이중 트라이를 이용한 한국어 단어 검색

김철수, 배우정, 이용석
전북대학교 전자계산학과

(The Searching of Korean Words Using Double Trie)

Cheol-Su Kim, Woo-Jeong Bae, Yong-Seok Lee
Dept. of Computer Science, Chonbuk National University

요약

한국어 정보처리를 효율적으로 수행하기 위해서는 단어의 검색시간을 최소화하여야 한다. 그러나 기존의 방법들은 단어의 삽입과 삭제가 불가능하거나 검색시간이 길다는 단점을 가지고 있다. 본 논문에서는 탐색시간을 최소화하기 위해서 이중 배열을 가지는 이중 트라이를 이용하여 음절 및 자소단위의 검색방법에 관하여 논의한다. 검색시간에 있어서는 음절단위의 방법이 자소단위의 방법보다 빠르지만 기억장소는 자소단위의 방법이 음절단위의 방법보다 효율적이다. 자소단위의 방법에서 하나의 트라이를 여러개로 분할하여 저장함으로써 기억장소를 절반으로 줄일 수 있어 기억장소를 보다 효율적으로 이용할 수 있다.

1. 서 론

컴퓨터 사용 범위의 확산과 더블어 편리하고 신속한 정보처리가 요구되고 있다. 컴퓨터를 이용한 정보 처리를 위해서는 전자 사전의 탐색이 필수적으로 수반되며, 탐색 시간의 최소화는 정보 처리 시간의 단축과 밀접한 관계가 있다. 따라서 신속한 단어 검색을 위해서는 효율적이고 적합한 사전 구성은 필수적이고 기본적인 요소라 할 수 있다. 이러한 신속한 검색을 위해 해싱방법, 단어 사용 빈도에 따른 계층적 구조, 리스트 구조, 트라이 구조등 다양한 방법들이 제안되었다 트라이[1,2] 구조는 신속한 검색이 가능한 구조이다.

트라이 및 디지털 탐색[3]은 키들 사이의 비교에 기초한 방법과 달리 10진수나 문자의 순서를 이용하여 표현하는 방법이다. 검색 과정은 주어진 단어에 대하여 단어 전체가 아닌 단어를 구성하는 문자 단위의 비교에 의하여 다음 분기할 위치를 결정하는 구조가 재귀적으로 적용된다. 이렇게하여 루트 노드에서 각각의 터미널 노드까지의 경로상에 존재하는 문자들이 하나의 단어를 이룬다. 이 방법은 길이가 다양한 자료들을 쉽게 처리할 수 있도록 해 줄 뿐만 아니라 키들의 갯수에 무관하게 키의 길이에만 의존하여 문자의 검색 시간이 결정되어 신속한 검색이 가능하다. 뿐만 아니라 단어들의 숫자가 동적인 경우에도 잘 적용된다는 장점을 가지고 있다.

그러나 기존의 방법들은 삽입과 삭제가 어렵거나 검색 속도가 느리다는 단점이 있다.

본 논문에서는 사전의 참조 시간을 줄이고 기억공간을 줄이기 위하여 배열을 이용한 이중트라이[4] 방법을 이용하여 우리 한글 특성에 맞는 자소 및 음절 단위의 한글 전자 사전 구성에 관하여 알아 본다. 하나의 이중 트라이로 구성시 단어 검색 시간에 있어서는 음절 단위 방법이 자소단위 방법보다 빠르지만, 기억 장소는 자소 단위 방법이 음절 단위 방법보다 효율적으로 사용한다. 자소 단위 방법에서 기억 공간을 보다 효율적으로 이용하기 위하여 트라이를 여러개로 분할하여 저장하므로써, 기억 공간을 절반정도로 줄일 수 있다. 자소 단위 방법과 기존 방법을 비교분석한 결과 기억 장소는 비슷하게 요구하면서, 빠른 검색을 나타냈다.

본 논문 구성은 다음과 같다. 2장에서는 이중 배열을 이용한 트라이에 대하여 소개하고, 3장에서는 한국어 사전 구성에 관하여, 제 4장에서는 실험 평가에 대해서 알아보고, 마지막으로 5장에서 결론을 논한다.

2. 이중 배열을 이용한 트라이

2.1 단일 트라이

빠른 검색을 위해 트라이 구조를 배열에 표현하며, 배열 공간을 효율적으로 이용하기 위한 방법의 하나로 디지털 검색 방법을 이용하여 이중 배열에 효율적으로 표현할 수 있는 방법을 제안했다[3,4].

디지털 탐색은 **M** = (S, I, g, s1, F),

S : state들의 유한 집합으로 양의 정수,

I : 입력 심볼들의 유한 집합,

g : goto function from S×I to S∪ (fail),

s1 : 루트 노드이거나 S 내의 초기 상태로 정수 1로 표현,

F : accept state 의 유한 집합(F ⊆ S)

의 5개의 항들로 정의되며, 디지털 탐색 트리에 관한 몇 가지 개념들을 다음과 같이 정의한다.

정의 1

. 분할노드 - 입력 심볼 t에 의해 상태 s1에서 s2로의 전이 함수(goto 함수)를 g(s1,t) = s2로 정의하면, g(s1, xa) = s_r 에서 단어들의 집합 K에 포함된 단어 xay에 대해 집합 K내의 다른 모든 단어들과 xay를 구분하기 충분하다면 s_r를 분할 노드(S_P)라 한다.

. 다중노드 - 초기 노드에서 분할 노드까지의 경로 상에 존재하는 노드(S_M),

. 단일노드 - 분할 노드에서 accept 노드까지의 경로 상에 존재하는 노드(S_I)로

$S_P = S_M \cap S_I$가 성립한다.

정의 2. $s_r \in S_P$ 이고 $s_t \in F$ 에 대해 goto 함수 g(s_r , x) = s_t 를 만족하는 문자열 x 를 단일 문자열이라 부르며, STR[s_r]로 정의한다.

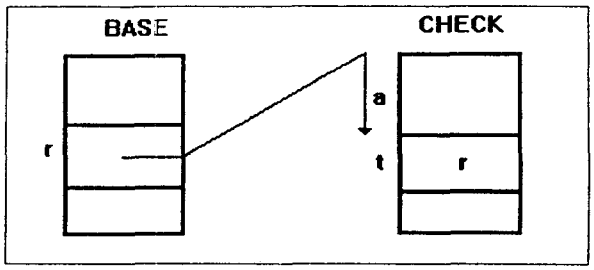
정의 3. 디지털 탐색 트리에서 다음 조건을 만족하는 단어 K에 대해 이중 배열과 TAIL을 정의하면,

. s_r and $s_t \in S_M$에 대해 g(s_r , a) = s_t에 대한 필요충분 조건은

BASE[r] + a = t, CHECK[t] = r.

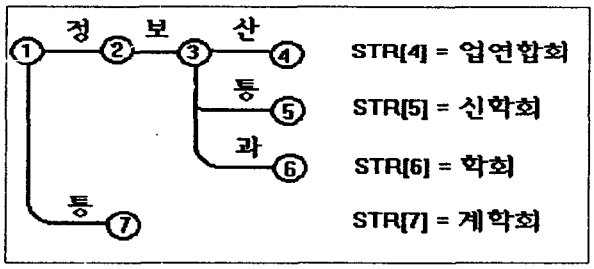
. $s_r \in S_p$에 대한 STR[s_r] = b_1.....b_m(m ≥ 0) 이면, BASE[r] < 0 이고, P = - BASE[r]에대해 TAIL[p]=b_1, TAIL[p+1] = b_2,..... TAIL[p+m-1] = b_m.

입력 심볼 a에 의해 상태 r에서 상태 t로의 전이 함수 g(r,a) = t 에서, 만약 전이가 존재하지 않으면 fail이 된다. 검색 머신 L(M)의 Accept 문자열 K는 L(M) = {y|g(1,y) = s, s ∈ F, y ∈ I* } 로 정의한다. BASE와 CHECK라는 두개의 배열 상에 goto 함수를 이용한 구조는 <그림 2.1>과 같다.



<그림 2.1> g(r,a)=t에 대한 이중 배열 구조

<그림 2.1>에서 g(r,a) = t 함수가 성립되기 위해서는 정의 3에서 처럼 BASE[r] + a = t, CHECK[t] = r 조건을 만족하여야 한다. 이러한 조건을 이용하여 트라이 구조의 자료들을 배열상에 조밀하게 표현할 수 있다. 이중 배열을 이용한 디지털 탐색 트리의 삽입, 삭제, 검색 알고리즘은 [3]을 참조하기 바란다. 단어 "정보산업연합회", "정보통신학회", "정보과학회", "통계학회"에 대한 단일 트라이를 음절단위로 표현하면 <그림 2.2>와 같다.



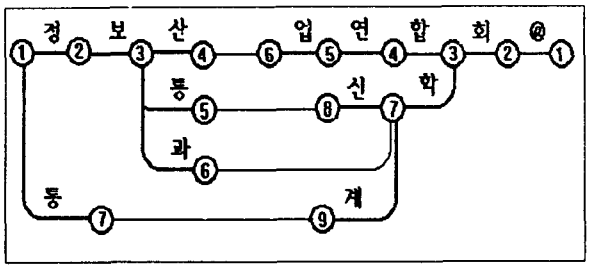
<그림 2.2> 단일 트라이의 예

<그림2.2>에서 다중 노드 ①에서 분할 노드 ⑦까지는 BASE와 CHECK 배열에 저장되고 단일 노드 STR[i] 내용들은 TAIL[ ] 배열에 저장된다. 노드 ①, ②, ③은 다중 노드로 여러개의 단어에서 공통적으로 중복되어 나타나 압축됨을 볼 수 있으며, 노드 ④, ⑤, ⑥, ⑦은 분할 노드이다.

2.2 이중 트라이

트라이의 기본적인 성질이 <그림 2.2>에서 처럼 단어

들의 공통된 전위부분(prefix)에 대한 압축이 가능하다는 점이다. 검색 과정에서 사용되는 단어 가운데 단어의 공통된 전위부분도 많이 나타나지만 공통된 후위부분(suffix)도 많이 나타난다. 이러한 공통 후위 부분들에 대해서도 압축을 시도하는 개념으로, 먼저 전위 부분에 대한 압축의 일환으로 앞절의 방법을 이용하여 다중 노드와 분할 노드에 대해서 트라이를 구성하게 되는데, 이를 LH-트라이라 부르고 LH-BASE, LH-CHECK 배열에 저장한다. 그런 다음 STR[s_i] 문자열(단일노드)들에 대해 역방향 문자열을 만든 후 이들 문자열에 대해 동일한 방법으로 또 하나의 트라이를 구성하게 되는데 이를 RH-트라이라 부르며, RH-BASE, RH-CHECK 배열에 저장한다. 그리고 LA[ ] 배열을 통하여 이 두개의 트라이를 연결하여 표현하는 방법[4]을 말한다. 이렇게 표현하므로써 공통된 전위부분은 LH-트라이에서 공통된 후위 부분은 RH-트라이에서 각각 압축을 가능하게 해 준다. 이처럼 두개의 트라이를 이용한 방법을 이중 트라이라 하며, <그림 2.2>의 단일 트라이를 이중 트라이 구조로 표현하면 <그림 2.3>과 같다.



<그림 2.3> 이중 트라이를 이용한 구조

<그림 2.3>에서 실선의 왼쪽이 LH-트라이를 오른쪽이 RH-트라이를 나타내며, 실선은 두 트라이의 연결을 의미한다.

3. 한국어 사전 구성

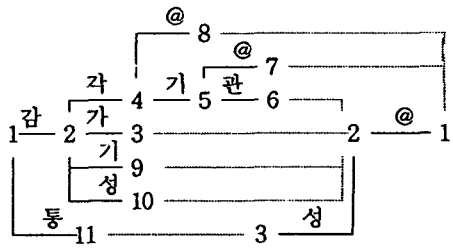
우리가 사용할 수 있는 11,172개(열린음절 19*21, 닫힌음절 19*21*27)의 음절을 이용하여 나타날 수 있는 모든 단어를 트라이로 표현하는 경우, 하나의 음절을 하나의 노드로 표현하는 음절 단위 방법의 방법과 하나의 음절을 초성, 중성, 종성으로(이하 자소 단위로 표기) 분리하여 각각의 자소가 하나의 노드를 구성하는 자소 단위의 방법이 가능하다.

한글 어휘 형태소 사전을 구성하는 단어들의 평균 음절수는 2.77음절로 2음절과 3음절이 대부분(81%)을 차지한다[7]. 하나의 음절을 각각의 자소로 분리하여 이중 배열에 저장하는 경우 하나의 단어를 검색하기 위한 평균 검색 비교 횟수는 8.31(2.77*3)이다. 가장 긴 단어의 길이는 11음절로 최악의 경우 33(11*3)번의 비교로 검색이 가능하다는 것을 알 수 있다. 이는 비교 횟수가 레코드 갯수가 아닌 하나의 단어에 대한 문자 비교 횟수이므로 매우 신속한 검색이 이루어 진다.

임의의 단어가 다른 단어에 내포되어 나타나는 경우(감,감기)를 구분하기 위하여 각 단어에 대한 종료 기호로 "@"을 사용한다고 가정하고, 몇 개의 단어(감가, 감각, 감각기, 감각기관, 감기, 감성, 통성)들에 대해서 음절 및 자소 단위의 이중 트라이를 구성해 보자

3. 1 음절단위 방법

한글을 저장하고 검색하기 위해 입력된 단어 각각의 음절을 하나의 노드로 표현하는 방법으로 완성된 하나의 음절에 하나의 디지털값을 부여하여 하나의 노드를 이룬다. 이에 따라 음절을 분리하거나 조합하지 않고 하나의 완성된 음절 자체가 하나의 노드를 구성하므로 음절에 대한 조작 연산 시간을 절약할 수 있다. 대상 단어들에 대해 이중 트라이로 표현하면 <그림 3.1>과 같다.



<그림 3.1> 음절단위 이중트라이

모든 음절을 사용하는 것은 아니므로 디지털값 부여를 용이하게 하기 위하여 KS 완성형을 이용한다고 하면, 각각의 음절에 1에서 2,350까지의 디지털 값을 부여하여 저장 규칙에 따라 배열에 저장한다.

그러나 완성형 코드에 포함되어 있지 않은 음절들을 다룰 수 없다는 문제점이 있다. 이러한 단점을 해결하기 위해 조합형 코드를 이용할 수 있으나 디지털값을 부여하는 과정에서 연산 시간이 증가할 수 있다.

임의의 단어를 검색하는데 걸리는 시간은 단어의 음절수에 비례하므로 최대 문자 비교 횟수는 12번(최장 검색 Key의 길이+@)으로 신속하게 할 수 있다. 이는 빠른

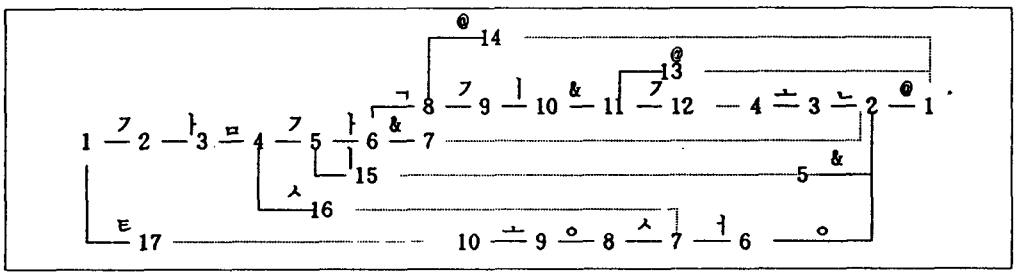
검색이 가능함을 잘 보여주고 있다.

3.2 자소단위 방법

저장, 검색의 기본 단위가 자소 단위로써, 하나의 음절을 저장하기 위해서는 음절을 자소 단위로 분리하여 분리된 자소 각각을 하나의 노드에 저장, 검색하는 방법이다. 자소단위의 취급을 함으로써 철자 검사가 가능하고 한글의 고유 특성을 살린다는 점과 모든 음절을 표현할 수 있으며, 자소단위로의 분리가 용이하다는 장점을 가지고 있다. 문제점으로는 하나의 단어를 저장하는데 각 음절을 3개의 자소로 분리하여 저장하므로 검색 과정에서 노드수가 음절 단위 방법의 3배이다. 따라서 이론적인 검색 시간은 음절 단위 방법의 3배가 된다. 이와 더불어 자소를 저장하기 위한 노드들도 증가하여 음절 단위 방법보다 많은 기억 장소를 요구한다. 음절 단위의 트라이 구조를 자소 단위의 트라이 구조로 표현하면 <그림 3.2>와 같으며 &은 열린 음절에서의 종성을 의미한다.

이 구조로 표현했을때, 레코드 정보를 고려하지 않고 단어만을 저장하는데 소요되는 공간을 그림으로 표현하면 <그림 5.1>과 같다.

이론적으로는 자소 단위 방법이 음절단위 방법에 비해 3배의 기억 장소를 요구하지만 실제로는 정 반대로 자소단위 방법이 음절 단위 방법의 2/3정도 밖에 소요되지 않음을 <그림 5.1>에서 처럼 알 수 있다. 이는 각 방법에서 사용하는 노드들에 부여하는 디지털값에 따라 다음 자소 및 음절을 저장하는데 요구하는 기억장소가 다르고, 이와더불어 할당된 공간들에 대한 사용율이 많은 차이가 있으며, 공통으로 사용되는 노드들의 숫자(압축율) 차이가 크기 때문이다. 할당된 전체 공간에 대한 사용율을 살펴보면 음절 단위 방법의 경우 240,700여개의 노드를 할당받고 할당받는 노드중 105,200여개 를 사용하여 43% 정도만을 사용하고 있으며, 자소단위 방법의 경우 152,000여개를 할당하여 99% 이상을 사용하고 있다. 사용된 노드수 만을 고려한다면 음절단위 방법이 자소단위 방법보다 적지만 음절단위 방법은 사용되지 않은 노드수가 많아 정 반



<그림 3.2> 자소 단위의 이중트라이

4. 실험 평가

실험을 위한 시스템으로는 SPARC 1+를 이용하였으며, 입력 단어들은 동아출판사의 동아 새 국어사전[8]에 등록되어 있는 단어들을 대상으로 하였다. 80,000여개의 단어를 저장하는데 242,4000여개의 음절이 사용되어 하나의 단어당 평균 음절 길이는 3.03으로, 3.1절에서의 2.7음절보다 높게 나타났다. 이는 단어(어휘) 입력 과정에서 입력 어휘를 형태소 단위로 완전하게 정제하지 않은 상태로 입력된 어휘들이 존재하여 불필요한 음절들이 저장되었기 때문이다.

4.1 기억장소 효율성

각각의 배열 요소들을 4Byte씩 할당하여 하나의 트라

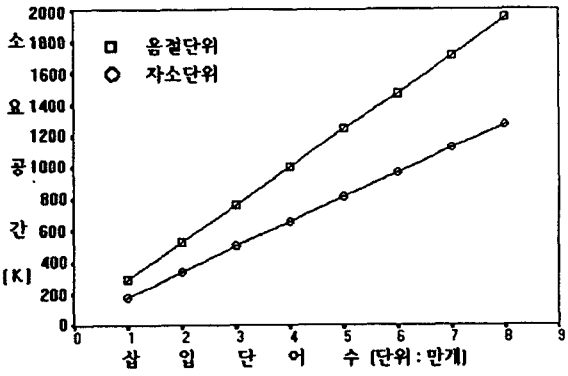
대 현상이 발생한다.

이와더불어 압축율을 보면 전체 242,400개의 음절을 저장하는데 음절 단위 방법의 경우 240,700 여개의 노드를 할당받으므로 99%(240,700÷242,400)로 압축되어 압축 효과가 거의 없으며 실제로 사용한 노드들에 대해서만 압축율을 고려한다면 43% (105,200÷242,400)까지 압축되었다. 자소단위 방법에서는 전체자소 수 727,000(전체음절*3)개를 저장하는데 152,300개의 노드를 할당받아 20.95%(152,300÷727,000)까지 압축되었다.

공통 후위부분에 사용되는 엔트리수를 보면, 8만개의 단어 삽입시 음절단위에서는 5,370여개이나, 자소 단위의 경우에는 13,280여개로 공통으로 사용되는 후위부분 엔트리수가 많은 차이가 난다. 공통으로 사용되는 엔트리수가 적음은 LH-트라이에 저장되는 노드수의 증가와 관계가 있다. 자소단위 방법에서 LH-트라이에서 할당된 노드수가

129,270 여개이고 음절단위의 경우는 228,600여개로 음절 단위 방법이 자소 단위 방법보다 0.768배의 노드수가 더 존재한다.

노드들이 차지하는 전체 공간과 삽입 단어수 증가에 따라 차지하는 기억 공간 사이의 관계를 보면, <그림 5.1>에서 처럼 자소 단위 및 음절 단위 방법 모두 직선의



<그림 5.1> 요구하는 기억장소의 크기

방정식에 거의 수렴함을 볼 수 있다. 삽입 단어수를 X라 하고 단어 삽입에 따라 차지하는 공간을 Y Byte라고 하면

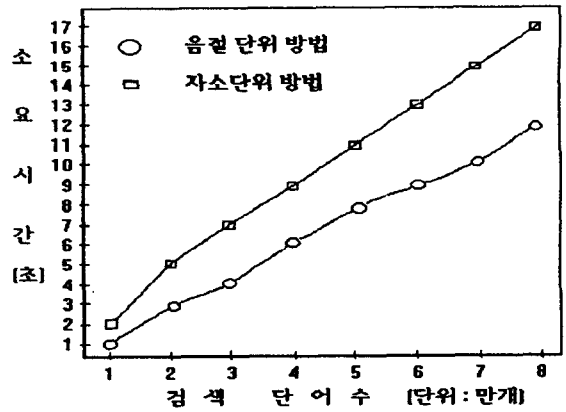
음절 단위 방법 : Y = 23.65 X + 53,700
자소 단위 방법 : Y = 15.53 X + 29,400

이라는 식이 성립된다. 직선의 방정식에서 보는 것처럼 하나의 단어를 추가로 삽입하는데 요구되는 공간은 원래의 예상과는 달리 음절 단위 방법이 8바이트 정도를 추가로 요구함을 볼 수 있다. 이는 동일 갯수의 단어를 저장하는데 실제로 사용되는 노드수만을 고려하면 음절 단위 방법이 적지만, 사용되지 않는 노드수가 많은 관계로 요구하는 노드수가 상대적으로 많기 때문이다. 11만개의 단어 삽입시 자소 단위 방법의 경우 1,737K Byte 정도, 음절단위 방법의 경우는 2.6M Byte 정도를 요구한다. 자소 단위 방법에서 전체 요구하는 공간의 크기를 줄이기 위하여 LH-트라이를 초성수(19)로 분할하여 저장하므로써 4Byte 씩 할당하던 배열을 2 Byte씩 할당하여 저장하므로써 절반정도로 줄일 수 있었다. 따라서 기억 공간에서는 음절 단위가 자소 단위보다 3배 정로를 요구하여 예상과는 정반대로 자소단위가 매우 효율적임을 알 수 있다.

4.2 검색 시간의 효율성

검색 시간에 대해서 살펴보면, 단어의 음절 길이를 K라고 하면 음절단위 방법의 경우는 K번, 자소 단위 방법의 경우는 3K번의 노드 검색으로 하나의 단어를 검색하므로 이론적으로는 자소단위 방법이 3배 정도의 검색 시간이 걸릴것으로 예상된다. 그러나 실제로는 <그림 5.2>에서 처럼 자소 단위 방법이 음절 단위 방법에 비해 1.5배 정도 소요된다. 이는 자소 단위 방법이 음절 단위 방법보다 노드 검색 횟수는 3배이지만 검색 단어들에 대한 디지털값 부여 과정에서, 자소 단위 방법에서는 한 음절을 자소 단위로 분리하는 과정이 용이한 반면, 음절단위 방법의 경우에는 완성형 코드에 대한 디지털값을 부여하는 과정에서의 산술 연산에 따른 시간이 소요되기 때문이다.

단어 삽입 과정의 소요 시간은 반대의 양상으로 음절 단위 방법이 적게 걸리는데 이는 다음 음절을 저장하기 위한 과정에서 음절 단위 방법에서 음절에 부여된 디지털 값이 자소 단위 방법에서 부여된 디지털값보다 커서 배열상에 비어 있는 공간이 상대적으로 많아 충돌이 적게 발생하여 삽입 시간은 적게 걸린다.



<그림 5.2> 소요되는 검색시간

음절단위의 리스트 구조를 이용한 방법[5]의 경우 14,168개의 단어를 검색하는데 SPARC 2 시스템에서 69.5초가 소요 되었다. 우리가 사용한 방법에서는 SPARC 1+ 시스템에서 8만개의 단어를 자소 단위로 검색하는데 17초가 소요되었다. 하나의 단어를 검색하는데 소요 시간은 [5]의 경우 $4.9*10^{-3}$초가 소요되었으며, 자소 단위 방법은 $2.125*10^{-4}$초가 소요되어 [5]방법보다는 23배 정도 빨랐다. 리스트 구조를 배열에 구현한 경우, 삽입 단어의 갯수가 증가하므로써, 형제 노드가 증가하게 되어 검색 과정에서 비교해야 하는 노드수도 증가하게 된다. 이에 따라 소요되는 검색 시간도 증가하게 된다. 즉, 검색 과정에서 트

라이의 성질을 완전히 발휘하지 못하고 있기 때문에 리스트 구조를 배열에 표현한 경우 80,000단어 정도를 저장한 후 검색한다면 하나의 단어당 평균 검색 시간이 $8.6*10^{-4}$ 초 보다 길어질 것으로 예상된다. 그러나 본 방법에서는 트라이 성질을 그대로 발휘하고 있으므로 검색 시간을 항상 일정하게 된다. 이는 리스트 구조를 배열에 이용한 경우보다 훨씬 효율적이라는 것을 알수 있다.

5. 결 론

한국어 정보 처리를 위해 필수적으로 요구되는 사전 참조에서 검색 시간의 최소화는 정보 처리 시간의 단축과 밀접한 관계를 가지는 중요한 요소중의 하나이다.

일반적으로 삽입 단어수의 증가에 따라 검색 시간도 증가하지만 트라이 성질을 이용하므로써 단어의 갯수에 무관하게 일정한 검색 시간을 가지며, 배열에 저장.검색하므로써 신속한 검색이 가능하다. 하나의 이중 트라이로 구성시 검색 속도는 음절단위 방법이 자소 단위 방법의 70%가 소요되어 음절단위가 효율적이었다. 기억 장소에서는 음절단위 방법이 자소단위 방법보다 0.5배를 추가로 요구할 뿐만 아니라 할당된 기억 장소도 비효율적으로 사용된다. 자소 단위 방법에서 주기억 장치의 차지 공간을 줄이기 위하여 어절의 첫번째 음절에 나타나는 초성수로 트라이를 분할하여 구성하므로써, 기억 공간을 절반 수준으로 줄였다. 이에 따라 예상과는 정반대로 분할한 자소 단위 방법에 비하여 음절 단위 방법이 3배의 기억 장소를 요구한다. 즉, 기억 장소는 자소단위 방법이 매우 효율적으로 이용하고 있다.

특히, 트라이 성질을 완벽하게 유지하므로 삽입 단어수에 무관하게 검색 시간이 항상 일정하므로 삽입 단어 숫자가 증가할 경우 더욱 유리하다.

문제점으로는 삽입 시간이 다른 방법들보다 많이 걸린다는 점이며, 검색 시간이 증시되어 음절단위 방법을 이용한다면 할당 공간을 효율적으로 이용할 수 있는 방법이 요구된다.

참 고 문 헌

[1] E. Fredkin, "Trie Memory", Comm. ACM, 3, pp. 490-500, 1960.

[2] D. E. Knuth, The art of Computer Programming, Vol.III, sorting and searching, pp.481-505, 1973.

[3] J. I. Aoe, "An Effienent Digital search algorithm by using a Double - Array Structure," IEEE Transactions on S/W Eng., Vol. 15, No. 9, Sept., pp. 1066-1077, 1989.

[4] K. Morimoto, H. Iroguchi and J. I. Aoe, "A Retrieval algorithm of Dictionaries by Using Two Trie structures," 일본 전자공학회 논문집 D-II Vol. J76-D-II No. 11, pp.2374-2383, 1944.

[5] 최기선외 5인, 한국어 철자 및 띄어쓰기 교정 시스템에 관한 연구(II), 최종보고서, 과학기술처, pp 23-35, July,1992.

[6] J.I.Aoe, K.Morimoto," An Efficient Implementation of Trie Structurs," S/W Prac. and Exp., Vol. 22, No. 9, Sep., pp.695-721. 1992.

[7] 강승식, "음절 정보와 복수어 단위 정보를 이용한 한국어 형태소 분석", 서울대학교 공학박사 학위 논문, 1993.

[8] 동아출판사 동아 새 국어사전, 동아출판사,1994.