한국음향학회지 제22권 제6호 pp. 434~442 (2003)

다국어 음성 인식을 위한 자동 어휘모델의 생성에 대한 연구

A Study on the Automatic Lexical Acquisition for Multi-lingustic Speech Recognition

지 원 우*, 윤 춘 덕*, 김 우 성*, 김 석 동*
(Won-Woo Ji*, Chun-Duk Youn*, Woo-Sung Kim*, Suk-Dong Kim*)

*호서대학교 컴퓨터학부

(접수일자: 2002년 5월 10일; 수정일자: 2003년 7월 11일; 채택일자: 2003년 7월 23일)

특정한 언어 (영어)로 구현된 소프트웨어를 다른 언어 (한국어, 중국어 등)에서 처리할 수 있도록 하는 과정인 소프트웨어의 국제화는 음성 기술 분야에 적용할 때 매우 복잡해진다. 그 이유는 음성 자체가 언어와 많은 연관 관계를 갖기 때문이다. 그러나 어떠한 언어라 해도 그 나라의 언어표현은 ASCII코드나 혹은 그 나라 고유의 코드 기반으로 소프트웨어를 처리한다. 영어의 경우는 ASCII코드의 코드체계로 이루어지지만 다른 나라 언어인 경우 다른 형태의 언어코드를 사용하는 것이 일반적이다. 음성 처리에서 언어의 본질적 특성은 어휘모델에 나타난다. 어휘모델은 문자집합, 음소집합, 발음규칙으로 구성된다. 본 논문에서는 다국어 음성인식처리를 위한 어휘모델을 자동으로 생성하기 위하여, 4단계로 나누어 처리하는 어휘모델 구축 방법을 제안한다. 우선 전처리 과정으로 특정한 언어로 표현한 단어를 유니코드로 변환한다.
(1단계) 유니코드로부터 중간 형태 코드로의 변환 (2단계) 발음 형태를 기본으로 하는 표준화된 규칙 적용
(3단계) 음소 규칙들에 의한 문자소 구현 (4단계) 음운론을 적용하는 순서로 구성된다.

핵심용어: 해석기, 음소식별, 어휘모델, 문자집합, 발음규칙
투고분야: 음성처리 분야 (2,6)

Software internationalization, the process of making software easier to localize for specific languages, has deep implications when applied to speech technology, where the goal of the task lies in the very essence of the particular language. A greatdeal of work and fine-tuning has gone into language processing software based on ASCII or a single language, say English, thus making a port to different languages difficult. The inherent identity of a language manifests itself in its lexicon, where its character set, phoneme set, pronunciation rules are revealed. We propose a decomposition of the lexicon building process, into four discrete and sequential steps. For preprocessing to build a lexical model, we translate from specific language code to unicode.
(step 1) Transliterating code points from Unicode. (step 2) Phonetically standardizing rules.
(step 3) Implementing grapheme to phoneme rules. (step 4) Implementing phonological processes.

Keywords: *IPE, PLI, Lexicon, Character set, Phoneme set, Pronunciation rules, Software internationalization*
ASK subject classification: *Speech signal processing (2,6)*

I. 서 론

하나의 언어를 다른 언어 음성시스템에 적용시킬 때 여러 가지 문제점이 발생한다. 대부분의 가정은 단일 언어로 설계된다는 것이며 다른 언어에 적용할 때는 고려하지 않고 있다. 언어에 따라 문자, 음운 그리고 발음 규칙이 서로 다르다. 하나의 시스템을 대상으로 다른 언어로 구현되도록 하는 연구가 몇몇 연구자들에 의하여 수행하였다[1,2]. 이식 가능한 크로스 언어[3]를 시스템에 적용

책임저자: 김석동 (sdkim@office.hoseo.ac.kr)
330-713 충남 아산시 배방면 세출리 산 29-1
호서대학교 컴퓨터학부
(전화: 041-540-5705; 팩스: 041-540-5214)

하여 만족할만한 결과를 얻고 있다. 이러한 접근 방식은 매우 가치있다. 시스템을 보다 빠르게 개발하기 위하여 현존하는 시스템을 활용하면 효율적이고 개발시간도 단축될 것이다.

본 논문에서는 음성 인식에 필요한 다국어 어휘모델을 자동으로 구현하는 방법을 제시한다. 언어와 무관하게 문맥 내에서의 글자를 발음 형태로 표현하는 방법을 택하였으며 대상 언어로 영어와 한국어를 택하였다. 음성 인식 기술에서 언어에 따라 변화되는 요인은 크게 3가지로 나누어 볼 수 있다. 첫째로 언어에 따라 음성 모델의 변화가 있다. 예를 들어 영어의 "beer"를 발음할 때 나오는 모음 "이" 한국어의 "이사"를 발음할 때 나오는 모음 "이"는 음성모델로 달리 표현한다. 둘째로 언어에 따라 언어모델이 변화한다. 보통 언어모델은 연속 음성을 대상으로 인식할 때 후반부에 적용되는 기술로 언어에 따라 단어의 조합이 다르다는 것은 누구나 알 수 있다. 마지막으로 언어에 따라 제일 많이 변화되는 모델로 어휘 모델을 들 수 있다. 어휘모델은 사람에 따라 발음사전이라고도 한다. 예를 들어 "신라의 달밤"을 우리는 "실라의 달빰"으로 발음한다. 음성모델을 적용하여 인식한 결과가 "실라의"로 나왔더라도 우리는 "신라의"로 표현한다. 이렇게 문자로 표현하기 위해 사용하는 모델이 바로 어휘모델로, 이는 언어에 따라 의존도가 매우 높다. 어휘모델은 음성 시스템에서는 한 부분이지만 국제화하기 위해서는 중요한 구성요소로 볼 수 있다. 어휘모델을 작성하는 방법으로 기계 학습[4]을 이용하여 처리하는 방법이 있다. 이 방법은 처리 대상의 자료 양과 질에 의존한다. 본 논문에서는 언어정보를 이용하여 어휘모델을 구축한다. 이를 위해 본 논문에서는 다음과 같은 4단계로 나누어 처리하는 방법을 제시한다.

(1) 유니코드로부터 중간 형태 코드로의 변환
(2) 발음 형태를 기본으로 하는 표준화된 규칙 적용
(3) 음소 규칙들에 의한 문자소 구현
(4) 음운 규칙 적용

유니코드가 이들 4단계를 거치게 되면 그에 대응되는 음성 스트링을 생성해 낸다. 각 단계를 실행하기 위하여 기본 문법을 표현하는 음소식별 (PLI: Phonetic Language Identity)를 만들어 적용시켰으며, 음소식별 해석기인 IPE (International Phoneticizing Engine)를 JAVA로 구현하였다. 적용 결과 영어와 한국어 인식률을 비교하였다.

II. 음소 구성을 위한 4단계

그림 1은 어휘모델을 구축하는 전체적인 방법을 나타내고 있다. 여기서 보인 바와 같이 첫 번째 단계는 유니코드를 중간 코드로 변환하는데서부터 시작해서 총 4 단계로 구성되어 있다. 각 단계마다 처리과정의 규칙을 정의하는 음소식별 섹션을 수반하고 있다. 언어의 음성 표현은 이들 4가지 생성 규칙으로 구성된다[부록 참조].

기계적인 학습에 근거한 접근들[5]과는 달리 본 논문의 목적은 어휘 모델을 자동화하기 위한 방법만이 아니라 다국어 처리 언어기술까지 확장될 수가 있다.

2.1. 유니코드에서 중간코드로의 변환

유니코드 (Unicode)가 일반적으로 공인된 규범의 세계적 문자 코드로 표준화가 되고 있으며 코드 체계가 개방되어 있다. 유니코드는 크기가 고정된 문자 집합이며, 각

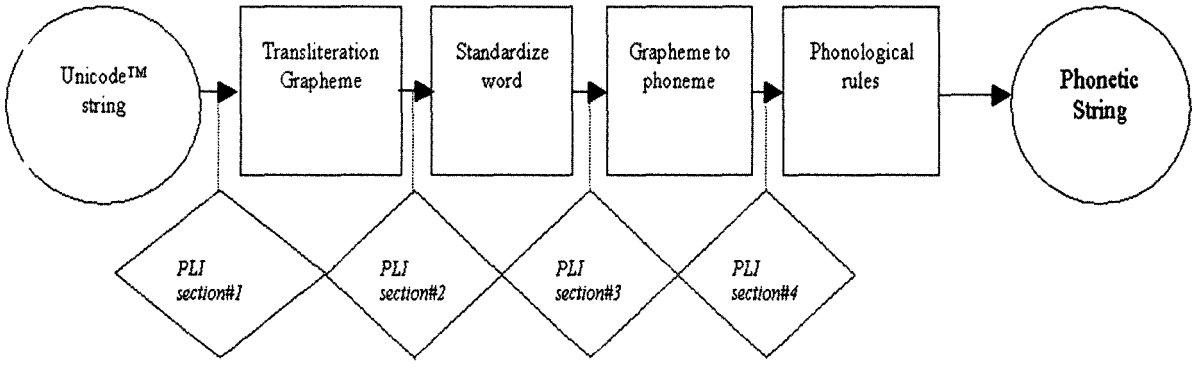

그림 1. 전반적인 구조
Fig. 1. Language independent phoneticization in four step.

엘리먼트는 16비트로 (UCS-2) 암호화되어 있다. 이것은 서로 다른 형태의 언어에서도 균일성을 갖는다. 더 중요한 것은 유니코드 협회[6]는 ASCII에 (예를 들면 양방향 알고리즘으로, 한글 음절 decomposition/composition 알고리즘) 의해서 만들어진 내용을 보완하여 확장시켰으며 음성 기술에 도움이 되는 다수 스크립트들의 처리에 대하여 표준화 집합으로 구성되어졌다. 이러한 이유들 때문에 유니코드는 언어 독립적인 어휘로 구성되어 있다.

유니코드를 읽고 나서 그에 해당하는 ASCII 문자열을 치환한다. 이 처리는 특정 언어에 대하여 유니코드 공간을 규정하여 지정한 유니코드값을 그에 해당하는 ASCII 매핑을 통해 중간 코드 값을 생성한다. 중간 코드로 표현하는 방법으로 프랑스어 모음들은 발음 구별 부호를 기준으로 작성하고, 한글에서는 음절을 기준으로 하여 기본 엘리먼트를 4개의 자모 (즉 다시 말하면, 단음들)로 구성한다. 문자 변환은 기본적으로 문자열에 근거하여 내부 문자로 변환하되 ASCII값으로 표현하여 기존의 ASR (Automatic Speech Recognition) 시스템들을 그대로 적용할 수 있도록 한다.

음성 기술이란 말로 표현하는 단어와 그것에 해당하는 문자사이의 관계성을 다룬다. 하지만 모든 언어들은 음성을 표시하는 스크립트를 가지고 있지 않다. 번역 기술이란 텍스트 엘리먼트 수준에서 하나의 언어를 또다른 언어로 표현되는 문서로 다시 생성하는 기술이다.

2.2. 표준화 처리

간혹 소리와 관계된 정자법은 아주 직관적이지 않다 (즉, 영어의 knight 소리는 nite, 프랑스어의 paon의 소리는 pan, 한국어의 같이는 가치). 또한 동음인 경우에 정자법 표기에 따라 의미론적인 차이가 있다 (즉, 영어인 경우 know, no). 기본적인 형태의 단어를 표준적으로 발음되는 음성을 생성한다. 그러나 불규칙적으로 발음되는 현상을 처리하기 위해 규칙보다는 현상위주로 표현한다. 이 과정은 또한 음성이 부분단어 단위로 처리하여 문맥적으로 독립적인 발음에 밀접한 영향을 고려했다. 그리고 다음 단계 음성 처리 과정에서 가지는 부담을 덜도록 하였다.

2.3. 음소의 변환

세 번째 단계에서는 음운 매핑의 기본적인 문자소의 구현을 의미한다. 모든 문맥 발음 조합은 이 단계를 거치는 동안에 각각의 코드값을 그에 해당하는 위치에 나열시킨다.

2.4. 음운 규칙 적용

어떠한 특별한 언어에서는 정자법에 관계없이 몇몇 발음 규칙들이 소리에 기초하여 발음되는 경우가 있다. 프랑스어로 몇몇 동일한 음운들이 반복될 때 단지 한가지로 발음되는 경우가 있다 (즉, "tourette": T UW R EH T T -> T UW R EH T). 이 단계에서는 또한 이음들 사이에서 의미를 갖는데, 그들의 정확한 음성을 표시한다.

III. 음소식별 (Phonetic Language Identity) 와 해석기 (International Phoneticizing Engine)

3.1. 음소식별 형식

이장에서는 각 단계에서 사용한 규칙을 정의한 음소식별 형식을 다룬다. 음소식별의 일반적인 형태는 다음과 같이 매핑 테이블과 유사하다.

Source_substring [tabulation mark] Target_substring

음소식별 규칙은 유니코드를 음성의 나열로 전환시키기 위해 4단계으로 나누어진다. 각 단계는 키워드 "1", "#2", "#3", 그리고 "#4"로 분리한다. 부록 1과 2에 우리가 사용한 한국어와 영어에 대한 음소식별 내용을 수록하고 있다.

3.1.1. 섹션

1 단계: 유니코드가 2 Byte이므로 이를 중간 코드 형태의 ASCII문자열로 변환한다. 16비트 코드를 참조하여 ASCII 형태로의 변환을 위하여 음소식별 형식을 참조한다. 다음에 1단계에서 사용 예를 나타내었다. 앞부분은 유니코드의 값이며 뒷부분은 그에 해당하는 한글을 중간 코드로 표현하고 있다.

D4AD {pVt}

2 단계: 앞부분은 단어를 구성하는 음절영역이고 뒷부분은 표준화 발음을 나타낸다. 즉 다음의 예에서 음절 영역의 의미는 앞단어의 종성자음이 "ㄴ"으로 끝나고 뒷단어가 모음으로 시작된다. 표준화 발음 부분은 앞단어의 종성 자음이 탈락되고 탈락된 자음이 뒷 단어의 초성자음으

로 옮겨진다.

n{X _}{n

3 단계: 앞부분은 표준화 발음을 나타내고 뒷부분은 그 발음에 해당하는 어휘모델로 표현한다.

t:ı [TH]
i⟨vowel⟩ [AY]⟨vowel-sound⟩

4 단계: 특별한 경우나 예외경우를 표현한다.

[G] [G][G]
[K] [K][S]

3.1.2. The comments

모든 문자열이 키워드 //로 시작하면 그들의 의미는 무시되며 음소식별 편집시 사용된다.

// The following rules take care of
th [TH] // This rule phoneticizes th sound.

3.1.3. 공간표시

그 키워드 "+"은 공간을 표현하기도 하며 혹은 글자 중간에서 음성학 상호작용의 표현으로도 사용된다. 이것은 음소식별이 예외 사전과 규칙 양쪽 모두가 문법 시스템을 사용하기 때문이다.

s+⟨vowel⟩ [Z]+⟨vowel-sound⟩

3.1.4. 널음소

특별한 케이스 음운, #은 인쇄되지 않는다. 이 음운은 발음되지 않은 문자소에 편리하게 사용될 수 있다:

[HH]+ #+

3.1.5. 그룹화 변수

그룹화 변수들을 사용하는 목적은 단위의 클래스의 활동을 시뮬레이션하는 것이다. PLI가 처리될 때, 이 변수들은 그들을 열거한 세트들로 확장된다. 열거된 단위들의 순서는 각각을 열거할 때 그 위치에 의해서 결정된다.

하나의 그룹화 변수를 설정한다:
⟨variable_name⟩ = {unit1 unit2 unit3 unit4 ...}

하나의 그룹화 변수를 다른 그룹 변수를 사용한다:
ph⟨variable_name1⟩ f⟨variable_name2⟩

그룹화 변수의 선언은 사용전에 (표 1 참고) 선행이 필요로 하다. 또한 2개의 그룹화 변수를 동시에 사용할 때 즉 (⟨variable_name1⟩ ⟨ variable_name2⟩)형태는 각각의 그룹에 속한 개별 엘리먼트들은 일대일로 매핑된다. 이 규칙에 대한 예외는 그룹 변수에 속한 실제 변수들이 없는 경우이다. 이런 경우에는 다음과 같이 표현한다.

표 1. 음소식별에 사용한 그룹화 변수
Table 1. Grouping variable and space marker syntax in PLI.

```
//Grouping Variables
<consonants> = {b c d f g h j k l m n p q r s t v w x y z }
//groups the consonants in one variable
<consonant-sound> = {B K D F G HH JH K L M N P K R S T V W X Y Z }
//groups the consonant sounds in one variable
i<consonant>     [IY]<consonant-sound>
//using the grouping variable to simulate the sound of "i"

//Space Marker
s+<vowel>  [Z]+<vowel-sound>
//implements the French inter-word "liaison"
+colonel+  +kernel+ //PLI can be used as an exception dictionary
```

표 2. 해석기의 결과
Table 2. Output of the IPE in trace mod.

```
#TRANSLITERATION FROM UNICODE
  internationalize
#TEXT TO STANDARDIZED TEXT
  internashahnal[AY]z
    477 ize+      ->  [AY]z+
    558 tion      ->  shahn
#TEXT TO PHONEMES
[IH][N][T][ER][N][AH][SH][AH][N][AH][L][AY][Z]
    689 sh        ->  [SH]
    693 er        ->  [ER]
    704 ah        ->  [AH]
    712 l         ->  [L]
    712 n         ->  [N]
    712 t         ->  [T]
    712 z         ->  [Z]
    713 a         ->  [AH]
    713 i         ->  [IH]
#SOUND TO SOUND
[IH][N][T][ER][N][AH][SH][AX][N][AH][L][AY][Z]
    799 [SH][AH][N]       ->   [SH][AX][N]
```

o〈variable_name1〉 o#

혹은

o〈variable_name1〉 o

음소식별 파일을 생성할 때 음절을 기본으로 작성한다. 예를 들면 음절 '한'이 {han}으로 표현하며 괄호를 양쪽 끝에 사용한다. 영어인 경우 음운들의 문맥 독립의 부족으로 인하여 음소식별 섹션 #2에 사용된다. 그룹화 변수의 사용할 때 변수명은 사용자가 임의로 정의한다.

3.2. 해석기

해석기는 앞에서 언급한 음소식별을 참조하여 어휘모델 생성기로서 자바로 구현하였다. 특정 언어로 입력한 유니코드 파일과 음소식별 파일을 이용하여 작성한 응용프로그램이다. 이것은 입력으로 받아드린 내용과 음소식별을 연관시켜 해석한다. 사용자는 출력 형식을 임의로 지정할 수 있다. 즉 4단계로 구성된 모듈에서 각 단계마다 임의로 출력을 선택할 수 있다. 사용자가 해석기가 올바르게 동작하는지 단계마다 추적을 (표 2 참조) 할 수가 있다. 각 단계에 있는 모든 규칙들은 소스 문자열의 길이에 의해서 내림차순으로 정렬시켰다.

IV. 실험 및 결과

음소식별들을 생성하기 위하여 일반적인 문장들로부터 얻은 자료에서 빈번한 단어들을 선택한다. 어휘모델의 대부분의 자료들은 위에서 얻은 자료의 부분 집합이 된다. 인식 및 오인식 단어를 포함하기 위해 인식 대상의 단어 뿐 아니라 일반적인 단어를 포함시키게 되면 좀더 효율적인 어휘모델이 될 수 있다. 한국어 음소식별 생성에 관련한 내용이 다음과 같다.

PLI section #1:

한국어 음절을 표시하는 유니코드 값과 그에 해당하는 우리말 음절을 1:1로 대응시켰다. 총 11,179 개를 사용하지만 매우 단순한 스크립트의 사용으로 빠른 속도로 자동적으로 수행된다.

PLI section #2:

한글은 정규 언어이다. 음절을 이루는 3-자모 (초성자음, 중성모음, 종성자음)가 위치에 따라 발음이 달라진다. 음절의 영향을 고려하여 발음 규칙을 이곳에서 정의하였다. 발음 형태는 음절의 문맥에 따라 규칙적으로 변화하므로 이를 지식구조로 구축하였다. 예를 들어 "삶이"

표 3. 오인식률 비교
Table 3. Error rate using Korean PLI with ASR system.

LM text corpus size: 14358 Unique words in transcript: 310 Dictionary size: 8550 words	Trigram Language Model Perplexity: 6.25 Entropy: 2.64 bits	Bigram Language Model Perplexity: 38.48 Entropy: 8.40 bits	Unigram Language Model Perplexity: 1895.80 Entropy: 10.89 bits
Word Error Rate (%)	8.45	15.67	25.25
Syllable Error Rate (%)*	5.54	9.70	16.61

과 같이 앞 음절의 종성자음이 "ㄹㅁ"으로 끝나고 다음 음절에 초성자음이 없고 바로 모음으로 시작된다면 다음과 같은 음소식별로 표현된다.

M) X -> r)(m)

즉 실제 발음은 앞음절의 종성자음은 "ㄹ"이고 뒷음절의 초성자음은 "ㅁ"으로 이루어진다는 의미가 된다.

PLI section #3:

자모의 위치에 따라 3가지가 가능한 발음을 나타낸다.

PLI section #4:

예외의 경우, 추가, 삭제등과 같이 특수한 경우를 쉽게 나타낸다.

PLI 섹션 #1은 기본적으로 확장성이 용이하다. 즉 코드 체계의 변화에 쉽게 적응될 수 있도록 구성하였다. 한글에 대한 조합형이나 완성형이나 관계없이 임의의 형태들을 하나로 통합해서 표현한다. 섹션 #1이 유니코드의 코드 공간을 규정하기 때문에 유연성이 좋다. 만약 기본 자료형을 ASCII 코드로 표현해도 이 부분에서 아스키 코드에 해당하는 값만 변화시켜 쉽게 구성할 수 있다. PLI 섹션 #2는 많은 목적이 있다, 그중에서 가장 중요한 것은 언어의 관계성을 표현하기 위하여 소리와 묘사성의 모호성들을 제거하는 것이다. 소리의 관계성을 표현하기 위하여 고든 경우를 생각하여 음운들을 뽑아내는 것이 가장 힘든 작업이다. 이 섹션에 적용되는 규칙들이 많기 때문에 여러 개의 규칙들이 충돌하는 현상이 발생했다. PLI 섹션 #3 은 음운 혹은 음운의 순서를 표현한다. 모든 언어는 PLI 섹션 #2후에 언어의 스크립트가 표준화에 맞춰지는 것이 가정에 바탕을 두기 때문에 이 섹션은 가장 단순한 섹션이다. PLI 섹션 #4의 음소의 예외 현상을 (강세, 비음화 등) 다룬다.

4.1. 음성인식 시스템

음성인식 시스템에 사용한 음성모델은 일반적인 HMM 모델을 사용하였고, 언어모델은 CMU-Cambridge 언어모델 툴 킷[7]을 사용했다. CMU-Cambridge 언어모델은 유니코드로 되어 있지 않아 PLI 섹션 #1에 적용하기 전에 전처리 과정을 하였다. 전처리 과정에 사용한 데이터는 중간 코드로 ASCII코드를 기반으로 작성하였다. 언어 모델과 음성 데이터의 수집은 인터넷을 통해 신문이나 잡지 온라인 싸이트를 이용하였다. 텍스트는 KSC 완성형을 사용했고 유니코드 (UCS-2)로 변환되었다.

음성 모델을 작성하기 위한 훈련 데이터는 162명 (여자 70명, 남자 92명)이 1시간에서 3시간 발음을 사용하였다. 실험에 사용한 어휘모델은 우리가 제안한 방법인 IPE에 의해서 작성하였다. 음성모델들의 훈련과 인식 양쪽 모두 IPE로 작성한 어휘모델을 사용하였으며 인식 실험에 사용한 음성은 훈련에 참여 하지 않았다.

표 3에서 보는 바와 같이 언어모델을 3가지로 나누어 인식실험을 하였다. 물론 인식률은 언어모델의 구성에 따라 다름을 알 수 있으며 단어 인식률이 세음절 구조 (Trigram)인 경우 약 92% 정도 비교적 높은 인식률을 얻었으며 이에 따라 우리가 제안한 방법에 따라 작성한 어휘모델이 잘 적용된다는 것을 보여주고 있다.

4.2. 영어와의 비교

ASR 시스템들의 서로 다른 언어를 비교하는 것은 그 인수들이 무수하기 때문에 단순비교는 어렵다. 그러나 우리가 제안한 어휘모델의 견고성을 시험하기 위해서 두 언어를 비교해 보았다. 그림 2에서 볼 수 있듯이 단어인식률이 영어가 전체적으로 다소 높다. 인식 형태는 언어모델의 복잡도 (Perplexity)에 따라 유사하다. 영어로 구성한 언어 모델이 100 이하에서는 한국어 모델보다 오인식률이 낮게 나타났으며, 100 이상에서는 한국어 언어 모델이 약간 적게 감소하는 것으로 나타났다. 어휘 모델에서

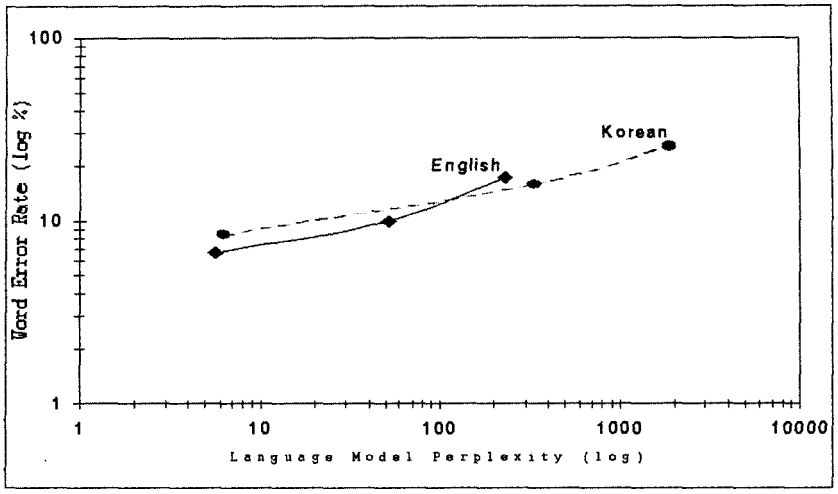

그림 2. 언어 모델의 복잡도에 따른 음성 인식률의 비교
Fig. 2. Affect of the language model perplexity on the word Error rate of Korean and English ASR system.

한국어 단어들이 영어 단어보다 더 발음대가 길게 나타났다. 실험에 참여한 한국어 어휘모델에서는 단어당 음절이 평균 8.31로 관찰되었으나 영어 어휘모델에서는 단어당 6.24로 관찰되었다.

V. 결 론

음성학적으로 언어독립으로 처리하여 어휘모델을 생성하였고 이를 실현하기 위하여 문법을 확립했다. 모든 언어에 사용될 유니코드로 다국어의 모든 관련한 정보를 프로세스 내에 저장하여 음성처리를 국제화 음성처리 기술로 사용할 방법을 보였다. 충돌을 피하기 위한 음성 규칙을 3단계 나누어 분석하여 생성하였다. 음성 시스템의 서로 다른 언어 처리의 비용을 줄였으며, 제안한 방식대로 구현한 어휘모델을 영어와 한국어에 적용하여 우수한 음성 인식률을 얻을 수가 있었다.

참 고 문 헌

1. L.-S. Lee, C.-Y. Tseng, H.-Y. Gu, F.-H. Liu, C.-H. Chang, S.-H. Hsieh, and C.-H. Chen, "A *real-time* mandarin dictation machine for chinese language with unilimited texts and very large vocabulary," *ICASSP '90*, 65-68, 1990.
2. T. Matsuoka, K. Ohtsuki, T. Mori, S. Furui, and K. Shirai, "Large-vocabulary continuous- speech recognition using a japanese business newspaper (NIKKEI)," *Proc. Of the ARPA Workshop on Spoken Language Technology*, Austin TX, Morgan Kaufmann, Cohen, Ed., 1996.
3. L. Deng, "Integrated-multilingual speech recognition using universal features in a functional speech production model," *ICASSP '97*, 1007-1010, 1997.
4. T. J. Sejnowski, and C. R. Rosenberg, "Netltalk: a parallel network that learns to read aloud", The Johns Hopkins University Electrical Engineering and Computer Science Technical Report JHU/EECS-86/01, 1986.
5. R. I. Damper, "Self-learning and connectionist approaches to text-to-phoneme conversion," Connectionst Models of Memory and Language, Levy J., Bairaktaris J., Bullinaria J., and Cairns P. (eds.), UCL Press, London, 117-144, 1995.
6. The Unicode Consortium, "The unicode standard, version 2.0," Addison-Wesley Publishing Company, 1996.
7. P. Clark, and R. Rosenfeld, "Statistical language modeling using the CMU-cambridge toolkit," *EUROSPEECH '97*, 2707-2710, 1997.

저자 약력

● 지 원 우 (Won-Woo Ji)
1987년 2월: 호서대학교 전자계산학과 학사 졸업
1983년 2월: Texas A&M 대학 석사 과정 졸업
2001년 3월~현재: 호서대학교 컴퓨터학부 박사과정
※ 주관심분야: 음성 인식

● 윤 춘 덕 (Chun-Duk Youn)
1989년 2월: 서울 산업대학교 전자계산학과 학사 졸업
1992년 2월: 동국대학교 석사 과정 졸업
2003년 2월: 호서대학교 공학박사
1992년 3월~현재: 호서 전산 전문대학 교수
※ 주관심분야: 음성인식

● 김 우 성 (Woo-Sung Kim)
1980년 2월: 서강대학교 전자공학과 학사 졸업
1983년 2월: Texas A&M 대학 석사 과정 졸업
1993년 2월: 서강대학교 공학박사
1987년 3월~현재: 호서대학교 컴퓨터학부 교수
※ 주관심분야: 영상 신호 처리

● 김 석 동 (Suk-Dong Kim)
1982년 2월: 아주대학교 전자공학과 학사 졸업
1984년 2월: 아주대학교 석사 과정 졸업
1993년 2월: 아주대학교 공학박사
1986년 3월~현재: 호서대학교 컴퓨터학부 교수
※ 주관심분야: 음성인식

```
//************************** Korean PLI ****************************
//      Some Grouping Variables declarations
//      The List of all "Jamo"s
<jamo> = { a A c j e E U h i N k g x m n l L o O p b q r H G M z B Z D s C K P
S T t d u w W v V J y Y f F I Q R X _ }

//      The List of the phonemes of these JAMOS
//      when in initial position of the syllable
<initial-sound> = { [AA] [AE] [CH] [JH] [EH] [AO] [EU] [HH] [IY] # [K] [G] [K] [M]
[N] [N] [N] [OW] [W][EH] [P] [B] [P] [R] [*] [*] [*] [*] [*] [*] [*] [S] [JJ] [KK] [PP]
[SS] [TT] [T] [D] [UW] [W][AA] [W][AE] [W][EH] [W][AO] [W][IY] [Y][AA] [Y][AE]
[Y][EH] [Y][AO] [EU][IY] [Y][OW] [Y][UW] # # }

//      The List of the phonemes of these JAMOS
//      when in medial position of the syllable
<medial-sound> = { [AA] [AE] [CH] [JH] [EH] [AO] [EU] [HH] [IY] # [K] [G] [K] [M]
[N] [N] [N] [OW] [W][EH] [P] [B] [P] [R] [*] [*] [*] [*] [*] [*] [*] [SS] [JJ] [KK] [PP]
[SS] [TT] [T] [D] [UW] [W][AA] [W][AE] [W][EH] [W][AO] [W][IY] [Y][AA] [Y][AE]
[Y][EH] [Y][AO] [EU][IY] [Y][OW] [Y][UW] # # }

//      The List of the phonemes of these JAMOS
//      when in final position of the syllable
<final-sound> = { [AA] [AE] [TC] [TC] [EH] [AO] [EU] [TC] [IY] [NG] [KC] [KC] [S]
[M] [N] [CH] [HH] [OW] [W][EH] [PC] [PC] [S] [L] [*] [*] [*] [*] [*] [*] [*] [TC] #
[KC] # [TC] # [TC] [TC] [UW] [W][AA] [W][AE] [W][EH] [W][AO] [W][IY] [Y][AA]
[Y][AE] [Y][EH] [Y][AO] [EU][IY] [Y][OW] [Y][UW] # # }

#1
//      Transliterating all the Hangul Syllable Codepoints (AC00-D7A3)
[...]
CA03 {CAB} //
CA05 {CAD} //
CA01 {CAG} //
CA07 {CAH} //
[...]

#2
//      Standardizes Hangul in a phonetically intuitive pseudolanguage
g}{X  _}{g
n}{X  _}{n
M}{X r}{m
[...]
#3
//      The Basic Phonetic Rules in Korean are the Jamo
//      position rules expressed below
(<jamo><jamo><jamo>)        <initial-sound><medial-sound><final-sound>

#4
//      Sound to Sound rules
[AA][NG]      [AN] // Vowel nasalization
[G][Y][EH]    [G][EH]
[...]
```

부록1: 한국어 PLI의 예

```
//**********************English PLI*********************
<vowel> = { a e i o u }
<vowel-sound> = { [AH] [EH] [IH] [AO] [UH] }
<consonant> = { b c d f g h j k l m n p q r s t v w x y z }
<consonant-sound> = { [B] [K] [D] [F] [G] [HH] [JH] [K] [L] [M] [N] [P]
[K] [R] [S] [T] [V] [W] [K][S] [Y] [Z] }

#1
0041 a  //A
0042 b  //B
0043 c  //C
[...]
0061 a  //a
0062 b  //b
0063 c  //c

#2
+<consonant>ay+    +<consonant>[EY]+
+<consonant>ays+   +<consonant>[EY][Z]+
<vowel>ry+  <vowel>ree+
<consonant>ay+     <consonant>ey+
<consonant>ays+    <consonant>eyz+
+chair       +[CH][EH][R]
ause+        [AO][Z]+
+through+    +throo+
+how+        +h[AW]+
+off+        +[AO]f+
+four+       +faur+
+many+       +[M][EH][N][IY]+
+being       +bee[IH][N][G]
+into+       +intoo+
[...]

#3
//   Many English phoneme to sound rules present.
<vowel>      <vowel-sound>
<consonant> <consonant-sound>
aw    [AW]
edd   [EH][D]
[...]

#4
//   Deals with english Phoneme-to-Phoneme interaction
[EH][EY]     [IY]
[S][Z]       [S][EH][Z]
[L][F]+      [F]+
[...]
```

부록 2: 영어 PLI 의 예