DAWG에 의한 한글단어사전의 구성 및 실험

신 성 효*, 김 상 운*
* 명지대학교 컴퓨터공학과

Implementation and Exprimentation for Hangul Word Dictionary via DAWG

Seong-Hyo Shin*, Sang-Woon Kim*
* Dept. of Computer Eng., Myongji univ.

요 약

한글 전자사전은 많은 양의 데이타를 저장할 수 있어야 하며, 빠른 검색 속도를 제공해야 한다. 기존의 트라이는 공통접두사만을 압축하기 때문에 사전의 크기가 방대하다는 단점이 있다. 본 논문에서는 DAWG(*Directed Acyclic Word Graph*)를 이용하여 공통접미사까지 압축하였고, 검색과 기억장소의 효율을 위하여, 링크드리스트 구조의 DAWG를 유형별 배열 구조로 바꾸었다. 전국의 각 학교 이름들을 대상으로 실험한 결과, 본 논문에서 제안한 DAWG를 이용한 배열 구조의 사전은 트라이와 비교하여 볼 때, 검색 연산의 성능은 동일하게 유지하면서 기억 장소의 효율과 압축율에서 효과적이었다. 또한, 트라이보다 주기억장치와 보조기억장치와의 블럭입출력횟수를 줄임으로써 전체 검색 시간을 낮출 수 있었다.

Ⅰ. 서론

문자인식에서의 후처리 및 문서처리의 철자교정, 도서목록의 탐색, 한자변환 등에서 이용하는 전자화사전은 고속검색이 중요한 요소가 되며, 이러한 사전의 기억구조로는 디지탈 탐색이 가능한 트라이(trie)가 적합한 구조로 알려져 있다. 그런데 트라이는 노드의 링크필드에 공백포인터가 많기 때문에 기억공간의 이용효율이 저조하다는 단점이 있다.

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 디스크로 부터 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.

따라서 이 논문에서는 한글단어 사전을 DAWG(*Directed Acyclic Word Graph*)[1]로 구성하여 공통접두사 뿐만아니라 공통의 접미사 까지도 통합하여, 대용량 사전을 고속으로 검색할 수 있도록, 사전의 화일 크기를 압축시킨다. 또한 구성된 DAWG를 배열구조[2]로 변환하여 저장함으로서, 보조기억장치와의 입출력 시 주소변환이 용이하도록하여 전체 검색 시간을 줄인다.

Ⅱ. DAWG의 구성

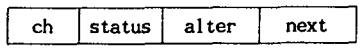
2.1 DAWG의 데이타 구조

DAWG는 공통의 접두사를 통합할 수 있는 트라이보다 검색시간을 저하시키지 않고 공통의 접미사 까지도 통합시킬 수 있는 저장구조로서, 트라이와 DAWG를 위한 노드구조는 <그림1>과 같다.

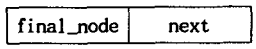
DAWG는 트라이의 특성을 만족시키면서 공통접미사를 통합시킬 수 있는 방법이다. 공통접미사를 검색하기 위해서는 우선, DAWG내의 문자열들의 가장 마지막 문자와 입력한 문자열의 마지막 문자가 같은가를 검색하여야 한다. 그런데 트라이 구조를 그대로 사용할 경우 마지막 문자를 검색하기 위하여는 DAWG내부를 모두 탐색하여야 만 한다. 따라서 수행시간의 효율을 위해 DAWG내의 가장 마지막 노드만을 가리키는 별도의 구조(end_node)를 준비한다(<그림1>(b)참조). 마지막 문자가 같으면 같지 않은 문자를 만날 때까지 백트랙하여 공통접미사

부분을 탐색한다. 따라서 트라이의 노드 구조에 백트랙 기능을 가진 노드(p_alter, p_next)를 추가한다(<그림1>(c)참조).

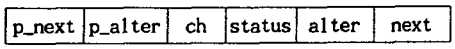
예를들어 단어집합 K = { 명지대학교, 명지전문대, 서울대학교, 서강대학교 }를 <그림1>의 구조로 구현한 트라이와 DAWG는 각각 <그림2>(a),(b)와 같다. 여기서 좌상단의 첫 번째 노드 <명>이 시작노드로서 root노드에 해당하며, 각 가지의 마지막 노드 <교>,<대>는 한 경로의 끝을 나타내는 노드로서 status필드가 'E'의 값을 갖는다. 또한, 노드 </>는 단지 앞뒤노드를 연결하기 위한 의사노드(dummy node)이다.



(a) 트라이의 노드구조

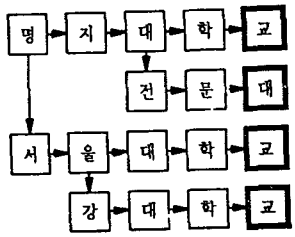


(b) end_node의 노드구조

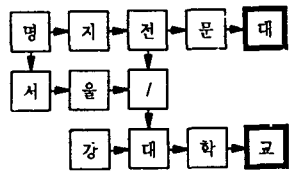


(c) DAWG의 노드구조

<그림 1> 트라이와 DAWG의 노드구조



(a)트라이



(b)DAWG

<그림 2> 2진링크드리스트로 구현한 트라이와 DAWG

2.2 DAWG의 연산

DAWG에서 사용하는 연산은 검색(search), 삽입(insert), 삭제(delete)가 있다[3].

DAWG 검색연산은 트라이의 검색연산과 같다. DAWG의 루트 노드와 검색할 문자열을 입력으로 받아, 현재노드(state)의 문자와 입력한 문자열의 해당 문자가 서로 같을 경우 next를 따라 전진하고, 같지 않을 경우는 alter를 따라 간다. 패스가 존재하지 않을 때 검색에 실패하게 된다.

삽입연산은 입력, 공통접미사의 검색, 그리고 합병 및 분리 단계로 나누어진다.

(1) 입력된 문자열을 DAWG내에 트라이의 삽입 알고리즘로 삽입한다.
(2) 공통접미사를 검색한다.
(3) 통합(merge)처리를 수행한다.
(4) 분리(split)처리를 수행한다: 단계(3)의 결과 입력하지 않은 문자열이 존재하는 경우가 있다. 따라서 이러한 부작용을 없애기 위해서는 분리 처리를 하여야만 한다.

삭제연산은 다음과 같이 세단계로 나누어 수행한다.

(1) 검색연산을 사용하여 삭제할 문자열에 해당하는 DAWG내의 경로를 찾아낸다.
(2) 삭제할 첫문자를 찾는다.
(3) 삭제할 마지막문자를 찾아 삭제한다.

DAWG 삽입 알고리즘에 의해 구성된 단어 사전은 링크드 리스트 구조를 갖는다. 주소 변환의 용이성과 불필요한 링크필드의 사용을 줄이기 위하여 링크드리스트 구조의 DAWG를 노드의 링크필드 갯수에 따라 서로다른 유형의 배열로 변환한다. 즉, 한 노드의 링크필드 두개를 모두 사용하고 있으면 유형_2(이하, A2라 함)에 저장한다. 링크 필드 하나만을 사용하고 있는 두가지 경우 중, 자식링크만 사용할 경우 유형_11(A11)에, 형제링크만 사용하고 있으면 유형_12(A12)에 저장한다. 그리고 하나도 사용하지 않는 단말노드는 유형_0(A0)에 저장한다. 단어집합 K에 대한 <그림 2>의 트라이와 DAWG를 배열로 변환한 구조는 <그림 3>과 같다.

A2

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

(배열명/인덱스)

A11

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

(a)트라이에 대한 배열 구조

A2

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

A11

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) DAWG에 대한 배열 구조

<그림 3> 트라이와 DAWG의 배열 구조

III. DAWG에 의한 사전구성 및 실험

3.1 실험 데이타

이 논문에서 제안한 DAWG알고리즘의 성능 평가를 위하여 세가지 데이타를 가지고 UNIX환경에서 실험하였다. 전국의 290개 대학명(이하 DATA 1이라 함), 경기도 관내의 초,중,고등학교명 1326개(DATA 2), 전국 유치원, 도서관, 초,중,고등학교명 16195개(DATA 3)를 데이타로 선정하였다. 데이타의 입력 단어 수, 전체 문자 수와 단어의 평균 길이를 <표 1>에 나타낸다. 이 세가지 데이타를 각각 입력으로 하여 DAWG로 사전을 구성한 다음, 네 종류의 배열구조로 변환하는 실험을 하였다.

<표 1> 입력 데이타의 특성

특성 / 데이타	단어수	문자수	평균단어길이
DATA 1	290	1884	6.4965
DATA 2	1326	8728	6.5822
DATA 3	16195	130192	8.0390

3.2 DAWG의 압축 효율

DATA 1, DATA 2, DATA 3를 각각 트라이로 구현하였을 경우와 배열의 크기를 비교한 결과는 <표 2>와 같다. 실험에 사용한 데이타에서는 A12의 경우가 발생하지 않기 때문에 <표 2>의 내용에서 제외하였다.

<표 2> 배열 크기 비교

BYTE

구현방법 \ 배열유형		A2	A11	A0	합계
DATA 1	화일크기				4636
	트라이	2556	5904	855	9315
	D A W G	2556	2562	9	5127
DATA 2	화일크기				17879
	트라이	11871	22524	3960	38375
	D A W G	11871	11088	24	22983
DATA 3	화일크기				
	트라이	116298	310494	38745	465537
	D A W G	116298	153102	96	269493

트라이와 DAWG가 공통으로 가지는 특성은 공통접두사의 압축이다. 이 압축 과정에서 생겨나는 노드 타입은 alter와 next를 모두 가지는 노드이다. <표 2>의 내용을 살펴보면, A2의 경우 트라이와 DAWG가 같은 수의 엔트리를 가진다. 반면에 DAWG에서만 가지는 특성인 공통접미사의 압축으로 인하여 A11과 A0의 경우 기억 장소의 효율에 있어서 많은 효과를 볼 수 있다. DAWG의 A11의 크기는 트라이의 절반 가량으로 줄어들었고, A0는 입력된 문자열의 마지막 문자의 종류로 줄어들었다. DAWG를 사용한 경우 배열의 크기는 전체 5127바이트이고, 트라이를 사용한 경우는 9315바이트이다. 따라서 DAWG는 트라이의 55.04%의 배열 크기만을 사용함으로써 기억 장소의 사용에 있어서 44.96%가 효율적이다. DATA 2에서는 40.11%, DATA 3에서는 57.89%를 절약할 수 있어 효과적이었다.

DATA 1의 경우, 입력한 데이타의 실제 크기는 4636바이트이다. 이 데이타를 트라이로 구축하였을 경우, 링크 필드와 상태 필드의 존재로 크기가 약 두 배로 증가하였다. DAWG로 구축하였을 경우, 공통된 문자열들을 압축시킴으로써 크기가 입력한 데이타 화일의 크기에 접근하였다.

입력한 데이타를 DAWG로 사전을 구축하였을 때, 입력한 문자열에 대하여 압축율은 데이타의 종류에 따라 그 정도가 상이하다. 트라이와 DAWG를 이용하여 사전을 구축하였을 때의 노드 수와 입력 데이타의 문자수를 비교하여 입력한 데이타들의 종류에 따른 압축율을 살펴보면 <표 3>과 같다. 압축율은 아래와 같이 구하였다.

$$\text{압축율} = \frac{\text{데이타 문자수} - \text{노드수}}{\text{데이타 문자수}}$$

<표 3> 데이타 종류별 압축율 비교

데이타 \ 비교		노드수	압축율
DATA 1	트라이	1553	0.1757
	DAWG	714	0.6210
DATA 2	트라이	6393	0.2675
	DAWG	3175	0.5744
DATA 3	트라이	77586	0.4041
	DAWG	38471	0.7045

트라이의 경우는 공통접두사의 압축 효과로써 평균 0.2824의 압축율을 보이고, DAWG는 공통접두사의 압축율과 공통접미사의 압축율을 합하여 0.6333의 압축율을 보였다. 트라이에 비해 평균 0.3509의 압축 효과를 볼 수 있었다. DATA 1의 경우, 문자열의 수는 290개이며, 전체 문자의 수는 1,884개이고, 따라서 문자열 당 평균 단어의 길이는 6.4965이다. 트라이를 이용하여 사전을 구축하였을 경우, 전체 노드의 수는 1,553개로서 331개의 문자가 공통접두사로써 압축되었다. DAWG를 사용할 경우, 전체 노드수는 741개이고 문자열 당 평균 2.5552개의 노드를 필요로 한다. 공통접두사의 압축은 트라이와 같으므로 331개의 문자가 공통접두사로써, 831개의 문자가 공통접미사로써 압축되었다. 따라서 DATA 1의 경우 0.6210, DATA 2의 경우는 0.5744, DATA 3의 경우는 0.7045로 압축되었다.

3.3 입출력 효율

주기억장치와 보조기억장치의 입출력 시간은 탐색시간의 대부분을 차지하므로, 사전의 고속검색은 보조기억장치와의 입출력 횟수를 줄임으로써 가능하다. 따라서, 네가지 유형의 배열을 일정한 크기의 블럭으로 나누고, 네개의 배열에 대하여 각각 번갈아 가면서 검색에 필요한 해당 블럭들을 주기억장치에 적재하면서 검색을 실행하였다. DATA 1을 가지고 구성한 사전에 대하여, 블럭의 크기를 달리하여 주기억장치와 보조기억장치의 블럭 입출력 횟수를 비교한 결과를 <표 4>에 나타내었다. 배열마다 5개의 블럭을 주기억장치에 적재시킨 후, LRU에 의하여 블럭 교환을 실행하였다.

<표 4>에서 보듯이, 배열의 크기가 작을수록 블럭의 입출력 횟수는 줄어들게 된다. 트라이와 DAWG의 A2는 크기가 서로 같기 때문에 블럭의 입출력 횟수가 같다. 그러나, DAWG의 A11

<표 4> 블럭 크기별 입출력횟수 비교

배열유형 / 블럭크기	트라이				DAWG			
	A2	A11	A0	전체	A2	A11	A0	전체
10	927	292	183	1402	927	199	0	1126
20	465	219	136	820	465	141	0	606
30	275	198	97	570	275	117	0	392
40	175	165	87	427	175	71	0	246
50	69	147	55	271	69	62	0	131

은 트라이보다 크기가 작고, 공통접미사가 자주 참조되기 때문에 항상 메모리에 적재되어 있으므로, 블럭의 입출력 횟수가 트라이에 비해 평균 50%가 작아졌다. DAWG의 A0는 크기가 블럭의 크기보다 작기 때문에 입출력이 일어나지 않았다.

사전의 크기가 커질수록 블럭의 입출력 시간은 전체 검색 시간에 많은 영향을 미친다. 검색할 문자열의 평균 검색 시간은 주기억 장치 내의 사전 검색 시간과 블럭 입출력 시간을 합한 것이 된다. 트라이와 DAWG의 검색 연산은 서로 같기 때문에 평균 사전 검색 시간은 같다. 그러나 블럭 입출력 횟수는 DAWG가 훨씬 작기 때문에 전체 검색 시간에 대해서는 DAWG가 트라이에 비해 30% 효율적이다.

3.4 연산시간 및 성능분석

3.4.1 검색 시간

입력된 문자열을 검색하는 시간은 alter링크를 따라가는 시간에 next링크를 따라가는 시간을 더한 것이다. 여기서 next링크를 따라가는 시간은 입력한 문자열의 길이와 같다. 트라이와 DAWG에서 A2의 엔트리 갯수가 같기 때문에 alter링크의 갯수는 동일하다. 따라서, 트라이와 DAWG의 평균 검색 시간은 동일하다.

3.4.2 삽입 시간

DAWG를 배열로 변환할 경우 DAWG의 노드 구조 중, 백트랙을 하기 위한 p_next와 p_alter 링크필드는 배열에 저장하지 않는다. 따라서 배열 구조로 변환된 DAWG는 공통접미사를 찾기 위하여 백트랙을 할 수 없다. 그런데 공통접미사를 탐색하기 위하여 사전의 모든 내용을 검색하는 것은 더욱 비효율적이므로, 트라이의 삽입 알고리즘을 그대로 사용하여 공통접미사를 압축하지 않은 상태로 저장한다. 왜냐하면 백트랙링크를 배열 구조에 첨가하는 것보다 압축을 하지 않는 것이 효율적이기 때문이다. 사전을 구축한 이후 대량의 문자열을 사전에 삽입하려 할 경우 트라이에 대비되는 DAWG의 압축효율은 저하된다.

3.4.3 삭제 시간

배열 구조에서 하나의 문자를 삭제하기 위하여 해당 배열 요소를 삭제하고, 그 이후의 배열 엔트리를 이동하는 것은 공간적인 효율에 비해 시간적인 비효율이 더욱 크다. 즉, 삭제 연산은 배열 요소의 삭제가 아니라 해당 배열의 링크 필드를 없애는 것이다. 공통접두사의 마지막 배열 요소의 링크 필드를 없앰으로써 그 이 후의 배열 요소를 액세스하지 못하도록 하면 된다. 따라서 트라이에서와 같은 시간을 소비하면서 문자열을 삭제할 수 있다.

IV. 결론

본 논문에서는 기존의 트라이에 공통접미사의 압축 알고리즘을 추가한 DAWG를 구현하여 사전의 크기를 압축하였다. 검색효율과 기억장소의 효율성을 기하기 위해 링크드리스트로 구현된 DAWG를 유형별 배열 구조로 바꾸어 이용하였다. 트라이와 DAWG의 성능을 상호 비교하기 위하여 기억 장소 효율, 압축율, 블럭 입출력 횟수의 세가지 파라메타를 사용하였다. DAWG를 트라이 경우와 비교해 볼 때 검색 시간은 동일하게 유지하면서, 세가지 파라메타 중, 기억 장소의 사용은 평균 47.65%가 효율적이었고, 블럭 입출력 횟수는 평균 30%, 평균 압축율은 0.3509가 효율적이었다. 그리고, 대용량의 사전을 구축하였을 경우, 보조기억장치와의 입출력 횟수가 줄어들게 되어 전체 검색 시간은 줄어들었다.

DAWG의 배열 구조에서 검색과 삭제 연산은 DAWG의 압축된 구조를 변형시키지 않고 실행이 된다. 그러나 삽입 연산의 경우 백트랙이 불가능하므로, 삽입되는 문자열은 공통접미어가 압축되지 않은 상태로 배열에 저장된다.

이에 배열구조에 백트랙킹 기능을 추가하여 삽입 시에도 공통접미사가 압축될 수 있는 배열의 구조가 연구되어져야 하겠다.

참 고 문 헌

1. 青江 順一, 他 "トライ構造 共通接尾辭の壓縮 ", 電子情報通信學會論文誌, Vol.J75-DII, No.4, pp.770-779, 1992.
2. 이승선 외, "TRIE구조를 이용한 한국어 전자사전을 위한 데이타베이스 인덱스구조", 한국정보과학회 학술발표논문집, Vol.21, No.1, pp.849 - 852, 1994.
3. 신성효,이성희,김상운,"DAWG에 의한 한글단어사전의 압축방법", 한국정보처리응용학회 추계학술발표논문집, Vol.1, No.2, pp.91 - 94, 1994.