효율적인 한자 순위 변환과 복합한자 변환 알고리즘

이태헌**, 박기홍*
군산대학교 컴퓨터과학과

An Efficient Algorithms of HANJA Conversion Program and a Compound HANJA Conversion.

Tai Heon Lee, Ki Hong Park
Dept. of Computer Science, Kunsan National University

요 약

한글 워드프로세서의 한자 사용이 필수적인 요소이다. 일상 생활에서 의사를 전달하기 위해서는 한국어로 사용하지만, 글이나 작문을 이용하여 의사를 전달할 때는 한글이 주를 이루고 문맥 안에 함축된 의미로 한자를 사용하면 정확하게 의사가 전달된다. 현재의 한글 워드프로세서에서는 한자 순위 변환 시스템의 키의 검색 횟수가 많고 또한 복합한자 변환 시스템에서는 실행 횟수가 많은 단점을 가지고 있다. 본 논문에서는 보다 효율적인 키의 검색 횟수를 줄이기 위해 효율적인 한자 순위 변환 알고리즘과 실행횟수를 줄이기 위해 개선된 복합한자 변환 알고리즘을 사용한다. 이러한 알고리즘은 검색과 실행 속도를 빠르게 함은 물론 한글·한자 변환시스템 처리에 도움을 줄 수 있다.

1. 서론

일상 생활에서 한자의 비중은 과거에 비하여 많이 줄어들고 있다. 그러나 일부 문장에서 의미를 함축적으로 표현하는 제목이나 문장의 중간 중간에 한자 단어들이 사용하는 것을 쉽게 볼 수 있다. 그러나 기존의 한글 워드프로세서에서 한자 변환 시스템에서 사용함으로서 매우 불편함 점을 발견하여 본 논문은 한글 워드프로인 "한글 프로 96"의 한자 변환 시스템 체제를 기본으로 하였다. 한글 워드프로세서에서는 찾고자 하는 한자가 메뉴의 맨 상위가 아닌 맨 하위에 있으면 매번 그 한자를 찾고자 마우스나 키보드로 이동하여 실행을 하든지 그렇치 않으면 마우스를 이용하여 메뉴의 맨 상위로 이동해서 사용해야 하는 불편한 점과 또한 복합명사의 경우 수동으로 등록을 하지 않고 다음에 사용하면 변화가 없이 그대로 나타나기 때문에 수동으로 등록을 하여서 사용해야하는 불편한 점이 있다.

본 논문에서는 이러한 문제점을 해결하기 위하여 두 가지 알고리즘이 사용한다.

첫째는 한자 순위 변환 알고리즘으로 메뉴의 최하위 한자를 실행하여도 다음에 재실행하면 최상위의 메뉴로 이동하게 된다. 즉 한번 실행한 한자는 메뉴의 어디에 위치했던 재실행하면 메뉴의 최상위인 첫번째에 위치하다.

둘째는 복합 한자 변환 알고리즘으로 명사와 명사를 붙여서 사용하면 수동으로 등록하지 않아도, 자동으로 복합 한자로 등록 변환되어 사전에 등록되어 다음 실행하고자 할 때에 자동으로 나타나게 된다.

예를 들어 한자 순위 변환 알고리즘은 한글 워드프로세서에서 한자사전에 등록된 순서로 "가사"라는 단어에서 "佳詞"는 1번째이고 "歌詞"는 11번째일 때 11번째인 "歌詞"를 사용하고 다시 실행하면 변화없이 "歌詞"가 11번째 위치하나 한자 순위 변환 알고리즘을 사용하면 "歌詞"가 1번째이고 "佳詞"는 2번째로 바뀌게 되어 바로 엔터에 의해서 실행할 수 있고, 또한 한글 워드프로세서에서 "부산시"의 경우 "부산+시"로서 "부산시"를 수동을 등록해야 만이 다음 실행 할 때부터 "부산시"가 "釜山市"로 나타나게 되나 복합 한자 알고리즘의 경우는 "釜山"과 "市"를 공간없이 사용하기만 하면 다음 실행 할 때부터 "釜山市"가 자동으로 등록되어 나타나게 되어서 한자변환 알고리즘의 경우에서는 적은 검색 횟수가 사용되고 복합한자 변환 알고리즘의 경우에서는 적은 실행 횟수가 사용된다. 이를 위해서 한자는 일상 생활에서 사용하는 상용한자 3,000자를 대부분 포함하는 KS C 5601 코드(4,888자)를 기준으로 하였다.

본 논문의 구성은 2절에 한자 변환 시스템의 기본 체제를 3절에서는 한자 순위 변환과 복합한자 변환 알고리즘을 4장 실험 및 평가에서는 기존의 변환 시스템과 본 논문의 변환 알고리즘을 비교 분석하고 5장에서는 결론으로 본 논문 알고리즘의 효율성과 한자 변환에 활용되는 측면을 설명한다.

2. 한자 변환 시스템의 기본 체제

현재 한자 변환 시스템의 기본 체제 중 2가지를 살펴보면 한자 변환 시스템에서의 위치 변환이고 다른 하나는 복합한자 변환 시스템에서 복합명사 처리이다.

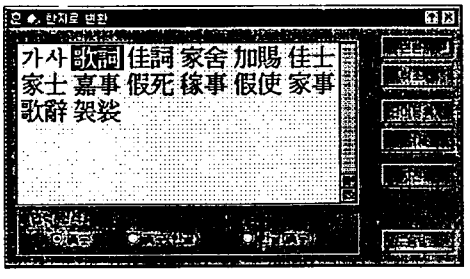
2.1 한자 변환 시스템에서의 위치 변환

현재의 한자 변환 시스템에서 한자를 사용할 때 메뉴의 중간 또는 최하위에 위치한 한자를 실행할 경우나 차후에 같은 한자를 재실행하여도 변화가 없이 실행되었던 장소에 위치하게 된다. 즉 한번 실행한 한자는 메뉴의 어디에 위치했던 다시 실행하면 메뉴의 그 장소의 그위치에 위치하기 때문에 마우스를 이용하여 최상위로 이동하여 사용해야 다음 실행 할 때부터 최상위에 나타나게 된다. 그렇지 않으면 매번 사용했던 위치로 가서 선택해야 할 한자를 마우스나 키보드로 실행을 해야하는 어려움이 발생하게 된다.

예를 들어보면 "가사"란 단어 중에서 "노랫말"이라는 뜻을 가진 한자어를 찾을 경우에 대해서 알아보면 <그림 2.1>과 같이 나타나게 된다. 여기서 "노랫말"이라는 뜻을 가진 한자어 "歌詞"의 위치는 메뉴에서 11번째에 위치해 있음을 알 수 있고 실행하기 위해서 마우스나 키보드로 이동하여 "歌詞"를 찾아서 실행한다. 키보드의 경우에는 종으로 1회 횡으로 5회를 총 6회를 이동하여 찾아서 실행해야 한다.<그림 2.1> 그런 다음 차후에 같은 한자로 노랫말이라는 "가사"를 재실행하여도 변화가 없이 <그림 2.1>과 같이 그 장소에 다시 위치하게 되어 불편성이 제공된다.

<그림 2.1 노랫말이라는 "歌詞"의 첫 실행 위치>

그렇지 않으면 수동으로 "歌詞"란 단어를 마우스로 메뉴의 최상위에 옮겨 놓아야(드래그) 다음에 실행할 때는 "歌詞"가 메뉴의 최상위에 위치한다. <그림 2.2> 그리고 다음부터 "가사"를 재실행하면 <그림 2.2>와 같이된다 [1].



<그림 2.2 마우스로 이용한 상태>

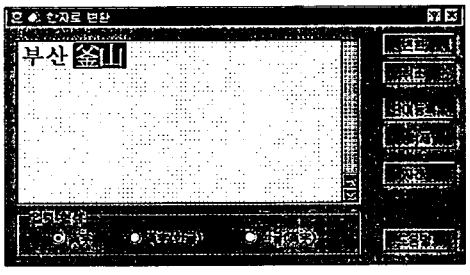
만약 마우스로 "歌詞"를 옮겨 놓지 않고 사용을 하면 다음에 재실행을 했을 경우는 앞에서의 경우와 마찬가지로 다시 "歌詞"를 찾기 위해서 키보드로 종 1회 횡으로 5 총 6회를 이동하거나 마우스로 이동하여 사용한다는 점에서 사용자에게 매번 사용할 때마다 키보드나 마우스로 이동하여 그 한자에 맞는 한자를 실행을 하거나 그렇지 않으면 사용할 한자를 마우스로 드래그를 해야하는 불편성이 있는 것이 단점이다. 그래서 한자 순위 변환할 때 키의 검색 회수를 줄여 바로 실행할 수 있는 개선책이 필요하다.

2.2 한자 변환 시스템에서의 복합한자 변환

한자 변환 시스템에서의 복합한자 변환할 때 복합 명사를 처리하는 방법 두 가지가 있다. 첫째는 등록을 하지 않고 각각 명사를 한자로 처리하여 등록을 하지 않고 사용하는 방법과 두 번째는 명사와 명사를 "한자 단어 등록"에 수동으로 등록하여 다음 실행할 때부터 수동으로 등록되어 있는 사전에서 그 복합한자를 실행하는 방법이다. 만약에 한자 사전에 등록을 하지 않고 사용을 하게 되면 매번 사용 할 때마다 명사와 명사를 각각 실행해야 한다는 불편성이 있는데 본 논문에서는 후자의 경우 한자 단어 등록을 해서 사용하는 경우에 대해서 알아본다.

예를 들어 "부산시"를 등록 하고자 할 때의 "한자 단어 등록"에 수동으로 등록하는 경우를 알아보자.

다음 <그림 2.3>은 "부산시"를 수동으로 등록하고자 할 때 "부산시"를 실행 할 때의 첫 화면이다[1].



<그림 2.3 "부산시"를 실행 할 때의 첫 화면>

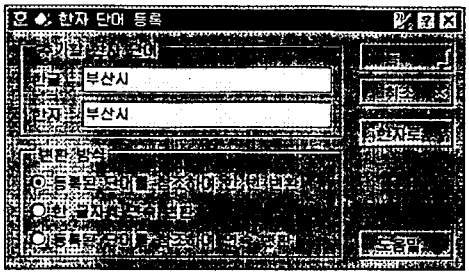
다음의 <그림 2.4>는 부산시를 수동으로 한자 단어를 등록하고자 할 때의 두 번째 화면이다[1].

다음은 한글 프로 96의 워드프로세서에서 "부산시"를 수동으로 등록하여 다음 실행 할 때부터 자동으로 등록되어 나타낼 수 있는 3가지의 방법을 설명한다.

첫째는 등록된 단어를 참조하여 한 번 변환된 경우이며,

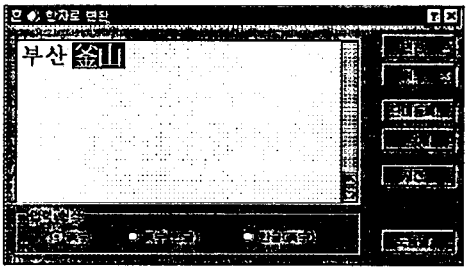
두번째는 한 글자씩 연속 변환한 경우이고,

세 번째는 등록된 단어를 참조하여 연속 변환 경우 등이 있는데 본 논문에서는 편의상 보편적으로 사용자가 많이 사용하는 첫번째의 경우 등록된 단어를 참조하여 한번 변환된 경우에 대해서 알아본다.



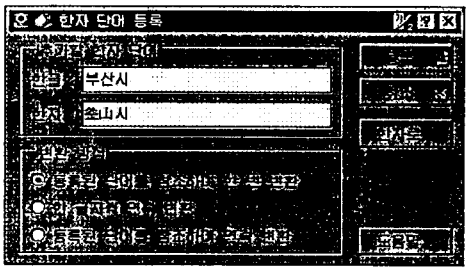
<그림 2.4 "부산시"를 수동으로 실행 할 때의 둘째 화면>

다음 <그림 2.5>는 "부산시"를 수동으로 한자 단어를 등록하고자 할 때 "부산시"를 수동으로 아래의 세 가지 메뉴 중 첫 번째의 경우 등록된 단어를 참조하여 한번 변환된 경우를 선택하여 등록하고자 할 때의 "부산"만 나타낸 화면의 상태이다[1].



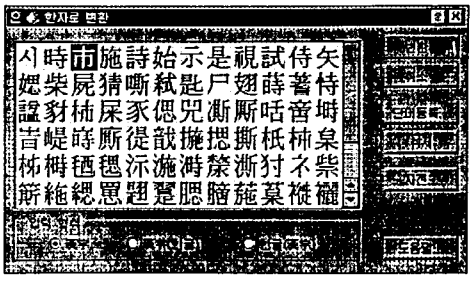
<그림 2.5 "부산시"를 수동으로 실행 할 때의 셋째 화면>

다음 <그림 2.6>은 "부산"을 등록하고 난 다음 "시"를 사전에 등록하는가를 물어 찾아서 "시"를 자동으로 등록되게 하는 화면 상태를 나타내고 있다.



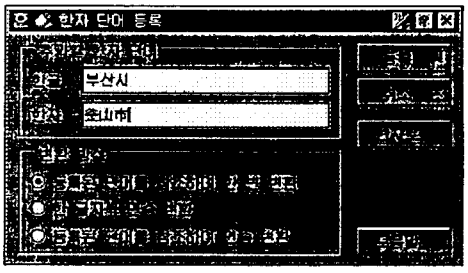
<그림 2.6 "부산시"를 수동으로 실행 할 때의 넷째 화면>

다음 <그림 2.7>은 "부산"을 등록하고 난 다음 "시"를 사전에서 찾아가서 "시"를 자동으로 등록되게 하는 화면 상태를 나타낸 것인데 2.1절에 한자 순위 변환 시스템의 경우와 마찬가지로 키보드의 경우는 횡으로 1회를 이동하거나 마우스의 경우는 "市"에 맞는 한자를 찾아서 실행하고 있는 상태의 화면이다.



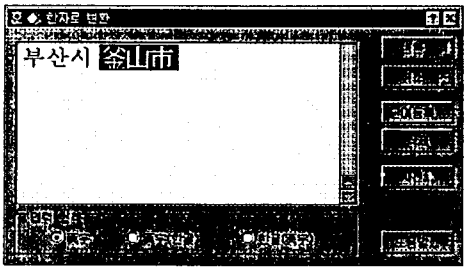
<그림 2.7 "부산시"를 수동으로 실행 할 때의 다섯째 화면>

다음 <그림 2.8>은 "부산시"를 수동으로 한자 단어 사전에 등록하기 위해서 마지막으로 사용자에게 물어보는 상태의 화면이다[1].



<그림 2.8 "부산시"를 수동으로 실행 할 때의 여섯째 화면>

다음 <그림 2.9>는 "부산시"를 수동으로 한자 단어 사전에 최종적으로 등록되어 다시 한번 사전에 등록 할 것인가를 물어봐서 등록하는 화면의 실행 상태이다[1].



<그림 2.9 "부산시"가 수동으로 완성된 화면>

앞의 경우를 각각 실행 횟수를 검사해 보면 등록된 단어를 참조하여 한 번 변환 할 때 실행 횟수 6번이고 한 글자씩 연속 변환 할 때 실행 횟수 7번이며 등록된 단어를 참조하여 연속 변환시 실행 횟수 6번이었으며, 등록을 하지 않고 사용할 경우에는 명사와 명사 그리고 복합명사를 실행해 실행 횟수가 3번 임을

알 수가 있다.

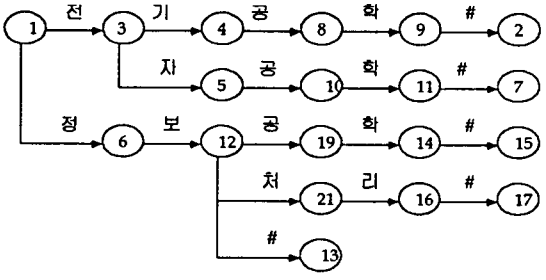
위에서 볼 수 있듯이 한자 순위 변환 시스템에서의 복합 한자의 복합명사 처리는 수동으로 한자의 단어를 등록해야 하고 또 등록하는데 실행 횟수가 최소 6번이거나, 등록을 하지 않고 사용할 경우에는 명사와 명사 그리고 복합명사를 실행해 실행 횟수가 3번임을 알 수가 있게 되나 반면 다음에도 같은 복합 한자를 실행하게 되면은 다시 명사와 명사 그리고 복합명사를 재실행하여야 하기 때문에 다시 한번 실행 횟수가 3으로 늘어나므로 사용자에게는 실행횟수가 많아지므로 사용을 하는데 매우 불편함을 느낀다. 그래서 한자 변환 시스템에서의 복합명사 처리는 등록횟수를 줄이고 다음 같은 복합 한자를 사용할 때 자동으로 등록 될 수 있는 개선책이 필요하다.

3. 한자 순위 변환과 복합 한자 알고리즘

3.1 한글 사전의 구성

본 장에서는 한자 순위 변환과 복합 한자 알고리즘을 구현하기 위해서 한자 사전을 트라이에 의한 이중 배열을 이용해 구현하는 방법이 있다[5,6].

STATE : 현재 처리중인 상태번호
t : 전이선의 상태번호
POS :입력 문자열의 현재 처리 위치를 표시한다.



<그림 3.1 트라이>

문자	#	전	기	자	정	보	공	학	처	리	사
내부 표현치	1	2	3	4	5	6	7	8	9	10	11

index	1	2	3	4	5	6	7	8	9	10
BASE	1	0	1	1	3	6	0	1	1	3
CHECK	21	9	1	3	3	1	11	4	8	5

index	11	12	13	14	15	16	17	18	19	20	21
BASE	6	12	0	14	0	16	0	0	6	0	6
CHECK	10	6	12	19	14	21	16	0	12	0	12

<그림 3.2 트라이에 대한 더블 배열>

그림 3.2는 리스트 구조에 대한 더블 트라이에서의 키 「정보공학#」의 검색을 나타낸다.

● 더블 배열에 의한 검색 알고리즘

```
절차(1) : { 변수의 초기화 }
    STATE ← 1 ;
    POS ← 0 ;
절차(2) : { 트라이 검색 }
    repeat
        POS ← POS + 1;
        t ← BASE[STATE] + apos;
        if(t > CHECH[1] or CHECH[T]  STATE)
            then return(FALSE);
        STATE ← t;
    until (BASE[STATE] ≤ 0);
    return (TRUE);
```

● 그림 3.2의 더블배열에서 키 「정보공학#」의 검색은 다음과 같다.

```
절차(1) : { 변수의 초기화 }
  검색에 사용되는 변수를 아래와 같이 초기화한다.
    STATE ← 1 ;
    POS ← 0 ;
절차(2) : { 트라이 검색 }
  트라이의 초기 상태 1부터 시작하면
        POS ← POS + 1 = 0 + 1 = 1;
        t ← BASE[STATE] + apos; = BASE[1] + a1
                                = BASE[1] + 전
                                = 1 +2
                                = 3
        CHECH[t] = CHECH[3]
                 = 1
                 = STATE 가 확인되고
        g(1, '전') = 3
된 전이가 정의되어 있는 것을 알 수 있다. 다음에
        STATE ← t = 3;
        BASE[STATE] = BASE[3]
                    = 1 ≠ 0
으로 되기 때문에 계속해 더블 배열을 사용한 검색을 행한다.
        BASE[3] + '기' = 1 + 3 = 4;
        CHECK[4] = 3;
        BASE[4] = 1 ≠ 0;
        BASE[4] + '공' = 1 + 7 = 8;
```

CHECK[8] = 4;
BASE[8] = 1 ≠ 0;
BASE[8] + '학' = 1 + 8 = 9;
CHECK[9] = 8;
BASE[9] = 1 ≠ 0;
BASE[9] + '#' = 1 + 1 = 2;
CHECK[2] = 9;
BASE[2] = 0;

로 되기 때문에

g(3, '기')=4, g(4, '공')=8, g(8, '학')=9, g(9, '#')=2

된 천이가 순서적으로 확인되고, BASE[2]=0에서, 상태 2가 최종 상태이기 때문에 검색은 성공하게 되며, return(TRUE)를 행한다.

3.2 한자 순위 변환 알고리즘

이번에는 사용하고자 하는 한자 단어를 실행만 하면 한자의 위치가 자동으로 메뉴의 어디에 위치했던 그 한자가 메뉴의 맨 상위에 위치하도록 바뀌어 줌으로서 다음 실행 할 때부터 검색 속도를 빠르게 하는 알고리즘이다.

● **한자 순위 변환 알고리즘**

한자 사전내의 단어 구성은 연결 리스트로 구현한다.
BASE[STATE] - 마지막 "#"의 BASE 값의 음의 값을 가진다

[노드의 첫 주소지]

NODE.left -BASE[STATE];
NODE.right NextAddress;
NODE HanjaCode;

해당 한자 단어의 첫 주소와 종결을 표시하기 위해 한글 사전의 종결 표시인 "#"의 BASE값은 첫번째 주소지 값인 음의 값이 저장한다.

[수정된 트라이 검색]

```
repeat
        POS ← POS + 1;
        t ← BASE[STATE] + apos;
        if(t > CHECH[1] or CHECH[T]  STATE)
            then return(FALSE);
        STATE ← t;
until (BASE[STATE] ≤ 0);
return (TRUE);
```

순위 변환은 연결된 주소를 바꿈으로 조정한다.

Node1 : 탐색 순위가 내려갈 노드
Node2 : 탐색 순위가 올라갈 노드
Node2.left ↔ Node1.left
Node2.right ↔ Node1.right

예를 들어보면은 "가사"란 단어 중에서 "노랫말"이라는 뜻을 가진 한자어를 찾았을 경우 키보드의 경우에는 종으로 1회 횡으로 5회 총 6회를 이동하여 찾거나 마우스로 이동해 찾아야 한다. 다음에 가사를 재실행 할 때 한글워드프로세서의 경우는 앞에서 말한 동작을 다시 해야한다. 즉 키보드로 최소 6회 마우스로 1회를 이동해서 실행해야하나 한자 순위 변환 알고리즘에서는 "歌詞"가 제일 상위에 위치해 있기 때문에 키보드나 마우스의 이동이 없이 바로 실행하면 되므로 키의 검색 횟수를 0으로 줄일 수 있어서 한자 순위 변환 알고리즘은 검색 횟수를 줄여서 검색하는 속도를 빠르게 하는 장점이 있다.

3.3 복합한자 변환 알고리즘과 복합명사 두음법칙 처리

3.3.1 복합한자 변환 알고리즘

이번에는 한자 단어와 한자 단어를 공간 없이 사용했을 경우 수동으로 등록하지 않고 자동으로 등록되어 등록된 복합 한자의 실행 횟수를 빠르게 하는 복합 한자 알고리즘을 제안한다.

● **복합한자 변환 알고리즘**

먼저 복합 한자에 해당하는 한글 단어를 이중 배열에 추가한다. 복합 한자를 하나의 단어와 같이 처리하는 추가 알고리즘이다.

[절차 BRANCH(r, ax)]

```
(a-1)  t ← BASE[r] + a;
(a-2)  if CHECK[t] ≠ 0 then
         begin
(a-3)      LISTR ← SET-LIST(r);
(a-4)      LISTH ← SET_LIST(CHECK[t]);
(a-5)      if N)LISTH) > N(LISTR) + 1
           then
(a-6)        r ← MODIFY(r, r, a, LISTR)
           else
(a-7)        r ← MODIFY(r, CHECK[t], ∮, LISTH);
         end;
(a-9)  INS_STR(r, ax)
(절차 LH_INSERT 끝)
```

[함수 MODIFY(current, r, a, LIST)]

```
(m-1)  oldbase ← BASE[r];
(m-2)  BASE[r] ← X_CHECK(LIST ∪ {a});
       for each c in LIST do
         begin
(m-3)      t ← oldbase + c;
           t' ← BASE[r] + c;
```

```
(m-4)      CHECK[t'] ← r;
           BASE[t'] ← BASE[t];
(m-5)      if BASE[t] > 0 then
             begin
(m-6)          for each q such that CHECK[q[ = t do
(m-7)                        CHECK[q] ← t';
(m-8)          if t = current then current ← t';
             end;
(m-9)      BASE[t] ← 0;
           CHECK[t] ← 0;
         end;
       return(current)
(함수 MODIFY 끝)
```

[절차 INS_STR(r, b_1b_2 ..... b_m)]

```
(s-1)      t ← BASE[r] + b1;
           CHECK[t] ← r;
           r ← t;
(s-2)      for I := 2 to m do
                 begin
                   BASE[r] ← X_CHECK({b1});
                   t ← BASE[r] + b1;
                   CHECK[t]  ← r;
                   r ← t;
                 end;
(절차 INS-STR 끝)
```

복합 한자를 하나의 단어와 같이 처리하는 추가 알고리즘은. 추가한 한글 단어의 종결 표시인 "#"의 BASE값에 새로운 Linked List의 첫번째 주소지의 값이 음의 값으로 저장한다.

```
[수정된 트라이 검색]
repeat
        POS ← POS + 1;
        t ← BASE[STATE] + apos;
        if(t > CHECH[1] or CHECH[T]  STATE)
            then return(FALSE);
        STATE ← t;
until (BASE[STATE] ≤ 0);
return (TRUE);
```

새로운 Linked List에 추가한 한글 단어에 해당하는 복합 한자를 등록하고 등록 할 때 첫 주소지는 BASE " # "의 마지막 BASE 값이 음의 값을 가진다.

```
[노드의 첫 주소지]
NODE.lfet  -BASE[STATE];
NODE.right  NextAddress;
NODE  HanjaString
```

복합 한자의 순위 변환도 연결된 주소를 바꿈으로 조정한다.

Node1 : 탐색 순위가 내려갈 노드
Node2 : 탐색 순위가 올라갈 노드
Node2.left ↔ Node1.left
Node2.right ↔ Node1.right

예를 들면 "부산시"를 한글 워드프로세서에서 "부산시"를 수동으로 등록 할 수 있는 3가지의 경우의 실행 횟수는 첫째는 한 번 변환 할 때의 실행 횟수는 6번이고, 둘째 한 글자씩 연속 변환 할 때의 실행 횟수는 7번이며, 셋째 등록된 단어를 참조하여 연속 변환시의 실행 횟수는 6번이다. 즉 실행수가 적어도 최소 6번으로 등록을 하든지 아니면 등록을 하지 않고 단어와 단어를 실행하여 사용한다. 그리고 다음 사용 할 때는 전에 사용했던 복합 한자가 나타나면 등록을 해서 사용하든지 단어와 단어를 실행하여 사용하든지 해야하는데, 복합 한자 알고리즘을 사용한 경우에는 등록을 하지 않고 단어와 단어를 실행하기 때문에 실행 횟수가 3으로 사용하였을 알 수 있고, 다음부터 "부산시"를 재실행할 때 완전한 "釜山市"가 실행되므로 한자를 등록해야 하는 불편한 점을 제거하였을 뿐만 아니라 복합 한자를 검색하는 속도를 빠르게 한다는 장점이 있다.

3.3.2 복합명사의 두음법칙 처리

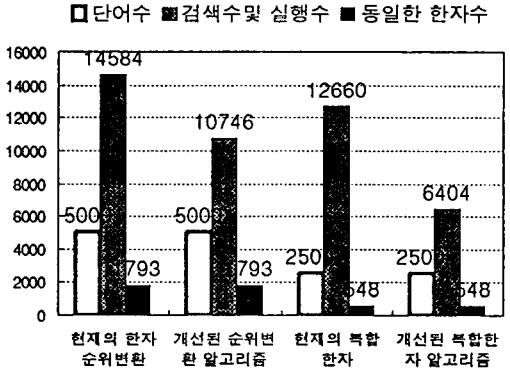
복합 한자를 처리할 때 두음법칙에 의해서 한글에서 한자의 변환은 맞게 변환이 되는데, 반해 한자에서 한글로 변환은 두음법칙이 적용되어 실패를 하게된다.

예를 들어 설명을 하면은 한글에서 한자로 변환하는 경우의 "연간"과 "년간"을 한자로 변환하면 "연간"은 "年間"으로 년간도 "年間"으로 맞게 변환이 되는데 한자에서 한글로 변환하는 경우의 "연간"인 "年間"은 연간으로 되지만 년간인 "年間"은 "연간"으로 잘못 실행된다. 그래서 본 논문은 이중배열을 이용한 한글 사전에서 한자 사전 사전을 구축 할 때 세 가지의 테이블 값이 있는데 첫 번째의 테이블 값, 두 번째의 테이블 값, 세 번째의 테이블 값이 있어 첫 번째의 테이블 값은 한글의 코드 값이고, 두 번째의 테이블 값은 한자 수의 값이고, 세 번째의 테이블 값은 누적된 한자 수의 값을 갖게 되므로 한자표를 테이블을 두음법칙이 적용되는 한자에만 네 번째 테이블 값을 주어서 마지막의 테이블은 실행된 테이블의 값으로 정하여 한번 사용한 한자의 코드 값이면 체크를 다음에 한자에서 한글로 변환 할 때 전에 읽은 사실이 있으면 한글 코드의 값을 다시 읽어서 화면에 보내주면 된다. 단 본 논문에서는 어떠한 알고리즘을 제시하는 것이 아니고 사용하다 보니 문제점이 발생되어 이러한 방향을 제시해 본다. 결론에서 향후의 연구방향으로 제시해야 하나 보다

자세하게 발표하고자 이장에서 다루게 되었다.

4. 실험 및 평가

본 실험에서는 한자를 위치를 변환하는 검색 횟수와 한자를 등록하기 위해 실행되는 실행 수에 초점을 맞추어 실험하였다. 실험을 위해서 한글 워드 프로세서를 한자 변환 시스템을 이용하여 개인용 컴퓨터 팬티엄 166Hz, RAM 32MB, HDD 2GB에서 언어는 터보 C에서 시뮬레이션 하였으며 대상 단어는 한글과 한자의 문장으로 되어있는 전공 서적인 실용 소프트웨어 공학론을 이용하여 표준 한자 교본에 등록되어 있는 3,000 한자에 맞게 적용하였는데[2,3] 여기서 알 수 있는 것은 한번 사용한 한자가 다음에도 계속하여 나옴을 알 수 있었다, 현재의 기본체제로 되어 있는 한자 변환 시스템에서의 위치 변화할 때 키의 검색 횟수와 한자 변환 시스템에서 복합명사 처리 때의 실행 횟수와 본 논문에서 제안한 한자 변환 알고리즘과 복합 한자 변환 알고리즘을 적용한 키의 검색 횟수와 실행 횟수<그림 4.1>를 비교해 보았다



<그림 4.1 기존의 한자변환과 변경된 알고리즘의 비교>

본 그림에서 알 수 있듯이 한자 순위 변환에서의 5,000 단어로 시험 결과 현재의 한자 순위 변환은 14,584번의 검색 횟수를 알았고 개선된 순위 변환은 10,746번의 검색 횟수를 알 수 있어 개선된 순위 변환 알고리즘이 보다 더 효율적이고 동시에 동일한 한자 수가 많아질수록 개선된 순위 변환 알고리즘의 검색 횟수가 점점 감소한다는 것을 알 수 있다, 그래서 개선된 한자 변환 알고리즘이 효율적이다.

복합 한자는 2500 단어로 시험 결과 현재의 복합 한자는 시험 한자(2500)에서 재실행 한자수(548)를 빼고 그 값에 현재 복합한자에서 실행되는 횟수(6)를 곱하고 다시 재실행 한자수(548)를 더해서 나온 실행 횟수가 12,660번을 가졌고 개선된 복합 한자는 2500 단어로 시험 결과 복합 한자 알고리즘은 시험 한자(2500)에서 재실행 한자수(548)를 빼고 그 값에 현재 복합한자에서 실행되는 횟수(3)를 곱하고 다시 재실행 한자수(548)를 더해서 나온 실행 횟수가 6,404번을 실행하고 현재의 복합 한자와 개선된 복합 한자 알고리즘은 약 2배 이상의 실행 차가 나므로 개선된 복합 한자 알고리즘이 보다 효율적이다.

5. 결론

본 논문은 한자 변환에서 보다 더 실행 횟수를 줄여 검색 속도를 빠르게 하기 위해서 한자 변환 알고리즘과 복합 한자 변환 알고리즘을 구현하여 한자 변환을 효율적으로 활용한다. 한자 변환 알고리즘과 복합 한자 변환 알고리즘은 한글 워드프로세서에서 수동으로 해야 하는 점을 없애고 한 번만 사용을 하면 자동으로 변환되거나 자동으로 등록된다는 점에서 사용자에게 편리성을 제공하면서 다음과 같은 특징을 갖는다.

첫째, 한자 변환 알고리즘의 경우는 마우스의 이동이 필요없이 자동으로 변환하기 때문에 검색 시간을 줄일 수 있다.

둘째, 복합 한자 변환 알고리즘의 경우는 한자를 수동으로 등록해서 사용해야 하는데 복합 한자 변환 알고리즘은 이러한 등록이 필요 없이 자동으로 등록된다.

셋째, 한글·한자 변환 시스템을 구현하는데 한자 변환 알고리즘과 복합 한자 변환 알고리즘을 적용하여 효율적으로 이용한다.

앞으로의 향후 방향은 이중 배열을 이용한 이중 트라이로 한자 사전을 구축하여 본 논문의 알고리즘을 적용하여 검색 할 때 불필요한 요소를 제거하는데 있고, 또한 이 알고리즘은 기억장소의 낭비가 크기 때문에 기억장소를 줄이는데 있다.

참고문헌

[1] 호글96. 도서출판 호글과컴퓨터. 1997.

[2] 실용 소프트웨어 공학론. 도서출판 법영사. 1997.

[3] 3천자 표준한자 교본. 도서출판 이상사, 1995.

[4] 홍성혁, 김철수, 이용석 "한문 혼용 문장의 형태소 분석을 위한 사전 구성" 한국정보과학회 논문지 23권 2호, pp. 541-544, 1996.

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

[6] K. Morimoto, H. Iroguchi and J. I. Aoe, "A Retrieval algorithm of Dictionaries by Using Two Trie structures", 일본전자공학회논문집 D-II Vol. J76-D-11, No.11, pp. 2374-2383, 1994.