한국음향학회지 제28권 제3호 pp. 239~244 (2009)

한국어 음소결합확률 계산기 개발연구

A Study of Development for Korean Phonotactic Probability Calculator

이 찬 종*, 이 현 복**, 최 훈 영***
(Chan-Jong Lee*, Hyun-Bok Lee**, Hun-Young Choi***)

*강원대학교 인문과학연구소, **서울대학교 언어학과, ***강원대학교 컴퓨터과학과
(접수일자: 2009년 2월 10일; 채택일자: 2009년 3월 5일)

본 연구는 현대 한국어 단어의 말소리가 결합하는 음소결합확률 (Phonotactic Probability)을 예측하는 계산기 엔진 개발에 관한 연구이다. 한국이 음소결합확률계산기 (이하, KPPC)는 첫째로, 한국어의 주어진 단어에서의 음소와 그 음소의 음소결합의 빈도를 예측하여 말소리가 단어내의 특정위치에서 특정 분절음이 나타나는 빈도 값, 두 음소간의 결합의 빈도값, 그리고 세 음소간의 결합의 빈도 출현율을 예측하여 계산한다. 둘째로 한국어의 주어진 단어에서 말소리 하나만 다르면서 실제로 존재할 수 있는 근접밀도 (neighborhood density)의 값을 계산한다. University of Kansas에서 개발된 음소결합계산기는 영어 20,000단어의 D/B를 대상으로 위치별 분절음빈도와 두 음소간의 음소결합률 빈도를 컴퓨터가 읽을 수 있는 반음기호를 통해서만 가능하다. 본 연구에서는 분절음빈도와 두 음소간의 빈도뿐만 아니라 세 음소간의 결합률 빈도와 근접밀도율을 예측할 수 있고 입력할 때 반음기호뿐만 아니라 단어를 입력하면 확률값을 얻을 수 있다. 이 엔진은 67,284 단어의 한국어 표준발음을 D/B로 구축하여 고빈도 음소결합확률, 저빈도 음소결합확률, 고빈도 근접빈도, 저빈도 근접빈도의 값을 예측할 수 있다.

핵심용어: 음소결합확률, 음소결합확률 계산기, 근접밀도, 고빈도, 저빈도
투고분야: 말소리 생성 및 인지 분야 (12.4)

This paper is to develop the Korean Phonotactic Probability Calculator (KPPC) that anticipates the phonotactic probability in Korean. KPPC calculates the positional segment frequency, position-specific biphone frequency and position-specific triphone frequency. And KPPC also calculates the Neighborhood Density that is the number of words that sound similar to a target word. The Phonotactic Calculator that was developed in University of Kansas can be analyzed by the computer-readable phonemic transcription. This can calculate positional frequency and position-specific biphone frequency that were derived from 20,000 dictionary words. But KPPC calculates positional frequency, positional biphone frequency, positional triphone frequency and neighborhood density. KPPC can calculate by korean alphabet or computer-readable phonemic transcription. This KPPC can anticipate high phonotactic probability, low phonotactic probability, high neighborhood density and low neighborhood density.

Keywords: Phonotactic Probability, Phonotactic Probability Calculator, Neighborhood Density, High Frequency, Low Frequency

ASK subject classification: Speech Production and Perception (12.4)

I. 서 론

본 연구의 목적은 한국어 음소결합확률 계산기 알고리즘 개발에 관한 연구이다. 67,284 단어의 방대한 단어 D/B를 대상으로 한국이 대상의 음소결합확률 및 근접밀도를 측정할 수 있는 계산기 소프트웨어가 국내에서 개발되지 않은 상황에서 본 연구는 그 의의가 크다고 하겠다.

국외에서 개발된 음소결합확률계산기 (Phonotactic Probability Calculator)로 미국 캔자스대학교 (University of Kansas)의 [1,2]가 있다. 이 계산기는 영어사전의 20,000 단어를 대상으로 하여 분절음의 위치별 확률, 두 음소간의 출현율을 계산할 수 있다. 그러나 이 계산기의 경우 확률을 계산할 때 컴퓨터가 읽을 수 있는 반음기호로만

책임저자: 이 찬 종 (cjlee@kangwon.ac.kr)
200-701 강원도 춘천시 강원대학길 1강원대학교 인문대학 2호관 309호 인문과학연구소
(전화: 033-250-7252; 팩스: 033-250-7257)

음소결합확률 처리가 가능하였다. 그러나 본 KPPC는 한국어에 대해 발음기호뿐만 아니라 한글발음을 입력할 수 있도록 사용자에게 보다 편한 환경을 구축하였고 세 음소간의 음소결합확률 및 근접밀도까지 계산이 가능하게 하였다.

KPPC는 다음의 기능을 수행할 수 있다.

첫째, 말소리가 주어진 단어내에서 특정 분절음이 나타나는 빈도 값을 계산한다.

둘째, 단어내에서 특정위치에서의 두 음소간의 출현률을 예측하여 계산한다.

셋째, 단어내에서 특정위치에서의 세 음소간의 출현률을 예측하여 계산한다.

넷째, 입력된 단어와의 근접밀도 (Neighborhood Density)의 값을 계산하여 근접 단어의 수를 계산한다.

II. 프로그램 개발

프로그램 개발을 위해 [3]의 한국어 표준발음사전을 대상으로 67,284개의 단어를 한국어 사전 데이터로 사용하였다. 프로그램에서 사용되는 데이터는 사전에 수록된 단어들을 Excel로 전환한 것을 사용하였다.

OS는 MS WindowsXP이며, 프로그램은 Delphi7로 작성하였다. 전체적인 프로그램 구성도는 그림 1과 같다. 표 1은 프로그램에 사용된 알고리즘과 표기법을 나타낸다.

알고리즘은 크게 4개의 기능으로 이루어져 있다. 표 2는 그 중에서 가장 처음 단계인 사전 등록을 담당하는 절차이다. 현재 본 프로그램은 1인 사용자를 가정하고 만들어진 프로그램이다. 하지만 차후 네트워크를 통해 동시에 많은 사용자가 사용할 수 있는 환경, 가령 웹과 같은 환경에서도 사용하기 위해 위에 보이는 사전등록절차 (registration dictionary procedure) 부분에서 많은 일을 담당하도록 설계가 되어 있다. 웹과 같은 환경은 기본적으로 서버-클라이언트 (server-client)로 동작이 이루어지며, 이는 서버에 많은 부담을 준다. 따라서 사전을 등록할 때, 차후 이루어질 계산에 필요한 부분을 최대한 미리 작업을 하여 이후 많은 사용자가 동시에 작업을 서버에 요구하더라도 서버는 최소한의 계산만을 하도록 설계하였다. 따라서 사전 등록이 이루어질 때 서버에서는 계산이 많이 이루어지지만 이는 사전을 등록할 때 단 1번만 이루어지기 때문에 지속적인 부담을 주지 않는다는 큰 장점을 지닌다.

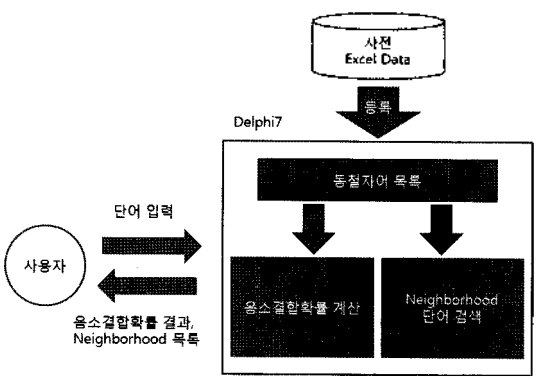

그림 1. KPPC 프로그램 구성도
Fig. 1. KPPC Program.

표 1. 표기법
Table 1. Algorithm code.

기호	정의
search_string	사용자에 의해 입력된 단어로 type은 한글 단어 (dictionary), 컴퓨터 가독형 음성기호 (computer readable phonetic symbol)로 구성
homograph	사용자가 입력한 단어와 동철자인 단어들의 집합
frequency	말소리가 전체 단어내에서 특정 분절음이 나타나는 빈도 값
biphone_frequency	전체 단어내에서 특정위치에서의 두 음소의 출현률 예측 값
triphone_frequency	단어내에서 특정위치에서의 세 음소의 출현률 예측 값
neighborhood_density	입력된 단어와의 근접밀도 (Neighborhood Density)의 값
neighborhood	근접 (Neighborhood) 단어 집합
dictionary	사전 데이터로 k개 단어들의 집합 ($1 \le x \le k$), 하나의 단어는 사전 단어 (word), 컴퓨터 가독형 음성기호 (computer readable phonetic symbol), 한글발음기호 (korean phonetic symbol)로 구성
ChangeTable[t]	한글 발음 컴퓨터 가독형 음성기호 변환표 ($1 \le t \le s$)
WordList[x][j]	memory에 올려진 사전 데이터 및 기초자료 ($1 \le x \le k$), j는 총 4개로 구성되어 있으며 0에는 사전 단어, 1에는 컴퓨터 가독형음성기호, 2에는 한글발음기호, 3에는 한글발음기호 길이가 수록
PositionValue[t+1][y]	사전에 수록된 단어에 포함된 각 음소들의 위치별 count ($1 \le t \le s$, $1 \le t \le$ 사전에 등록된 단어 중에서 가장 긴 단어의 길이), PositionValue[t+1][y]에는 y번째 위치에 음소가 있는 모든 단어의 count가 수록
ListValue[x][n]	사전에 등록된 단어와의 음소결합빈도 값과 neighborhood density count ($1 \le x \le k$), n은 총 4개로 구성되어 있으며 0에는 positional segment frequency 값, 1에는 biphone frequency 값, 2에는 triphone frequency 값, 3에는 neighborhood density count가 수록

표 2를 좀 더 자세히 설명하겠다. 사전 등록 절차는 사전 파일과 사전에 사용된 컴퓨터 가독형 음성변환표를 입력받는다.

라인 3~9는 사전 파일에 수록된 데이터를 컴퓨터 메모리에 탑재하는 과정이다. 사전 파일의 데이터는 WordList에 수록된다. WordList의 1차원은 사전에 있는 단어들의 수와 동일한 크기를 가진다. 다음으로 2차원에는 각 단어들에 대한 데이터가 수록된다. 0번째에는 사전에 등록된 한글 데이터, 1번째에는 사전에 해당 단어의 컴퓨터가독형음성기호, 2번째에는 해당 단어의 한글발음기호, 3번째에는 해당 단어의 한글발음의 길이가 기록된다.

표 2. 사전등록절차
Table 2. Dictionary registration procedure.

```
input: dictionary, ChangeTable[t]
output: WordList[x][j], PositionValue[t+1][y], ListValue[x][n]
01  procedure registration_dictionary (dictionary, ChangeTable[t])
02  begin
03    for x:=1 to k do
04      read dictionaryx;
05      WordList[x][0]:= dictionaryx.word;
06      WordList[x][1]:= dictionaryx.english_phonetic_symbol;
07      WordList[x][2]:= dictionaryx.korean_phonetic_symbol;
08      WordList[x][0]:= length (dictionaryx.korean_phonetic_symbol)
- ( ':', '(', ')', after all symbol '/');
09    end
10    for x:=1 to k do
11      for l:=1 to length (WordList[x][1]) do
12        search o when WordList[x][1]l = ChangeTable[o];
13        PositionValue[o][l]++;
14        PositionValue[t+1][l]++;
15      end
16    end
17    for x:=0 to k do
18      for l:=1 to length (WordList[x][1]) do
19        search o when WordList[x][1]l = ChangeTable[o];
20        ListValue[x][0]:= ListValue[x][0] + PositionValue[o][l] /
PositionValue[t+1][l];
21      end
22      for l:=1 to length (WordList[x][1])-1 do
23        search o when WordList[x][1]l = ChangeTable[o];
24        search p when WordList[x][1]l+1 = ChangeTable[p];
25        ListValue[x][1]:= ListValue[x][1] + min (PositionValue[o][l],
PositionValue[p][l+1]) /
min (PositionValue[t+1][l], PositionValue[t+1][l+1]);
26      end
27      for l:=1 to length (WordList[x][1])-2 do
28        search o when WordList[x][1]l = ChangeTable[o];
29        search p when WordList[x][1]l+1 = ChangeTable[p];
30        search q when WordList[x][1]l+2 = ChangeTable[q];
31        ListValue[x][2]:= ListValue[x][1] + min (PositionValue[o][l],
PositionValue[p][l+1], PositionValue[q][l+2]) /
min (PositionValue[t+1][l], PositionValue[t+1][l+1],
PositionValue[t+1][l+2]);
32      end
33    end
34    for x:=0 to k do
35      for u:=0 to k do
36        if length (WordList[x][1]) = length (WordList[u][1]) then
37          if less then 1 different symbol between WordList[x][1]
and WordList[u][1] then
38            ListValue[x][3]++;
39    end
40  end
```

라인 10~16은 사전에 포함된 단어의 각 음소의 위치별 빈도를 계산하는 과정으로 계산된 결과는 PositionValue에 저장된다. PositionValue의 1차원은 사전의 모든 음소를 고려해야 하므로 컴퓨터가독형 음성변환표에 수록된 음소들의 수와 동일한 크기를 가진다. 다음으로 2차원은 해당 음소가 단어에서의 위치 정보를 저장한다. 따라서 기본적으로 사전에 수록된 단어 중에서 최대 길이의 위치도 저장해야하므로 최대 길이의 크기를 가지며, 추가적으로 하나의 크기를 더 주어서 해당 위치에서 모든 음소들의 총 빈도를 저장한다.

라인 17~33은 사전에 포함된 각 단어들의 음소결합빈도를 계산하는 과정으로 계산된 결과는 ListValue에 저장된다. ListValue의 1차원은 사전에 있는 단어들의 수와 동일한 크기를 가진다. 다음으로 2차원에는 각 단어들의 음소결합빈도 값이 저장된다. 0번째에는 위치별 분절음 빈도 (positional segment frequency) 값이 저장되고, 1번째에는 위치별 두 음소빈도 (position-specific biphone frequency)값이 저장되며, 2번째에는 위치별 세 음소빈도 (position-specific triphone frequency) 값이 저장된다.

위치별 분절음 빈도를 계산하는 과정은 18~21 라인에 나타나 있다. 위치별 분절음 빈도는 말소리가 주어진 단어에서 특정 분절음에 나타나는 빈도를 나타내며, 단어의 빈도/특정 위치에서의 음소를 갖는 단어의 빈도로 표현할 수 있다. 이는 기존에 작성되었던 위치값을 이용하여 손쉽게 계산할 수 있다. 다음으로 위치별 두 음소빈도를 계산하는 과정은 22~26 라인에 나타나 있으며, 단어 내에서 특정 위치에서의 두 음소의 출현률을 계산한다. 이는 위치별 분절음 빈도를 계산하는 방법과 기본적인 원리가 동일하므로 자세한 설명은 하지 않도록 하겠다. 단, 두 음소를 대상으로 하므로 연속된 두 음소의 위치별 값에서 더 작은 값을 사용하여 계산하는 부분만 추가되었다. 마지막으로 위치별 세 음소빈도를 계산하는 과정은 27~32 라인에 표현되어 있다.

라인 34~39는 사전에 포함된 각 단어들의 근접밀도 (neighborhood density)를 계산하는 과정으로 계산된 결과는 List Value의 2차원의 3번째에 저장된다. 비교하는 단어와 음소 하나 이하의 차이일 경우 해당 단어는 근접

밀도어휘로 판단하며, 근접밀도값 (neighborhood density count)을 증가시킨다.

표 3은 4개의 알고리즘 중에서 검색을 담당하는 단계로 사용자가 입력한 단어를 등록된 사전 데이터를 바탕으로 음소결합빈도의 계산을 보여준다. 본 프로그램에서는 사용자가 검색하고자하는 단어를 입력할 때 두 가지 형태로 입력할 수 있도록 고안되었다. 하나의 형태는 사용자에게 익숙한 한글 단어이고 다른 하나는 컴퓨터 가독형 음성기호이다. 한글 단어의 경우 사용자에게 편리하다는 장점을 가지고 있지만, 한국어의 복잡한 발음형태를 고려하지 못한다는 단점을 가지고 있다. 이를 극복하기 위해서 본 프로그램에서는 사전에 등록된 단어에 한해서 동철자어를 처리할 수 있도록 고려하였다. 다음으로 기본적으로 음소결합 계산에서 사용되는 컴퓨터가독형 음성기호로도 입력할 수 있도록 사용자가 원하는 정확한 단어에 대한 검색 부분도 고려하였다.

표 3을 좀 더 자세히 설명하겠다. 검색절차 (search procedure)는 사용자가 입력한 검색어와 사전에 사용된 컴퓨터 가독형 음성기호 변환표를 입력받으며, 음소결합 계산의 수행을 위해 사전 등록 절차에서 생성한 단어목록과 위치값을 입력받는다.

라인 3~11은 사용자가 입력한 단어를 어휘목록과 비교하여 동음이의어가 있는지 비교하는 과정을 보여준다. 만약 동철자어가 있다면 동철자어 리스트에 추가하고, 없다면 사용자가 입력한 단어만을 기준으로 음소결합 계산을 할 준비를 한다.

라인 12~33은 각 동철자어나 사용자가 입력한 단어에 대해서 위치별 분절음 빈도, 위치별 두 음소빈도, 위치별 세 음소빈도를 계산하고 출력하는 과정을 보여준다. 알고리즘에 나타나듯 사전 등록 절차에서 생성된 PositionValue를 사용하여 특정 위치에 음소가 나타나는지 검색하는데 걸리는 시간을 단축하여 빠른 검색이 가능해진다.

표 4는 4개의 알고리즘 중에서 사전에 등록된 모든 단어에 대해서 음소결합빈도를 출력하는 과정을 보여준다. 알고리즘에서도 나타나듯이 이미 사전 등록 절차에서 생성한 WordList와 PositionValue를 이용하여 별도의 계산이 없이 즉각적으로 메모리에 저장된 내용을 단순히 출력하는 것으로 모든 과정이 종료된다.

표 3. 검색절차
table 3. Search procedure.

```
input: WordList[x][i], ChangeTable[t], PositionValue[t+1][y],
search_string
output: positional_frequency, biphone_frequency, triphone_frequency
01  procedure search (WordList[x][i], ChangeTable[t], PositionValue[t+1][y],
search_string )
02  begin
03     for x:=1 to k do
04        if search_string.type = dictionary then
05           input x into homograph when WordList[x][0] =
search_string;
06        else
07        begin
08           input x into homograph when WordList[x][1] =
search_string;
09           if homograph.count = 0 then input search_string into
homograph
10        end
11     end
12     for z:=1 to homograph.count do
13        for i:=1 to length (string_size) do
14           search o when homograph[z]i = ChangeTable[o];
15           positional_frequency:= positional_frequency +
PositionValue[o][i] / PositionValue[t+1][i];
16        end
17        print (positional_frequency);
18        for i:=1 to length (string_size)-1 do
19           search o when (homograph[z]i = ChangeTable[o]) and
(homograph[z+1]i = ChangeTable[p]);
20           search o when homograph[z]i = ChangeTable[o];
21           search p when homograph[z]i+1 = ChangeTable[p];
22           biphone_frequency:= biphone_frequency +
min (PositionValue[o][i], PositionValue[p][i+1]) /
min (PositionValue[t+1][i], PositionValue[t+1][i+1]);
23        end
24        print (biphone_frequency);
25        for i:=1 to length (string_size)-1 do
26           search o when (homograph[z]i = ChangeTable[o]) and
(homograph[z+1]i = ChangeTable[p]);
27           search o when homograph[z]i = ChangeTable[o];
28           search p when homograph[z]i+1 = ChangeTable[p];
29           search q when homograph[z]i+2 = ChangeTable[q];
30           triphone_frequency:= triphone_frequency +
min (PositionValue[o][i], PositionValue[p][i+1],
PositionValue[q][i+2]) / min (PositionValue[t+1][i],
PositionValue[t+1][i+1], PositionValue[t+1][i+2]);
31        end
32        print (triphone_frequency);
33     end
34  end
```

표 4. 빈도 목록 절차
table 4. Frequency list procedure.

```
input: WordList[x][i], ListValue[x][n]
01  procedure frequency_list (WordList[x][i], ListValue[x][n] )
02  begin
03     for x:=1 to k do
04        print (WordList[x][0], WordList[x][2], WordList[x][3],
WordList[x][1], length (WordList[x][1]),
ListValue[x][0], ListValue[x][1], ListValue[x][2], ListValue[x][3])
05     end
06  end
```

표 5. 근접밀도 목록 절차
table 5. Neighborhood list procedure.

```
input: WordList[x][j], homograph
01  procedure neighborhood_list (WordList[x][j], homograph )
02  begin
03     for i:=1 to homograph.count do
04        for x:=1 to k do
05           if less then 1 different symbol between WordList[x][1] and
homograph then print (WordList[x][1]);
06        end
07     end
08  end
```

마지막으로 표 5는 4개의 알고리즘 중에서 사용자가 입력한 단어의 근접밀도 어휘를 출력하는 부분이다. 알고리즘에서도 나타나듯이 검색절차에서 생성된 동철자어 (homograph)와 사전 등록절차에서 생성한 단어목록을 이용하여 근접밀도어휘를 손쉽게 출력하는 것을 알 수 있다. 사전에 있는 모든 단어에 대해서 근접밀도 어휘목록을 미리 사전 등록 절차에서 생성하는 것도 가능하다. 하지만 사용자가 검색하는 단어는 사전에서 일부분임을 고려해볼 때, 출력되지도 않을 단어들의 근접밀도 어휘목록을 전부 보유하는 것은 메모리 사용에 있어 비효율적이기 때문에 본 프로그램에서는 모든 단어의 근접밀도 어휘목록을 미리 계산하여 메모리에 탑재하는 것을 피하였다. 하지만 메모리에 적재된 사전 데이터만을 가지고도 빠르게 근접밀도 어휘를 출력하기 때문에 수행에 별다른 영향을 미치지는 않는다.

그림 2는 위에서 설명한 알고리즘들의 전체 흐름도를 간략화하여 표현한 것이다. 그림에서 보이듯이 사전 데이터 (dictionary)와 한글받음 컴퓨터 가독형기호 변환표 (Change Table)를 사용하여 우선 사전에 수록된 단어들에 대해 근접밀도 및 음소결합빈도를 수행한 이후 결과를 저장한다. 결과는 저장되어 있으므로 이후 사전 단어에 대한 근접밀도 및 음소결합빈도는 추가적인 계산 없이 저장된 값만 출력할 수 있다. 단, 흐름도에서 알 수 있듯이 사용자가 검색한 단어가 사전에 없는 경우는 추가적인 계산을 해야만 한다. 하지만 이 경우에도 중간 과정에서 계산해야 할 값들에 대해 사전에 수록된 음소들의 통계값을 이용하여 최소한의 계산만을 하도록 구성되어있다.

마지막으로 그림3은 본 논문에서 개발한 프로그램의 화면을 보여준다. 그림을 보면 위에서 서술한 알고리즘의 각 모듈별로 설명을 참조하여 분리가 되어있는 것을 알 수 있다. 그림3에서도 확인할 수 있듯이 사용자가 한글을 입력할 경우 동철자어가 발생하는 경우 모든 동철자어에 대해서 음소결합확률 계산 및 근접밀도어휘 목록이 출력되는 것을 확인할 수 있다.

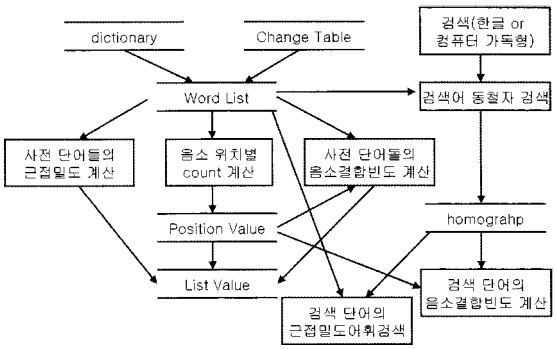

그림 2. 프로그램 흐름도
Fig. 2. Program flow chart.

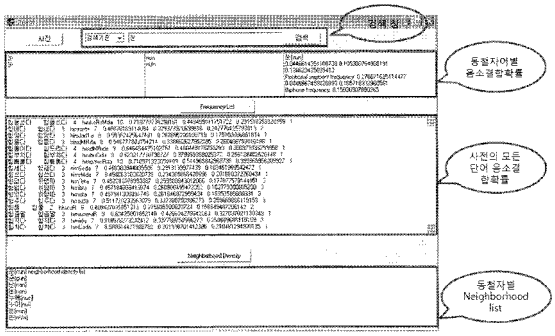

그림 3. 프로그램 실행 결과
Fig. 3. Program demo.

III. 결 론

본 연구는 현대 한국어 말소리의 음소결합확률 계산기 개발에 관한 연구이다. KPPC에서는 한국어의 음소결합확률을 위치별, 두 음소간, 그리고 세음소간 음소결합확률을 계산하였고 근접밀도값을 구하였다. 이 KPPC 엔진은 향후에 언어치료 (Speech-Language Therapy) 및 언어교육 (Language Educaion) 분야, 인문치료 (Humanities Therapy), 청능치료 및 보청기 개발 분야, 음성합성, 음성인식 등 통신공학 분야, 컴퓨터언어학 등 언어처리 분야에서 대규모로 사용될 수 있을 것으로 예상된다. 이 엔진을 적용할 경우 음성인식률의 향상을 기대할 수 있고 언어치료나 언어교육시 효과적인 교육을 기대할 수 있다. [4,5,6,7]

감사의 글

이 논문은 2007년 정부 (교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF-2007-361-AM0056). 본 논문과 관련하여 이메일로 세밀

한 지도와 도움을 주신 University of Kansas의 Storkel 교수님께 감사를 드린다.

참 고 문 헌

1. M. S. Vitevitch and P. A. Luce, "A Web-based interface to calculate phonotactic probability for words and nonwords in English", *Behavior Research Methods, Instruments, and Computers*, vol. 36, no. 3, pp. 481-487, 2004.
2. http://www.bncdnet.ku.edu/cgi-bin/DEEC/post_ppc.vi
3. 이현복, *한국어 표준발음사전*, 서울대학교출판부, 서울, 2003.
4. H. L. Storkel, "Learning new words: Phonotactic probability in language development", *Journal of Speech, Language, and Hearing Research*, vol. 44, no. 6, pp. 1321-1337, 2001.
5. H. L. Storkel, Learning New Words II: Phonotactic Probability in Verb Learning, *Journal of Speech, Language and Hearing Research*, vol. 46, no. 6, pp. 1312-1323, 2003.
6. H. L. Storkel, J. Armbrüster and T. P. Hogan, Differentiating Phonotactic Probability and Neighborhood Density in Adult Word Learing, *Journal of Speech, Language, and Hearing Research*, vol. 49, no. 6, pp. 1175-1192, 2006.
7. M. S. Vitevitch and P. A. Luce, Increases in phonotactic probability facilitate spoken nonword repetition, *Journal of Memory & Language*, vol, 52, pp. 193-204, 2005.

저자 약력

•이 찬 종 (Chan-Jong Lee)

강원대학교 인문과학연구소 HK조교수
한국외국어대학교 언어인지과학과 (언어학 박사)
한국외국어대학교 영어과 (문학사)
※관심분야 : 음성학, 인문치료, 언어치료, 음성진단

•이 현 복 (Hyun-Bok Lee)

런던대학 대학원 음성학과 졸 (1965 : M.A.)
런던대학 대학원 음성·언어학과 졸 (1969 : Ph.D.),
서울대학교 명예교수 (음성·언어학)
※관심분야 : 한국어·영어·프랑스어 음성학

•최 훈 영 (Hun-Young Choi)

강원대학교 컴퓨터공학과 공학사
강원대학교 컴퓨터교육학과 교육학석사
강원대학교 컴퓨터과학과 박사과정
강원대학교 정보통신연구소 연구원
※관심분야 : 데이터 웨어하우스, OLAP, 데이터마이닝, 컴퓨터교육