論文 95-11-11-16

한글 인식 후처리용 단어사전의 비교연구

正會員 金 商 雲*, 愼 晟 孝*

A Comparative Study of Word Dictionary for the Postprocessing of Hangul Recognition

Sang-Woon Kim*, Seong-Hyo Shin* Regular Members

이 연구는 1995년도 명지대학교 산업기술연구소의 재정지원에 의한 결과임

要 約

이 논문에서는 한글 인식 후처리용 단어사전의 성능을 비교 분석하였다. 단어사전은 트라이와 DAWG의 저장 형태를 유형별 배열과 이중 배열, 그리고 저장 단위를 자소단위와 음절단위로 각각 나누어 구현한 8종류로 하였다. 전국의 각급 학교 이름을 사전으로 구축하여 기억공간의 이용효율과 평균 검색시간을 비교 평가하였다.

실험 결과 검색시간은 DAWG와 트라이가 동일하였으며, 기억공간 이용율은 DAWG가 양호하였다. 또한 유형별 배열에서는 검색시간이 단어의 양에 비례하고, 이중 배열에서는 단어의 길이에만 비례하였으며, 소요되는 기억공간은 큰 규모의 사전에서 서로 비슷하였다. 그리고 음절단위의 방법이 자소단위의 방법보다 공간 이용율과 검색시간 모두 두 배 이상 양호하였다.

ABSTRACT

In this paper a performance evaluation of Korean word dictionary for the postprocessing subsystem of Hangul recognition is given. We construct eight kinds of word dictionary by means of the trie and DAWG(Directed Acyclic Word Graph) structure which are implemented with 3-type array or double array as the storage structure and phonemes or characters as the unit of representation. The efficiency of storage utilization and average search time for the dictionary are evaluated comparatively through all sorts of school names in Korea.

Simulation results show that the storage space for dictionary by DAWG can be compressed more efficiently than the dictionary by trie, keeping up equal search time. And the space for 3-type array and double array are much the same in large scale dictionary, and the search time for 3-type array is in proportion to word set size and word length, but the time for double array is in word length only. The space utilization and the search time of the dictionary by character representation method are two times better than those of dictionary by phoneme.

* 명지대학교 컴퓨터공학과
論文番號 : 95189-0524
接受日字 : 1995年 5月 24日

Ⅰ. 서 론

문자인식에서의 후처리 및 문서처리의 철자교정, 도서 목록의 탐색, 한자변환 등에서 이용하는 전자화사전은 고속검색이 중요한 요소가 되며, 이러한 사전의 기억구조로는 공통되는 접두어를 한번만 저장하면서 디지탈 탐색이 가능한 트라이(trie)가 적합한 구조로 알려져 있다[1,2,3]. 그리고, 전체 노드의 갯수를 줄이기 위하여 공통 접미어까지 압축하는 DAWG(Directed Acyclic Word Graph)가 있다[4,5].

한편, 미등록율을 감소시키기 위해서는 어휘 수를 증가시켜야 하므로, 사전의 크기는 증가하게 된다. 따라서 사전으로 구축할 대용량 어휘를 분야별, 품사별, 용도별 또는 빈도수 별 등으로 분할하여 사전을 구축한 다음 디스크 등의 보조기억장치에 저장하였다가 접근요구가 발생할 때 주기억장치에 적재하여 검색하는 방법을 생각할 수 있다. 이러한 분할구조 사전의 검색시간은 디스크로부터 데이타 화일을 입출력하는 시간과 적재한 구조의 탐색시간의 합이 되며, 입출력시간은 화일의 전송시간과 주소변환시간 등으로 결정된다. 따라서 고속검색을 위해서는 구축할 사전의 입출력 화일크기 및 입출력 횟수를 감소시켜야 하며, 주기억장치와 디스크사이에서 주소변환이 용이한 구조를 이용해야 한다.

이러한 특성을 고려해 볼 때, 링크드리스트 구조의 사전을 노드별 유형에 따른 배열로 변환하여 사용하는 방법을 생각할 수 있다[3]. 이 방법은 공백포인터를 삭제함으로써 기억공간을 절약할 수 있고, 입출력할 때 주소변환이 용이하다는 장점을 갖고 있다.

그리고, 트라이는 입력되는 단어의 양이 많아질수록 형제 링크의 수가 증가하게 되어 검색 시간이 입력되는 단어의 양에 비례하여 증가한다는 단점이 있다. 이에 이중 배열을 이용하여 형제 링크를 없앰으로써 검색 시간을 단축시킬 수 있다[4,6].

이 논문에서는 트라이와 DAWG를 유형별 배열과 이중 배열을 이용하여, 한글의 특성에 맞는 자소단위 및 음절단위 한글 사전을 구성하는 방법에 대하여 설명한다. 세가지 데이타를 입력하여 각각의 사전을 구축한 후, 각 사전의 성능을 평가하기 위하여 기억 장소의 효율과 검색 시간을 상호 비교한다.

국내의 한글 인식을 위한 후처리 기법은 초보적인 연구 단계를 거쳐 현재 부분적으로 실용화 단계에 이르는 기술적 진보를 보이고 있으며[7,8], 거의 대부분의 방법이 전자 사전을 이용하고 있다. 따라서, 효율적인 사전 구성 방법의 선택은 한글 인식의 후처리 단계에 소요되는 시간을 크게 단축할 수 있다.

이하 제 2장에서는 사전의 구축 방법 네 가지와 한글 저장 단위인 음절단위 방법과 자소단위 방법에 대하여 설명한다. 제 3장에서는 2장에서 설명한 네 방법들의 성능을 이론적으로 분석한 결과를 나타내고 제 4장에서는 실험 결과를 통하여 각 방법의 장단점을 분석한다. 마지막으로 제 5장에서 결론을 맺는다.

Ⅱ. 사전 구축 방법

문자 인식 후처리용 단어 사전을 구축하는 방법에 있어서 트라이나 DAWG로 구성한 사전을 유형별 배열이나 이중 배열로 저장하는 방법을 생각할 수 있다. 이 장에서는 단어집합 K = {명지대학교, 명지전문대, 서울대학교, 서강대학교}에 대한 사전을 구현하는 방법에 대하여 설명한다.

2.1. 트라이-유형별 배열에 의한 구축 방법 (방법-1)

이 방법은 트라이를 구성하는 각 노드를 유형별로 나누어 배열로 구현하는 방법으로써, 기억 장소의 효율과 보조기억장치와의 입출력 효율을 증가시킬 수 있는 방법이다[3].

우선, 단어 집합 K를 링크드리스트 구조로 구현한 음절별 트라이는 그림 1과 같다. 여기서 각 노드의 첫번째 자료 필드에는 한글 한 글자를 저장하고, 두번째 자료필드에는 그 글자의 상태에 관한 정보를 저장한다. 한 경로의 마지막 노드일 경우 'E'의 값을 가지며 이외의 노드는 'C'의 값을 갖는다. 세번째 필드는 형제 노드(sibling node)를 가리키는 형제 링크 필드이며, 네번째 필드는 자식 노드(child node)를 가리키는 자식 링크 필드이다.

트라이를 그림 1과 같이 링크드리스트를 이용하여 구현할 경우 세 가지의 단점이 존재한다. 첫째, 트라이 구조는 각 노드의 크기를 항상 일정하게 구현하여야 하는데, 각 노드에서 모든 링크필드를 사용하는 것은 아니므로 기억장치 사용에 있어서 비효율적이다. 그림 1에서 노드는 17개이며, 전체 34개의 링크 필드가 존재한다. 그러나 사용되는 링크 필드는 16개로써 사용율이 47%

에 지나지 않으므로, 50%이상의 링크 필드가 낭비되고 있다. 둘째, 단어 당 평균 검색 시간은 입력된 단어의 양에 비례한다. 입력되는 단어의 양이 많아질수록 사전 내에 형제 링크를 사용하는 노드의 갯수가 증가하게 되므로, 대량의 사전 구축 시 검색 시간에 있어서 문제가 된다. 셋째, 형제, 자식 링크 필드는 주기억장치 내의 주소이므로, 보조기억장치로 저장할 때 주소를 변환하여야 한다.

이에 사용하지 않는 링크 필드와 주소 변환의 복잡성을 해결하기 위하여 유형별 배열을 사용하는 방법이 제안되었다. 유형별 배열이란 각 노드의 링크 필드 사용여부에 따라 노드를 배열에 저장하는 방법이다. 링크 필드 두 개 모두 사용하는 노드를 위한 유형-2 배열, 하나의 링크만을 사용하는 노드를 위한 유형-1 배열, 하나도 사용하지 않는 노드를 위한 유형-0 배열을 이용하여 그림 2와 같이 저장하는 방법이다. 따라서, 사용되지 않는 링크 필드를 없앨 수 있기 때문에 기억 장소의 낭비를 줄일 수 있으며, 형제 노드와 자식 노드의 위치는 배열의 인덱스로 저장되기 때문에 보조기억장치로의 저장 및 주기억장치로의 적재 시 주소 계산이 필요치 않게 된다. 그러나, 단어 당 평균 검색 시간이 입력되는 단어의 양에 비례한다는 단점은 해결하지 못한다.

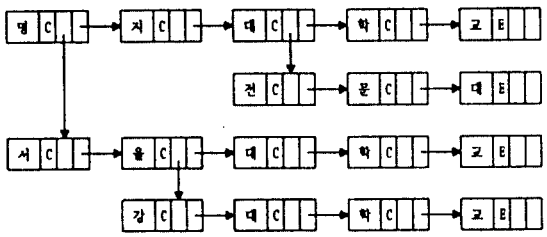


그림 1. 단어 집합 K에 대한 트라이의 예
Fig. 1. An example of trie for word set K

A2

명	C	11/0	11/4
대	C	11/1	11/2
울	C	11/5	11/7

A1

지	C	2/1
학	C	0/0
전	C	11/3
문	C	0/1
서	C	2/2
대	C	11/6
학	C	0/2
강	C	11/8
대	C	11/9
학	C	0/3

A0

교	E
대	E
교	E
교	E

그림 2. 그림 1.의 트라이를 유형별 배열로 구현한 예
Fig. 2. 3-type array implementation for the trie of Fig. 1

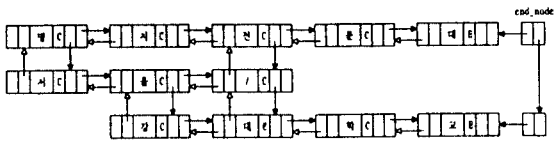
2.2. DAWG-유형별 배열에 의한 구축 방법(방법-2)

이 방법은 DAWG를 유형별 배열로 구현하는 방법으로써[5], 공통접두어뿐만 아니라 공통접미어도 압축시킴으로써 방법-1보다 전체 노드의 갯수를 줄일 수 있다.

단어 집합 K에 대하여 링크드리스트와 유형별 배열로 구현한 음절별 DAWG는 각각 그림 3의 (a),(b)와 같다. 그림 3의 (a) 중, 노드 〈/〉는 단지 앞뒤 노드를 연결하기 위한 의사 노드(dummy node)이다. 그림 1과 그림 3을 비교하여 볼 때, 트라이를 이용하였을 경우 전체 노드는 17개이지만, DAWG를 사용하였을 경우 전체 노드가 11개로써 트라이의 65%정도의 노드만을 필요로 한다. 그러나 첫째, 유형별 배열로 변환한 이후에 삽입되는 단어는 공통접미어가 압축되지 않은 상태로 저장되며, 둘째, 공통접미어를 압축함으로써 인덱스로써의 기능을 상실한다는 문제점이 있다.

2.3. 트라이-이중 배열에 의한 구축 방법(방법-3)

이 방법은 빠른 검색과 기억 공간의 효율을 위하여 트라이 구조를 이중 배열(double array)로 표현하여 사



(a)

A2

명	C	11/0	11/2
실	C	11/1	11/4
울	C	11/4	11/3

A1

지	C	2/1
문	C	0/0
서	C	2/2
강	C	11/4
대	C	11/5
학	C	0/1

A0

대	E
교	E

(b)

그림 3. 단어 집합 K에 대한 DAWG의 예
(a) 링크드리스트에 의한 DAWG
(b) 유형별 배열로 구현한 DAWG
Fig. 3. An example of DAWG for word set K
(a) A DAWG by linked list
(b) 3-type array implementation for DAWG

전을 구축하는 방법이다[4]. 이중 배열은 디지탈 검색을 이용한 방법으로써 그림 4와 같은 구조적 특징을 갖는다. 여기서 키 a에 대하여

$$BASE[s]+a = t;\ s = CHECK[t] \quad (1)$$

의 관계식이 성립한다.

단어 집합 K를 방법-3으로 구현한 단어 사전은 그림 5와 같다. 유형별 배열에서 사용하는 트라이 구조는 노드에 문자 정보를 표현하지만, 이중 배열에서 사용하는 트라이 구조는 그림 5와 같이 링크에 문자 정보를 키값(key-value)으로 디지탈화한 후 저장한다. 링크의 괄호안 숫자는 문자를 키값으로 변환시킨 숫자를 나타내며, 표현을 간단하게 하기 위하여 임의의 키값을 부여하여 그림을 도시하였다. 각 단어의 삽입, 삭제, 검색은 항상 s = 1부터 시작한다. 검색에 있어서 트라이의 경우, 노드의 문자와 해당 문자가 일치하면 자식 노드로, 일치하지 않으면 형제 노드로 탐색하게 된다. 따라서, 하나의 단어를 검색할 때 검색할 노드의 수는 단어의 길이에 검색되는 형제 노드의 길이를 더해야 한다. 그러나, 이중 배열을 사용할 경우 하나의 노드에서 분기할 수 있는 경로는 키가 가질 수 있는 값의 범위와 같다. 즉, 해당 노드의 BASE값에 키값를 더하여 다음 노드를 구한 후, 다음 노드의 CHECK값이 해당 노드와 같은지를 판별하여, 같으면 검색을 계속하고, 그렇지 않으면 검색을 중단하기 때문에 검색할 노드의 수는 단어의 길이에 비례하게 된다.

2.4 DAWG-이중 배열에 의한 구축 방법 (방법-4)

이 방법은 DAWG를 이중 배열로 표현하는 방법으로써, 방법-3과 비교하여 볼 때 검색 시간의 효율은 저하시키지 않으면서 공통접미어도 압축시켜 전체 기억 장소의 효율을 높이는 방법이다[4].

단어 집합 K를 방법-4로 구현한 단어 사전은 그림 6과 같다. 그림 6에서 BASE[s] < 0일 경우는 노드 s는 다른 노드와 같은 노드임을 표시하는 것으로써 통합 처리의 결과로써 발생한다. 예로 BASE[10] = -9 이므로 노드 10은 노드 9와 같은 노드임을 나타낸다.

그리고, CHECK[s] < 0일 경우(노드 13), 노드 s로 분기하는 노드의 갯수가 2이상임(노드 9와 노드 3)을 표시하며, 이 경우 BASE[t] = -s인 노드 t(노드 12)가 존재한다. 이 노드 또한 통합 처리의 결과로써 발생한다. 검색 시 BASE[s] < 0인 노드를 만나면 s와 통합된 노드 -BASE[s]로 상태를 이전시킨 후 검색 연산을 계속한다.

방법-3과 배열의 크기를 상호 비교하여보면, 방법-3은 전체 21개의 슬롯을 할당받아 18개의 슬롯을 사용하였고 방법-4는 19개의 슬롯을 할당받은 후 13개의 슬롯만을 사용하였다. 따라서, 방법-4를 사용함으로써 전체

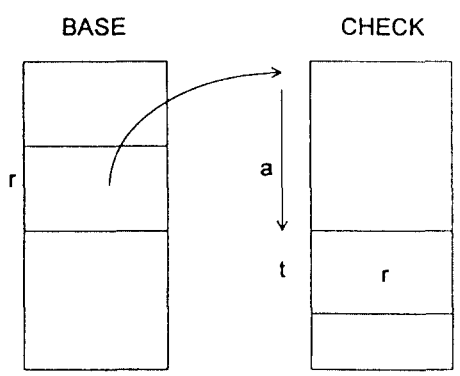
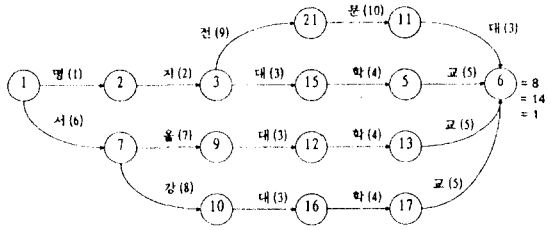


그림 4. 이중 배열의 구조[4]
Fig. 4. The structure of double array



index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
BASE	1	1	12		1	0	2	-6	9	13	5	9	9	-6	1	13	13	-6			1
CHECK	21	1	2		15	-5	1	11	7	7	21	9	12	13	3	10	16	17			3

(b)

그림 5. 단어 집합 K를 이중 배열에 의한 트라이로 구현한 예
(a) 단어집합 K에 대한 트라이
(b) 이중 배열로 구현한 트라이
Fig. 5. Double array implementation of trie for word set K
(a) A trie for word set K
(b) Double array implementation for trie

배열의 크기를 줄일 수 있다.

그리고, DAWG를 이중 배열로 표현할 경우 방법-2의 첫번째 문제점을 해결할 수 있으나 두번째 문제점은 해결하지 못한다.

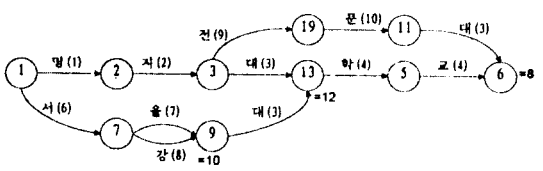
이중트라이를 이용한 방법[6]에서는 한 단어를 다중 노드, 분할노드, 단일노드의 세부분으로 나눈 후, 단일노드 상에 존재하는 공통접미어만을 압축함으로써 두번째 문제점을 해결할 수 있다. 다중노드는 공통접미어 부분의 노드들, 단일노드는 다중노드 이후의 노드들에 해당하며, 분할노드는 첫번째 단일노드에 해당한다.

2.5. 음절단위 방법과 자소단위 방법

한글은 초, 중, 종성의 3성으로 구성된 조합문자이기 때문에 음절단위로 기억시키는 방법과 3성의 자소단위로 기억시키는 방법이 가능하다.

2.5.1 음절단위 방법

한글을 저장하고 검색하기 위해 입력된 단어 각각의 음절을 하나의 노드에 표현하는 방법이다. 유형별 배열을 사용할 경우 노드의 문자 저장 필드에 2바이트를 부여한다. 따라서, 문자 저장 필드 2바이트, 상태 필드 1바이트, 형제, 자식 필드에 각각 5바이트를 할당하기 때문에 각 노드 당 전체 13바이트가 사용된다.



(a)

index	1	2	3	4	5	6	7	8	9	10	11	12	13	···	19
BASE	1	1	10		1	0	2	-6	9	-9	5	-13	1	···	1
CHECK	19	1	2		13	-5	1	11	-7	7	19	9	-3	···	3

(b)

그림 6. 단어 집합 K를 이중 배열에 의한 DAWG로 구현한 예
(a) 단어집합 K에 대한 DAWG
(b) 이중 배열로 구현한 DAWG

Fig. 6. Double array implementation of DAWG for word set K
(a) A DAWG for word set K
(b) Double array implementation for DAWG

이중 배열을 사용할 경우에는 문자에 대한 정보는 링크에 저장되며, 노드는 오프셋을 나타내는 BASE와 문자의 키값과 오프셋을 더하여 생성되는 분기의 진위 여부를 나타내는 CHECK로 구성된다.

입력되는 단어의 음절들을 이용하여 분기를 결정하기 위해서는 각 음절에 해당하는 키값을 부여해야 한다. 모든 음절을 다 사용하는 것은 아니므로 디지탈 값을 용이하게 부여하기 위하여 KS완성형을 이용한다고 하면, 각각의 음절에 1에서 2,350까지의 디지탈 값을 부여하여 저장 규칙에 따라 배열에 저장한다. 그러나, 완성형 코드에 포함되어 있지 않은 음절들을 다룰 수 없다는 문제점이 있다. 이 단점을 해결하기 위하여 조합형을 사용할 경우, 초, 중, 종성에 해당하는 키값을 각각 나누어 계산한 후, 다시 하나의 키값으로 변환시켜야 하는 번거로움이 생긴다.

2.5.2 자소단위 방법

한글을 구성하는 기본 단위로써 하나의 음절을 초, 중, 종성의 자소단위로 분리한 후, 각각의 자소를 하나의 노드에 저장하는 방법이다. 자소단위로 저장함으로써 한글 철자 검색 및 모든 가능한 음절을 표현할 수 있다는 장점이 있다. 그러나, 하나의 단어를 저장하는 데 각 음절을 3개의 자소로 분리하기 때문에, 음절단위보다 3배의 노드수를 필요로 한다는 단점이 있다.

유형별 배열을 사용할 경우 문자 저장 필드 1바이트만 사용하고 나머지 필드는 음절단위 방법과 같은 크기를 사용하므로 노드당 12바이트를 할당받는다.

이중 배열을 사용할 경우에는 초, 중, 종성 각각의 값을 디지탈화하여 하나의 노드에 하나의 자소를 저장하면 된다. 따라서 소요되는 용량은 음절단위 방법의 3배가 된다.

Ⅲ. 이론적 성능 평가

이 장에서는 제 2장에 기술한 사전 구축 방법에 대한 기억 장소, 검색과 삽입 연산의 성능을 이론적으로 분석 평가한다. 여기서 단어의 길이는 단어를 이루는 문자의 갯수 n으로 한다.

3.1 단어 당 필요한 노드 수

트라이를 이용하는 방법-1과 방법-3은 하나의 단어가

입력되었을 때 공통접두어 이후의 문자들에 대해서만 새 노드를 할당받는다. 따라서, 하나의 단어 당 필요한 노드의 수는 단어의 길이에서 공통접두어의 길이를 제외한 수가 된다.

DAWG를 이용한 방법-2와 방법-4는 단어의 길이에서 공통접두어와 공통접미어의 길이를 제외한 수만큼의 노드를 필요로 하게 된다. 따라서, DAWG를 사용하는 방법이 트라이를 사용한 방법보다 적은 수의 노드를 필요로 하기 때문에 전체 기억 장소의 효율을 증가시킬 수 있다.

3.2 노드 당 필요한 기억 용량

트라이를 이용한 방법-1과 방법-2에서는 노드에 문자, 상태, 분기의 정보를 모두 저장하고 있다. 노드 당 필요로 하는 공간은 음절단위일 때, 유형-2의 경우 13바이트, 유형-1의 경우 8바이트, 유형-0의 경우 3바이트를 소비한다. 그리고, 트라이 내의 유형-2에 해당하는 노드의 수는 유형-0에 해당하는 노드의 수와 거의 같기 때문에, 노드 당 평균 기억 용량은 8바이트가 된다. 자소단위의 경우는 음절단위보다 한 바이트가 적은 노드 당 7바이트를 필요로 한다.

방법-3과 방법-4와 같이 이중 배열을 사용할 경우, 문자 정보는 해당 노드에서 다음 노드로의 분기에만 사용될 뿐이며 노드 내에는 이전 노드와 다음 노드의 인덱스만을 저장한다. 따라서 한글의 저장 단위에 상관없이 BASE와 CHECK에 각각 4바이트씩을 할당하여 노드 당 8바이트를 필요로 한다.

3.3 평균 검색 시간

트라이를 사용할 경우 한 노드에서 분기할 수 있는 다음 노드들은 각 노드들의 형제 링크로 연결되어 있기 때문에, 다음 해당 노드를 찾아내기 위하여 형제 링크를 따라 검색하는 시간을 소비해야 한다. 따라서 길이 n인 단어의 최대 검색시간은 n×a가 된다. 여기서 a는 형제 링크를 따라 검색하는 데 소요되는 시간이다.

그리고, DAWG를 사용하여 공통접미어를 압축한다는 것은 경로의 길이를 줄이거나 늘이지 않고 공통된 경로를 하나의 경로로 통합시키는 것이기 때문에 검색 시간에는 영향을 미치지 않아 트라이와 사전내를 검색하는 속도가 같다.

유형별 배열을 사용할 경우, 사전의 크기가 커질수록 형제 링크를 사용하는 유형-2의 노드가 많아지게 되어 a의 값은 커지게 된다. 즉 입력되는 단어의 양에 비례하여 a의 값이 변하고 따라서 검색 시간도 증가하게 된다. 소규모 단어 사전에서의 검색 시간은 a의 값이 작기 때문에 단어의 길이에 많은 영향을 받으므로 자소단위 방법이 음절단위 방법보다 불리하다. 그러나 단어의 양이 많아질수록 음절단위 방법보다 자소단위 방법이 효율적이다. 예로 음절단위 트라이의 경우, 다음 노드를 찾기 위하여 최대 2,350개의 형제 노드를 검색해야 할 수도 있어 최악의 경우 2,350×n의 검색 시간을 필요로 한다. 자소단위에 있어서는 최악의 경우 32×3n의 검색 시간을 필요로 한다.

이중 배열을 이용할 경우, 한 노드 이후 노드들은 키 값에 의하여 분기해 갈 노드가 결정되기 때문에 단어의 길이에 해당하는 시간을 소비한다. 검색되는 시간은 입력되는 단어의 양에 상관없이 입력되는 단어의 길이에 비례하므로 음절단위 방법이 자소단위 방법보다 이론적으로 3배가 빠르다.

3.4 평균 삽입 시간

단어 당 평균 삽입시간은 각 방법에 있어서 많은 차이를 보인다. 방법-1의 경우 검색시간과 거의 같은 시간에 삽입할 수 있는 빠른 삽입 연산 시간을 보인다. 방법-2는 방법-1의 삽입시간에 공통접미어를 압축하는 시간이 더 소요된다. 공통접미어를 압축하는 데 걸리는 시간은 입력한 단어의 마지막 문자와 서로 같은 사전 내의 마지막 문자를 찾아내는 시간과 그 문자로부터 같지 않은 문자를 만날 때까지 백트랙 시간으로 구성된다. 즉, 압축 시간은 다음과 같이 구성된다.

$$\text{압축 시간} = \frac{A}{2} + B \tag{2}$$

여기서,

A = 사전 내 마지막 문자의 종류,

B = 공통접미어의 길이

에 해당한다.

방법-3의 경우 단어의 길이에 충돌을 해결하는 시간이 추가된다. 충돌은 다음과 같은 경우에 발생한다.

$$t = BASE[s] + key\_value,\quad s \neq CHECK[t] \text{ and } CHECK[t] \neq 0 \tag{3}$$

만약 s = CHECK[t]이면 해당키가 이미 사전 내에 존재함을 의미하고, 이는 공통접두어임을 나타낸다. 그러나, s ≠ CHECK[t], CHECK[t] ≠ 0이면 공통접두어가 아니면서 노드 t가 비어 있지 않은 경우이다. 이를 충돌이라 하며 이러한 문제점을 해결하기 위해서는 s의 BASE값을 조정하여 s에서 해당키를 입력하였을 때 노드 t로 분기할 수 없도록 하든지, 노드 t의 위치를 옮기어 노드 s에서 해당키를 입력할 수 있도록 하여야 한다. 노드 t의 위치를 변경하기 위해서는 노드 t의 이전 노드의 BASE값을 변경하여야 하므로 노드 s의 BASE값을 변경하는 작업과 같은 작업이 이루어진다.

충돌 해결을 위하여 변경하여야 할 노드를 r이라 하자. 우선 노드 r 이후에 존재하는 모든 노드를 조사하여 리스트에 저장한다. 리스트 내의 노드들의 위치를 변경할 때 또다시 충돌이 일어나지 않도록 노드 r의 BASE를 변경한다. 그리고, 노드들의 위치가 바뀌었으므로 리스트의 노드 이후에 나타나는 다음 노드들의 CHECK값을 바꾸어야 한다. 음절단위 방법에서 최악의 경우 r 노드 이후에 바꾸어야 할 노드가 2,350개, 그리고 각 노드의 다음 노드들의 CHECK값 변경이 각각 2,350번, 따라서 n×2,350×2,350번의 연산이 일어난다. 이는 방법-1과 방법-2보다 2,350배의 시간을 더 소비한다.

방법-4의 경우는 방법-3보다 공통접미어를 압축시키는 시간이 더 소요된다. 이는 방법-2와 마찬가지로 마지막 문자를 검색하는 시간과 백트랙하는 시간으로 구성된다. 따라서 여러 가지 방법 중 삽입시간이 가장 길다.

방법-1,2,3,4로 구축한 사전에서 단어당 필요한 노드수(node/word), 노드당 필요한 기억 용량(byte/node), 단어당 평균 검색시간(search-time/word), 단어당 평균 삽입시간(insert-time/word)은 각각 표 1과 같다.

Ⅳ. 실험 및 평가

4.1 실험 데이타

앞에서 고찰한 사전 구축 방법의 성능 평가를 위하여 전국의 290개 대학명(이하 DATA 1이라 함), 경기도 관내의 초,중,고등학교명 1,326개(DATA 2), 전국 유치원, 도서관, 초,중,고등학교명 16,195개(DATA 3)를 실험 데이타로 선정하였다. 데이타의 입력 단어 수, 전체 문자 수와 단어의 평균 길이는 표 2와 같다. 이 세 가지 데이타를 각각 입력으로 하여 네 가지 방법으로 사전을 구성한 다음, 기억 장소 효율 및 검색 시간을 측정하였다. 사용한 컴퓨터는 SUN sparc-20이고, C 언어를 이용하였다.

표 2. 입력 데이타의 특성
Table 2. Characteristics of experimental data

특성 / 데이타	단어수	문자수	평균단어 길이	화일 크기 (byte)
DATA 1	290	1,884	6.5	4,636
DATA 2	1,326	8,728	6.6	17,879
DATA 3	16,195	130,192	8.0	276,748

표 1. 방법별 이론적 성능 비교
Table 1. Comparisons of theoretical performance for each method

방법 / 특성	방법-1		방법-2		방법-3		방법-4	
	음절	자소	음절	자소	음절	자소	음절	자소
node/work	n-p	3n-p	n-p	3n-p	n-p	3n-p	n-p	3n-p
byte/node	7	8	7	8	8	8	8	8
serch-time/word	n×a	3n×a	n×a	3n×a	n	3n	n	3n
insert-time/word	n×a	3n×a	n×a+m	3n×a+m	n×a	3n×a	n×a+m	3n×a+m

n : 단어의 길이(자소단위의 경우 3×n), a : 형제 노드의 수, m : 통합에 걸리는 시간,
c : 충돌 해결에 걸리는 시간, p : 공통접두어의 길이, s : 공통접미어의 길이

4.2 기억 장소의 효율 비교

4.2.1 트라이와 DAWG의 비교

DATA 1, DATA 2, DATA 3을 각각 제 2장에서 설명한 네 가지 방법으로 구현한 후에, 각 방법에 의해 할당된 노드의 갯수와 실제 저장 공간을 비교한 결과는 표 3과 같다.

트라이와 DAWG가 공통으로 가지는 특성은 공통접두어의 압축이며, 이 압축 과정에서 생겨나는 노드 타입은 형제 링크와 자식 링크를 모두 가지는 유형 2의 노드이다. 표 3의 방법-1과 방법-2의 내용을 살펴보면, 유형-2의 경우 트라이와 DAWG가 같은 수의 엔트리를 가진다. 반면에 DAWG에서만 가지는 특성인 공통접미어의 압축은 유형-1과 유형-0의 노드들 중 같은 문자를 갖고 있는 노드들을 통합하는 것이므로, 유형-1과 유형-0의 경우 기억 장소의 효율에 있어서 많은 효과를 볼 수 있다. DAWG의 유형-1의 크기는 트라이의 절반 가량으로 줄어들었고, 유형-0은 입력된 문자열의 마지막 문자의 종류로 줄어들었다. 이중 배열의 경우도 이와 상응되는 노드의 수가 감소하였다.

입력한 데이타를 DAWG로 사전을 구축하였을 때, 압축율은 데이타의 종류에 따라 그 정도가 상이하다. 트라이와 DAWG를 이용하여 음절단위 사전을 구축하였을 때의 노드 수와 입력 데이타의 문자수를 비교하여 입력한 데이타들의 종류에 따른 압축율을 살펴보면 표 3과 같다. 압축율은 다음과 같이 구하였다. 단 자소단위의 방법의 경우 입력화일의 자소수를 기준으로 하였다.

$$\text{압축율} = \frac{A}{B} \times 100 \qquad (4)$$

여기서,

A = B - 주어진 구조로 구현하였을 경우의 노드 수,

B = 입력화일의 문자(자소)수

에 해당한다.

DATA 1을 사용하여, 방법-1에 의한 음절단위 단어 사전을 구축하였을 경우, 전체 노드의 수는 1,553개로서 331개의 문자가 공통접두어로써 압축되어 문자열 당 평균 17.57%를 줄일 수 있었다. 방법-2를 사용할 경우, 전체 노드수는 741개이고 문자열 당 평균 2.56개의 노드를 필요로 한다. 공통접두어의 압축은 트라이와 같으므로 331개의 문자가 공통접두어로써, 831개의 문자가 공통접미어로써 압축되었다. 따라서 DATA 1의 경

표 3. 기억 장소의 효율 비교
Table 3. Comparison of storage efficiencies

방법-1

데이타 / 특성	DATA 1		DATA 2		DATA 3	
	자소	음절	자소	음절	자소	음절
유형-0	285	285	1,320	1,320	12,916	12,916
유형-1	3,940	984	15,874	3,754	200,344	51,670
유형-2	284	284	1,319	1,319	12,915	12,915
합 계	4,509	1,553	18,513	6,393	226,175	77,501
압축율(%)	20.22	17.57	29.30	26.26	42.29	40.47
기억용량 (byte)	27,050	10,867	111,074	44,297	1,357,046	542,503

방법-2

데이타 / 특성	DATA 1		DATA 2		DATA 3	
	자소	음절	자소	음절	자소	음절
유형-0	3	3	8	8	32	32
유형-1	1,782	427	7,678	1,848	100,214	25,517
유형-2	284	284	1,319	1,319	12,915	12,915
합 계	2,069	714	9,005	3,175	113,161	38,464
압축율	63.40	62.10	65.61	63.62	71.63	70.45
기억용량	15,888	7,117	65,590	31,955	856,542	372,097

방법-3

데이타 / 특성	DATA 1		DATA 2		DATA 3	
	자소	음절	자소	음절	자소	음절
배열크기 (빈슬롯)	4,510 (0)	3,956 (2,402)	18,514 (0)	8,214 (1,820)	226,179 (4)	78,910 (1,408)
압축율	20.22	-109.98	62.38	34.77	69.72	62.19
기억용량	36,080	31,648	126,512	65,712	1,809,432	631,280

방법-4

데이타 / 특성	DATA 1		DATA 2		DATA 3	
	자소	음절	자소	음절	자소	음절
배열크기	2,567 (9)	3,956 (3,020)	9,851 (15)	5,693 (2,696)	118,260 (5)	489220 (4,642)
압축율	54.58	-110.03	62.38	24.77	69.72	62.19
기억용량	20,536	31,656	78,808	45,544	946,080	393,760

우 62.10%의 압축율을 보였다. DATA 2의 경우는 63.62%, DATA 3의 경우는 70.45%의 압축율을 가진다.

트라이를 사용한 방법-1의 음절단위인 경우는 공통접두어만 압축하는 효과로써 평균 28.24%의 압축율을 보인 반면, DAWG를 사용한 방법-2는 공통접두어와 공통접미어가 압축되어 65.39%의 압축율을 보여 트라이보다 평균 37.15%이상을 압축할 수 있다. 자소단위인 경우 압축율이 각각 25.26%와 66.88%로써 트라이보다 41.62%이상을 압축할 수 있었다.

4.2.2 유형별 배열과 이중 배열의 비교

방법-1에서 음절단위 방법의 경우, DATA 1, DATA 2, DATA 3에서의 압축율이 각각 17.57%, 26.75%, 40.47%인데 비하여, 방법-3의 경우는 각각 -109.98%, 5.89%, 39.39%로 방법-1과 많은 차이의 압축율을 보인다. DATA 1의 경우는 -109.98%로써 중간에 많은 빈 슬롯의 갯수가 존재하여 약 2배 이상으로 늘어난 것을 볼 수 있다. 그러나, 입력되는 단어의 양이 많아질수록 빈 슬롯의 수가 적어짐에 따라 유형별 배열과 이중 배열의 압축율이 서로 비슷해져 감을 알 수 있다.

유형별 배열은 키가 입력되면 현재 사전의 제일 마지막 엔트리에 키를 저장한 후, 링크 필드의 값을 조정하게 된다. 따라서 중간에 비어있는 경우가 발생하지 않는다. 그러나 이중 배열의 경우, 한 노드와 다음 노드의 관계는 노드의 BASE값과 키의 값의 합에 대응되므로, 다음 노드는 반드시 이 합의 위치만큼 떨어져 존재하여야 한다. 따라서 중간에 비어있는 공간이 존재한다.

자소단위 방법을 사용하여 사전을 구축할 때 실제 차지하는 기억 공간을 비교하여 보면, 이중 배열은 모든 노드가 유형별 배열의 7바이트보다 1바이트가 큰 8바이트를 사용하기 때문에 전체적으로 기억 공간을 많이 차지하게 된다.

방법-4에서 이중 배열에서 중간에 비어 있는 공간을 제외하고 실제 값이 저장되어 있는 노드의 숫자를 계산하면, 방법-2의 노드의 숫자보다 큰 것을 알 수 있다. 이는 공통접미어를 통합하게 되면 그림 6의 노드 9와 노드 10같이 공통접미어의 가장 첫 문자 노드는 두 개의 노드로 표현되는 한편, 유형별 배열은 하나의 노드만으로 표현되기 때문이다.

4.2.3 자소단위 방법과 음절단위 방법의 비교

자소단위 트라이의 경우 한 노드의 크기는 음절단위 트라이보다 1바이트가 작지만, 자소단위는 단어의 길이가 음절단위의 3배이기 때문에 기억 공간도 약 3배를 차지한다.

이중 배열의 경우, 노드의 크기는 자소단위와 음절단위가 동일하다. 음절단위의 경우 키값이 크기 때문에 적은 양의 단어가 입력될 때, 중간에 비는 노드가 많이 생긴다. 예를 들어 표 3의 방법-3 중 입력되는 단어의 양이 가장 적은 DATA 1의 경우, 자소단위 방법은 중간에 비는 노드가 하나도 없다. 그러나 음절단위 방법에서 전체의 60%에 해당하는 노드가 할당을 받고 쓰여지지 않고 있다. 따라서, 자소단위 방법이 음절단위 방법보다 3배의 기억 장소를 필요로 한다는 관계는 성립되지 않는다.

그러나, 대량의 단어가 입력되면 그 중간 노드가 거의 사용되기 때문에 대량의 단어를 입력하게 되면, 자소단위의 방법이 음절단위의 방법보다 3배의 공간을 차지하게 된다.

4.3 검색 시간의 효율 비교

네 가지 방법을 사용하여 사전을 구축한 후, 입력한 화일을 다시 검색하였을 때 검색을 완료하는 데 걸리는 시간은 표 4와 같다. 방법-1과 방법-2, 방법-3과 방법-4는 서로 시간이 비슷하기 때문에 하나의 표로 나타내었다.

표 4의 방법-1,2와 방법-3,4에서 단어당 평균 검색 시간을 비교하면 이중 배열을 이용하는 방법이 검색에 있어서 효율적임을 알 수 있다. 트라이와 DAWG의 검

표 4. 검색 시간의 비교
Table 4. Comparison of search times 10^{-6}sec.

데이타 \ 방 법		방법-1, 방법-2	방법-3, 방법-4
		전체(평균)	전체(평균)
DATA 1	자소	180,116(621.30)	160,280(552.97)
	음절	85,785(285.92)	61,723(212.42)
DATA 2	자소	849,623(641.75)	690,547(521.32)
	음절	451,544(380.84)	283,669(214.71)
DATA 3	자소	11,385,645(702.15)	9,838,000(606.72)
	음절	8,823,023(510.02)	3,841,430(236.74)

색 시간 차는 사전내의 형제 링크의 갯수에 의해 결정된다. 트라이의 검색시간은 단어의 길이와 형제 링크의 갯수에 비례하며, 형제 링크는 입력되는 단어의 갯수에 비례한다.

방법-1,2의 음절단위 방법에서 DATA 1,2,3에 대하여 각각 295×10-6초, 0.340×10-6초, 0.510×10-6초로써 입력한 단어의 양과 길이에 비례하여 당 평균 검색 시간이 증가하는 것을 알 수 있다. 그러나 이중 배열의 경우는 각각 0.212×10-6초, 0.214×10-6초, 0.236×10-6초로 입력한 단어의 길이에 비례하는 검색 시간을 갖는다.

자소단위 방법은 음절단위 방법보다 키의 길이가 3배 더 길기 때문에, 검색 시간이 약 3배 더 걸릴 것으로 예상된다. 그러나 유형별 배열을 이용할 경우, 자소단위 방법은 노드를 구성하는 1바이트의 문자 필드를 검색하고, 음절단위 방법은 2바이트의 문자 필드를 검색하기 때문에 약 2배의 차를 보인다. 이중 배열의 경우는 검색 연산시 문자의 비교가 아니라 배열의 인덱스 비교로써, 자소단위 방법과 음절단위 방법이 서로 같기 때문에, 약 3배의 검색 시간 차이를 보인다.

V. 결 론

이 논문에서는 한글 인식 후처리용 단어사전의 구축방법을 트라이와 DAWG구조를 유형별 배열과 이중 배열로 저장하는 방법, 그리고 한글 표현 단위를 자소단위와 음절 단위로 저장하는 방법 등 모두 8가지로 나누어 고찰하였다. 또한 전국의 각급 학교 이름의 사전을 구축하고, 기억 장소의 효율과 평균 검색 시간을 측정하여 사전의 성능을 비교 분석하였다.

DAWG방법을 트라이 방법과 비교하여 볼 때, 검색 시간은 동일하게 유지하면서 한 단어당 필요로 하는 기억 장소를 줄일 수 있었다. 또한 적은 수의 단어에 대한 사전을 구축할 때, 이중 배열을 이용하면 중간에 비어있는 공간이 많아 기억 장소의 효율면에서 유형별 배열에 비하여 떨어지나, 입력되는 단어의 양이 많아질수록 거의 동등하게 되어감을 볼 수 있었다.

유형별 배열에서의 검색 시간은 입력되는 단어의 양에 비례하여 증가하지만, 이중 배열에서는 입력되는 단어의 양에 무관하며 단지 단어의 길이에 비례하는 검색시간을 갖고 있어 보다 효율적이다. 그러나, 이중 배열의 경우, 유형별 배열에 비하여 삽입에 소요되는 시간이 많이 걸린다는 단점이 있으나 실제 응용에서 문제가 될 정도는 아닌 것으로 사료된다.

마지막으로, 자소단위 방법은 음절단위 방법에 비하여 노드의 갯수가 약 3배로 증가하여 전체 기억 장소의 사용은 약 2.5배가 더 소요되고, 검색 시간 또한 약 2.5배가 더 걸리는 것을 알 수 있었다.

앞으로는, 실험 데이타를 어떤 분야에 국한하지 않고 임의로 선정한 데이타에 대해서도 실험할 필요가 있으며, 또한 주소들과 같이 문장을 저장할 때 단어들간의 계층구조를 효율적으로 이용하는 사전 구조에 대한 연구가 필요하다. 그리고, 이 논문에서는 사전의 엔트리를 음절이나 자소만으로 하였으나 여러 음절의 조합으로 이루어진 접두사 및 접미사를 하나의 엔트리로 취급할 수 있는 사전 구조를 생각할 수 있으며 이에 대한 연구도 필요하다.

참고문헌

1. E. Fredkin, "Trie Memory", *Commun. ACM.*, Vol. 3, No. 9, pp.490-500, 1960.
2. 金商雲, 青木 由直, "한글 인식 후처리용 단어사전의 기억구조", **한국통신학회 논문지**, Vol. 19, No. 9, pp.1702-1709, 1994.
3. 이승선 외 4명, "Compact TRIE Index(compTI) : 한국어 전자 사전을 위한 데이타베이스 색인 구조", **한국정보과학회 논문지**, Vol. 22, No. 1, pp.3-12, 1995.
4. 青江 順一 他, "トライ構造における 共通接尾辭の壓縮アルコリズム", **일본전자정보통신학회논문지**, Vol. J75-DII, No. 4, pp.770-779, 1992.
5. 신성효, 김상운, "DAWG에 의한 사전 구성 및 실험", **제6회 한글 및 한국어정보처리 학술발표논문집**, pp.207-210, 1994.
6. 김철수, 배우정, 이용석, "이중 트라이를 이용한 한국어 단어 검색", **제6회 한글 및 한국어정보처리 학술발표논문집**, pp.113-118, 1994.
7. 이성환, 김은순, "한글 주소의 오인식 수정을 위한 효율적인 후처리 알고리즘", **제4회 한글 및 한국어정보처리 학술발표 논문집**, pp.555-565, 1992.
8. 이영식, 채영숙, 윤애선, 권혁철, "한국어 철자 교정 시

스템", 인공지능 연구회 춘계 학술발표논문집, pp.25-37, 1993.

金 商 雲(Sang-Woon Kim) 정회원

1956년 3월 13일생
1978년 2월 : 한국항공대학교 통신정보공학과 졸(공학사)
1980년 2월 : 연세대학교 대학원 전자공학과 졸(공학석사)
1988년 2월 : 연세대학교 대학원 전자공학과 졸(공학박사)
1992년 12월~1993년 12월 : 일본 홋카이도대학 정보공학과 (Post-Doc.)
1984년 4월~1989년 3월 : 한국방송통신대학교 전자계산학과 조교수
1989년 4월~현재 : 명지대학교 컴퓨터공학과 부교수
※주관심 분야 : 패턴인식 및 학습

愼 晟 孝(Seong-Hyo Shin) 정회원

1970년 1월 25일생
1992년 2월 : 명지대학교 전자계산학과 졸(공학사)
1994년 2월 : 명지대학교 대학원 전자계산학과 졸(공학석사)
1994년 3월~현재 : 명지대학교 대학원 컴퓨터공학과 박사과정 재학중
※주관심 분야 : 패턴인식 및 학습