
본 시스템은 한국어 15 만개의 명사형 단어를 수록한 자료구조 에서의 검색을 행하였으며 Windows 2003 환경의 Visual Studio 6.0 을 이용하여 구현되었고 작성된 언어는 C 언어 이다. 일반적으로 색인어 추출 시스템은 많은 양의 문서에서 빠르고 정확하게 색인어를 추출해야 하므로 실행시간과 추출된 색인어의 수에 초점을 맞추어 실험을 하였다.

주제 / 자료 구조	유비쿼터스	미들웨어	블루투스	애플리케이션	와이브로
TRIE (추출 단어)	48537	12554	9668	67585	25824
Binary(추출 단어)	26312	7000	5723	39051	14116
HASH (추출 단어)	26312	7000	5723	39051	14116

<표 1> 데이터 구조에 따른 주제별 추출 단어수

실험에 사용한 테스트 셋은 유비쿼터스, 미들웨어, 블루투스, 애플리케이션, 와이브로 이 5 개의 주제로 뽑은 기사들의 집합으로 원본문서로 사용하였다.

정리되어 있는 결과표에 보면 각각의 문서들의 파일의 크기(kbyte), 문장 분리 알고리즘에 의해 나올 문장의 수(개) 각 자료구조를 이용하여 색인어를 추출하였을 때 걸린 시간 (초) 추출된 단어의 수(개)의 결과가 정리되어 있다.



(그림 13) 데이터 구조별 성능 비교

5. 결론 및 향후 방향

정보검색 시스템에서 색인어 추출시스템은 가장 중요한 기능을 담당하는 부분이다. 대부분의 색인어 추출 시스템이 형태소 분석을 이용하고 있고 그 형태소 분석에 의해서 추출되는 색인어들도 결과적으로는 명사형이라는 점에서 착안하여 색인어 추출의 과정을 버리고 미리 색인어를 사전에 정의해두고 부분 및 완전 일치법에 의하여 색인어를 추출하는 시스템을 제안하였다. 각 자료구조에 맞게 부분, 완전 일치법을 사용하였고 각각의 자료구조의 실행 속도와 추출되는 색인어의 숫자를 실험을 통하여 비교해 보았다.

본문에서는 언급하지 않았지만 이분 탐색에서는 사전의 정렬 시간이, 해쉬 탐색에서는 저장 시간이 필요하고 TRIE 의 구조에서는 입력되는 사전의 단어들에 대한 구조화의 시간이 필요하다.

그 모든 것을 감안하여 볼 때 이분 탐색은 역시 빠른 속도를 보여주었다. TRIE 는 복합 명사의 처리가 가능한 자료 구조였고 해쉬는 실행시간이 의외로 많이 걸려서 성능 면에서 세 가지 자료구조 중 가장 떨어졌다. HASH 함수의 알고리즘을 향상 시키고, TRIE 의 구조를 이중배열구조나 더 나은 구조로 구현한다면 적재 시간이나 공간 검색 시간을 줄일 수 있을 것이다. 불용어 처리를 통해서 시스템 전반적으로 속도가 향상될 수 있을 것이라 생각되고 이분 검색이나 HASH 의 경우 복합명사의 처리가 안 되는 문제와 TRIE 의 경우 메모리를 너무 많이 차지한다는 문제가 발생하였다. 복합명사처리에의 알고리즘을 TRIE 의 경우 구조적인 수정으로 많은 것이 해결되고 더 나은 시스템으로 바뀔 수 있을 것이라고 확신한다.

참고문헌

[1] 최성필,서정현,채영숙 “자동 색인을 위한 한국어 형태소 분석기의 실제적인 구현 및 적용”,정보처리학회논문지 2002

[2] 최재혁, “형태소 분석을 통한 한.영 자동 색인어 추출 시스템”, 정보과학회논문지, 1996

[3] 강승식,권혁일,김동렬, “한국어 자동 색인을 위한 현태소 분석 기능”,한국정보과학회, 1995

[4] Richard Neapolitan,Kumarss Naimipour 저 도경구 역,"Foundations of Algorithms Using C++ Psedocode", 2004

[5] 이준호, 안정수, 박현주, 김명호 “한글 문서의 효과적인 검색을 위한 n-Gram 기반의 색인 방법”, 정보관리학회 , 1996

[6] 이상호,박승수 공저, “C 언어에 의한 자료구조론”,1995

[7] 김철수, 배우정, 이용석 “이중 배열 트라이 구조를 이용한 한국어 전자 사전의 구축, 정보과학학회논문지, 1996

[8] 강승식, “한국어 형태소 분석과 정보 검색”,2002

[9] 조현양,최성필,최재황 “어절 분석 기반 형태소 분석 시스템 개발에 관한 연구”, 정보관리학회,2001

[10] 이승선,송주원,황규영,최기선 “TRIE 구조를 이용한 한국어 전자 사전을 위한 데이터 베이스 인덱스 구조”, 한국정보과학괴, 1994