BTI 사전 색인을 이용한 유사단어 검색

정연수°*, 조영환*, 김길창*
한국과학기술원 전산학과*

A method for similar-word retrival based on BTI dictionary indexing mechanism

Yeon Su Jung°*, Young Hwan Cho*, Gil Chang Kim*
Computer Science Dept. of KAIST*

요약

유사단어의 추정에 있어서 사전 검색에 드는 비용, 즉 사전 탐색 횟수는 효율성의 문제와 직결된다. 본 논문에서는 BTI(Bidirectional TRIE Index)사전 색인을 이용하여 한 글자의 변형요소가 있는 유사단어들을 효율적으로 검색하는 방법을 제안한다. BTI방식은 정방향, 역방향 표제어를 모두 저장하는 방법이다.

BTI방식으로 사전 표제어를 색인하면 표제어에 대한 사전 탐색 도중에 사전에 존재하는 prefix와 postfix를 모두 검색할 수 있다. 이러한 정보를 이용하면 유사 단어에 대한 정확한 변형 위치를 결정할 수 있다. BTI사전 색인은 사전 표제어에 대한 정보없이 유사단어를 추정한 후에 사전 검색을 통하여 확인하는 방법보다 사전 검색에 드는 비용이 적다. 추가적으로 유사단어 후보들에 대한 우선순위를 정하기 위하여 corpus에서 추출한 각 표제어의 발견 빈도를 이용하였다.

1. 서론

본 논문에서는 철자 오류를 포함하고 있는 단어에 대하여 사전에 등록되어 있는 유사한 단어를 효율적으로 검색하는 방법에 관하여 설명한다. 유사단어의 추정에 있어서 사전 검색에 드는 비용, 즉 사전 탐색 횟수는 효율성의 문제와 직결된다.

사전 색인어를 Indexing하는 방법에는 보통 ISAM을 이용하는 방법[1]과 Hashing을 이용하는 방법[2], 그리고 TRIE구조를 이용하는 방법[2,3,4]이 있는데, 이중에 TRIE 구조는 동일한 prefix를 갖는 모든 사전 표제어를 한번의 사전 탐색으로 검색하는 것이 가능하여 보편적으로 사용되고 있다. 음성인식이나 문자인식 시스템의 후처리의 용도로 오인식 단어의 교정을 위한 여러가지 방법들이 제안되었다. 어절 단위의 철자 교정 방법에는 양방향 접근법을 통한 오류 위치의 판별[2]과 N-GRAM을 이용하여 hashing에 의하여 유사어를 검색하는 방법[5]과 말뭉치를 이용하는 방법[6]이 제안되었다. 그러나, 이러한 교정 방법들은 어절 단위의 한국어 처리를 위하여 형태소 분석 방법에 기인한 오류의 발견과 교정의 방법으로 제안되었고 유사 단어 혹은 형태소를 효율적으로 검색하는 방법에 대한 연구는 미약하였다.

단어수준의 유사단어의 검색을 위하여는 연상메모리를 이용하는 방법[7]이 제안되었다. 이 연구는 1000단어 수준의 사전상의 단어간 유사도 측정을 위하여 자판 배열 특성을 이용하였는데, 단어의 갯수만큼 신경망의 노드가 필요하게 되어 다량의 사전의 표제어에 대한 연상메모리의 구성에 있어서는 문제가 있다.

본 논문에서는 명사에 대하여 BTI(Bidirectional TRIE Index)사전 색인을 이용하여 한 글자의 변형 요소가 있는 유사단어들을 효율적으로 검색하는 방법을 제안한다. 유사단어를 추정한 후에 후보에 대한 사전 검색을 하는 일반적인 방식에서 탈피하여 사전 인덱스를 직접 탐색하면서 사전내에 존재하는 유사어를 검색하게 되므로 사전 검색에 드는 비용이 상대적으로 낮다. [8]에 의하면 철자의 오류에 있어서 1자소가 삭제되어 발생한 경우가 39.4%, 1자소가 추가된 경우가 29.1%, 1자소가 변경된 경우가 23.2%인 것으로 조사되었다. 따라서, 여기서는 한 글자 혹은 자소의 변형 요소가 있는 유사단어들을 효율적으로 검색하는 BTI 방법을 제안한다

BTI방식은 정방향, 역방향 표제어를 동일한 TRIE 색인으로 저장하는 방법이다. BTI방식으로 사전 표제어를 색인하면 표제어에 대한 사전 탐색 도중에 사전에 존재하는 prefix와 postfix를 모두 검색할 수 있다. 이러한 정보를 이용하면 유사 단어에 대한 정확한 변형 위치를 결정할

수 있다. BTI사전 색인은 사전 표제어에 대한 정보없이 유사단어를 추정한 후에 사전 검색을 통하여 확인하는 방법보다 사전 검색에 드는 비용이 적다.

본 논문에서는 우선 BTI사전 색인을 글자별로 구성하는 방법과 자소별로 구성하는 방법에 대하여 설명한다. 그리고 이들 각각을 이용하여 오류 요소가 있는 단어에 대한 유사단어를 검색하는 방법을 설명한다.

2. BTI방식

검색하고자 하는 Key의 길이가 일정하지 않을 때에 유용한 자료구조로 TRIE가 있다. TRIE는 한 레벨에서 다음 레벨로의 분기의 결정이 전체 Key에 의하지 않고, Key의 일부로 결정되는 tree이다. TRIE 구조에 의한 색인 방식의 장점은 동일한 prefix를 갖는 표제어를 모두 검색할 수 있다는 것인데 이러한 특징을 prefix closed 특성이라 한다. 이와 유사한 특성인 postfix closed 특성은 동일한 postfix를 갖는 표제어를 모두 검색할 수 있는 것을 의미한다. BTI 방식은 prefix closed 특성과 postfix closed 특성을 함께 갖는다고 할 수 있다. 왜냐하면 BTI는 한 표제어에 대하여 정방향 문자열과 역방향 문자열을 모두 같이 저장하고 있기 때문이다. 즉, 표제어 X에 대하여 X와 뒤집기(X)를 모두 인덱스로 저장하고 있다. 예를들면, 사전 표제어 '컴퓨터'에 대해서 '컴퓨터', '터퓨컴'을 모두 표제어로 하고 있다. TRIE 색인의 가장 큰 단점은 불필요한 공간의 사용이 과도하다는 것이다. 이것은 특히 말단 노드로 갈 수록 사용하지 않는 공간이 생기는 데, BTI 방법에서는 효율적인 공간의 활용을 위하여 표제어가 나타나는 공간을 linked list로 연결하는 방법을 사용하여 불필요한 공간 사용의 소지를 최대한 제거하였다. 따라서, BTI방법은 TRIE 구조를 글자 단위 혹은 자소 단위의 이진 tree형태로 변환하고 사전 색인어에 대하여 정방향 문자열과 역방향 문자열을 동일한 TRIE 구조에 저장한 변형된 TRIE 구조를 말한다.

2.1 글자별 BTI

글자별 BTI란 앞에서 설명한 BTI를 구성하되, TRIE의 저장단위가 글자로 이루어진 것을 말한다. 즉, '컴퓨터'라는 표제어에 대해서 Key는 '컴'+'퓨'+'터'로 이루어 진다. 동일한 level의 글자열에 대하여 linked list로 연결하여 불필요한 중간 노드를 유지하고 있는 것을 방지한다. 글자별 BTI의 추상적인 자료구조는 다음과 같다.

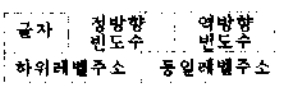

그림-1 BTI 노드의 구조

그림-1은 BTI 색인의 노드 구조를 나타낸다. 정방향 빈도와 역방향 빈도는 corpus상에서 발생한 단어의 빈도를 나타낸다. 역방향과 정방향 빈도의 값은 사전 표제어가 아닌 경우 0, 사전 표제어인 경우에는 1보다 큰 값으로 발견 빈도를 나타내며 이 값들은 이후에 유사단어들의 우선순위 결정에 사용된다. 어느 노드의 정방향(혹은 역방향)빈도가 1보다 크거나 같다면 현재까지의 path는 사전 표제어임을 의미하고, 만약 정방향(혹은 역방향)빈도가 0인 경우는 현재까지의 path는 사전 표제어가 아님을 의미한다.

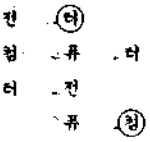

그림-2 글자별 BTI의 색인방법

그림-2는 글자별 BTI를 이용하여 글자를 단위로 TRIE 구조를 구성하는 방법을 나타낸다. 두개의 단어 '컴퓨터'와 '터전'에 대하여 9개의 노드를 통하여 순방향과 역방향 표제어, '전터'와 '퓨터컴'을 포함하는 BTI색인을 만들 수 있다.

다음은 서로 다른 양의 단어들을 사용하여 정방향의 TRIE와 정방향과 역방향 색인어를 모두 포함하는 BTI에 대하여 각각의 경우의 노드의 수와 그들의 비율을 조사한 표이다.

단어의수	10000	20000	30000	40000	50000	60000	70000
화일크기	67432	134184	201366	268352	335870	402776	469908
TRIE노드	18325	33468	46900	58828	69912	80747	91646
BTI노드	34278	62747	86290	111653	134058	155461	176539
BTI/TRIE	1.87	1.87	1.68	1.90	1.91	1.92	1.92
TRIE/크기	0.27	0.25	0.23	0.22	0.21	0.20	0.19
BTI/크기	0.51	0.47	0.44	0.42	0.40	0.39	0.37
평균길이	2.86	2.86	2.86	2.86	2.86	2.86	2.86

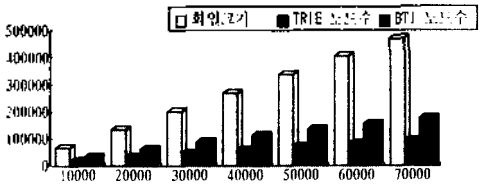

그림-3 단어크기의 증가에 따른 글자별 BTI의 규모 증가

글자별 BTI를 이용하여 사전 색인을 하게 되면 그림-3과 같은 규모의 노드를 갖게 된다. 그림-3에서 각 규모의 사전은 7만 단어의 명사 사전에서 임의로 선택한 1만단어에서 7만단어까지 일만단어를 기준으로 하여 만들어졌다. 위의 표에서 알수 있듯이 화일의 크기에 대한 TRIE의 크기의 비율은 0.19이므로 노드의 크기가 5바이트 이내라면 화일의 크기와 유사한 정도의 크기로 TRIE 색인을 구성할 수 있고, BTI의 노드수와의 비율은 7만 단어의 경우 0.37이므로 노드와 크기가 3바이트 정도라면 화일의 크기와 유사한 정도 크기로 TRIE 색인을 구성할 수 있다. TRIE에 비하여 BTI는 1.87배에서 1.92배의 크기를 갖는다. 그리고, 평균 단어의 길이는 사전의 크기에 상관없이 2.86 글자인 것으로 조사되었다.

2.2 자소별 BTI

자소별 BTI는 글자별 BTI와는 달리 TRIE의 저장과 검색이 글자단위가 아니라 자소단위로 이루어지는 것을 말한다. 다시말해서, '컴퓨터'라는 표제어에 대해서 정방향 표제어의 Key는 'ㅋ+ㅓ+ㅁ+ㅍ+ㅠ+ㅌ+ㅓ'이고 역방향 표제어의 Key는 'ㅓ+ㅌ+ㅠ+ㅍ+ㅁ+ㅓ+ㅋ'이다.

앞 절에서 설명했던 글자별 BTI에서 들었던 예로 자소별 BTI를 구성해보면 다음과 같다.

그림-4 자소별 BTI의 색인방법

'컴퓨터'와 '터전'에 대한 자소별 BTI의 자료구조는 그림-4와 같으며, 모두 22개의 노드로 색인이 구성된다.

단어의 수	10000	20000	30000	40000	50000	60000	70000
화일크기	113923	227629	341677	455441	570053	683800	797723
TRIE노드	59041	105721	146810	183166	216724	249721	282547
BTI노드	106563	191458	267738	336748	401683	464662	526384
BTI/TRIE	1.80	1.81	1.82	1.84	1.85	1.86	1.86
TRIE/크기	0.52	0.46	0.42	0.40	0.38	0.36	0.35
BTI/크기	0.93	0.84	0.78	0.74	0.70	0.67	0.65
평균길이	10.4	10.4	10.4	10.4	10.4	10.4	10.4

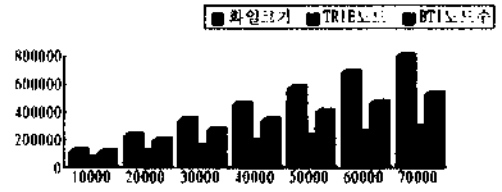

그림-5 단어크기의 증가에 따른 자소별 BTI의 규모 증가

자소별 BTI를 이용하여 사전 색인을 하게 되면 그림-5와 같은 규모의 노드를 갖게 된다. 화일의 크기에 대한 TRIE의 크기의 비율은 0.35이므로 노드의 크기가 3바이트 이내라면 화일의 크기와 유사한 정도 크기로 TRIE 색인을 구성할 수 있고, BTI의 노드수와의 비율은 7만 단어의 경우 0.65이므로 노드의 크기가 2바이트 정도라면 화일의 크기와 유사한 정도 크기로 TRIE 색인을 구성할 수 있다. TRIE에 비하여 자소별 BTI는 1.80배에서 1.86배의 크기를 갖는다. 그리고 평균 단어의 길이는 10.4 자소인 것으로 조사되었다.

3. BTI를 이용한 유사단어 검색

TRIE기법을 사용하지 않는 대부분의 사전 검색 시스템에서는 일정한 Key에 대해서 오직 Key와 완전히 일치하는 표제어가 있는가, 없는가에 대해서만 답할 수 있을 뿐이었다. 이에 반해 TRIE기법은 문자열에서 각 글자에 대한 인덱싱 방법이므로 Key인 X를 찾기 위해서는 X의 글자 하나하나를 순서대로 비교하며 탐색하여 X의 마지막 글자까지 검색한 후에야 비로소 X가 인덱스로 존재한다고 판단한다. 그러므로, X의 prefix인 표제어들도 모두 X의 탐색 도중에 자동적으로 탐색되기 때문에 한번의 X의 검색으로 prefix인 모든 표제어를 찾을 수 있다.

BTI에서는 하나의 Key 'X'에 대해 정방향색인과 역방향색인을 함께 저장하고 있으므로, 단 두번의 검색만으로 X의 prefix와 postfix를 모두 찾을 수 있다. 그러고 나서 이제 이 둘을 서로 비교분석하면 'X'의 정확한 오류의 위치를 추정할 수 있다.

만약 "abcdef"라는 단어를 잘못 입력하여 "abgdef"라는 단어로 입력했다고 하자.

BTI에서 "abgdef"라는 단어를 검색하여 prefix와 postfix를 구하면 다음과 같은 결과가 나올 수 있다.

prefix	postfix
a	f
ab	ef
abg	def
abgd	gdef
abgde	bgdef

그림-6 "abgdef"의 prefix와 postfix

그림-6에서 보면 "abgdef"에 대한 색인이 없으므로 이 단어는 오류를 포함하고 있음을 알수있다.

prefix와 postfix를 조합하여 유사단어를 추정해 보자. 일단 오류는 오직 한 부분에서만 일어난다고 가정하자.

그림-6에서 "abgdef"의 유사단어가 될수 있는 조합은 다음과 같다.

prefix	postfix	result
a	gdef	a?gdef
ab	def	ab?def
abg	df	abg?ef
abgd	f	abgd?f
abgde		abgde?
	bgdef	?bgdef

그림-7 "abgdef"의 유사단어가 될 수 있는 조합들

그림-7은 유사단어가 될수 있는 후보들이므로 이들을 사전에서 검색하면 된다. 이때 단어의 prefix나 postfix가 길어지면 길어질수록 같은 prefix나 postfix를 가진 단어의 갯수가 작아지므로 prefix-postfix 조합에서 둘중에 길이가 긴쪽을 기준으로 사전을 검색하는 것이 유리하다. 즉, "a"와 "gdef"의 조합에서는 역방향검색으로 "gdef"앞에 "b"를 제외한 임의의 글자가 오고 그 앞에 "a"가 오는 표제어가 있는 지를 검색하고, "abgd"와 "f"의 조합에서는 정방향검색으로 "abgd"다음에 "e"를 제외한 임의의 글자가 오고 그 다음에 "f"가 오는 표제어가 있는 지를 검색한 후에 그러한 표제어가 있으면 이를 유사단어로 추정한다.

3.1 글자별 BTI

글자별 BTI에서는 어느 표제어 X에 대하여 유사한 표제어 X'는 X와 한 글자만 다른 것이라고 가정한다.

글자별 BTI에서 유사단어 검색을 위한 알고리즘은 다음과 같다.

- 우선 단어 X(αxβ)에 대하여 모든 prefix와 postfix를 찾는다. 찾아진 prefix와 postfix에서 유사단어가 될 수 있는 모든 prefix-postfix 조합들을 구한다.
- 찾아진 prefix중에 "α"가 있고 postfix중에 "β"가 존재한다고 하자. 그러면, "αxβ"에서 "x"를 다른 글자인 "y"로 변형하면 "αyβ"는 "αxβ"의 유사단어가 될 수 있으므로 [α - β] 조합은 유사단어의 후보가 될 수 있다.
- 이렇게해서 구해진 prefix-postfix조합에서 길이가 긴 쪽에서부터 처음 한글자만 변형된 표제어가 사전에 있는지 검색하고 표제어가 있으면 이것을 유사단어로 추정한다.

예를 들어 "시뮬베이션"이라는 단어의 유사단어를 찾는 과정을 살펴보자.

prefix	postfix
시	션
시뮬	이션
	베이션

그림-8 "시뮬베이션"의 prefix와 postfix

글자별 BTI에서는 오직 한글자의 변형만 허용하므로 그림-8에서 유사단어가 될 수 있는 조합은 [시 - 베이션]과 [시뮬 - 이션]이다.

그러므로, "베이션"앞에 "뮬"외의 글자가 오고 그 앞에 "시"가 오는 표제어와 "시뮬"다음에 "베"외의 글자가 오고 그 다음에 "이션"이 오는 표제어가 발견되는 지를 검색한다.

그러면, 사전에 "시뮬레이션"이 표제어로 들어있으므로, "시뮬레이션"이 "시뮬베이션"의 유사단어가 된다.

3.2 자소별 BTI

한글에서는 한글자가 초성+중성 혹은 초성+중성+종성으로 이루어지므로 한 단어가 오직 한글자만 오류를 가지고 있다고 해도 이는 자소가 변형되었을 수도 있지만 이외에도 하나의 자소가 추가되거나 삭제되었을 가능성도 매우 크다. 따라서, 자소별 BTI를 이용하여 유사단어를 검색하고자 할 때는 자소의 변형외에도 한자소가 추가되었거나 삭제되었을 경우도 고려해 주어야만 한다.

예를 들면, "컴퓨터"라는 단어를 "커뮤터"라고 입력했을 경우 이 단어는 자소 "ㅍ"이 삭제되었다고 봐야 한다. 더우기 한 자소에 발생한 오류가 "커뮤터"에서처럼 두글자에 걸쳐 나타나는 경우가 많다.

서론에서 이미 밝힌 바와 같이 실제로 철자 오류에 있어서는 한자소가 변형된 경우보다 오히려 한자소가 추가되거나 삭제된 경우가 많은 것으로 나타났으며, 이 3가지의 경우만 고려하면 91.0%의 철자 오류를 교정할 수 있음을 알 수 있다.

자소별 BTI를 이용하여 유사단어를 검색하는 방법은 다음과 같다.

- 단어 X(αxβ)에 대하여 모든 prefix와 postfix를 찾는다. 찾아진 prefix와 postfix에서 유사단어가 될 수 있는 모든 prefix-postfix 조합들을 구한다.

1) 자소를 변형해야 하는 경우

- 찾아진 prefix중에 "α"가 있고 postfix중에 "β"가 존재한다고 하자. 그러면, "αxβ"에서 "x"를 다른 자소인 "y"로 변형하면 "αyβ"는

'αxβ'의 유사단어가 될 수 있으므로 [α - β] 조합은 유사단어의 후보가 될 수 있다.

- 이렇게해서 구해진 prefix-postfix조합에서 길이가 긴쪽에서부터 처음 한자소만 변형된 표제어가 사전에 있는지 검색하고 그런 표제어가 있으면 이것을 유사단어로 추정한다.

2) 자소를 삽입해야 하는 경우

- 찾아진 prefix중에 'αx'가 있고 postfix중에 'β'가 존재한다고 하자. 그러면, 'αxβ'에서 'αx'와 'β'사이에 임의의 자소 'y'를 삽입하면 'αxyβ'는 'αxβ'의 유사단어가 될 수 있으므로 [αx - β]조합은 유사단어의 후보가 될 수 있다.

- 이렇게해서 구해진 prefix-postfix조합에서 길이가 긴쪽에서부터 처음 한자소만 삽입된 표제어가 사전에 있는지 검색하고 그런 표제어가 있으면 이것을 유사단어로 추정한다.

3) 자소를 삭제해야 하는 경우

- 찾아진 prefix중에 'αx'가 있고 postfix중에 'xβ'가 존재한다고 하자. 그러면, 'αxβ'에서 자소 'x'를 삭제하면 'αβ'는 'αxβ'의 유사단어가 될 수 있으므로 [αx - xβ]조합은 유사단어의 후보가 될 수 있다.

- 이렇게해서 구해진 prefix-postfix조합에서 길이가 긴쪽에서부터 처음 한자소만 삭제된 표제어가 사전에 있는지 검색하고 그런 표제어가 있으면 이것을 유사단어로 추정한다.

철자 오류를 포함한 "커뮤터"라는 단어 경우에 대하여 유사단어를 검색하는 것은 다음과 같다.

prefix	postfix
ㅋ	ㅓ
ㅋㅓ	ㅌㅓ
ㅋㅓㅁ	ㅠㅌㅓ
ㅋㅓㅁㅠ	

그림-9 "커뮤터"의 prefix와 postfix

그림-9에서 유사단어가 될 수 있는 prefix-postfix조합은 다음과 같다.

1) 자소변형의 경우

prefix	postfix	result
ㅋㅓ	ㅠㅌㅓ	ㅋㅓ?ㅠㅌㅓ
ㅋㅓㅁ	ㅌㅓ	ㅋㅓㅇ?ㅌㅓ
ㅋㅓㅁㅠ	ㅓ	ㅋㅓㅁㅠ?ㅓ

2) 자소삽입의 경우

prefix	postfix	result
ㅋㅓㅁ	ㅠㅌㅓ	ㅋㅓㅁ?ㅠㅌㅓ
ㅋㅓㅁㅠ	ㅌㅓ	ㅋㅓㅁㅠ?ㅌㅓ

3) 자소삭제의 경우

prefix	postfix	result
ㅋㅓㅁㅠ	ㅠㅌㅓ	ㅋㅓㅁㅠㅌㅓ

그림-10 "커뮤터"의 유사단어가 될 수 있는 조합

그림-10의 조합들로부터 자소변형의 경우에는 prefix와 postfix사이의 자소를 변형한 단어를, 자소삽입의 경우에는 한자소를 삽입한 단어, 자소삭제의 경우에는 한자소를 삭제한 단어를 사전에서 검색한다.

여기서는 자소 삽입의 경우에 자소 'ㅛ'를 삽입한 단어인 '컴퓨터'가 사전에 있으므로 이것이 유사단어가 된다.

4. 결론

본 논문에서 우리는 BTI를 이용하여 효율적인 사전 검색으로 오류를 포함한 단어에 대한 유사단어 검색 방법을 제안하였다. 특히 글자별 BTI와 자소별 BTI에 대하여 각각의 구성 방법과 유사단어 검색을 위한 알고리즘을 설명하였다.

BTI 사전색인을 이용한 유사단어 검색 방법은 한 단어 내에 한 글자나, 한 자소, 혹은 한 뭉치의 자소열에 오류가 있는 경우 효율적인 유사단어 추정 방법을 가지지만, 여러자소에 걸쳐 오류가 흩어져 있는 경우에는 발생한 오류 단어의 prefix와 postfix 정보 만으로는 오류의 위치를 추정할 수 없으므로 부리가 따른다.

참고문헌

[1] 박혁로 외, "한글 문서를 위한 자동 색인 시스템 개발", 우리말 정보화 간지 '91 논문집, 1991

[2] 한국과학기술원, 한국어 철자 및 띄어쓰기 교정 시스템에 관한 연구 II, 과학기술처, 1992

[3] 강재우, 접속정보를 이용한 한글 철자 및 띄어쓰기 검사기의 설계 및 구현, 한국과학기술원 석사학위논문, 1990

[4] 조영환 외, "확장사전 환경에서의 한국어 형태소 해석과 생성", 제 5회 한글 및 한국어정보처리 학술대회, 1993

[5] 이종연 외, "N-GRAM 한글 사전을 이용한 오인식 단어의 교정 알고리즘", 제 5회 한글 및 한국어정보처리 학술대회, 1993

[6] 이병훈 외, "말뭉치를 기반으로 한 한국어 철자 교정기의 구현", 제 5회 한글 및 한국어정보처리 학술대회, 1993

[7] 정한민 외, "자판 배열 특성을 이용한 Neuro-Fuzzy 한국어 철자 교정기의 구현", 제 5회 한글 및 한국어 정보처리 학술대회, 1993

[8] 조영환, 한글 맞춤법 교정기의 설계 및 구현, 한국과학기술원 석사학위논문, 1991