자소 및 음소 정보를 이용한 영어-한국어 음차표기 모델

(An English-to-Korean Transliteration Model based on Grapheme and Phoneme)

오 종 훈 † 최 기 선 ††
(Jong-Hoon Oh) (Key-Sun Choi)

요 약 최근 정보검색과 기계번역과 같은 자연언어응용에서 영-한 자동 음차표기에 대한 관심이 높아지고 있다. 지금까지의 영-한 자동 음차표기 연구에는 <영어자소→한글자소>의 직접방식, <영어자소→음소→한글자소>의 피봇방식이 있다. 기존의 영-한 음차표기 연구들은 주로 직접방식에 대한 연구가 활발히 진행되어 왔다. 하지만, 음차표기는 직접방식에서 사용하는 단순한 자소 대 자소변환 작업이라기보다는 자소의 음성적 변환 작업이라고 할 수 있다. 따라서 자소 뿐만 아니라 음소 등 음성적 정보가 매우 중요하다. 본 논문에서는 이러한 특성을 이용하여 자소 정보뿐만 아니라 음소 정보를 이용한 음차표기 기법을 제안한다. 주어진 자소와 음소 및 자소와 음소의 문맥정보를 이용하여 한국어 음차표기를 생성하는 본 논문의 기법은 약 60%의 단어 정확도를 나타내었다.

키워드 : 음차표기, 외래어, 음소, 자소, 정보검색

Abstract There has been increasing interest in English-to-Korean transliteration recently. Previous works are related to a direct method like <English graphemes → Korean graphemes> and a pivot method like <English graphemes → English phoneme → Korean graphemes>. Though most of the previous works focus on the direct method, transliteration, however, is a phonetic process rather than an orthographic one. In this point of view, we present an English-Korean transliteration model using grapheme and phoneme information. Unlike the previous works, our method uses phonetic information such as phonemes and their context. Moreover, we also use graphemes corresponding to phonemes Our method shows about 60% word accuracy.

Key words : transliteration, foreign word, phoneme, grapheme, information retrieval

1. 서 론

음차표기란 외국어의 단어를 글자나 발음을 이용하여 자국어로 표기하는 것으로 정의된다[1,2]. 한국어에서 음차표기된 단어들은 주로 영어에 기원을 두고 있기 때문에, 한국어에서 음차표기는 주로 영-한 음차표기를 지칭한다. 전문분야문서와 같이 외국어원의 용어가 많이 포함된 문서처리에서 음차표기를 올바르게 파악하고 이를 효과적으로 처리하는 것은 매우 중요하다. 이는 전문용어의 많은 부분이 외국어에 기원을 두고 있고, 한국어 문서에 나타날 때 음차표기되거나 원어 그대로 사용되는 경우가 많기 때문이다. 전문분야 문서에서 음차표기와 원어의 혼재는 정보검색과 같은 자연언어응용에서 단어불일치 문제[1]를 야기한다. 음차표기로 인한 단어 불일치 문제를 해결하기 위한 방법으로 영-한 음차표기 대역쌍을 포함하는 사전을 이용하는 방법이 있다. 하지만 음차표기되는 용어들이 대부분 사전에 등재되지 않는 경우가 많기 때문에 이를 처리하기 위한 방법으로 자동음차표기에 대한 연구가 활발히 진행되어 왔다[1, 3-9].

영-한 음차표기에서 영어단어는 크게 입말 표기(spoken word transliteration)와 글말표기(written word transliteration)의 두 가지 방식으로 음차표기된다. 입말표기

· 본 연구는 과학기술부와 한국과학재단 특성장려연구사업(R21-2003-000-10042-0)의 지원으로 수행되었습니다.
† 학생회원 : 한국과학기술원 전산학과 박사과정
rovellia@world.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 정교수
kschoi@world.kaist.ac.kr
논문접수 : 2004년 10월 6일
심사완료 : 2005년 2월 11일

1) 단어 불일치 문제라 같은 의미의 단어가 다른 형태로 나타날 경우 다른 용어로 취급하는 문제를 지칭한다.

는 영어단어의 발음을 이용하여 음차표기하는 방법으로 '표준 외래어 표기법'[10]과 같이 음소에 대응되는 한글 자소를 생성하는 음차표기 방식이다. 따라서 입말표기는 음소기반 음차표기 방식이다. 예를 들어, 영어 단어 *impactite*는 입말표기에 의해서 올바른 음차표기인 '임팩타이트'로 음차표기된다(글말표기에 의해서는 '임팍티테'). 즉, *impactite*의 음소 /IH/, /M/, /P/, /AE/, /K/, /T/, /AY/, /T/[2]는 각각 한글자소 '이', 'ㅁ', 'ㅍ', '애', 'ㄱ', 'ㅌ', '아이', '트'로 음차표기된다. 반대로 글말표기는 자소기반 음차표기 방식으로 영어 자소에 대응되는 한글 자소를 생성하는 방법이다. 예를 들어, 영어 단어 *adenoid*는 글말표기에 의하여 올바른 음차표기인 '아데노이드'라고 음차표기되는데, *adenoid*의 자소 *a, d, e, n, o, i, d*에 의해 각각 '아', 'ㄷ', '에', 'ㄴ', '오', '이', 'ㄷ' 라고 음차표기된다(입말표기는 '애더노이드').

이러한 글말표기와 입말표기의 상이성으로 인하여, 영-한 음차표기에서 올바른 음차표기를 생성하기 위해서는 **"글말 및 입말표기방식에 따른 음차표기의 생성"**, **"글말 및 입말표기 방식의 선택"**이라는 두 가지 문제를 효과적으로 처리해야 한다.

"글말 및 입말 표기 방식에 따른 음차표기의 생성" 문제는 같은 영어 단어라도 음차표기 방식에 따라 다른 음차표기로 생성되기 때문에 나타나는 문제이다. **음소기반의 입말표기**와 **자소기반의 글말표기** 방식은 음소와 자소라는 서로 다른 정보를 이용하여 음차표기를 생성하는 기법이다. 따라서 기계적으로 각 방식의 음차표기를 생성하기 위해서는 서로 다른 메커니즘을 이용하여야 한다. 이재성[1]은 이러한 입말표기와 글말표기를 처리하기 위한 모델로서 피봇방식과 직접방식의 음차표기 모델을 제안하였다. 직접방식은 글말표기를 효과적으로 처리하는 모델로서 영어단어를 한국어 음차표기로 직접 변환하는 모델이다. 직접방식은 영어단어만으로 음차표기를 생성할 수 있기 때문에 발음지식을 필요로 하는 피봇방식에 비해 비교적 간단하게 음차표기를 생성할 수 있다는 장점이 있다. 즉, 주어진 영어와 영어에 대응되는 한국어 음차표기 데이타에서 나타나는 영-한 자소간 변환 규칙을 파악하고 이를 이용하여 음차표기를 수행하는 모델이다. 입말표기를 효과적으로 처리하는 음차표기 방식인 피봇방식에 의한 음차표기는 주어진 영어 단어에 대한 **'발음생성 단계'**와 생성된 발음을 이용하여 **'한국어 음차표기를 생성하는 단계'**로 이루어진다. 피봇방식에 의해 올바른 음차표기를 생성하기 위해서는 다음과 같은 두 가지 사항을 효과적으로 처리해야 한다. 첫째, **'발음 추정 문제'**이다. 영어단어에 대한 발음은 주로 영어-발음 사전을 이용하여 생성될 수 있지만, 발음사전에 등록되지 않은 영어단어에 대해서는 발음을 자동으로 생성하기 위한 규칙이 필요하다[3]. 둘째, **'순차처리에 의한 오류 파급 문제'**이다. 피봇 방식이 발음생성과 음차표기 생성의 두 단계 과정을 거치기 때문에 발음생성의 오류가 음차표기 생성의 오류로 파급되는 문제점이 있다. 이러한 오류의 파급을 최소화하기 위해서는 발음생성의 오류를 최소화하거나 발음생성의 오류에 견고한 음차표기 생성기법이 필요하다. 피봇방식의 두 가지 문제점으로 인하여 기존의 영-한 음차표기 연구[1][5-9]에서는 피봇방식보다는 비교적 간단한 직접방식에 기반한 연구들이 주로 이루어져 왔다. 하지만 올바른 음차표기를 생성하기 위해서는 글말 뿐만 아니라 입말표기도 효과적으로 처리해야 한다.

"입말 및 글말표기 방식의 선택" 문제는 올바른 음차표기를 생성하기 위해 음차표기 방식을 결정하는 문제이다. 상기에 기술한 바와 같이 입말표기와 글말표기는 서로 다른 정보에 기반하여 음차표기를 생성한다. 따라서 글말표기와 입말표기로 생성된 음차표기는 ***'bold* /B OW L D/'**와 같이 영어단어의 음소가 자소와 유사한 형태로 나타나는 경우를 제외하고는 일반적으로 서로 다른 형태를 나타낸다. 따라서 주어진 영어단어에 대한 음차표기 방식을 선택이 올바른 음차표기를 생성하기 위해서는 매우 중요하다. 실제 본 논문에서 사용한 실험집합[4]을 분석한 결과 글말표기와 입말표기로 생성할 수 있는 음차표기의 적용률은 표 1과 같이 나타났다. 표 1에서 글말표기의 적용률은 약 30%를 나타내며, 입말표기의 적용율은 60%를 나타낸다. 또한 글말표기와 입말표기를 모두 고려했을 경우의 적용율은 약 81%~84%로 두 가지 음차표기 방식을 올바르게 선택한다면 보다 효과적으로 음차표기를 생성함을 알 수 있다.

본 논문에서는 상기에 기술한 두 가지 문제점을 **자소 및 음소 기반 음차표기 모델**을 이용하여 해결하고자 한다. 자소기반의 직접방식과 음소기반의 피봇방식과 달리 본 논문의 기법은 자소와 음소를 모두 이용하여 음차표기를 수행한다.

첫째, **"글말 및 입말표기방식에 따른 음차표기의 생**

2) 본 논문에서 발음 및 음소를 표현하기 위하여 ARPAbet 기호를 사용한다. ARPAbet 기호는 음소를 ASCII 기호로 표현하기 위한 방법이다. ARPAbe의 체계는 www.cs.cmu.edu/~laura/pages/arpabet.ps에 자세히 나타나 있다.

3) 본 논문에서 사용한 실험집합에 대하여 발음사전의 커버율은 67%였다.

4) 본 논문에서 데이타 분석 및 실험을 위하여 두 가지 실험집합을 사용하였다. 실험집합I은 [1]에서 사용한 1,650개 영-한 음차표기 쌍으로 구성된 실험집합으로 [1][5-9]의 영-한 음차표기 연구의 성능평가에 사용된 실험집합이다. 실험집합II[31]는 7,185개 영-한 음차표기 쌍으로 구성되어 있으며 [5][6][9]의 영한 음차표기 연구의 성능평가에 사용된 실험집합이다. 본 논문에서는 실험집합 I과 실험집합 II의 음차표기를 올바른 음차표기로 기술한다

표 1 실험집합에 대한 음차표기방식의 적용률

음차표기 방식	실험집합 I	실험집합 II
글말	28.67%	33.1%
입말	60.00%	60.00%
글말 ∩ 입말	7.33%	8.9%
글말 ∪ 입말	81.33%	84.2%
글말 및 입말의 혼개	18.67%	15.8%

성"문제는 자소 및 음소를 모두 사용하기 때문에 글말 및 입말표기 방식의 음차표기를 모두 처리할 수 있다. 또한 피봇방식에서의 **"발음생성 문제"**를 발음 사전과 발음생성 규칙을 이용하여 해결하였으며, **"순차적 오류 문제"**는 자소와 음소를 모두 사용하여 음소의 오류에 비교적 견고한 음차표기 생성을 수행한다. 즉 음소의 오류를 대응되는 자소가 보완하여 줌으로써 잘못된 음차표기를 생성할 가능성을 낮추어 줄 수 있다.

둘째, **"입말 및 글말표기 방식의 선택"**문제는 자소와 음소에 기반한 자소 단위 음차표기 변환규칙을 이용함으로써 해결한다. 즉, 영어의 자소와 대응되는 음소에 대하여 입말표기방식에 의해 한글 자소가 생성되는지 혹은 글말표기방식에 의해 한글 자소가 생성되는지를 결정하는 문제로 생각할 수 있다. 본 논문에서는 각 자소에 대한 자소기반 음차표기를 수행하는지 혹은 음소기반 음차표기를 수행하는지에 대한 규칙을 이용하여 각 자소의 음차표기를 생성한다.

따라서 본 논문의 자소 및 음소기반 음차표기 모델은 기존의 자소기반 방식 또는 음소기반 방식들의 한계점을 보완하고 해결함으로써, 효과적으로 음차표기를 수행하는 모델이라 할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대하여 기술하고 3장에서는 자소 및 음소정보를 이용한 음차표기 방법에 대하여 기술한다. 4장에서는 실험 및 결과를 기술하고 5장에서는 결론을 맺는다.

2. 관련연구

기존의 영-한 음차표기 연구들은 대부분 직접방식에 기반한 연구들로 크게 **확률기반 모델, 결정트리기반 모델, 음차표기 네트워크 기반 모델**로 분류될 수 있다. 본 장에서는 각 모델에 대하여 기술한다.

2.1 확률기반 모델

[1,8]에서는 직접방식의 확률기반 음차표기 모델을 제안하였으며, 식 (1)을 이용하여 주어진 영어 단어 E에 대한 한국어 음차표기 K를 생성하였다.

$$K^* = \arg\max_K p(K \mid E) = \arg\max_K p(K)p(E \mid K)$$

$$P(K) \cong p(kpu_1)\prod_{i=2}^{n} p(kpu_i \mid kpu_{i-1})$$

$$P(E \mid K) \cong \prod_{i=1}^{n} p(epu_i \mid kpu_i) \quad (1)$$

여기에서, E는 영어단어, K는 한국어 단어를 나타내며, epu_i는 i번째 영어발음단위를 kpu_j는 j번째 한국어 발음단위를 나타낸다.

발음단위(Pronunciation Unit: PU)를 음소(phoneme)에 매핑될 수 있는 알파벳의 덩어리로 정의하고, 발음단위에 기반한 음차표기를 수행하였다. 영어 단어 E로부터 영어 발음 단위의 열 $E_1,E_2,\cdots,E_n$을 생성할 수 있고 Ei에 대응하는 모든 가능한 한국어 발음단위들의 조합으로 $K^i_1,K^i_2,\cdots,K^i_m$를 생성할 수 있다. 여기에서 $E_i=epu_{i1}, epu_{i2}, \cdots, epu_{in}$는 $K^i_j=kpu^i_{j1}, kpu^i_{j2}, \ldots, kpu^i_{jn}$이다. 예를 들어, 영어 단어 *board*로 생성할 수 있는 i번째 발음단위의 열 E_i ={(epu_{i1}=*b*), (epu_{i2}=*oa*), (epu_{i3}=*r*), (epu_{i4}=*d*)}라 가정하자. 그러면 E_i에 대응되는 한국어 발음단위의 조합은 K^i_1={<kpu^i_{11}=ㅂ>, <kpu^i_{12}=오>, <kpu^i_{13}=르>, <kpu^i_{14}=드>}, K^i_2={<kpu^i_{21}=ㅂ>, <kpu^i_{22}=오오>, <kpu^i_{23}=르>, <kpu^i_{24}=드>} 등으로 표현할 수 있다.

이재성은 모든 가능한 영어의 발음단위의 열과 이에 대응하는 한국어 발음단위의 열을 고려하고, 이 중에서 식 (1)을 최대화하는 한국어발음단위의 열을 선택하였다. 예를 들어 영어단어 *data*로부터 *d:at:a*, *da:ta*, *d:a:t:a* 등의 영어 발음단위의 열을 생성할 수 있다. 이재성의 음차표기 기법은 이들 조합 중 가장 적합한 영어의 발음단위열과 대응되는 한국어의 발음단위열을 파악하는 작업으로 정의된다.

김정재[7]는 이재성[1,8]의 방법을 확장한 확률기반 음차표기 모델을 제안하였다. 이재성과 마찬가지로 식 (1)을 사용하여 음차표기를 수행하였지만, $P(E|K)$의 추정을 위하여 식 (2)와 같이 kpu_{i-1}와 kpu_{i+1}의 추가적인 정보를 이용하였다. 또한 식 (2)는 신경망을 이용하여 추정되었다.

$$P(E \mid K) \cong \prod_{i=1}^{n} p(epu_i \mid kpu_{i-1}, kpu_i, kpu_{i+1}) \quad (2)$$

확률기반 음차표기 모델은 '발음단위'라는 자소수준의 음성적 정보를 사용하였다는 장점을 가진다. 하지만 영어 및 한국어에 대한 가능한 '발음단위'를 모두 고려하기 때문에 과도한 '발음단위'의 생성이 음차표기의 성능을 저하시킬 뿐만 아니라 시간복잡도를 높이는 결과를 초래한다. 만약, 영어단어로부터 생성할 수 있는 영어 발음단위열의 평균개수가 N개이고, 영어발음단위에 대응되는 한국어 발음단위조합의 평균 개수가 M개라고 하면, $N\times M$개의 가능한 조합이 생성될 수 있다.

2.2 결정트리 기반 모델

[5,6]에서는 결정트리에 기반한 영-한 음차표기 기법을 제안하였다. 결정트리 기반 모델도 영어 자소를 한글

자소로 변환(영-한 자소 변환)하는 직접방식에 기반한 방법이다. 결정트리기반 모델에서는 영-한 자소변환을 위하여 각 영어 자소에 대한 결정트리를 구성하고 이를 이용하여 각 자소에 대한 음차표기를 수행한다. 결정트리 구성을 위하여 사용되는 특성자질은 좌우 세 개의 영어자소 정보를 사용하였다.

표 2 data에 대한 결정트리 기반 음차표기

L3	L2	L1	C0	R1	R2	R3	K
$	$	$	d	a	t	a	'ㄷ'
$	$	d	a	t	a	$	'네이'
$	d	a	t	a	$	$	'ㅌ'
d	a	t	a	$	$	$	'ㅏ'

표 2는 영어 단어 *data*에 대한 결정트리 기반 음차표기 과정을 나타낸다. 표 2에서 *C0*는 한국어 자소로 변환할 영어 자소를 나타내고 *L1, L2, L3*는 왼쪽 문맥을 나타내는 세 개의 영어자소를, *R1, R2, R3*는 오른쪽 문맥을 나타내는 세 개의 영어자소를 각각 나타낸다. 그리고 *K*는 결정트리에 의해 변환된 한국어 자소를 나타낸다. 표 2에서 *$*는 단어의 처음과 끝을 나타내는 기호이다.

결정트리 기반 방법은 다른 연구에 비해 넓은 범위의 문맥을 고려했다는 장점이 있다. 하지만 음차표기에 중요한 음성적 정보를 전혀 사용하지 않아 입말표기 음차표기를 효과적으로 처리하지 못하는 단점이 있다.

2.3 음차표기 네트워크 기반 모델

강인호[9]는 음차표기 네트워크를 이용한 영-한 음차표기 모델을 제안하였다. 음차표기 네트워크 기반 모델도 기존의 다른 방법과 마찬가지로 직접방식에 의한 음차표기 기법이다. 음차표기 네트워크는 그림 1과 같이 구성된다. 우선 주어진 영어단어에 대하여 모든 가능한 영어자소열을 생성한다. 그리고 이를 이용하여 음차표기 네트워크를 생성한다. 음차표기 네트워크에서 각 노드는 하나 이상의 영어자소와 해당 자소들에 대응되는 한글 자소로 구성된다. 또한 각 아크는 식 (3)에 의해 가중치가 부여된다.

식 (3)의 가중치 기법은 문맥(context)과 출력 (output)으로 구성되어 있다. 문맥은 특정 출력을 나타내는 히스토리 정보(historical information)를 나타내며, 출력은 문맥에 의해 생성되는 한국어자소를 나타낸다.

$$weight\ (context\ ,output\ ) = \frac{C(output\ )}{C(context\ )} \quad (3)$$

그림 1과 같이 음차표기 네트워크가 구성되면, 가중치의 합이 가장 높은 경로를 찾아 음차표기를 생성한다. 이 때, 경로를 파악하기 위하여 비터비(Viterbi) 알고리즘[11]과 트리-트렐리(Tree-Trellis) 알고리즘[12]을 사용하였다.

음차표기 네트워크 기반 모델은 이재성의 연구에서의 '발음단위'와 유사하게 하나 이상의 영어자소를 고려하여 자소수준에서의 음성적 특성을 반영하였다. 또한 이재성의 연구와 달리 '발음단위'의 생성과 음차표기 파악의 과정을 음차표기 네트워크라는 모델을 이용하여 하나의 단계로서 처리하였다. 하지만 음차표기 네트워크 기반 모델도 기존의 다른 연구와 마찬가지로 음소정보를 사용하지 않았기 때문에 입말표기를 효과적으로 처리하는데는 한계가 있다.

2.4 기존연구의 문제점

기존의 연구들은 자소기반의 직접방식 음차표기 모델을 사용하였다. 따라서 영-한 음차표기에서 나타나는 두 가지 문제에 대한 효과적인 처리를 하지 못한 한계점이 있다. 즉, **"음소기반 입말표기에 대한 처리"**와 **"글말 및 입말표기의 선택 문제"**에 대한 고려 없이 영어자소 대 한국어 자소의 변환규칙만으로 음차표기를 수행하였다. 본 논문에서는 이러한 한계점을 극복하고 영-한 음차표기에서 나타나는 현상들을 고려한 자소 및 음소기반 영-한 음차표기 모델을 제안한다.

3. 자소 및 음소 기반 영-한 음차표기 모델

3.1 시스템 구조도

그림 2는 본 논문에서 제안하는 자소 및 음소 기반 영-한 음차표기 모델의 전체시스템 구조도와 영어단어 *board*에 대한 음차표기 생성과정을 나타낸다. 자소 및 음소 기반 영-한 음차표기 모델은 **"전처리 단계"**와 **"영-한 음차표기 단계"**로 구성된다.

"전처리 단계"는 **"영어-발음-한글 정렬"**로 구성된다. 영어-발음-한글 정렬은 영어-발음-한글음차표기가 대응된 학습데이타에서 영어자소, 음소, 한글자소 간의

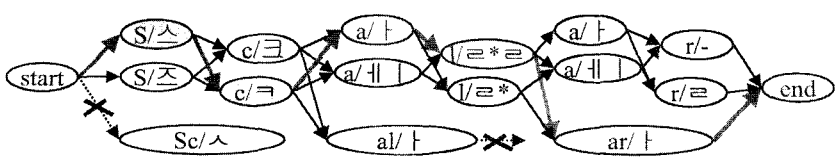


그림 1 영어단어 scalar에 대한 음차표기 네트워크의 구성 예

대응관계를 파악하는 과정이다. 정렬결과는 발음생성단계의 발음추정과 음차표기생성을 위한 학습데이타로 사용된다.

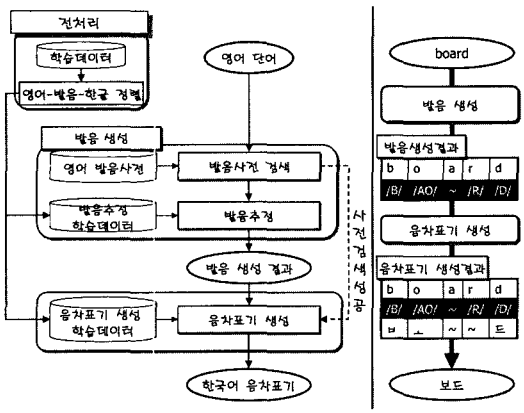


그림 2 영-한 음차표기 시스템 구조도

"영-한 음차표기 단계"는 크게 **"발음생성"**, **"음차표기 생성"**단계로 구성된다. 주어진 영어단어는 **"발음생성"**단계를 통하여 해당 영어의 발음을 생성한다. **"발음생성"**단계에서 생성된 발음은 주어진 영어단어와 함께 다음 단계인 **"음차표기생성"**단계에서 한국어 음차표기를 생성하는데 사용된다.

첫 번째 단계인 **"발음생성"**단계 에서는 주어진 영어단어의 각 자소에 대응하는 음소를 파악한다. 예를 들어, 영어 단어 *board*의 각 자소 *b, o, a, r, d*에 대응되는 음소 *b-/B/, o-/AO/, a-/~/*[5], *r-/R/, d-/D/*를 생성한다(그림 2의 우측 발음생성결과). **"발음생성"** 단계에서는 **"발음사전검색"**과 발음사전에 미등록된 단어의 **"발음추정"**을 통하여 영어단어의 발음을 생성한다. 발음생성 단계에서는 우선 정확한 발음을 포함하는 영어 발음사전을 검색한 후 영어 발음사전에 존재할 경우 해당 발음을 영어 단어의 발음으로 생성한다. 만약 해당 영어단어가 영어 발음사전에 미등록된 단어인 경우에는 **"발음추정"** 단계에서 발음추정을 통하여 발음을 생성한다. 영어 발음사전 검색을 위하여 약 120,000 영어-발음쌍을 포함하는 CMU 발음사전을 사용한다. 발음 추정은 영어-발음이 정렬된 학습데이타를 이용하여 기계학습방법으로 발음을 생성한다. 사용한 기계학습 방법은 결정트리 기법과 메모리 기반 학습 방법이다.

두 번째 단계는 영어와 발음에 대응되는 한국어 음차표기를 생성하는 단계이다. **"음차표기생성 단계"**에서는 발음생성단계의 결과를 입력으로 하여 음차표기를 생성한다. 이를 위하여 메모리기반학습 방법과 결정트리를 이용한다. 예를 들어 발음생성단계의 결과인 *b-/B/, o-/AO/, a-/~/, r-/R/, d-/D/*에 대하여 음차표기생성 단계에서는 각 음소와 자소에 대응하는 한글 자소를 *'b-/B/-ㅂ', 'o-/AO/-오', 'a-/~/-~', 'r-/R/-~', 'd-/D/-드'*와 같이 생성하고 (그림 2의 우측 음차표기 생성결과), 이를 통하여 한국어 음차표기 '보드'를 생성한다.

3.2 영어-발음-한글 정렬: 발음추정 및 음차표기 생성을 위한 학습데이타 구축

발음 추정과 한국어 음차표기 생성을 위한 규칙을 작성하기 위해서는 학습데이타가 필요하다. 발음 추정(δ_p)은 각 자소 $e \in E$가 생성하는 음소들 $p=\{p_1,\cdots,p_n\} \in 2^P$를 파악하는 작업 δ_p: $E \rightarrow 2^P$이며, 한국어 음차표기 생성(δ_k)은 각 자소 $e \in E$와 자소에 대응되는 음소 $p=\delta_p(e)$를 이용하여 한국어 자소 $k=\{k_1,\cdots,k_m\} \in 2^K$를 파악하는 작업이며 δ_p: $E \times 2^P \rightarrow 2^K$이다. 여기에서, E는 영어자소의 집합, P는 음소의 집합, K는 한글자소의 집합을 나타낸다. 따라서 발음생성 및 음차표기생성을 위해서는 *'b-/B/-ㅂ', 'o-/AO/-오', 'a-/~/-~', 'r-/R/-~', 'd-/D/-드'*와 같이 각 영어 자소에 대응하는 음소 및 한국어 자소가 대응된 데이타가 필요하다. 하지만 이를 수작업으로 구성하는 것은 많은 시간이 소요될 뿐만 아니라 학습데이타의 일관성 유지가 힘든 문제점이 있다.

본 논문에서는 이러한 학습데이타를 자동으로 구축하기 위한 알고리즘을 제안하고 이를 이용하여 학습데이타를 구축한다. 필요한 학습데이타는 발음추정을 위한 영어자소와 음소가 정렬된 데이타와 음차표기 생성을 위한 영어자소와 이에 대응되는 음소 및 한글자소가 정렬된 데이타이다. 이를 위하여 영어자소-음소-한글자소 간 정렬 알고리즘을 구축하였다. 정렬알고리즘은 영어자소-음소를 정렬하는 알고리즘(EP 정렬 알고리즘)과 음소-한글자소를 정렬하는 알고리즘(PK 정렬 알고리즘) 그리고 영어자소-음소-한글자소를 정렬하는 알고리즘(EPK 정렬 알고리즘)으로 구성된다.

3.2.1 EPK 정렬 알고리즘

EPK 정렬 알고리즘은 *EP* 정렬 알고리즘과 *PK* 정렬 알고리즘에 의해 생성된 정렬결과를 이용한다. 본 논문에서 사용한 *EPK* 정렬 알고리즘은 음소를 피봇으로 사용하여 영어자소-음소-한글자소간의 음운적 대응관계를 파악한다. 즉, 영어자소에 대응하는 음소와 해당 음소에 대응되는 한글자소 간의 대응관계를 이용하여 영어자소-음소-한글자소간의 대응관계를 파악한다.

예를 들어, 표 3과 4에서 영어자소 b는 음소 /B/와 대응되고 음소 /B/는 한글자소 'ㅂ'과 대응된다. 따라서

5) 본 논문에서 '/~/'를 묵음(silence)으로 정의하고, 하나의 음소로 취급하며, '~'는 음차표기가 되지 않음을 나타내는 기호로 사용한다

표 3 EP 정렬 알고리즘에 의해 생성된 *board*의 영어자소-음소 간 대응 관계

영어자소	b	o	a	r	d
음소	/B/	/AO/	/ㄱ/	/R/	/D/

표 4 PK 정렬 알고리즘에 의해 생성된 *board*의 음소-한글자소 간 대응 관계

음소	/B/	/AO/	/R/	/D/
한글자소	ㅂ	ㅗ	~	ㄷ

표 5 EPK 정렬 알고리즘에 의해 생성된 board의 영어자소-음소-한글자소 간 대응 관계

영어자소	b	o	a	r	D
음소	/B/	/AO/	/~/	/R/	/D/
한글자소	ㅂ	ㅗ	~	~	ㄷ

영어자소와 음소간 대응관계(*b*→/B/)와 음소와 한글자소 간 대응관계(/B/→ 'ㅂ')에 의해 표 5와 같이 *b*, /B/. 'ㅂ' 간의 대응관계를 파악할 수 있다(if '*b*'→'/B/' and '/B/' →'ㅂ' *then* '*b*'→ 'ㅂ').

영어 단어 *board*에 대한 발음과 한국어 음차표기가 */B AO R D/*와 '보드'와 같이 주어지면, 표 3, 4, 5와 같이 영어자소-음소, 음소-한글자소, 영어자소-음소-한글자소간의 정렬결과를 파악할 수 있다. *EPK* 정렬 알고리즘을 도식화하면 그림 3과 같이 나타내어진다.

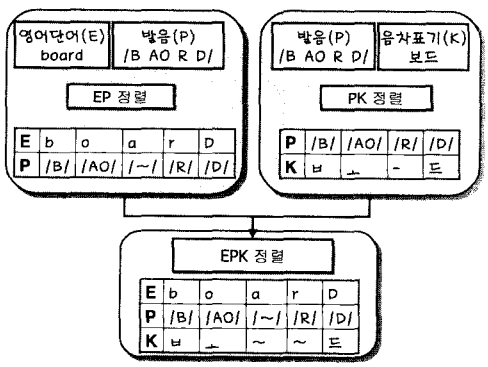


그림 3 EPK 정렬 알고리즘의 도식화

3.2.2 EP 정렬 알고리즘과 PK 정렬 알고리즘

EP 정렬 알고리즘과 *PK* 정렬 알고리즘은 강병주[6]가 제안한 영어자소-한글자소간 정렬 알고리즘을 기반으로 한다. 강병주[6]의 정렬 알고리즘은 Covington [14]의 알고리즘을 기반으로 한 것이다. Covington은 정렬을 두 단어가 주어졌을 때, 왼쪽에서 오른쪽으로 한 글자씩 '일치시키 (match)'와 '건너뛰기(skip)' 연산을 반복하면서 나아가는 것으로 생각하였다. 하지만 Covington의 알고리즘은 글자사이의 일대일 대응만을 가정하였기 때문에 *1:N*(일대다), *N:1*(다대일), *N:M*(다대다) 정렬관계를 파악할 수 없다. 강병주[6]는 Covington의 알고리즘을 확장하여 '앞쪽으로 묶기(forward bind)'와 '뒤쪽으로 묶기(backward bind)'라는 두 가지 연산을 추가하였다. 이를 통하여 *1:N*(일대다), *N:1*(다대일), *N:M* (다대다) 정렬이 가능하게 하였다.

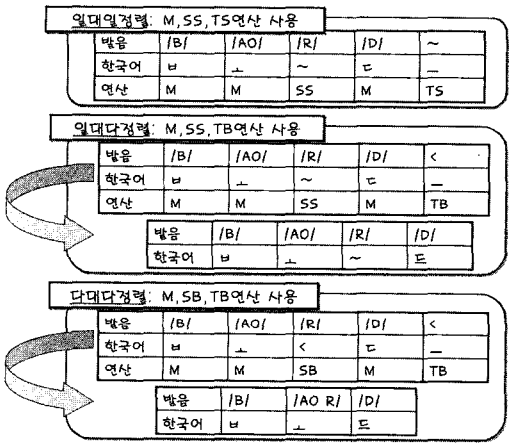


그림 4 발음 /B AO R D/와 음차표기 '보드'에 대한 일대일, 일대다, 다대다 정렬의 예

그림 4는 발음 /B AO R D/와 음차표기 '보드'에 대한 일대다, 다대일, 다대다 정렬의 예를 나타낸다. 그림에서 발음을 원어(source language)로 한국어를 목적어(target language)로 정의하면 연산은 **'일치시키기(M)'**, **'원어 건너뛰기(SS)'**, **'목적어 건너뛰기(TS)'**, **'원어묶기(SB)'**, **'목적어묶기(TB)'**가 사용된다. **'일치시키기'**는 원어의 정렬단위와 목적어의 정렬단위가 정렬된다는 것을 의미한다. **'건너뛰기'**에는 원어의 정렬단위가 목적어의 정렬단위와 일치되는 것이 없을 경우 정렬하지 않는 **'원어 건너뛰기'**와 목적어의 정렬단위가 원어의 정렬단위와 일치되는 것이 없을 경우 정렬하지 않는 **'목적어 건너뛰기'**가 있다. **'묶기'**에는 원어의 정렬단위가 목적어의 정렬단위와 일치되는 것이 없을 경우 목적어의 가장 가까운 앞 뒤 정렬단위와 정렬하는 **'원어 묶기'**와 목적어의 정렬단위가 원어의 정렬단위와 일치되는 것이 없을 경우 원어의 가장 가까운 앞 뒤 정렬단위와 정렬하는 **'목적어 묶기'**가 있다. '묶기'는 방향성에 따라 **'앞쪽 묶기'**('<')와 **'뒤쪽 묶기'**('>')로 구별된다. 그림에서 정렬단위는 음소와 한글자소가 된다. 따라서, 일대일 정렬은 하나의 음소와 하나의 한글자소가 대응하는 정렬을 나타내고, 일대다 정렬은 하나의 음소와 하나이상의

한글자소가 대응되는 정렬을 나타낸다. 그리고 다대다 정렬은 하나이상의 음소와 하나이상의 한글자소가 대응되는 정렬을 나타낸다.

이들 연구에서 정렬은 각 연산으로 얻어질 수 있는 모든 가능한 대응관계 중에서 최적의 대응 관계를 찾는 문제로 정의된다. 최적의 대응관계는 각 연산과 조건에 대한 벌점이 명시된 벌점 행렬을 이용하여 결정되며, 가장 벌점이 작은 대응관계가 최적의 대응관계라고 정의된다. 본 논문에서는 최적의 대응관계를 찾기 위한 평가행렬을 표 6과 같이 정의하였다. 벌점은 강병주[6]에서 사용한 평가행렬의 벌점값을 변형하여 사용하였다. 표 6에서 '유사한 자음/자음'과 '유사한 모음/모음'은 영어자소-음소간 정렬테이블 (EP 정렬테이블)과 음소-한글자소간 정렬테이블 (PK 정렬테이블)에 의해 결정된다. 만약 영어자소-음소가 *EP* 정렬테이블에 존재하면 유사하다고 판단하고, 존재하지 않으면 유사하지 않다고 판단한다. 표 7과 8은 *EP* 정렬테이블과 *PK* 정렬테이블의 예를 보여준다.

표 6 벌점 행렬

연산	조건	벌점
일치시키기 (Match:M)	유사한 자음/자음	0
	유사한 모음/모음	0
	모음/반모음, 자음/반모음	30
	유사하지 않은 자음/자음	240
	유사하지 않은 모음/모음	100
	자음/모음, 모음/자음	250
묶어주기 (Bind:SB,TB)	유사한 자음/자음	20
	유사한 모음/모음	20
	모음/반모음, 자음/반모음	50
	유사하지 않은 자음/자음	190
	유사하지 않은 모음/모음	120
	자음/모음, 모음/자음	200
건너뛰기 (Skip: SS, TS)	모음	40
	자음	40

표 7 EP 정렬테이블의 예

영어자소	음소
A	"AA","AE","AH","EY"
B	"B"
C	"CH", "K","S"
D	"D"

표 8 PK 정렬테이블의 예: 여기에서 '*'는 종성자음을 나타낸다.

음소	한국어자소
AA	"아", "아"
EH	"에", "에"
K	"ㅋ", "ㄱ*"
DH	"ㄷ", "ㅈ"

기존의 자소 정렬 기법은 비교적 높은 성능을 나타내지만, 높은 시간 복잡도를 가진다. 기존의 방법은 모든 가능한 정렬관계를 고려하기 위하여 *n*갈래 탐색트리(n-way branching search tree)를 구성한 후 깊이 우선 탐색 방법(Depth First Search)을 이용하여 최적의 대응관계를 파악한다. Covington의 연구와 강병주의 연구에서는 각각 3갈래 탐색트리(M, SS, TS연산)와 3~5갈래 탐색트리(M, SS, TS, SB, TB 연산)를 구성하여 최적의 정렬을 탐색하였다. 하지만 이러한 방법은 단어의 길이에 따라 정렬의 탐색공간이 엄청나게 커질 수 있으며, 이로 인하여 높은 시간 복잡도를 가진다. 이들 방법에서 길이 *l, m*의 두 단어에 대한 *n*갈래 탐색트리의 탐색을 위한 시간 복잡도는 $O(n^{(l+m)})$이다. 이러한 높은 시간 복잡도는 대용량의 데이타를 처리할 때 문제점으로 나타나며, 본 논문에서는 이를 해결하기 위하여 동적 프로그래밍 기법을 이용하였다.

본 논문의 자소 정렬 알고리즘은 편집거리[6]를 계산하는 알고리즘을 기반으로 표 9와 같이 표현된다. *EP* 정렬 알고리즘과 *PK* 정렬 알고리즘은 같은 알고리즘으로 표현될 수 있으므로 본 논문에서는 *EP* 정렬 알고리즘을 중심으로 알고리즘을 기술한다. 표 9의 정렬 기법은 정렬을 위한 행렬을 구성하고 행렬을 탐색함으로써 영어자소-음소 또는 음소-한국어 자소간의 정렬을 수행한다. 따라서 길이 *l, m*의 두 단어에 대한 정렬을 위해서는 $l \times m$의 행렬에 대한 처리만이 필요하므로 $O(l \times m)$의 시간 복잡도를 가지며 기존기법의 $O(n^{(l+m)})$보다 시간복잡도를 줄였다. 표 9의 기법은 기존 기법과 같은 연산을 사용하므로 기존기법과 마찬가지로 일대다, 다대일, 다대다 형태의 정렬이 가능하다.

표 9의 알고리즘은 크게 5단계의 과정으로 정렬을 수행한다. 1~3단계까지는 정렬을 위한 행렬 *D*를 초기화하는 과정이며, 4단계는 행렬 *D*의 각 셀에 대한 연산과 각 연산에 대한 벌점을 할당하는 과정이다. 또한 5단계는 1~4단계에서 구성된 행렬을 이용하여 최적의 정렬을 찾는 과정이다. 표 10은 영어단어 *board*와 발음 /B AO R D/에 대하여, 표 9에 의해 생성되는 정렬 행렬과 최적의 정렬관계를 나타낸다. 최적의 정렬관계는 '*'이 표시된 경로를 탐색하여 생성된다. 탐색은 *d[n,m]*에서부터 시작되며, 해당 셀의 연산이 M일 경우에는 *d[i-1,j-1]*로, SS일 경우에는 *d[i-1,j]*로, TS일 경우에는 *d[i,j-1]*로의 탐색을 수행한다. 예제에서 *d[n,m]*은 *d[5,4]*

6) 편집거리(Edit distance)또는 루빈스타인 거리(Levenshtein distance)는 두 문자열간의 유사도를 측정하는 기법으로 하나의 문자열을 다른 문자열로 변환하기 위한 삽입, 삭제, 치환의 개수로 정의된다[15].

표 9 EP, PK 정렬 알고리즘

단계	각 단계의 처리
1	Let s be source side input string and t be target side input string Set n to be the length of s and set m to be the length of t. Construct an $n \times m$ matrix (D) with s and t. Let d[i,j].penalty be a penalty value of d[i,j] and d[i,j].op be a operation which derives d[i,j].penalty
2	Initialize penalty score • □ d[0,0].penalty = 0; d[i,0].penalty = $300 \times i$; d[0,j].penalty = $300 \times j$;
3	Examine each character of s (i from 1 to n) and t (j from 1 to m).
4	Set cell d[i,j].penalty and d[i,j].op of the matrix • □ d[i,j].penalty = min(a,b,c); d[i,j].op = op(arg min(a,b,c)); • □ a = d[i-1,j].penalty + penalty(SB(s_i,t_j) or SS(s_i,t_j)); op=SB or SS • □ b = d[i-1,j].penalty + penalty(TB(s_i,t_j) or TS(s_i,t_j)); op=TB or TS • □ c = d[i-1,j-1].penalty + penalty(M(s_i,t_j)); op=M
5	Find the best alignment result by traversing matrix from d[n,m] to d[0,0] Retrieve cells with the following manners: • □ d[i-1,j] if d[i,j].op=SB or SS • □ d[i,j-1] if d[i,j].op=TB or TS • □ d[i-1,j-1] if d[i,j].op=M

이며, 탐색은 $d[5,4] \rightarrow d[4,3] \rightarrow d[3,2] \rightarrow d[2,2] \rightarrow d[1,1] \rightarrow d[0,0]$의 순으로 이루어진다. 그리고 이들 탐색경로와 각 경로의 연산으로부터 자소-음소간의 대응관계를 파악한다.

표 10 board와 /B AO R D/의 EP 정렬 알고리즘에 의해 정렬 행렬 및 정렬 결과. 여기에서는 SS, TS, M 연산만을 사용한 정렬 예를 나타낸다.

d	j=0	/B/ (j=1)	/AO/ (j=2)	/R/ (j=3)	/D/ (j=4)
i=0	**0***	300	600	900	1200
b (i=1)	300	**0 [M]***	40 [TS]	80 [TS]	120 [TS]
o (i=2)	600	40 [SS]	**0 [M]***	40 [TS]	80 [TS]
a (i=3)	900	80 [SS]	**40 [SS]***	80 [SS]	120 [TS]
r (i=4)	1200	120 [SS]	80 [SS]	**40 [M]***	80 [TS]
d (i=5)	1500	160 [SS]	120 [SS]	80 [SS]	**40 [M]***

영어	b	o	a	r	d
음소	/B/	/AO/	~	/R/	/D/
연산	M	M	SS	M	M

본 논문에서는 약 7,000개 영어단어-발음-한국어음차표기 단어에 대한 정렬을 수행하였다. 100개의 결과를 임의적으로 추출하여 평가한 결과, 강병주의 기법과 본 논문의 기법의 정확률은 98%로 동일하였다. 하지만 시간 복잡도는 현저히 줄어들어, 두 개의 *Ultra-Sparc II 296MHz CPU*와 *1024MB* 메모리에서 기존의 강병주의 방법이 9.12초에 정렬을 수행한 반면 본 논문의 기법으로는 0.12초만으로 정렬을 수행하여 약 75배의 성능향상을 나타내었다.

3.3 발음 생성 및 음차표기 생성

본 절에서는 발음 생성 및 음차표기 생성에 대하여 기술한다. 3.3.1절과 3.3.2절에서는 발음생성 및 음차표기 생성에 사용되는 기계학습 방법인 메모리 기반 학습(memory-based learning)[16]과 결정트리(decision tree) 학습[17][18]에 대하여 간략히 기술하고, 3.3.3절과 3.3.4절에서 발음생성 및 음차표기 생성에 대하여 기술한다.

3.3.1 메모리 기반 학습

메모리 기반 학습 방법은 k-최근접 이웃 알고리즘(k-nearest neighborhood algorithm)에 기반한 분류 기법이다[19-21]. 메모리 기반 학습방법에서는 학습개체를 특성자질과 해당 특성자질의 값을 이용하여 벡터로 표현한 후 해당 벡터를 메모리에 저장하는 방식으로 학습을 수행한다. 학습된 개체를 이용하여 주어진 개체에 대한 분류는 주어진 개체와 메모리에 저장된 학습개체간의 유사도를 비교하여 이루어진다. 주어진 개체를 x라고 하고 메모리에 있는 모든 개체의 집합을 Y라고 하면 유사도는 거리척도 $\Delta(x,Y)$에 의해 계산된다. 거리척도는 정보이득 가중치법(information gain weighting), 수정된 상이값 척도(modified value difference metric), 제프리 발산척도(Jeffrey divergence metric), 카이제곱 특성자질 가중치법(chi-squared feature weighting) 등이 사용된다[16]. x의 클래스는 x와 가장 유사한 학습개체 또는 학습개체 군(k-최근접 이웃)내에서 가장 빈번하게 나타나는 분류 클래스로 할당된다. 메모리기반 학습은 학습개체의 정보를 그대로 이용하기 때문에 예제기반 학습(example based learning), 케이스 기반학습(case based learning), 개체 기반 학습(instance based learning) 등으로 불린다[21-25].

그림 5는 발음추정에 사용되는 메모리기반 학습의 예를 나타낸다. 그림에서 모든 학습개체는 벡터로 표현된다. 벡터의 특성자질 중 C0는 현재 영어자소를 나타

학습개체

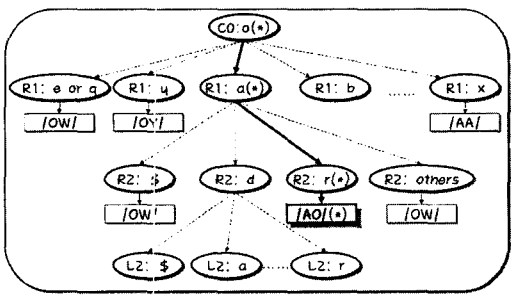
y_i	L3	L2	L1	C0	R1	R2	R3	δ_p	$\Delta(x,yi)$
1*	$	a	b	o	a	r	d	/AO/	0.93
2	h	a	b	o	$	$	$	/OW/	0.38
3	$	$	b	o	a	s	t	/OW/	0.81
4	$	$	b	o	a	t	$	/OW/	0.81
5	$	$	c	o	a	r	s	/AO/	0.73
6	$	e	d	o	a	r	d	/W/	0.75
7	c	k	s	a	w	$	$	/AO/	0.16
8	e	a	b	o	u	t	$	/UW/	0.51
9	$	$	b	o	a	n	$	/OW/	0.81
10	$	$	b	o	a	l	s	/OW/	0.81

x	L3	L2	L1	C0	R1	R2	R3		δ_p
	$	$	b	o	a	r	d	→	/AO/

그림 5 메모리기반 학습의 발음 추정 과정

내고, *R1, R2, R3*는 현재 영어자소의 오른쪽 자소들을 *L1, L2, L3*는 현재 영어자소의 왼쪽 자소들을 각각 나타낸다. 그림에서 각 학습개체를 y_i라고 정의하고 분류될 개체를 x라고 하면 y_i와 x간의 유사도인 $\Delta(x,y_i)$가 가장 높은 y_1의 분류클래스(δ_p) /AO/를 x의 분류클래스로 /AO/로 할당한다. 메모리기반 학습을 이용한 음차표기생성과정도 그림 5의 발음추정과정과 마찬가지 방식으로 학습된 개체 중 가장 유사한 개체의 분류클래스를 분류될 개체의 분류클래스로 할당한다.

3.3.2 결정 트리 학습



L3	L2	L1	C0	R1	R2	R3		δ_p
$	$	b	o	a	r	d	→	/AO/

그림 6 결정트리를 이용한 발음추정 과정의 예

본 논문에서는 결정트리기법으로 *ID3*계열의 귀납적 기계학습(inductive learning) 알고리즘인 *C4.5*를 이용하여 학습 예제들을 분류하는 결정트리를 만든다. *ID3*는 단순한 그리디(greedy) 하향식의 결정트리 구축 알고리즘으로 트리의 루트로부터 시작하여 재귀적으로 트리를 구축한다[17]. 그리고 각 노드에서 분기할 자질을 선택하고, 해당 노드의 예제들을 선택된 자질의 값에 따라 분류한다. 모든 예제가 같은 클래스 값을 가지거나 더 이상 테스트할 자질이 없을 때까지 분기과정이 반복된다. *ID3*에서 분기할 자질의 선택은 상호정보(mutual information)에 의해 이루어진다. *ID3*에서는 이를 정보이득(information gain)이라고 부른다[18]. 클래스 확률변수를 C, 자질 확률변수를 A라고 할 때 상호 정보 $I(A;C)$가 최대로 되는 자질부터 분기된다. 상호정보는 한 쪽의 확률변수에 대한 정보가 다른 쪽의 확률변수에 가져오는 불확실성의 감소분을 의미한다[26].

그림 6은 결정트리 기법에 의한 발음 추정과정을 나타낸다. 발음추정에서 정보이득에 의해 C→R1→R2→L2의 순으로 결정트리가 구성된다. 그림은 *board*의 자소 o에 대응되는 발음을 추정하기 위한 결정트리의 검색과정을 나타낸다. 결정트리의 검색은 실선과 같이 검색되며, (*)의 특성자질 조건에 의해 분기한다. 따라서 C:o(*)→R1:a(*)→R2:r(*)→/AO/와 같이 결정트리를 검색하여 o의 발음을 /AO/로 결정한다.

3.3.3 발음 생성

발음 생성(δ_p)은 영어단어의 각 자소 $e \in E$가 생성하는 음소들 $p=\{p_1,...,p_n\} \in 2^P$를 파악하는 작업 $\delta_p: E \rightarrow 2^P$로 정의된다. 여기에서, E는 영어자소의 집합, P는 음소의 집합을 나타낸다. 발음생성과정은 영어자소에 대응되는 음소를 할당하는 과정이기 때문에 발음생성의 결과는 각 자소와 각 자소에 대응되는 음소의 열로서 표현된다. 본 논문에서는 주어진 영어단어 $EW=e_1,e_2,...,e_n$에 대하여 발음생성의 결과를 $GP=gp_1,gp_2,\cdots,gp_n$로 표현한다. 여기에서 $gp_i=(e_i, \delta_p(e_i))$이며, e_i는 i번째 영어자소를 나태내고 $\delta_p(e_i)$는 e_i에 대응하는 음소를 나타낸다.

예를 들어 영어단어 *butyl*과 발음 /B Y UW T AH L/에 대하여, GP={gp_1=(b,/B/), gp_2=(u, /Y UW/), gp_3=(t,/T/), gp_4=(y, /AH/), gp_5=(l, /L/)}와 같이 표현할 수 있다.

표 11 butyl /B Y UW T AH L/에 대한 발음 생성 예

	gp_1	gp_2	gp_3	gp_4	gp_5
butyl	b	u	t	y	l
/B Y UW T AH L/	/B/	/Y UW/	/T/	/AH/	/L/

이러한 발음 생성 결과를 얻기 위하여, 본 논문에서는 "발음사전 검색" 방법과 "발음추정" 방법을 사용한다. 발음 사전은 주어진 단어에 대하여 정확한 발음을 미리 수작업으로 구축한 사전으로 주어진 단어에 대한 가능한 모든 발음을 수록하고 있다. 따라서 주어진 영어단어에 대한 발음을 생성할 경우 발음 사전을 우선적으로 검색한다. 사용한 발음 사전은 *CMU pronouncing dictionary*[28]로 약 120,000개 영어단어에 대한 발음을 포함하고 있다. CMU 발음 사전은 단어에 대한 발음을

포함하고 있기 때문에, 발음생성을 위해서는 단어와 발음의 대응관계를 자소와 음소의 대응관계로 변환하는 전처리가 필요하다. 본 논문에서는 3.2절에 기술한 *EP* 정렬 알고리즘으로 자소와 음소의 대응관계로 표현하고 이를 발음사전의 검색 결과로 사용한다.

표 12 발음 추정과 음차표기 생성을 위한 특성자질

표기	특성자질	특성자질값
E	영어자소	모든 영어 알파벳
P	음소	ARPAbet에 정의된 모든 음소
E'	영어자소의 일반화	자음, 모음
P'	음소의 일반화	자음, 모음, 반모음

하지만 발음사전이 모든 영어 단어에 대한 발음을 수록하고 있지 않기 때문에, 발음 사전에 등재되지 않은 단어에 대해서는 발음을 추정해야 한다. 본 논문에서는 발음추정을 위하여 기계학습방법인 메모리기반 학습[17]과 결정트리기법[18,19]을 사용하였다. 학습데이타는 CMU 발음사전의 영어-발음 쌍을 *EP* 정렬 알고리즘으로 정렬한 영어자소-음소간의 정렬결과를 사용하였으며, 학습을 위한 특성자질은 영어자소의 문맥정보를 사용하였다. 본 논문에서는 실험적으로 문맥의 크기를 3으로 한정하였다. 그리고 표 12의 특성자질 중 E와 E'을 사용하였다.

표 13는 *board*에 대한 발음추정과정을 나타낸다. 표 13에서 **L1, L2, L3**는 현재 자소의 왼쪽 문맥을 나타내며, **R1, R2, R3**는 현재 자소의 오른쪽 문맥을 나타낸다. 그리고 **C0**는 음소를 추정하고자 하는 현재자소를 나타내고 δ_p**(C0)**는 발음추정에 의해 생성된 음소를 나타낸다. 본 논문에서 $는 단어의 시작과 끝을 나타내는 기호로 사용하였다. 표 12의 결과는 다음과 같이 해석된다. 영어자소 *b*는 *b*가 생성하는 모든 가능한 음소집합 중에서 문맥정보 **L3, L2, L3, C0, R1, R2, R3**에 의하여 가장 적합한 음소 /B/를 발음추정함수 δ_p에 의해 생성한다. 영어자소 *o, a, r, d*도 마찬가지로 /AO/, /~/, /R/, /D/를 각각 생성한다. 여기에서 /~/는 묵음을 나타낸다.

3.3.4 한국어 음차표기 생성

한국어 음차표기 생성(δ_k)은 각 자소 e∈E와 자소에 대응되는 음소 $\delta_k(e)$를 이용하여 한국어 자소 $k=\{k_1,\cdots,k_m\}\in 2^K$를 파악하는 작업이며 $\delta_k: E\times 2^P \rightarrow 2^K$로 정의된다. 본 논문에서는 음차표기생성 결과를 $GK=gk_1, gk_2,\ldots,gk_n$로 표현한다. 여기에서 $gk_i=(gp_i, \delta_k(gp_i))$이며, gp_i는 i번째 영어자소에 대한 발음생성결과를 나태내고 $\delta_k(gp_i)$는 gp_i에 대응하는 한글자소를 나타낸다. 예를 들어, 영어단어 *butyl*, 발음 /B Y UW T AH L/, 한국어단어 '부틸'에 대하여, $GK=\{gk_1=$(b,/B/,ㅂ), $gk_2=$(u, /Y UW/,우), $gk_3=$(t,/T/,ㅌ), $gk_4=$(y, /AH/,이), $gk_5=$(l, /L/, ㄹ)와 같이 표현할 수 있다.

표 14 'butyl /B Y UW T AH L/ 부틸'에 대한 음차표기 생성 예

	gpk_1	gpk_2	gpk_3	gpk_4	gpk_5
butyl	b	u	t	y	l
/B Y UW T AH L/	/B/	/Y UW/	/T/	/AH/	/L/
부틸	ㅂ	우	ㅌ	이	ㄹ

음차표기생성(δ_k)는 발음추정과 마찬가지로 메모리기반 학습방법과 결정트리기법을 이용하여 학습한다. 학습데이타는 3.2절에 기술한 *EPK* 정렬 알고리즘에 의해 생성된 영어자소-음소-한글자소 정렬 데이타를 사용한다. 한글자소 할당함수 δ_k의 학습을 위하여 표 12의 *E, E', P, P'*의 특성자질을 사용하였다.

표 15는 영어 단어 *board*에 대한 한국어 음차표기 생성과정을 나타낸다. 표 15에서 **L1, L2, L3**는 왼쪽 문맥을 **R1, R2, R3**는 오른쪽 문맥을 나타내고, **C0**는 한글자소를 생성할 영어자소 및 음소를 나타낸다. 또한 δ_k(C0)는 음차표기생성함수 δ_k에 의해 생성된 한글자소를 나타낸다. $는 단어의 시작과 끝을 나타내는 기호이다. 표 15의 결과는 다음과 같이 해석된다. 영어자소 *b*와 대응되는 음소 /B/가 생성하는 모든 한글자소 집합

표 13 *board*에 대한 발음 추정의 예

특성자질	L3	L2	L1	***C0***	R1	R2	R3		δ_p***(C0)***
E	$	$	$	***b***	o	a	r	→	***/B/***
E'	$	$	$	***C***	V	V	C		
E	$	$	b	***o***	a	r	d	→	***/AO/***
E'	$	$	C	***V***	V	C	C		
E	$	b	o	***a***	r	d	$	→	~
E'	$	C	V	***V***	C	C	$		
E	b	o	a	***r***	d	$	$	→	***/R/***
E'	C	V	V	***C***	C	$	$		
E	o	a	r	***d***	$	$	$	→	***/D/***
E'	V	V	C	***C***	$	$	$		

표 15 *board*에 대한 음차표기 생성의 예

특성자질	L3	L2	L1	*C0*	R1	R2	R3		$\delta_k(C0)$
E	$	$	$	***b***	o	a	r	→	ㅂ
P	$	$	$	***/B/***	/AO/	~	/R/		
E'	$	$	$	***C***	V	V	C		
P'	$	$	$	***C***	V	~	C		
E	$	$	b	***o***	A	r	d	→	ㅗ
P	$	$	/B/	***/AO/***	~	/R/	/D/		
E'	$	$	C	***V***	V	C	C		
P'	$	$	C	***V***	~	C	C		
E	$	b	o	***a***	r	d	$	→	~
P	$	/B/	/AO/	~	/R/	/D/	$		
E'	$	C	V	***V***	C	C	$		
P'	$	C	V	~	C	C	$		
E	b	O	a	***r***	d	$	$	→	~
P	C	V	V	***C***	C	$	$		
E'	/B/	/AO/	~	***/R/***	/D/	$	$		
P'	C	V	~	***C***	C	$	$		
E	o	a	r	***d***	$	$	$	→	ㄷ
P	V	V	C	***C***	$	$	$		
E'	/AO/	~	/R/	***/D/***	$	$	$		
P'	V	~	C	***C***	$	$	$		

에 대하여, 음차표기 생성함수 δ_k는 L1, L2, L3, C0, **R1, R2, R3**의 정보를 이용하여 가장 적합한 한글자소 'ㅂ'을 생성한다. 마찬가지 방법으로 'ㅗ', '~', '~', 'ㄷ'가 생성되어 영어 단어 *board*와 발음 /B AO R D/에 대하여 음차표기 '보드'를 생성한다.

4. 실험 및 평가

4.1 실험환경

본 논문에서는 실험 및 평가를 위하여 두 가지 실험집합을 사용하였다. **실험집합 I**[1]은 1,650개 영-한 음차표기 쌍으로 구성된 실험집합으로 기존의 여러 연구에서 평가를 위한 실험집합으로 사용되었다[1,5-9]. 본 논문에서는 [1,5-9]와의 비교평가를 위하여 **실험집합 I**을 사용하였다. **실험집합 I** 중 1,500쌍은 학습데이타로 사용하고, 150쌍은 시험데이타로 사용하였다.

실험집합 II[6,13]는 7,185개 영-한 음차표기 쌍으로 구성되어 있으며, 6,185쌍은 학습데이타로 1,000쌍은 시험데이타로 사용하였다. **실험집합 II**는 [5,6,9]에서 사용한 실험집합으로 본 논문의 기법과 [5,6,9]의 성능평가를 위하여 사용한다.

평가 방법은 음차표기 평가 방법으로 널리 사용되는 단어정확도와 문자정확도를 이용한다. 단어정확도는 음차표기 시스템이 생성한 음차표기 중 올바르게 음차표기된 단어의 개수의 비율을 나타낸다. 문자 정확도는 음차표기 시스템이 생성한 음차표기의 문자열과 올바른 음차표기의 문자열간의 유사도를 나타낸다. 두 단어 사이의 글자간 유사도가 높을수록 올바른 음차표기에 근접한 결과를 생성한다고 판단한다. 따라서 단어정확도(Word Accuracy: W.A.)와 문자 정확도(Character accuracy: C.A.)가 높을수록 효과적인 음차표기 시스템이라 할 수 있다. 단어정확도와 문자정확도는 식 (4)와 같이 표현된다.

$$W.A. = \frac{\#of\ \ correct\ \ words}{\#of\ \ generated\ \ words}$$

$$C.A. = \frac{L-(i+d+s)}{L} \qquad (4)$$

여기에서 L은 원문자열의 길이를 나타내며, i,d,s는 각각 원문자열에서 목표문자열로 변환하기 위해 필요한 삽입, 삭제, 치환의 개수를 나타낸다. 만약 $L < (i+d+s)$이면 $C.A.$는 0으로 판단한다[28].

본 논문에서는 성능평가를 위하여 다음과 같은 네 가지 실험을 수행하였다.

1. 기존연구와의 성능비교 실험: 본 논문의 기법과 기존 연구와의 음차표기 성능을 비교 평가한다.
2. 발음생성방법에 따른 성능비교 실험: 발음사전 기반 발음생성방법과 발음추정에 의한 발음생성방법에 의해 생성된 발음에 대한 영-한 음차표기 성능평가를 수행한다.
3. 학습데이타 양에 따른 성능비교 실험: 학습데이타 양에 따른 영-한 음차표기의 성능을 비교 평가한다.
4. 문맥의 크기에 따른 성능비교 실험: 발음생성 및 음

차표기 추정에 사용한 문맥의 크기에 따른 영-한 음차표기의 성능을 비교 평가한다

4.2 실험결과

4.2.1 기존연구와의 성능비교 실험

표 16과 17은 **실험집합 I**과 **실험집합 II**에 대한 기존 연구와의 성능비교 실험 결과를 각각 나타낸다. 실험 결과에서 본 논문의 기법은 결정트리를 이용한 방법(DTree)과 메모리학습을 이용한 방법(MBL) 모두에서 기존의 방법보다 높은 성능을 나타낸다. 또한 **실험집합 I**에 대해서는 약 10%~70%의 W.A.의 성능향상을 나타내었으며, **실험집합 II**에 대해서는 약 5%~23%의 W.A.의 성능향상을 나타내었다. 그리고 본 논문의 기법에서 메모리 학습방법이 결정트리 방법보다 좋은 성능을 나타낸다. 이는 결정트리기법이 각 특성자질에 의한 분기방법인데 비해, 메모리기반 방법은 전체 특성자질간의 유사도를 비교하여 결과를 생성하기 때문으로 분석된다. 즉, 음차표기가 각 특성자질인 음소 또는 자소만의 변환 작업이 아니라 음소 및 자소의 통합적인 변환 작업이라는 측면에서 전체 특성자질을 비교하는 메모리기반 학습방법이 음차표기에 보다 유용함을 나타낸다.

표 16 실험집합 I에 대한 성능비교 실험 결과

방법	C.A.	W.A.
[1,8]	69.3%	40.7%
[7]	79.0%	35.1%
[5,6]	78.1%	37.6%
[9]	86.5%	55.3%
제안방법 (DTree)	89.35%	57.33%
제안방법 (MBL)	89.51%	60.67%

표 17 실험집합 II에 대한 성능비교 실험 결과

방법	C.A.	W.A.
[5,6]	81.8%	48.7%
[9]	89.05%	57.2%
제안방법 (DTree)	89.56%	58.4%
제안방법 (MBL)	89.29%	59.9%

실험결과는 직접방식에 기반한 기존의 자소 변환 방법보다 본 논문에서 제안한 자소 및 자소에 대응하는 음소 정보를 통합적으로 이용한 변환방법이 보다 효과적으로 한국어 음차표기를 생성함을 보여준다.

4.2.2 발음생성방법에 따른 성능비교 실험

발음생성방법에 따른 성능비교실험을 위하여 **실험집합 II**를 사용하였다. 표 18은 실험결과를 나타낸다. 표에서 '*R*'은 발음사전기반 발음생성방법을 이용한 영-한 음차표기 결과를 나타내며, '*NR*'은 발음추정기반 발음생성방법을 이용한 영-한 음차표기 결과를 나타낸다. 결과에서 '*R*'의 결과가 '*NR*'의 결과보다 높은 성능을 나타내는데, 이는 '*NR*'의 결과가 정확한 발음을 생성하는데 한계가 있기 때문으로 분석된다. 즉, 발음추정에 의한 방법은 표 19와 같이 60%~65%의 성능으로 발음을 생성한다. 올바르지 않은 발음은 정확한 한국어 음차표기를 생성하는데 노이즈로 작용하기 때문에, 보다 정확한 발음이 음차표기를 생성하는데 중요하다고 할 수 있다. 하지만, 발음추정의 성능에 비해 '*NR*'의 결과는 '*R*'의 결과에 비교해서 비교적 좋은 성능을 나타냄을 알 수 있으며, 두 가지 발음생성 방법에 의해 생성된 발음에 대하여 효과적으로 한국어 음차표기를 생성함을 알 수 있다.

표 18 발음생성방법에 따른 성능비교 실험 결과

Type	메모리기반학습		결정트리	
	W.A.	*C.A.*	*W.A.*	*C.A.*
R	64.77%	90.76%	63.90%	90.92%
NR	48.88%	86.09%	46.33%	86.59%

표 19 발음추정의 성능 평가 결과

	발음추정 정확률
결정트리학습	59.64%
메모리기반학습	64.63%

4.2.3 학습데이타 양에 따른 성능비교 실험

학습데이타 양에 따른 성능을 평가하기 위하여 **실험집합 II** 실험집합을 사용하였다. 실험은 학습데이타 양을 전체 학습데이타의 20%, 40%, 60%, 80%, 100%로 변화시키면서 학습을 수행한 후 성능을 비교평가 하였다. 실험결과에서 학습데이타 양이 증가함에 따라 성능이 향상됨을 알 수 있으며, 적은 양의 학습데이타에서도 비교적 높은 성능을 나타냄을 알 수 있다. 즉 20%의 학습데이타 만으로도 51%~53%의 *W.A.*를 나타낸다. 또한 학습데이타의 증가에 따른 성능의 향상을 보인다.

표 20 학습데이타 양에 따른 성능 평가 결과

Training Size	메모리기반학습		결정트리	
	W.A.	C.A.	W.A.	C.A.
20%	52.80%	86.88%	51.30%	87.68%
40%	55.10%	87.60%	52.90%	88.10%
60%	55.60%	88.11%	55.20%	88.52%
80%	58.50%	88.86%	57.70%	89.23%
100%	59.90%	89.29%	58.40%	89.56%

4.2.4 문맥의 크기에 따른 성능비교 실험

문맥의 크기에 따른 성능을 평가하기 위하여 **실험집합 II**를 사용하였다. 실험은 문맥의 크기를 2~4로 변화시키면서 수행하였으며, 가장좋은 성능을 나타내는 메모

리기반학습방법을 사용하였다. 표 21은 실험결과를 나타낸다. 실험결과에서 문맥의 크기가 3인 것이 가장 좋은 성능을 나타내었다. 문맥의 크기가 2인 경우의 성능 하락 요인은 음차표기를 생성하기 위한 충분한 정보를 제공하지 못했기 때문으로 분석된다. 예를 들어, '션'이라고 음차표기되는 *-tion*에서 자소 *t*의 올바른 발음과 음차표기를 생성하기 위해서는 *t*의 오른쪽 세 개의 자소 *i,o,n*가 필요하다. 문맥의 크기가 4인 경우 문맥의 크기 3보다 성능이 떨어지는 이유는 학습개체의 다양성이 증가하여 규칙의 일반화가 어려웠기 때문으로 분석되었다. 이러한 이유로 문맥의 크기가 3인 경우 가장 좋은 성능을 나타낸 것으로 분석되었다.

표 21 문맥의 크기에 따른 성능평가 결과

문맥크기	W.A.	C.A.
2	56.8%	89.45%
3	62.2%	92.45%
4	55.4%	88.93%

4.2.5 실험결과 요약

본 장에서의 실험결과는 다음과 같이 요약될 수 있다.

- 본 논문의 기법은 기존의 기법보다 높은 성능을 나타내며, 최고 70%의 성능향상을 나타내었다. 이를 통하여 자소 및 음소 정보가 음차표기에 유용함을 보였다.
- 발음생성방법에 따른 실험 결과, 올바른 발음이 정확한 음차표기를 생성하는데 도움을 줄 수 있음을 알 수 있었다. 또한 올바르지 않은 발음에 대해서도 본 논문의 기법은 비교적 효과적으로 음차표기를 생성함을 알 수 있었다.
- 본 논문의 기법은 적은 학습데이타에 대해서도 높은 성능을 나타내었다. 또한 학습데이타가 늘어날수록 성능의 향상을 나타내었다.

4.3 오류 분석

오류는 크게 음성적 동등 오류와 음성적 비동등 오류로 분류할 수 있다. 음성적 동등오류는 시스템이 생성한 음차표기가 정답집합에 나타난 음차표기와 형태는 다르지만 올바른 한국어 음차표기로 생각할 수 있는 것을 나타낸다. 음성적 비동등 오류는 시스템이 생성한 결과가 정답과 형태가 틀릴 뿐만 아니라 올바른 한국어 음차표기로 생각할 수 없는 것을 나타낸다. 표 22는 음성적 동등 오류의 예를 나타낸다. 음성적 동등오류는 표 22의 예와 같이 '악센트/액센트', '앰뷸런스/앰불런스'와 같이 주로 하나의 모음의 혼동에 의해 나타나는 경우와 '치누크/치눅', '아우트리치/아웃리치'와 같이 비슷한 음가의 자음이 종성에 사용되는 경우와 초성에 사용되는 경우(자음의 혼동)에 주로 나타난다. 음성적 동등 오류

표 22 음성적 동등 오류의 예

영어 단어	올바른 음차표기	시스템 생성 음차표기
accent	악센트	액센트
ambulance	앰뷸런스	앰불런스
Canter	캔터	칸터
Chinook	치누크	치눅
community	커뮤니티	코뮤니티
gel	젤	젤
minette	미넷	미네트
outreach	아우트리치	아웃리치
record	레코드	리코드

표 23 음성적 비동등 오류의 예

유형	영어단어	올바른 음차표기	시스템 생성 음차표기
모음만 잘못된 오류	actuator	액추에이터	악추에이터
	divergent	다이버전트	디버젠트
	italic	이탤릭	아이탤릭
자음만 잘못된 오류	fanfare	팽파르	판파르
	henna	헤나	헨나
	mammoth	매머드	매머트
모음과 자음이 모두 잘못된 오류	carat	캐럿	캐라트
	ermine	어민	에르민
	trachoma	트라코마	트레이초마

는 401개 오류 중 115개가 해당되었다. 유형별로는 모음의 혼동에 의한 것이 84개, 자음의 혼동에 의한 것이 31개였다. 음성적 동등 오류는 올바른 음차표기로 파악될 수 있기 때문에 정답으로 판단할 수 있다. 하지만, 본 논문에서는 정답집합에 나타난 단어가 아닌 경우는 잘못된 음차표기로 판단하였다.

음성적 비동등 오류는 총 286개로 이 중 143개가 모음만이 잘못된 오류, 50개가 자음만 잘못된 오류, 그리고 93개가 모음과 자음이 모두 잘못된 오류였다. 표 23은 각 유형별 오류의 예를 나타낸다. 모음의 오류가 비교적 많이 나타나는 이유는 음차표기시 모음의 애매성이 크기 때문으로 분석된다. 즉, 모음의 경우 문맥에 따라 다양한 형태로 음차표기되기 때문에 이를 효과적으로 처리하기 어렵기 때문이다. 예를 들어 표 23에서 모음자소 *i*는 영어 단어 *divergent*에서는 '아이'로 음차표기 되지만 영어단어 *italic*에서는 '이'로 음차표기된다. 따라서 보다 정확한 음차표기를 생성하기 위해서는 모음에 대한 보다 정교한 처리가 필요할 것으로 생각된다.

5. 결 론

본 논문에서는 자소 및 음소정보를 이용한 영-한 음차표기 모델을 제안하였다. 본 논문의 기법은 기존 연구와 달리 자소정보 뿐만 아니라 음소정보를 사용하였다.

즉, 직접방식과 피봇방식을 통합한 기법으로 높은 성능을 나타내었다. 또한 기존 방법보다 높은 성능을 나타내었으며, 최고 70%의 성능향상을 나타내었고, 적은 양의 학습데이타에 대해서도 비교적 높은 성능을 나타냄을 알 수 있었다.

향후, 최대엔트로피 모델(Maximum entropy model), 지지벡터기계(Support vector machine) 등 여러 가지 기계학습을 이용한 음차표기 방법에 대한 연구가 추가적으로 수행될 예정이며, 영-일 및 영-중 자동 음차표기에 대한 연구도 추가적으로 수행될 예정이다. 본 논문의 기법은 정보검색, 기계번역, 이중언어 사전 구축, 음성 인식 등 여러 자연언어응용에 유용하게 사용될 수 있을 것으로 기대된다.

참 고 문 헌

[1] 이재성, 다국어 정보검색을 위한 영-한 음차표기 및 복원 모델, 박사학위논문, 한국과학기술원 전산학과, 1999.

[2] 이희승, 안병희, 고친판 한글 맞춤법 강의, 신구문화사, 1994.

[3] Kang, Y. and A. A. Maciejewski, "An algorithm for Generating a Dictionary of Japanese Scientific Terms", Literary and Linguistic Computing, 11(2)., 1996.

[4] Knight, K. and J. Graehl, "Machine Transliteration". In Proceedings. of the 35th Annual Meetings of the Association for Computational Linguistics (ACL) Madrid, Spain, 1997.

[5] Kang B.J. and K-S. Choi, "Automatic Transliteration and Back-transliteration by Decision Tree Learning", In Proceedings of the 2nd International Conference on Language Resources and Evaluation, Athens, Greece, 2000.

[6] 강병주, 한국어 정보검색에서 외래어와 영어로 인한 단어불일치문제의 해결, 박사학위논문, 한국과학기술원 전산학과, 2001.

[7] 김정재, 이재성, 최기선, "신경망을 이용한 발음단위 기반 자동 영-한 음차 표기 모델", 한국인지과학회 춘계 학술대회 발표논문집, pp. 247-252, 1999.

[8] Lee, J. S. and K. S. Choi, English to Korean Statistical transliteration for information retrieval. Computer Processing of Oriental Languages, 12(1):17-37., 1998.

[9] Kang I.H. and G.C. Kim, "English-to-Korean Transliteration using Multiple Unbounded Overlapping Phoneme Chunks", In Proceedings of the 18th International Conference on Computational Linguistics, 2000.

[10] 문화부, 표준 외래어 표기법, 1995

[11] James A., Natural Language Understanding, The Benjamin/ Cummings Publishing Company, 1995.

[12] Soong Frank. K. and Eng-Fong Huang, A tree-trellis based fast search for finding the n best sentence hypotheses in continuous speech recognition. In IEEE International Conference on Acoustic Speech and Signal Processing, pp. 546-549, 1991.

[13] 남영신, 1997, 최신 외래어 사전, 국어사전 별책부록, 서울: 성안당 출판사.

[14] Covington, M. A., "An algorithm to align words for historical comparison", Computational Linguistics, 22,.1996.

[15] Levenshtein V. I.. Binary codes capable of correcting deletions, insertions and reversals. Doklady Akademii Nauk SSSR 163(4) p845-848, 1965.

[16] Daelemans W., Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch, Timble TiMBL: Tilburg Memory Based Learner, version 4.3, Reference Guide, ILK Technical Report 02-10, 2002.

[17] Quinlan, J.R., "Induction of decision trees," Machine Learning, Vol. 1 pp. 81-106., 1986.

[18] Quinlan, J.R., "C4.5: Programs for Machine Learning", Morgan Kauffman., 1993.

[19] Cover, T. M. and P. E. Hart. 1967. Nearest neighbor pattern classification. Institute of Electrical and Electronics Engineers Transactions on Information Theory, 13:2127, 1967.

[20] Devijver, P. .A. and J. Kittler. Pattern recognition. A statistical approach. Prentice-Hall, London, UK, 1982.

[21] Aha, D. W., D. Kibler, and M. Albert, Instance-based learning algorithms. Machine Learning, 6:3766, 1991.

[22] Stanfill, C. and D. Waltz. Toward memory-based reasoning. Communications of the ACM, 29(12): 12131228, December., 1986.

[23] Cost, S. and S. Salzberg, A weighted nearest neighbour algorithm for learning with symbolic features. Machine Learning, 10:5778, 1993.

[24] Kolodner, J., Case-based reasoning. San Mateo, CA: Morgan Kaufmann, 1993.

[25] Aha, D. W., Lazy learning: Special issue editorial. Artificial Intelligence Review, 11:710, 1997.

[26] Manning, C.D. and Hinrich Schutze, Foundations of Statistical natural language Processing, MIT Press, 1999.

[27] CMU, The Carnegie Mellon University (CMU) Pronouncing Dictionary v0.6, http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[28] Hall, P., and G. Dowling, "Approximate string matching," Computing Surveys, 12(4), 381-402, 1980.

오 종 훈

1998년 성균관대학교 정보공학과 졸업 (학사). 2000년 한국과학기술원 전산학과 졸업 (공학석사). 2000년~현재 한국과학기술원 전산학과 박사과정. 관심분야는 자연언어처리, 전문용어, 정보검색 등

최 기 선

1978년 서울대학교 수학과 졸업 (학사) 1980년 한국과학기술원 전산학과 졸업 (공학석사). 1986년 한국과학기술원 전산학과 졸업 (공학박사). 1985년~1986년 한국외국어대학교 전산학과 조교수. 1987년~1988년 일본 NEC C&C 정보연구소 초빙연구원. 1988년~현재 한국과학기술원 전산학과 교수 1998년~현재 한국과학기술원 전문용어언어공학연구센터 소장. 관심분야는 자연언어처리, 기계번역, 정보검색, 전문용어