論文 94-19-9-7

한글인식 후처리용 단어사전의 기억구조

正會員 金 商 雲* 非會員 青 木 由 直**

A Word Dictionary Structure for the Postprocessing of Hangul Recognition

Sang Woon Kim* *Regular Members*, Yoshinao Aoki** *Unregular Members*

要 約

한글인식 후처리에서 문맥정보의 저장구조는 인식율 및 인식속도를 결정짓는 중요한 요소이다. 단어사전의 형태로 문맥정보를 표현하기 위해서는 트라이(trie)를 주로 이용하지만, 기억공간 이용효율이 저조하다는 단점이 있다. 따라서 이 논문에서는 트라이의 장점을 유지하면서 공간효율을 향상시키는 기억구조를 제안한다.

한글은 조합문자이기 때문에 자모나 문자별로 기억시킬 수 있다. 그런데 자모단위로 기억시키면(P-모드) 검색시간은 빠르지만 공간효율이 나쁘고, 또한 문자단위로 기억시키면(C-모드) 공간효율은 좋지만 검색시간이 길어진다. 따라서 노드이용율과 분산율로 최적레벨을 선정한 다음, 입력단어의 시작자모부터 최적레벨까지는 자모 단위의 트라이로 기억시키고, 그 이상은 문자단위의 순차연결구조로 저장시켰다(H-모드). 6가지 단어집합에 대하여 실험한 결과, H-모드에서의 검색시간은 P-모드만큼 빠르면서, 공간효율은 C-모드와 같게되어 그 효용성을 확인할 수 있었다.

ABSTRACT

In the postprocessing of Hangul recognition system, the storage structure of contextual information is an important matter for the recognition rate and speed of the entire system. Trie in general is used to represent the context as word dictionary, but the memory space efficiency of the structure is low. Therefore we propose a new structure for word dictionary that has better space efficiency and the equivalent merits of trie.

* 明知大學校 컴퓨터工學科
Dept. of Computer Eng., Myong-Ji Univ.
** 北海道大學工學部情報工學科
Dept. of Information Eng., Hokkaido Univ.

論文番號 : 9423
接受日字 : 1994年 1月 19日

Because Hangul is a compound language, the language can be represented by phonemes or by characters. In the representation by phonemes(P-mode) the retrieval is fast, but the space efficiency is low. In the representation by characters(C-mode) the space efficiency is high, but the retrieval is slow. In this paper the two representation methods are combined to form a hybrid representation(H-mode). At first an optimal level for the combination is selected by two characteristic curves of node utilization and dispersion. Then the input words are represented with trie structure by P-mode from the first to the optimal level, and the rest are represented with sequentially linked list structure by C-mode. The experimental results for the six kinds of word set show that the proposed structure is more efficient. This result is based on the fact that the retrieval for H-mode is as fast as P-mode and the space efficiency is as good as C-mode.

I. 서 론

필기체 문자인식 시스템의 실용화를 위해서는 자유로운 필기방법과 사용문자에 대한 제한 등이 개선되어야 하나, 이를 위해서는 인식시간 및 인식율이 떨어지게 된다. 따라서 인식속도를 향상시키면서 유사한 문자나 흘려쓴 문자를 옳바르게 식별하기 위해서는 문맥정보(contexts)를 이용하는 후처리(postprocessing)가 필요하다[1].

후처리에서 가장 효율이 좋은 문맥정보는 해당문자의 앞뒤문자를 참조하는 단어수준의 정보이고, 이 정보의 효과적인 표현 및 이용방법은 사전참조(dictionary lookup)법이라는 연구보고가 있다[2]. 또한 단어사전의 기억구조는 디지탈탐색이 가능한 트라이(trie)가 적합한 구조로 알려져 있다[3].

트라이는 첫째, 공통접두사를 한번만 저장하면 될 뿐만 아니라 필요한 모든 부분패턴의 추출이 가능하고 둘째, 단어의 검색시간이 단어수에 관계없이 단어길이에 비례한다. 그리고 셋째, 다양한 길이의 단어들을 효율적으로 저장할 수 있으며 넷째, 다음 연결문자의 저장주소가 투명하게된다는 장점이 있는 반면, 공백포인터가 많기 때문에 기억공간의 이용효율이 저조하게 된다는 단점이 있다[4].

트라이의 단점을 제거하기 위하여 트라이의 노드들을 링크필드없이 순차적으로 저장하는 방법[5]과 트라이를 표현한 배열의 공백 부분을 압축하는 방법[6] 및 B-tree구조를 확장하여 문자별 검색이 가능하도록 하는 방법[7] 등이 제안되었다. 그러나 이 방법들은 검색효율은 양호하나, 삽입 및 삭제의 연산효율이 저조하다.

따라서 최근에는 입력문자수 만큼의 비트열을 사용하여 셋트되는 비트의 포인터만을 기억시키는 방법[8]과 트라이에서 분기가 없는 패스(다른 단어와 공유하는 문자가 없는 패스로서 SP(separate)절이라고 한다)를 한 노드로 압축하여 그 앞부분의 패스(다른 단어와 공유하는 문자들의 패스로서 MP(minimal prefix)절이라고 한다)분리하여 저장하는 방법[9] 등이 제안되었다.

이 논문에서는 한글인식 후처리에 이용할 수 있는 단어사전의 기억구조로서, 트라이의 장점을 모두 만족시키면서 기억공간 이용효율을 향상시킨 기억구조를 제안한다. 한글은 초, 중, 종성의 3성으로 구성된 조합문자이기 때문에 문자 단위로 기억시키는 방법과 3성의 자모단위로 기억시키는 방법이 가능하다. 그런데 문자단위로 기억시킬 경우 기억공간은 작지만 검색시간이 길고, 반면에 자모단위로 기억시킬경우 검색시간은 빠르지만 기억공간이 커진다. 따라서 효과적인 한글단어사전의 기억구조로 자모와 문자를 혼합하여 기억시키는 방법을 제안한다.

우선 사전의 검색효율 향상을 위하여 입력문자열을 자모단위로 기억시킨 다음, 자모를 저장하는 노드의 이용율과 분산율 특성곡선[10]을 이용하여 최적레벨을 선정하고, 최적레벨 이상에서는 분기가 발생하지 않는다는 점에 착안하여 문자단위의 연결 구조로 저장함으로서 공간효율을 향상시킨다. 이하, 제2장에서는 한글단어사전을 효율적으로 기억시키는 하이브리드 저장구조에 대하여 살펴보고, 제3장에서는 하이브리드 단어사전에 대한 검색, 삽입, 삭제 알고리즘을 기술한다. 그리고 제4장에서는 몇가지 단어집합에 대하여 실험한 결과에 대하여 고찰한 다음 제5장에서 결론을 맺는다.

II 한글단어사전의 저장구조

2.1 문자모드와 자모모드

한글은 초성자음, 중성모음, 종성자음의 3성으로 구성되는 조합문자이다. 따라서 한글단어사전의 저장구조는 그림1과 같이 문자노드 구조와 자모노드 구조를 생각할 수 있다.

(a) 문자노드

(b) 자모노드

그림 1. 문자노드와 자모노드
Fig 1. Character node and pheneme node

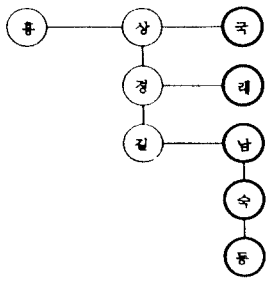
여기서 필드 ALTER와 NEXT는 트라이를 이진트리로 표현할때 각각 형제노드와 자노드를 위한 포인터이고, MARK는 단어의 끝을 나타내는 필드이다. 또한 C와 P는 각각 문자(초, 중, 종성)와 자모저장을 위한 필드이다.

문자노드를 이용한 저장구조를 C-모드(character mode)라 하고, 자모노드를 이용한 저장구조를 P-모드(phoneme mode)라 할때, 단어집합 W={홍길동, 홍길남, 홍길숙, 홍상국, 홍경래}에 대한 C-모드와 P-모드는 그림2와 같다.

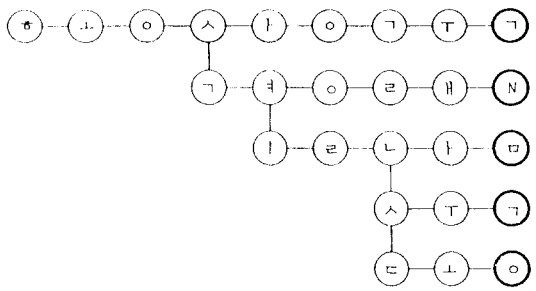
그림2에서 수평 선분은 다음문자를 나타내는 자노드 포인터에 해당하고, 수직 선분은 앞문자에 인접되는 형제노드의 포인터에 해당한다. 또한 굵은선의 노드는 단어의 마지막 문자노드를 의미한다.

그림2를 C언어로 구현할때 포인터 필드의 크기를 p바이트라고 하면, 문자노드를 위한 기억공간은 초, 중, 종성을 위한 3바이트와 MARK를 위한 1바이트 및 포인터 필드를 위한 2p를 합하여 4+2p바이트가 된다. 또한 자모노드의 기억공간은 자모저장을 위한 1바이트와 MARK 1바이트 포인터 필드 2p를 합하여 2+2p바이트가 된다. 따라서 그림2(a)를 구현하기 위한 기억공간 SPACE$_C$는 9×(4+2p)=36+18p 바이트이고, 그림2(b)를 위한 기억공간 SPACE$_P$는 26×(2+2p)=52+52p바이트이므로, 식(1)이 성립한다.

$$SPACE_C < SPACE_P \quad (1)$$



(a) 문자노드에 의한 저장(C-모드)



(b) 자모노드에 의한 저장(P-모드)

그림 2. 단어집합 W={홍길동, 홍길남, 홍길숙, 홍상국, 홍경래}의 저장구조
Fig 2. Dictionary structures for word set W={홍길동, 홍길남, 홍길숙, 홍상국, 홍경래}

또한 자음이나 모음 하나를 비교하는 시간단위를 t라고 하면 문자노드를 검색하는 시간단위는 3t가 되고, 자모노드의 시간단위는 t가 된다. 그림2(a)에서 '홍길동'을 검색하기 위한 검색경로의 길이는 7 이므로 최대검색시간 TIME$_C$는 7×3t=21t이고, 그림2(b)에서는 검색경로가 13으로 최대검색시간 TIME$_P$는 13×t=13t이다. 따라서 식(2)가 성립한다.

$$TIME_C > TIME_P \quad (2)$$

따라서 효율적인 한글단어의 기억구조는 고속검색이 가능한 P-모드와 공간이용율이 양호한 C-모드를 결합한 하이브리드(hybrid)구조임을 알 수 있다. 또한 기억공간을 줄이기위해서는 공통접두사(MP집)를 고려해야 함으로, 단어의 전반부는 P-모드로 기억시키고 후반부는 C-모드로 기억시켜야함을 알 수 있다.

2.2 최적레벨 선정

H-모드에서 문제가 되는 것은 단어의 몇번째 자모까지를 P-모드로 기억시킬것인가하는 트라이의 최적레벨(optimal level)의 선정이다. 이를 위하여 단어를 구성하는 자모열의 문자 레벨에 따라 어느 정도 분기가 발생하는가를 나타내는 함수로서 포인터 필드 이용율(utilization;fu)을 식(3)으로 정의한다.

$$f_U(i) = \frac{\text{레벨}i\text{까지}ALTER\text{가공백이아닌노드수}}{\text{레벨}i\text{까지의 전체노드수}} \quad (3)$$

여기서 fu(i)은 레벨 i에서 형제노드를 위한 포인터필드가 어느정도 이용되고 있는지를 나타내며, 분기가 많을수록 증가한다. 즉, 그림2(a)에서 레벨 2까지 노드수는 4이고, ALTER필드가 공백이 아닌 노드수는 2이므로, fu(2)=0.5이 되며, 이 값은 레벨 2에서 포인터필드가 50%만 이용되고 있음을 의미한다.

또한 입력단어에서 P-모드로 기억시키고 남은 나머지 문자열(이하 꼬리(tail)라고 이름한다)에서는 분기가 발생하지 않으므로 문자노드의 연결리스트구조로 기억시킨다. 이를 위하여 레벨 i에서의 노드 분산율(dispersion;fD)을 식(4)로 정의하며, 길이가 i보다 긴 패스가 없을 경우에는 fD(i)=1.0인 것으로 정의 한다.

$$f_D(i) = \frac{\text{레벨}i\text{까지}NEXT\text{가공백이아닌노드수}}{\text{길이가}i\text{보다긴패스수}} \quad (4)$$

식(4)는 레벨 i에서 연결리스트로 저장할 꼬리의 수를 나타내며, 식(4)의 값이 작을수록 기억시킬 꼬리의 수가 많아지고 따라서 기억공간이 증가한다. 즉, 레벨 2에서 길이가 2보다 긴 패스수는 5이고, NEXT 필드가 공백이 아닌 노드수는 3이므로, fD(2)=0.6이다.

저장구조의 효율을 높이기 위해서는 fu(i)가 큰값을 갖도록 최적레벨(이하 k_1이라 이름한다)을 선정해야 하고, 꼬리를 위한 기억공간을 최소화하기 위해서는 fD(i)도 큰값을 갖도록 해야 한다. 그런데 fu(i)는 i에 따라 증가하고 fD(i)는 i에 따라 감소하므로, fu(i)=fD(i)를 만족하는 i를 k_1으로 선정해야함을 알 수 있다.

2.3 하이브리드 저장구조

하이브리드 저장구조(이하 H-모드(hybrid mode)라고 이름한다)에서는 입력단어의 첫번째 자모부터 k_1번째 자모까지는 P-모드로 기억시키고, 나머지 k_1+1번째 자모부터 마지막 자모까지는 C-모드로 기억시킨다. 즉, $i \le k_1$에서는 검색효율이 양호한 P-모드를 이용하고 $i > k_1$에서는 공간효율이 양호한 C-모드를 이용함으로서, 전체적으로 기억공간을 축소하고 검색시간을 단축할 수 있도록 한다.

그런데 $i > k_1$의 C-모드에서는 분기가 더 이상 발생하지 않으므로, 그림1(a)의 문자노드에서 ALTER필드는 이용하지 않게된다. 따라서 C-모드를 ALTER필드를 제거한 문자노드의 연결리스트로 구현하면 검색시간을 증가시키고 않고 기억공간을 더욱 줄일 수 있다. 단어집합 W를 입력하였을때 최적레벨을 $k_1=6$로 한 H-모드는 그림3과 같다.

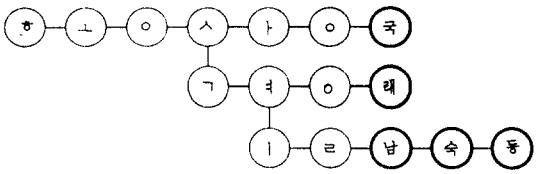


그림 3. 단어집합 W={홍길동, 홍길남, 홍길숙, 홍상국, 홍경래}의 H-모드

Fig 3. Hybrid structure for W={홍길동, 홍길남, 홍길숙, 홍상국, 홍경래}

그림3의 H-모드에서 W를 위한 기억공간 SPACEh는 11×(2+2p)+5×(4+2p)=42+32p 바이트이므로, 식(5)가 성립한다. 또한 '홍길동'을 검색하는 최대검색시간 TIMEh 은 8×t+3×3t=17t 이므로, 식(6)이 성립한다. 따라서 H-모드는 검색효율이 좋은 P-모드에 공간효율이 양호한 C-모드의 특성을 가미한 trade-off 저장구조임을 알 수 있다.

$$SPACE_C < SPACE_H < SPACE_P \quad (5)$$

$$TIME_C > TIME_H > TIME_P \quad (6)$$

즉, H-모드의 $i \le k_1$에서는 공통접두사의 MP절 길이를 증가시키는 만큼 탐색시간을 줄일 수 있으며, $i > k_1$에서는 연결리스트로 구현한 문자노드의 ALTER 필드 수 만큼의 기억공간을 줄일 수 있다. 그런데 k_1의 값을 크게하면 P-모드의 탐색특성은 좋아지나 기억공간이 증가하며, k_1을 작게하면 기억공

간효율은 좋아지나 순차검색에 의한 탐색 시간이 증가한다.

Ⅲ. H-모드에 대한 연산

3.1 검색연산

검색(find)은 가장 기본이되는 연산으로서, 검색할 단어를 구성하는 자모열에 해당하는 패스(path)가 사전에 존재하는가를 탐색하는 연산이다. 재귀적(recursive)형태로 구현한 검색알고리즘 h-find(p,i)은 〈알고리즘1〉과 같다. 단, p는 사전을 구성하는 노드의 포인터이고, i와 k는 각각 자모열 $w=x_1 x_2,\ldots,x_k$의 인덱스와 길이이며, k_1은 최적레벨이다. 또한 p->next와 p->alter는 p노드의 자노드와 형제노드 포인터이다.

〈알고리즘1〉 h-find(p,i)
단계 1. i>k이면 성공종료한다.
단계 2. p가 NULL이면 실패종료한다.
단계 3. $i\leq k_1$ 이면 다음을 수행하고,
3.1 p노드의 자모와 입력자모 x_i가 같으면 h-find(p->next, i+1)를 수행하고,
3.2 아니면 h-find(p->alter, i)를 수행한다.
단계 4. 아니면 다음을 수행한다.
4.1 꼬리 $x_i x_{i+1},\ldots, x_k$를 p노드로 부터 순차검색하여 성공하면 h-find(p, k+1)을 수행하고,
4.2 실패하면 p노드의 (k-i+1)번째 자노드 p'를 구하여 h-find(p', i)를 수행한다.

여기서 초기입력은 사전의 헤드노드를 가리키는 포인터 p=head와 검색할 단어 w를 구성하는 자소열의 인덱스 i=1이고, 출력은 검색에 성공하면 1을, 실패하면 0을 반환한다.

h-find()의 단계 3.은 디지탈탐색으로서 P-모드에 대한 검색이고, 단계 4.는 연결리스트의 순차검색에 해당하는 연산이다. 그런데 P-모드에서는 분기를 허용함으로서 i에 따라 형제노드가 증가하게되고, 검색은 넓이우선(breadth-first)탐색의 형태가 된다. 또한 $i > k_1$에서는 형제노드의 생성이 포화상태에 이르게되어 검색은 깊이우선(depth-first)탐색의 형태를 이루게 된다[11].

따라서 검색할 단어의 자모열 길이를 k라하고, 입력기호의 종류를 m, 사전의 전체 단어수를 n이라고 할때, 단계3. 의 최소 및 최대 수행횟수는 각각 $O(k_1)$와 $O(mk_1)$이고, 단계4. 의 최소 및 최대 수행횟수는 $O(k-k_1)$과 $O(m^{k-k_1})$이다. 따라서 검색연산 h-find()의 시간계산량은 최소 O(k)이고, 최대$O(m^k)$이다.

3.2 삽입연산

삽입(insert)은 주어진 단어의 자모열을 사전에 추가하는 연산으로서, 삽입알고리즘 h-insert(p,i)는 〈알고리즘2〉와 같다. 단, p는 사전을 구성하는 노드의 포인터 주소이고, i와 k는 자모열의 인덱스와 길이이며, k_1은 최적레벨이다. 또한 (*p)->next와 (*p)->alter는 *p노드의 자노드와 형제노드의 포인터 주소이다.

〈알고리즘2〉 h-insert(p,i)
단계 1. i≤k이면 *p노드의 부모노드에 종단기호를 저장한후 종료한다.
단계 2. $i<=k_1$이면 다음을 수행하고,
2.1 *p가 NULL이면 자모노드를 생성하여 *p로 하고 *p노드에 x_i를 저장한후 h-insert((*p)->next, i+1)를 수행하고,
2.2 아니면 *p노드의 자모가 입력자모 x_i와 같은지 조사하여 같으면 h-insert((*p)->next, i+1)를 수행하거나,
2.3 아니면 h-insert((*p)->alter, i)를 수행한다.
단계 3. 아니면 다음을 수행한다.
3.1 *p가 NULL이면 분자노드를 생성하여 *p로하고 *p노드에서부터 꼬리 $x_i x_{i+1},\ldots, x_k$를 저장한후 h-insert((*p)->next, i+3)을 수행하고,
3.2 아니면 꼬리 $x_i x_{i+1},\ldots, x_k$를 *p노드부터 순차검색하여 성공하면 h-insert(*p, k+1)을 수행하거나,
3.3 아니면 *p노드의 (k-i+1)번째 자노드 *p'를 구하여 h-insert(*p', i)를 수행한다.

여기서 초기입력은 사전의 헤드노드에 대한 포인터 p=&head 와 자모열의 인덱스 i=1이다.

h-insert()의 단계 1.에서 i가 k보다 크게되는 경우는 자모열을 모두 삽입한 경우에 해당한다. 그리고 단계 2.는 $i\leq k_1$의 자모열을 P-모드에 삽입하는 과정이며, 단계 3.은 $i>k_1$의 꼬리를 연결리스트형태로 삽입하는 과정이다.

입력단어를 삽입하기 위해서는 우선 삽입할 자모열 패스가 존재하는지를 조사하여(검색연산에 해당한다) 없을 경우에만 삽입하게 된다. 따라서 h-in-

sert()의 시간계산량은 검색연산 시간계산량에 노드 생성 및 저장시간(단계2.1, 단계3.1)을 추가한 것이 된다. 그런데 생성 및 저장시간은 k에 비례하게 되므로, h-insert()의 시간계산량은 검색연산의 시간계산량과 같게 된다.

3.3 삭제연산

삭제(delete)는 주어진 단어의 자모열을 사전으로부터 제거하는 연산으로서, 주어진 자모열의 패스를 찾은 다음, 패스에 속하는 노드를 삭제한다. 삭제알고리즘 h-insert(q, i)는 〈알고리즘3〉과 같다. 단, q는 삭제할 패스를 저장한 스택이고, i는 자모열의 인덱스이며, k_1은 최적레벨이다. 또한 p는 사전을 구성하는 노드의 포인터 주소이다.

〈알고리즘3〉 h-delete(q, i)
단계 1. i<1이면 종료한다.
단계 2. 스택 q에서 노드를 pop하여 *p로 한다.
단계 3. i≤k_1이면 다음을 수행하고,
3.1 p노드의 자노드가 NULL이면 p노드를 삭제한 후 h-delete(q,i−1)를 수행하고,
3.2 아니면 h-delete(q, 0)를 수행한다.
단계 4. 아니면 *p노드를 삭제한후 h-delete(q, i−3)을 수행한다.

여기서 초기입력은 h-find()의 3.1과 4.1에서 찾아낸 자모노드 및 꼬리의 시작노드를 저장하고 있는 스택 q와 자모열의 길이 i=k이다.

h-delete()의 단계1. 에서 i가 1보다 작아지는 것은 자모열 패스를 모두 삭제한 경우이다. 또한 삭제할 때는 패스의 끝부터 삭제되며, 제거할 노드가 공통 접두사로서 다른 패스에 속하는 경우는 삭제하지 않는다.

h-delete()의 시간계산량은 삭제할 단어의 패스를 찾아내는 검색시간에 길이 k인 패스를 삭제하는 시간을 추가한 것이다. 따라서 h-delete()의 시간계산량은 검색연산의 시간계산량과 같게 된다.

Ⅳ. 실험 및 결과고찰

4.1 실험데이타

제안한 단어사전 기억구조는 Borland C++언어로 IBM-PC호환기종(CPU:486Dx, 58MHz)에서 구현하였으며, 실험을 위한 데이타로는 표1과 같이 6종류의 단어집합을 선정하였다.

표 1. 단어집합 특성
Table 1. Characteristics of word set

단어집합	바이트수	단어수	자모수	단어길이			비고
				최대	평균	최소	
ws1	1,472	144	1,305	18	9.1	6	
ws2	4,884	465	4,374	12	9.4	9	
ws3	9,615	1,207	8,268	15	6.9	6	
ws4	37,655	4,377	32,767	21	7.5	3	
ws5	71,999	10,000	60,000	6	6	6	
ws6	25,120	2,467	21,933	12	8.9	6	

여기서 단어집합1(WS1)은 전국 4년제 대학교의 이름 144단어이고, 단어집합2(WS2)는 전국 철도역의 이름 465단어이다. 또한 단어집합 3,4(WS3, WS4)는 각각 사전에 있는 명사(그부)의 1,207단어와 복합명사를 포함시킨 4,377단어이다. 그리고 단어집합5(WS5)는 난수발생으로 조합시킨 10,000단어이고, 단어집합6(WS6)은 우편번호부에 있는 전국의 읍, 면, 동이름 2,467단어이다.

4.2 특성곡선과 최적레벨

단어집합 WS1, WS2, WS3, WS4에 대한 노드이용율 $f_U(i)$와 분산율 $f_D(i)$은 그림4와 같다. 여기서 수평축은 트라이의 레벨 i이고, 수직축은 이용율과 분산율의 값으로서, 최소 0.0에서 최대 1.0의 값을 갖는다.

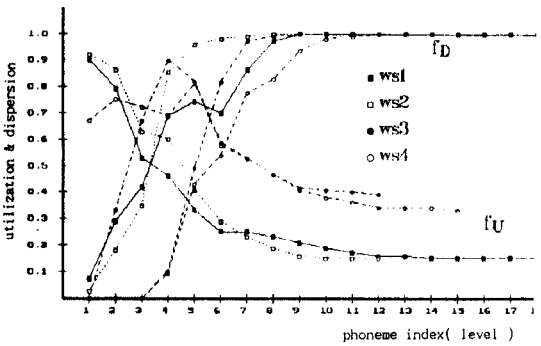


그림 4. 한글단어사전의 노드이용율 $f_U(i)$와 분산율 $f_D(i)$
Fig 4. Node utilization $f_U(i)$ & dispersion $f_D(i)$ for Hangul word dictionary

그림4에서 단어의 종류 및 단어집합의 크기에 관계없이, 레벨 i에 따른 노드이용율 $f_U(i)$의 감소특성과 분산율 $f_D(i)$의 증가특성이 유사하다. 따라서 이 특성을 이용하여 선정한 i값을 결합을 위한 최적레벨로 사용할 수 있음을 알 수 있다.

그리고 WS3과 WS4는 초성이 'ㄱ'인 단어만으로 구성되었기 때문에 그림4에서 노드 이용율 $f_U(i<k_1)$가 WS1, WS2의 경우와는 달리 작은 값에서 증가하는 특성을 갖게되었다.

또한 그림4에서 $f_U(i)=f_D(i)$를 만족하는 i의 정수값은 WS1과 WS2에서 3과 4이고, WS3과 WS4에서는 5과 6이다. 그런데 단어를 구성하는 문자들에 대한 i는 3의 배수가 되는 값을 갖게 된다. 따라서 최적레벨로 WS1과 WS2에서는 $k_1=3$, WS3과 WS4에서는 $k_1=6$의 값을 선정하였다.

4.3 결과고찰

표1의 단어집합에 대한 C-모드, P-모드, H-모드의 공간효율(space efficiency)과 검색연산에 대한 시간효율(time efficiency)을 각각 표2와 표3과 같다. 여기서 공간효율은 단어집합을 각각의 구조로 구현할 때 소요되는 메모리 바이트수를 소스화일의 바이트수로 나눈값이고, 시간효율은 검색연산을 수행할때 탐색한 노드 갯수이다.

표2와 표3에서 6개 단어집합에 대하여 공간효율은 C-모드가 P-모드보다 양호하고, 시간효율은 P-모드가 C-모드보다 양호함을 알 수 있다. 그런데 H-모드의 경우 최적레벨로 구현하면(괄호로 표기한 값) C-모드의 공간효율을 유지하면서 평균시간효율이 P-모드 만큼 양호한 것을 알 수 있다. 또한 최적레벨을 충분히 크게하면 H-모드의 공간효율은 P-모드와 같아지고, 시간효율은 C-모드와 같아지게됨을 알수 있다.

표2에서 6개 단어집합에 대한 H-모드의 평균공간효율은 1.41로서, Double-Array를 이용한 트라이 저장구조의 공간효율 1.2 보다는 공간효율이 다소 저조하지만 빈번히 발생하는 삽입연산에 대한 시간효율은 양호함을 알 수 있다.

표3의 시간효율에서 최소시간은 입력단어의 패스가 NEXT 포인터만으로 구성된 경우로서 가장 짧은 패스에 해당하며, 최대시간은 입력단어의 패스에 ALTER 포인터가 가장 많이 포함된 경우로서 최대길이 패스에 해당한다. 따라서 입력단어의 사용빈도수를 조사하여 사용빈도수가 높은 단어는 빈도수가 낮은 단어에 비하여 보다 짧은 패스를 갖도록 기억시킴으로써 시간효율을 향상시킬 수 있다.

이 실험에서 이용한 단어집합은 10,000단어 미만의 소규모 사전이다. 그러나 실제의 응용에서는 100,000단어 이상의 대규모 사전을 필요로하는 경우도 있다. 따라서 대규모사전을 구현할 경우에는 응용분야별, 품사별, 용도별 등으로 구분하여 각각의 사전을 주기억장치와 보조기억장치에 계층별로 구현하는 방법을 생각할 수 있으며, 앞으로의 연구과제이다.

표 2. 공간효율
Table 2. Space efficiency

단어집합	C-mode	P-mode	H-mode			비고
			$k_1=3$	$k_1=6$	$k_1=9$	
ws1	1.72	3.50	(1.64)	2.34	2.83	
ws2	1.88	3.85	(1.54)	2.56	3.70	
ws3	1.16	1.73	0.97	(1.31)	1.65	
ws4	1.11	1.79	1.03	(1.19)	1.56	
ws5	1.33	2.53	(1.01)	2.53	2.53	
ws6	1.36	2.73	1.09	(1.74)	2.71	

표 3. 시간효율
Table 3. Time efficiency

단어집합	C-mode			P mode			H mode								
							$k_1=3$			$k_1=6$			$k_1=9$		
	최대	평균	최소	최대	평균	최소	최대	평균	최소	최대	평균	최소	최대	평균	최소
ws1	189	87	9	28	15	7	53	(17.3)	6	38	14.4	7	25	14.7	7
ws2	501	248	9	36	20.1	9	72	(21.1)	6	34	19.7	9	36	20.1	9
ws3	543	232.0	9	33	19.2	7	723	265.3	6	33	(19.2)	7	33	19.2	7
ws4	900	276.5	9	44	22.0	6	1,326	349.8	6	114	(23.8)	6	41	22.0	6
ws5	5,871	2,701	6	46	24.8	6	64	(30.4)	6	46	24.8	6	46	24.8	6
ws6	798	252.4	9	36	21.1	8	234	46.6	6	37	(21.2)	8	36	21.1	8

V. 결 론

이 연구에서는 한글패턴인식 후처리에서 효율적으로 이용할 수 있는 단어사전의 기억구조를 제안하였다. 한글은 구성이 3성으로 이루어지고 단어 전반부에서는 분기가 많은 반면 후반부에서는 발생하지 않는다는 점에 착안하여, 단어의 전반부는 자모노드의 트라이구조로 기억시키고 후반부는 문자노드의 순차연결구조로 기억시키는 하이브리드 기억구조를 제안하였고, 결합을 위한 최적레벨을 선정하기 위하여 노드 이용율과 분산율 곡선을 이용하였다.

다양한 종류의 단어집합을 대상으로 사전을 구현하여 공간효율과 시간효율을 실험한 결과 제안한 방법의 효용성을 확인할 수 있었으며, 한글인식 후처리뿐만 아니라 철자교정, 도서목록탐색, 한자변환사전 등에 활용할 수 있을것으로 사료된다. 앞으로의 과제는 접두사뿐만 아니라 공통 접미사를 효율적으로 기억시킬 수 있는 구조와 문법 및 시맨틱스 등 상위수준의 문맥정보를 위한 사전구조의 연구이다.

"이 연구는 일본 홋카이도대학 정보공학과 Post-Doc. 연수중에 수행된 것으로, 한국과학재단과 명지대학교에 감사드린다."

참 고 문 헌

1. Sang-Woon Kim and Yoshinao Aoki, "A postprocessing of hangul recognitions using dictionary lookup", Proceedings of JTC-CSCC'93, No. 2, pp. 1013-1017, 1993.
2. 高尾 哲康, 西野 文人, "日本語文書 リータ後處理の實現と評價, 情報處理學會論文誌, Vol. 30, No. 11, pp. 1394-1401, 1989
3. 青江 順一, "トライとその應用", 情報處理, Vol. 34, No.2, pp.244-251, 1993
4. E. Fredkin, "Trie memory", Commun. ACM., Vol. 3, No.9, pp.490-500, 1960
5. K.Maly, "Compressed tries", Commun. ACM., Vol. 19, No.7, pp.409-415, 1976
6. M.Al-suwaiyel and E.Horowitz, "Algorithms for trie compaction", ACM Trans. Database Systems, Vol.9, No.2, pp.243-263, 1984
7. 日高 達, 他, "擴張B-treeと日本語單語辭典への應用", 電子情報通信學會論文誌, Vol. J67-D 4, pp.399-404, 1984
8. C.J.Wells, et al., "Fast dictionary look-up contextual word recognition", Pattern Recognition, Vol. 23, No.5, pp. 501-508, 1990
9. J. Aoe and K. Morimoto, "An efficient implementation of trie structures", Software-practice and experience, Vol. 22, No.9, pp. 695-721, 1992
10. R.M.K. Sinha, "Some characteristic curves for dictionary organization with digital search", IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-17, No. 3, pp. 520-527, 1987
11. 김상운, 이병래, 박규태, "상태공간탐색을 이용한 한글패턴 인식방법", 한국통신학회논문지, Vol. 15, No. 4, pp.267-277, 1990

金 商 雲(Sang Woon Kim) 正會員

1956년 3월 13일생

1978년 2월 : 한국항공대학 통신정보공학과졸(공학사)

1980년 2월 : 연세대학교 대학원 전자공학과졸(공학석사)

1988년 2월 : 연세대학교 대학원 전자공학과졸(공학박사)

1992년 12월~1993년 12월 : 일본홋카이도대학 정보공학과 (Post-Doc.)

1984년 4월~1989년 4월 : 한국방송통신대학 전자계산학과 조교수

1989년 4월~현재 : 명지대학교 컴퓨터공학과 부교수

※주관심분야 : 패턴인식 및 학습

青木 由直(Yoshinao Aoki) 非會員

1941年 9月 2日生

1964年 : 北海道大學 工學部電子工學科 卒

1966年 : 北海道大學 大學院 修士課程了

1969年~1971年 : Canada Laval 大學留學

1966年~現在 : 北海道大學 教授(情報工學科) 中國審陽工業大學, 黑龍江大學客員教授

※主關心分野 : 波動信號處理, Media工學, Pattern Recognition