論文99-36C-3-6

연속음성인식 후처리를 위한 음절 복원 rule-based 시스템과 형태소분석기법의 적용

(The syllable recovery rule-based system and the application of a morphological analysis method for the post-processing of a continuous speech recognition)

朴 美 星 *, 金 美 辰 *, 金 桂 成 *, 崔 宰 赫 **, 李 相 祖 *

(Mi Sung Park, Mi Jin Kim, Kye Sung Kim, Jae Hyuk Choi, and Sang Jo Lee)

요 약

한국어를 연속적으로 발음할 때 여러 가지 음운변동이 일어난다. 이러한 음운변동은 한국어 연속 음성 인식을 어렵게 하는 주요 요인 중의 한가지이다. 본 논문에서는 음운변동이 반영된 음성 인식 문자열을 규칙에 의하여 text 기반 문자열로 다시 복원시키는 rule-based 시스템을 제안한다. 그리고 복원 결과들은 형태소 분석되어 올바른 문자열만 생성된다. 복원은 4가지 rule 즉, 음절 경계 종성 초성 복원 rule, 모음처리 복원 rule, 끝음절 종성 복원 rule, 한 음절 처리 rule에 의거하여 이루어진다. 규칙 적용 과정 중에 효과적인 복원을 위해 x-clustering정보를 정의하여 사용하고, 형태소 분석기에 입력될 복원 후보수를 제한하기 위해 postfix음절 빈도정보를 구하여 사용한다. 본 시스템은 규칙기반 시스템이므로 대용량의 발음열 사전이나 음소열 사전을 필요로 하지 않고 문서 기반 형태소 분석기를 그대로 이용할 수 있다는 이점이 있다.

Abstract

Various phonological alteration occurs when we pronounce continuously in korean. This phonological alteration is one of the major reasons which make the speech recognition of korean difficult. This paper presents a rule-based system which converts a speech recognition character string to a text-based character string. The recovery results are morphologically analyzed and only a correct text string is generated. Recovery is executed according to four kinds of rules, i.e., a syllable boundary final-consonant initial-consonant recovery rule, a vowel-process recovery rule, a last syllable final-consonant recovery rule and a monosyllable process rule. We use a x-clustering information for an efficient recovery and use a postfix-syllable frequency information for restricting recovery candidates to enter morphological analyzer. Because this system is a rule-based system, it doesn't necessitate a large pronouncing dictionary or a phoneme dictionary and the advantage of this system is that we can use the being text based morphological analyzer.

* 正會員, 慶北大學校 컴퓨터工學科

(Dept. of Computer Engineering, Kyungpook National University)

** 正會員, 新羅大學校 컴퓨터教育科

(Dept. of Computer Education, Sila University)

接受日字:1998年9月18日, 수정완료일:1999年3月2日

I. 서 론

현재 자연언어 처리의 응용 분야 중 음성 인식 기술의 발달과 함께 음성을 처리 대상으로 하는 자연언어 인터페이스 시스템에 많은 관심이 모아지고 있다. 그 대표적인 시스템으로 사용자와의 대화를 통한 데이터베이스 정보 검색 시스템, 지능적 교수 시스템, 관광

안내 및 열차 예매와 같은 업무 수행을 위한 대화 시스템이 있다. 그리고 음성을 처리하는 시스템으로, 음성 자체로서의 의미보다는 음성이 문자의 형태로 변환되어야 의미를 가지는 경우의 응용 분야도 많다. 그 예로, 음성인식 시스템이 결합된 자동 전화 통역 시스템, 음성 번역 시스템, 국제 전화 교환 시스템, 우체국 전보 시스템, 받아 쓰기 시스템 등이다[1,10]. 이와 같은 발화되는 음성을 처리 대상으로 하는 시스템들은 지금까지의 고립 단어 인식 위주의 신호 처리 단계만으로는 처리가 불가능하고, 연속되는 음성을 인식할 수 있어야 한다.

그러나 발화되는 연속 음성은 텍스트 기반의 문장과는 달리 어절간의 경계가 모호하고, 발화되는 단어들이 연속적으로 발음되기 때문에 각각의 음소가 주변 음소에 영향을 끼치는 음운적 변화가 많아 음성 인식을 어렵게 하는 요인이 된다. 이러한 요인 중 한국어를 연속적으로 발음할 때 나타나는 이러한 음운적 변화를 한국어 음운변동이라 한다[2,3,4,5,6].

그러므로 연속된 한국어 음성을 인식하기 위해서는 한국어의 음운변동에 대한 처리가 우선 이루어져야 한다. 그리고 이 결과가 형태소분석, 구문분석, 의미분석까지 가능하게 되면 음성언어를 처리하는 여러 응용 시스템의 성능 향상에 많은 도움을 줄 수 있다[7,8,9].

음성 언어 처리를 위해 우선 처리되어야 하는 음운변동 처리에 관한 기존 연구로는 읽기 규칙을 역으로 적용한 방식[10]이 있다. 이 방식은 한국어를 발음할 때 발생하는 음운 규칙을 역으로 적용하여 복원을 수행한다. 복원을 위해 조사어미 사전, 음절사전, 선어말 어미 사전, 예외사전을 이용하였다. 이 방식은 규칙의 적용 순서가 복잡하여 알고리즘이 복잡하며, 예외 사전이 필요하다는 단점이 있다. 또 다른 연구로는 음성인식의 결과로 나온 음소의 열로부터 원래 음절로 복원하기 위해 음운변동을 고려한 음소열 사전을 이용한 방식[11], 발음열 사전을 이용한 방식[8,12]이 있다. 이 방식들은 인식결과로 나온 음소열을 표제어로 하여 그 음소열이 발음될 수 있는 모든 단어를 사전에 수록하거나 발음 규칙을 기반으로 발음열 사전을 만들어 사용하였다. 그러므로 정해져 있는 도메인을 확장할 경우 사전을 구축하는 시간이 많이 걸리고 사전량이 매우 커지는 단점이 있다.

또 다른 연구로는 자소 단위 사전을 이용하여 형태소 단계에서 음운변동을 처리한 방식[13]이 있다. 이 방식은 일부 음운 변동을 규칙으로 정의하여 처리하였으나 사잇소리, 경음화, 끝음절 대표음 처리와 같은 음운 변동은 처리하지 못했고, 자소 단위 사전 검색을 통해 형태소분석 단계에서 복원 후보를 결정하는 방식이기 때문에 사전 검색 횟수가 많다는 단점이 있다.

본 논문에서는 한국어가 연속적으로 발음될 때 일어나는 음운변동이 반영된 음성 문자열을 규칙에 의하여 text 기반 문자열로 다시 복원시키고, 복원 결과 후보를 형태소분석하여 올바른 문자열만을 생성하게 하는 시스템을 구현하였다. 본 논문에서 제안한 방법은 기존의 방법들과는 달리 음소변이 규칙 기반 시스템이므로 대량의 발음열사전이나 음소열사전이 필요없고, 읽기 규칙을 여러 단계로 역적용할 필요도 없다. 또한 이 시스템은 기존의 문서 기반 형태소 분석기를 그대로 이용할 수 있다는 이점이 있다.

Ⅱ. 음절복원 rule-based 시스템

한국어 음성인식 결과를 자연어처리 기법과 접목시키는 데에는 몇 가지 어려운 점이 있다[8]. 첫째는 말하는 단위와 문서의 띄어 쓰기 단위가 불일치한다는 것이다. 둘째는 각 형태소들이 형태소 내부에서 뿐만 아니라 형태소와 형태소 사이에서 음운변동이 일어난다는 것이다. 본 논문은 위의 두 번째 문제점을 해결하기 위한 것으로, 이를 위해 음절 복원 rule-based 시스템을 제안한다.

2.1 전체 시스템 구성도

입력된 어절은 앞으로 설명하는 각종 규칙에 따라 복원된 후에 형태소 분석기로 넘겨져 올바른 문자열만 생성되도록 하였다. 전체 시스템 구성도는 그림 1과 같다.

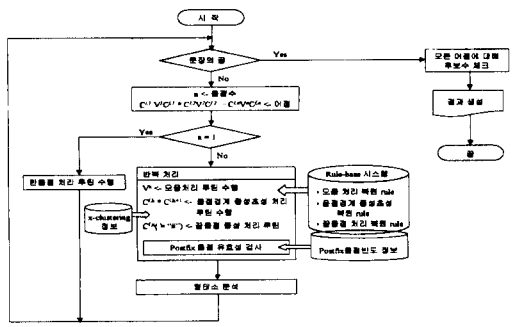


그림 1. 시스템 구성도
Fig. 1. Flowchart of the system.

2.2 음절복원 rule-based 시스템 정의

음성문자열은 여러 어절로 이루어져 있고, 각 어절은 하나 이상의 음절로 구성되는데, 본 논문에서는 한 음절을 C^iVC^f(C^i:초성자음, V:중성모음, C^f:종성자음)로 표현하고, 어절은 하나 이상의 음절로 이루어지므로 다음과 같이 표현한다.

$$C^{i,1}V^1C^{f,1} * C^{i,2}V^2C^{f,2} * \cdots * C^{i,k}V^kC^{f,k} * \cdots * C^{i,n}V^nC^{f,n}$$

여기서 i는 초성, f는 종성을 의미하며, 1,2,k,n은 한 어절 내의 각 음절 위치 즉 첫 번째, 두 번째, k 번째, n 번째 음절을 의미한다. 그러므로 $C^{i,1}$는 첫음절 초성을, V^1는 첫음절 중성을, $C^{f,1}$는 첫음절 종성을 가리킨다. 앞으로 설명될 각종 규칙의 정의 내에서 사용된 부호의 의미를 간단히 살펴보면, { }는 음소의 집합을 의미하는데 "->"의 오른쪽에 정의된 $C^{f,k}${ , , , } 와 $C^{i,k+1}${ , , , }속에 나열된 음소의 개수는 양쪽이 같고, 위치상으로는 $C^{f,k}${ , , , }의 첫 번째 음소와 $C^{i,k+1}${ , , , }의 첫 번째 음소가 복원의 쌍이 됨을 나타낸다. "#"(fill-code)은 종성이 없음을 의미하며 ","는 나열을, "*"는 음절간의 연결을 의미한다. 그리고 rule내의 음소에 위첨자로 "+"가 붙은 경우는 한글 자모 발음에만 적용되는 rule을 의미한다. 각 rule에서 "->"의 왼쪽은 입력을 오른쪽은 출력 즉 복원부를 의미한다. 아래 [Rule001] 의 경우, 입력에 해당하는 $C^{f,k}$(#)*$C^{i,k+1}${ㄱ}은 한 어절 내의 k번째 음절의 종성이 무종성, k+1번째 음절의 초성이 "ㄱ"으로 입력되었음을 의미한다. 출력에 해당하는 $C^{f,k}$(ㄱ,ㄲ,ㄺ,ㅋ}*$C^{i,k+1}${ㅇ,ㅇ,ㅇ,ㅇ}는 입력에 대해 4개의 sub path {ㄱ}과 {ㅇ}, {ㄲ}과 {ㅇ}, {ㄺ}과 {ㅇ}, {ㅋ}과 {ㅇ} 중 전부 또는 일부 path에 의해 복원될 수 있음을 의미한다. 여기서 sub path란 한 rule에서 "->" 의 오른쪽에 정의된 음소 쌍의 개수로 [Rule001] 은 4개, [Rule002] 는 3개의 sub path를 가진다고 한다. 그리고 모든 rule의 정의에서 자신으로 변환되는 path는 생략했다. 예를 들어 "가고"라는 음성문자열을 복원할 경우 앞음절 "가" 의 종성이 무종성이고, 다음 음절 "고"의 초성이 "ㄱ"이므로 [Rule001] 에 의해 복원될 수 있다. [Rule001] 은 자기 자신 즉 "가고"와 4가지 sub path가 있지만 실제로는 2.3에 설명할 x-clustering 정보 참조로 가능한 2가지 sub path {ㄱ}과 {ㅇ}, {ㄺ}과 {ㅇ}에 의해서만 복원이 이루어진다. 이와 같은 방법으로 rule들이 적용된다. 본 논문에서 정의한 음절 복원 rule은 크게 4가지 종류, 즉, 음절 경계 종성 초성 복원 rule, 끝음절 종성 복원 rule, 모음처리 복원 rule, 한 음절 처리 rule로 나눈다. 이 rule들은 한글 표준 발음법[3]에 근거하여 한글 자모 철자 "기저형"을 주변 환경의 영향으로 변화되어 나타나는 음가 즉 "표면형"과 비교 분석하여, 어떻게 하면 변화되어 나타난 표면형이 다시 기저형으로 복원될 수 있을지를 규칙으로 정의한 것이다.

(1) 음절 경계 종성 초성 복원 rule

앞 음절 종성과 다음 음절의 초성 사이에 적용될 수 있는 rule로 음성문자열에서 종성으로 나타날 수 있는 것은 무종성(fill-code 또는 #)과 7개의 대표음 'ㄱ,ㄴ,ㄷ,ㄹ,ㅁ,ㅂ,ㅇ'이므로 이를 기준으로 rule을 정의한다. 여기에 해당하는 규칙 수는 85개의 main rule과 227개의 sub path이다.

1. fill-code(무종성)에 관한 복원rule

19개의 main rule과 55개의 sub path가 있다.

[Rule001] $C^{f,k}$(#)*$C^{i,k+1}${ㄱ}->$C^{f,k}${ㄱ,ㄲ,ㄺ,ㅋ$^+$} *$C^{i,k+1}${ㅇ,ㅇ,ㅇ,ㅇ$^+$}

[Rule002] $C^{f,k}$(#)*$C^{i,k+1}${ㄲ}->$C^{f,k}${#,ㄱ,ㄲ,ㅅ} *$C^{i,k+1}${ㄱ,ㄱ,ㅇ,ㄱ}

[Rule003] $C^{f,k}$(#)*$C^{i,k+1}${ㄴ}->$C^{f,k}${ㄴ,ㄶ}*$C^{i,k+1}${ㅇ,ㅇ}

[Rule004] $C^{f,k}$(#)*$C^{i,k+1}${ㄷ}->$C^{f,k}${ㄷ,ㅅ,ㅈ,ㅊ,ㅌ} *$C^{i,k+1}${ㅇ,ㅇ,ㅇ,ㅇ,ㅇ}

[Rule005] $C^{f,k}$(#)*$C^{i,k+1}${ㄸ}->$C^{f,k}${ㅅ}*$C^{i,k+1}${ㄷ}

[Rule006] $C^{f,k}$(#)*$C^{i,k+1}${ㄹ}->$C^{f,k}${ㄹ,ㄾ}*$C^{i,k+1}${ㅇ,ㅇ}

[Rule007] $C^{f,k}$(#)*$C^{i,k+1}${ㅁ}->$C^{f,k}${ㅁ}*$C^{i,k+1}${ㅇ}

[Rule008] $C^{f,k}$(#)*$C^{i,k+1}${ㅂ}->$C^{f,k}${ㅂ,ㅄ,ㅍ$^+$}*$C^{i,k+1}${ㅇ,ㅇ,ㅇ$^+$}

[Rule009] $C^{f,k}$(#)*$C^{i,k+1}${ㅃ}->$C^{f,k}${ㅅ}*$C^{i,k+1}${ㅂ}

[Rule010] $C^{f,k}$(#)*$C^{i,k+1}${ㅅ}->$C^{f,k}${ㄷ$^+$,ㅅ,ㅈ$^+$,ㅊ$^+$,ㅌ$^+$,ㅎ$^+$} *$C^{i,k+1}${ㅇ$^+$,ㅇ,ㅇ$^+$,ㅇ$^+$,ㅇ$^+$,ㅇ$^+$}

[Rule011] $C^{f,k}$(#)*$C^{i,k+1}${ㅆ}->$C^{f,k}${ㅅ,ㅆ,ㅎ}*$C^{i,k+1}${ㅅ,ㅇ,ㅅ}

[Rule012] $C^{f,k}$(#)*$C^{i,k+1}${ㅇ}->$C^{f,k}${ㅇ,ㅎ}*$C^{i,k+1}${ㅇ,ㅇ}

[Rule013] $C^{f,k}$(#)*$C^{i,k+1}${ㅈ}->$C^{f,k}${ㄷ,ㅈ}*$C^{i,k+1}${ㅇ,ㅇ}

[Rule014] $C^{f,k}$(#)*$C^{i,k+1}${ㅉ}->$C^{f,k}${ㅅ} * $C^{i,k+1}${ㅈ}

[Rule015] $C^{f,k}$(#)*$C^{i,k+1}${ㅊ}->$C^{f,k}${ㄷ,ㅈ,ㅊ,ㅌ,ㅎ} *$C^{i,k+1}${ㅎ,ㅎ,ㅇ,ㅇ,ㅈ}

[Rule016] $C^{f,k}$(#)*$C^{i,k+1}${ㅋ}->$C^{f,k}${ㄱ,ㅋ,ㅎ}*$C^{i,k+1}${ㅎ,ㅇ,ㄱ}

[Rule017] $C^{f,k}$(#)*$C^{i,k+1}${ㅌ}->$C^{f,k}${ㄷ,ㅅ,ㅈ,ㅊ,ㅌ,ㅌ,ㅎ} *$C^{i,k+1}${ㅎ,ㅎ,ㅎ,ㅎ,ㅇ,ㅎ,ㄷ}

[Rule018] $C^{f,k}$(#)*$C^{i,k+1}${ㅍ}->$C^{f,k}${ㅂ,ㅍ}*$C^{i,k+1}${ㅎ,ㅇ}

[Rule019] $C^{f,k}$(#)*$C^{i,k+1}${ㅎ}->$C^{f,k}${ㅎ}*$C^{i,k+1}${ㅇ}

2. ㄱ-종성에 관한 복원rule

5개의 main rule과 15개의 sub path가 있다.

[Rule020] $C^{f,k}$(ㄱ}*$C^{i,k+1}${ㄲ}->$C^{f,k}${ㄱ,ㄲ,ㄺ,ㅋ}

*C^{i,k+1}{ㄱ,ㄱ,ㄱ,ㄱ}

[Rule021] C^{f,k}{ㄱ}*C^{i,k+1}{ㄸ}->C^{f,k}{ㄱ,ㄲ,ㄱㄹ}
*C^{i,k+1}{ㄷ,ㄷ,ㄷ,ㄷ}

[Rule022] C^{f,k}{ㄱ}*C^{i,k+1}{ㅃ}->C^{f,k}{ㄱ,ㄱㅅㄹ}*C^{i,k+1}{ㅂ,ㅂ,ㅂ}

[Rule023] C^{f,k}{ㄱ}*C^{i,k+1}{ㅆ}->C^{f,k}{ㄱ,ㄱㅅ}*C^{i,k+1}{ㅅ,ㅇ}

[Rule024] C^{f,k}{ㄱ}*C^{i,k+1}{ㅉ}->C^{f,k}{ㄱ,ㄹㅋ}*C^{i,k+1}{ㅈ,ㅈ}

3. ㄴ-종성에 관한 복원rule

12개의 main rule과 29개의 sub path가 있다.

[Rule025] C^{f,k}{ㄴ}*C^{i,k+1}{ㄱ}->C^{f,k}{ㄴ}*C^{i,k+1}{ㄱ}

[Rule026] C^{f,k}{ㄴ}*C^{i,k+1}{ㄲ}->C^{f,k}{ㄴ,ㄴㅈ}*C^{i,k+1}{ㄱ,ㄱ}

[Rule027] C^{f,k}{ㄴ}*C^{i,k+1}{ㄴ}->
C^{f,k}{ㄴ,ㄴ,ㄴㅎ,ㄷ,ㅅ,ㅅ,ㅆ,ㅈ,ㅊ,ㅊ,ㅌ,ㅌ,ㅎ}
*C^{i,k+1}{ㄹ,ㅇ,ㄴ,ㄴ,ㄴ,ㅇ,ㄴ,ㄴ,ㄴ,ㅇ,ㄴ,ㅇ,ㄴ}

[Rule028] C^{f,k}{ㄴ}*C^{i,k+1}{ㄸ}->C^{f,k}{ㄴ,ㄴㅈ}*C^{i,k+1}{ㄷ,ㄷ}

[Rule029] C^{f,k}{ㄴ}*C^{i,k+1}{ㅁ}->C^{f,k}{ㅅ,ㅈ,ㅊ}*C^{i,k+1}{ㅁ,ㅁ,ㅁ}

[Rule030] C^{f,k}{ㄴ}*C^{i,k+1}{ㅃ}->C^{f,k}{ㄴ}*C^{i,k+1}{ㅂ}

[Rule031] C^{f,k}{ㄴ}*C^{i,k+1}{ㅆ}->C^{f,k}{ㄴ,ㄴㅎ}*C^{i,k+1}{ㅅ,ㅅ}

[Rule032] C^{f,k}{ㄴ}*C^{i,k+1}{ㅈ}->C^{f,k}{ㄴㅈ}*C^{i,k+1}{ㅇ}

[Rule033] C^{f,k}{ㄴ}*C^{i,k+1}{ㅉ}->C^{f,k}{ㄴ}*C^{i,k+1}{ㅈ}

[Rule034] C^{f,k}{ㄴ}*C^{i,k+1}{ㅊ}->C^{f,k}{ㄴㅈ}*C^{i,k+1}{ㅎ}

[Rule035] C^{f,k}{ㄴ}*C^{i,k+1}{ㅋ}->C^{f,k}{ㄴㅎ}*C^{i,k+1}{ㄱ}

[Rule036] C^{f,k}{ㄴ}*C^{i,k+1}{ㅌ}->C^{f,k}{ㄴㅎ}*C^{i,k+1}{ㄷ}

4. ㄷ-종성에 관한 복원rule

9개의 main rule과 34개의 sub path가 있다.

[Rule037] C^{f,k}{ㄷ}*C^{i,k+1}{ㄲ}->C^{f,k}{ㄷ,ㅅ,ㅆ,ㅈ,ㅊ,ㅌ}
*C^{i,k+1}{ㄱ,ㄱ,ㄱ,ㄱ,ㄱ,ㄱ}

[Rule038] C^{f,k}{ㄷ}*C^{i,k+1}{ㄸ}->C^{f,k}{ㄷ,ㅅ,ㅅ,ㅆ,ㅈ,ㅊ,ㅌ}
*C^{i,k+1}{ㄷ,ㄷ,ㄸ,ㄷ,ㄷ,ㄷ,ㄷ}

[Rule039] C^{f,k}{ㄷ}*C^{i,k+1}{ㅂ}->C^{f,k}{ㅅ}*C^{i,k+1}{ㅂ}

[Rule040] C^{f,k}{ㄷ}*C^{i,k+1}{ㅃ}->C^{f,k}{ㅅ,ㅆ,ㅈ,ㅊ,ㅌ}
*C^{i,k+1}{ㅂ,ㅂ,ㅂ,ㅂ,ㅂ}

[Rule041] C^{f,k}{ㄷ}*C^{i,k+1}{ㅅ}->C^{f,k}{ㅅ}*C^{i,k+1}{ㅅ}

[Rule042] C^{f,k}{ㄷ}*C^{i,k+1}{ㅆ}->C^{f,k}{ㅅ,ㅆ,ㅈ,ㅊ,ㅌ}
*C^{i,k+1}{ㅅ,ㅅ,ㅅ,ㅅ,ㅅ}

[Rule043] C^{f,k}{ㄷ}*C^{i,k+1}{ㅉ}->C^{f,k}{ㅅ,ㅅ,ㅆ,ㅈ,ㅊ,ㅌ}
*C^{i,k+1}{ㅈ,ㅉ,ㅈ,ㅈ,ㅈ,ㅈ}

[Rule044] C^{f,k}{ㄷ}*C^{i,k+1}{ㅊ}->C^{f,k}{ㅅ,ㅈ}*C^{i,k+1}{ㅊ,ㅊ}

[Rule045] C^{f,k}{ㄷ}*C^{i,k+1}{ㅌ}->C^{f,k}{ㅅ}*C^{i,k+1}{ㅌ}

5. ㄹ-종성에 관한 복원rule

16개의 main rule과 32개의 sub path가 있다.

[Rule046] C^{f,k}{ㄹ}*C^{i,k+1}{ㄱ}->C^{f,k}{ㄹㄱ}*C^{i,k+1}{ㅇ}

[Rule047] C^{f,k}{ㄹ}*C^{i,k+1}{ㄲ}->C^{f,k}{ㄹ,ㄹㄱ,ㄹㅐ}*C^{i,k+1}{ㄱ,ㄱ,ㄱ}

[Rule048] C^{f,k}{ㄹ}*C^{i,k+1}{ㄴ}->C^{f,k}{ㄹㅎ}*C^{i,k+1}{ㄴ}

[Rule049] C^{f,k}{ㄹ}*C^{i,k+1}{ㄸ}->C^{f,k}{ㄹ,ㄹㅐ,ㄹㅌ}*C^{i,k+1}{ㄷ,ㄷ,ㄷ}

[Rule050] C^{f,k}{ㄹ}*C^{i,k+1}{ㄹ}->C^{f,k}{ㄴ,ㄹ,ㄹ,ㄹㅌ,ㄹㅎ}
*C^{i,k+1}{ㄹ,ㄴ,ㅇ,ㄴ,ㄴ}

[Rule051] C^{f,k}{ㄹ}*C^{i,k+1}{ㅁ}->C^{f,k}{ㄹㅁ}*C^{i,k+1}{ㅇ}

[Rule052] C^{f,k}{ㄹ}*C^{i,k+1}{ㅂ}->C^{f,k}{ㄹㅐ}*C^{i,k+1}{ㅇ}

[Rule053] C^{f,k}{ㄹ}*C^{i,k+1}{ㅃ}->C^{f,k}{ㄹ}*C^{i,k+1}{ㅂ}

[Rule054] C^{f,k}{ㄹ}*C^{i,k+1}{ㅅ}->C^{f,k}{ㄹ}*C^{i,k+1}{ㅅ}

[Rule055] C^{f,k}{ㄹ}*C^{i,k+1}{ㅆ}->C^{f,k}{ㄹ,ㄹㅅ,ㄹㅌ,ㄹㅎ}
*C^{i,k+1}{ㅅ,ㅇ,ㅅ,ㅅ}

[Rule056] C^{f,k}{ㄹ}*C^{i,k+1}{ㅈ}->C^{f,k}{ㄹ}*C^{i,k+1}{ㅈ}

[Rule057] C^{f,k}{ㄹ}*C^{i,k+1}{ㅉ}->C^{f,k}{ㄹ,ㄹㅐ}*C^{i,k+1}{ㅈ,ㅈ}

[Rule058] C^{f,k}{ㄹ}*C^{i,k+1}{ㅊ}->C^{f,k}{ㄹㅌ,ㄹㅎ}*C^{i,k+1}{ㅇ,ㅈ}

[Rule059] C^{f,k}{ㄹ}*C^{i,k+1}{ㅋ}->C^{f,k}{ㄹㄱ,ㄹㅎ}*C^{i,k+1}{ㅎ,ㄱ}

[Rule060] C^{f,k}{ㄹ}*C^{i,k+1}{ㅌ}->C^{f,k}{ㄹㅌ,ㄹㅎ}*C^{i,k+1}{ㅇ,ㄷ}

[Rule061] C^{f,k}{ㄹ}*C^{i,k+1}{ㅍ}->C^{f,k}{ㄹㅐ,ㄹㅍ}*C^{i,k+1}{ㅎ,ㅇ}

6. ㅁ-종성에 관한 복원rule

8개의 main rule과 21개의 sub path가 있다.

[Rule062] C^{f,k}{ㅁ}*C^{i,k+1}{ㄱ}->C^{f,k}{ㄹㅁ}*C^{i,k+1}{ㄱ,ㄱ}

[Rule063] C^{f,k}{ㅁ}*C^{i,k+1}{ㄲ}->C^{f,k}{ㄹㅁ}*C^{i,k+1}{ㄱ,ㄱ}

[Rule064] C^{f,k}{ㅁ}*C^{i,k+1}{ㄴ}->C^{f,k}{ㄹㅐ,ㄹㅍ,ㅁ,ㅁ,ㅂ,ㅂ,ㅂㅅ}
*C^{i,k+1}{ㄴ,ㄴ,ㄹ,ㅇ,ㄴ,ㄹ,ㄴ}

[Rule065] C^{f,k}{ㅁ}*C^{i,k+1}{ㄸ}->C^{f,k}{ㄹㅁ}*C^{i,k+1}{ㄷ,ㄷ}

[Rule066] C^{f,k}{ㅁ}*C^{i,k+1}{ㅁ}->C^{f,k}{ㅁ,ㅂ,ㅂㅅ,ㅍ}
*C^{i,k+1}{ㅇ,ㅁ,ㅁ,ㅁ}

[Rule067] C^{f,k}{ㅁ}*C^{i,k+1}{ㅃ}->C^{f,k}{ㅁ}*C^{i,k+1}{ㅂ}

[Rule068] C^{f,k}{ㅁ}*C^{i,k+1}{ㅇ}->C^{f,k}{ㅂ}*C^{i,k+1}{ㅇ}

[Rule069] C^{f,k}{ㅁ}*C^{i,k+1}{ㅉ}->C^{f,k}{ㄹㅁ}*C^{i,k+1}{ㅈ,ㅈ}

7. ㅂ-종성에 관한 복원rule

8의 main rule과 23의 sub path가 있다.

[Rule070] C^{f,k}{ㅂ}*C^{i,k+1}{ㄲ}->C^{f,k}{ㅂㅅ,ㄹㅐ,ㄹㅍ,ㅍ}
*C^{i,k+1}{ㄱ,ㄱ,ㄱ,ㄱ}

[Rule071] C^{f,k}{ㅂ}*C^{i,k+1}{ㄴ}->C^{f,k}{ㅂㅅ}*C^{i,k+1}{ㄴ}

[Rule072] C^{f,k}{ㅂ}*C^{i,k+1}{ㄸ}->C^{f,k}{ㄹㅐ,ㄹㅍ,ㅂ,ㅂㅅ,ㅍ}
*C^{i,k+1}{ㄷ,ㄷ,ㄷ,ㄷ,ㄷ}

[Rule073] C^{f,k}{ㅂ}*C^{i,k+1}{ㅃ}->C^{f,k}{ㅂ}*C^{i,k+1}{ㅂ}

[Rule074] C^{f,k}{ㅂ}*C^{i,k+1}{ㅅ}->C^{f,k}{ㅂㅅ}*C^{i,k+1}{ㅇ}

[Rule075] C^{f,k}{ㅂ}*C^{i,k+1}{ㅆ}->C^{f,k}{ㄹㅐ,ㅂㅅ,ㅂㅅ,ㅂ,ㅍ}
*C^{i,k+1}{ㅅ,ㅅ,ㅇ,ㅅ,ㅅ}

[Rule076] C^{f,k}{ㅂ}*C^{i,k+1}{ㅉ}->C^{f,k}{ㄹㅐ,ㄹㅍ,ㅂ,ㅂㅅ,ㅍ}
*C^{i,k+1}{ㅈ,ㅈ,ㅈ,ㅈ,ㅈ}

[Rule077] C^{f,k}{ㅂ}*C^{i,k+1}{ㅍ}->C^{f,k}{ㅂ}*C^{i,k+1}{ㅎ}

8. ㅇ-종성에 관한 복원rule

8개의 main rule과 18개의 sub path가 있다.

[Rule078] C^{f,k}{ㅇ}*C^{i,k+1}{ㄲ}->C^{f,k}{ㅇ}*C^{i,k+1}{ㄱ}

[Rule079] C^{f,k}{ㅇ}*C^{i,k+1}{ㄴ}->C^{f,k}{ㄱ,ㄱ,ㄱ,ㄲ,ㄱㅅㄹ,ㅇ,ㅇ}
*C^{i,k+1}{ㄴ,ㄹ,ㅇ,ㄴ,ㅇ,ㄴ,ㄹ,ㅇ}

[Rule080] C^{f,k}{ㅇ}*C^{i,k+1}{ㄸ}->C^{f,k}{ㅇ}*C^{i,k+1}{ㄷ}

[Rule081] C^{f,k}{ㅇ}*C^{i,k+1}{ㅁ}->C^{f,k}{ㄱ,ㄱㅅㄹ,ㅋ}
*C^{i,k+1}{ㅁ,ㅁ,ㅁ,ㅁ}

[Rule082] $C^{f,k}\{ㅇ\}*C^{i,k+1}\{ㅃ\}->C^{f,k}\{ㅇ\}*C^{i,k+1}\{ㅂ\}$

[Rule083] $C^{f,k}\{ㅇ\}*C^{i,k+1}\{ㅆ\}->C^{f,k}\{ㅇ\}*C^{i,k+1}\{ㅅ\}$

[Rule084] $C^{f,k}\{ㅇ\}*C^{i,k+1}\{ㅇ\}->C^{f,k}\{ㅇ\}*C^{i,k+1}\{ㅇ\}$

[Rule085] $C^{f,k}\{ㅇ\} * C^{i,k+1}\{ㅉ\}->C^{f,k}\{ㅇ\}*C^{i,k+1}\{ㅈ\}$

(2) 모음처리 복원 rule

모음처리 복원 rule은 아래와 같이 4개의 main rule로 정의하였다. 모음의 경우에는 대부분이 [Rule101] 처럼 그대로 복원이 되고 특별한 경우가 [Rule102], [Rule103], [Rule104] 의 경우이다. [Rule102] 는 용언의 활용에 나타나는 음성문자열 "저, 쩌, 처"의 경우 "져, 쪄, 쳐" 로 복원이 되고, 나머지의 경우 "ㅕ"는 그대로 복원됨을 나타낸다. [Rule103] 의 "ㅖ"는 "ㅔ", "ㅖ", "ㅙ" 세 가지 경우로 복원될 수 있는데 첫 번째 경우는 그대로 복원되는 경우이고, 두 번째 경우는 "예, 례" 이외의 "ㅖ"는 "ㅔ"로 발음될 수 있다는 규칙에 의한 것이다. 그 예로 게집 [계집], 시게 [시계] 와 같은 경우가 여기에 해당한다. 세 번째의 경우는 조사 "의"는 "ㅔ" 로 발음할 수 있다는 규정에 의해 복원을 고려한 경우이다. [Rule104] 에서는 "ㅣ"가 "ㅣ", "ㅙ" 로 복원될 수 있음을 나타내는데 "ㅣ"는 그대로 복원 되는 경우와 희망 [희망], 유희 [유희] 과 같이 자음을 첫소리로 가지고 있는 음절의 "ㅢ"가 "ㅣ"로 발음되는 경우의 복원에 해당되고 "ㅢ"로의 복원은 주이 [주의], 혀비 [협의] 와 같이 단어의 첫음절 이외의 "의"가 "이"로 발음된 경우의 복원을 나타낸 것이다.

[Rule101] $V^k\{ㅏ,ㅐ,ㅓ,ㅔ,ㅗ,ㅚ,ㅜ,ㅟ,ㅡ,ㅣ,ㅑ,ㅒ,ㅕ,ㅖ,ㅘ,ㅙ,ㅛ,ㅝ,ㅞ,ㅠ,ㅢ\}$
$-> V^k\{ㅏ,ㅐ,ㅓ,ㅔ,ㅗ,ㅚ,ㅜ,ㅟ,ㅡ,ㅣ,ㅑ,ㅒ,ㅕ,ㅖ,ㅘ,ㅙ,ㅛ,ㅝ,ㅞ,ㅠ,ㅢ\}$

[Rule102] if $C^{i,k-1}=\{ㅈ,ㅉ,ㅊ\}$ then $V^k\{ㅕ\} -> V^k\{ㅕ\} \mid V^k\{ㅖ\}$
else $V^k\{ㅕ\} -> V^k\{ㅕ\}$

[Rule103] $V^k\{ㅖ\} -> V^k\{ㅔ\} \mid V^k\{ㅖ\} \mid V^k\{ㅙ\}$

[Rule104] if $C^{i,1} = \{ㅇ\}$ then $V^k\{ㅣ\} -> V^k\{ㅣ\}$
else $V^k\{ㅣ\} -> V^k\{ㅣ\} \mid V^k\{ㅙ\}$

(3) 끝음절 종성 복원 rule

한 어절의 마지막 음절에 종성이 있을 경우에 적용되는 rule로 5개의 main rule과 13개의 sub path로 구성된다. 예를 들면 "새삼"이 [새삶] 으로 복원될 때 어절의 끝음절 "삼"이 "삶"으로 복원되는 경우는 [Rule204] 의 rule에 의해 처리가 되는 경우이다. 그리고 "여덜"이 "여덟"으로 복원될 때 "덜"이 "덟"로 복원되는 경우는 [Rule203] 의 첫 번째 sub path에 의해 처리된 것이다.

[Rule201] $C^{f,k}\{ㄱ\} -> C^{f,k}\{ㄱ,ㄳ,ㄺ,ㅋ\}$

[Rule202] $C^{f,k}\{ㄷ\} -> C^{f,k}\{ㅅ,ㅈ,ㅊ,ㅌ\}$

[Rule203] $C^{f,k}\{ㄹ\} -> C^{f,k}\{ㄻ,ㄼ\}$

[Rule204] $C^{f,k}\{ㅁ\} -> C^{f,k}\{ㄻ\}$

[Rule205] $C^{f,k}\{ㅂ\} -> C^{f,k}\{ㅄ,ㅍ\}$

(4) 한 음절 처리 rule

한 음절 처리 rule은 한 어절이 한 음절인 경우에 적용되는 rule로 종성이 없는 경우는 그대로 복원이 되고, 종성이 있는 경우에는 대표음 처리에 관한 복원이 되어야 하므로 끝음절 종성 복원 rule을 동일하게 적용한다.

2.3 x-clustering 정보

x-clustering은 상용조합 2350자에서 "x 받침을 가질 수 있는 초성, 중성 쌍들의 집합"이라 정의하였는데, x는 무종성(#), ㄱ, ㄴ,…, ㅍ, ㅎ 중에 하나가 될 수 있다. 이 정보는 복원시 여러 개의 sub path를 지닌 임의 rule에서 올바른 음절을 생성시킬 수 있는 sub path만 적용하여 불필요한 복원 과정(path)을 거치지 않도록 하기 위해 사용된다. x-clustering의 종류는 28가지인 데 일부를 나타내면 표 1과 같다.

표 1. x-clustering의 종류
Table 1. Classification of the x-clustering.

종류(개수)	{초성,중성}쌍들의 집합
ㄱ-clustering(201)	{가개갸거겨고과괴구궈귀그 ..... 회회후훼휘휴흐히}
ㄲ-clustering(13)	{겨까꺼나다더바무보서소여파}
ㄳ-clustering(6)	{너모바사써싸}
:	:
ㅂ-clustering(207)	{가개거게겨계고과괘괴교구귀 ..... 표푸프하해허헤혀혜호회효휘흐희히}
ㅄ-clustering(3)	{가어여}
:	:
ㅍ-clustering(20)	{가거기노느니더라르부서수시아어여으이지토}
ㅎ-clustering(20)	{가거나너오다따떠라러마머무빠싸야여으조찌}

예를 들어 "바블"이란 음성 문자열을 복원하기 위해 앞 음절의 종성과 다음 음절의 초성 즉, "#"와 "ㅂ"이 입력 값이 되는데, 이 입력 값에 적용될 수 있는 규칙은 2.2절에 정의된 [Rule008] 이다. 이 rule에 의하면 다음의 세가지 복원 sub path가 존재함을 볼 수 있다.

"#" + "ㅂ" -> ① "ㅂ" + "ㅇ"
-> ② "ㅶ" + "ㅇ"
-> ③ "ㅍ" + "ㅇ"

이 때, 정의한 x-clustering 정보를 사용하지 않는다면 위 세가지 경우의 sub path로 모두 복원시켜 본 후 ①의 경우만 올바른 문자가 생성되고 ②, ③의 경우는 올바른 문자가 생성될 수 없음을 알게 되어 ②, ③의 복원 후보의 경우는 버리게 된다. 하지만, x-clustering정보(ㅂ-clustering,ㅶ-clustering,ㅍ-clustering)를 사용한다면 "바"의 복원 받침으로 ①의 "ㅂ"만 가능하고 ②의 "ㅶ" 과 ③의 "ㅍ"은 적용할 필요가 없음을 미리 알게 되어 ①번 sub path만 적용하게 된다. 이로 인해 좀 더 효과적인 복원이 가능하게 된다.

2.4 postfix 음절 빈도 정보

규칙에 의해 복원된 후보는 한 개부터 여러 개까지 나타났다. 후보들을 조사해 보니 한 음절 한 음절은 문자로 만들어질 수는 있지만 그 음절들이 모여서는 의미를 갖지 못하는 후보가 많이 나타났다. 예를 들어 음성문자열 [하나뿌니야] 의 복원후보들은 "한아뿌니야", "하낫부니야", "한아부니야", "한앗부니야", "하나뿐이야", "한아뿐이야", "하낫분이야", "한앗분이야" 8가지로 나왔다. 이처럼 많은 후보가 나온 경우에 모두를 형태소 분석한다면 형태소 분석기의 처리시간이 증가하게 되어 형태소분석기에 부담을 주게 된다. 그러므로 형태소분석기의 입력 후보수를 제한하는 것이 필요하다. 그래서 단어로 생성될 수 없는 후보를 미리 제한하기 위한 방안으로 임의의 한 음절 뒤에 바로 나타날 수 있는 문자의 빈도 값을 말뭉치로부터 구해서 이용한다. 말뭉치로 사용된 것은 신문사설, 초등학교 교과서, 소설로 약 60만 어절이다. 빈도값은 상용조합 2350자를 기준으로 하여 말뭉치로부터 획득하였는데 2350자 중 1777자가 자주 사용되는 문자로 나타났다. 아래 (a)는 [하나뿌니야] 의 복원 후보 각각에 대해 postfix음절 빈도정보 검색에 실패한 위치를 ^ 로 표기했다. ^의 앞음절과 뒤음절은 함께 나타날 수 없으므로 복원 과정 중에 제거되어 8개의 후보가 1개로 제한될 수 있음을 보여준다. 그리고 (b)는 음절 "아"에 대한 postfix음절 빈도정보의 일부를 보인 것이다. 약 60만 말뭉치에서 상용조합 2350자 중 374자가 "아" 뒤에 올 수 있는데, 그 중에 가장 많이 나타나는 문자는 "니"로 4271번 나타났고, 2순위는 "버"로 2287번, 3순위는 "이"로 1552번 나타났다.

(a) 8개의 복원후보에 대한 postfix음절 사용예

[한아^뿌니야] [하^낫부니야] [한아부^니야][한^앗부니야]
[하나뿐이야] [한아^뿐이야][하^낫분이야] [한^앗분이야]
==> 결과 : [하나뿐이야]

(b) 음절 "아"에 대한 postfix음절 374개의 빈도정보

가1469|간85|갈189|감5|갑32|갔148|갖1|개1|걸1|게1|겠2|경3|
계4|공3|과2|관2|교4|구3|국16|군7|궁8|귀32|규3|그13|글1|금1|
:
니4271|닌836|닐341|님41|닙340|닝3|다183|닥3|달6|담13|당26|
:
바3|받1|배1|버2287|번1|범8|법2|베3|벨2|병1|보1446|본58|
:
웃7|웅3|원12|유10|으6|을2|의216|이1552|인15|일13|임3|입13|
:

2.5 형태소분석

본 시스템은 복원된 어절을 기본 처리 단위로 하여 그 어절을 구성하는 형태소들을 찾아주는 형태소 분석 단계를 거친다. 음성인식 후처리의 형태소 분석은 복원 단계에서 처리되어 나온 복원 후보들에 대해서 형태소들이 결합하여 올바른 어절을 구성할 수 있는지를 검사하여 비문법적 후보 어절을 필터링 해준다. 예를 들어 "먹어라"의 음성문자열 "머거라"는 3가지 후보{머거라|먹을아|먹어라}로 복원이 되어 형태소 분석기로 넘어온다. 각 후보를 형태소 분석하면, "머거라"와 "먹을아"의 경우는 실패하게 되고, "먹어라"는 먹(어간)+어라(어미)로 형태소 분석에 성공한다. 그러므로 최종 결과는 "먹어라"만 생성된다. 본 논문에서는 양방향 최장일치 형태소 분석 방법[14]을 이용한다.

Ⅲ. 처리 알고리즘 및 적용예

음성문자열 복원에 관한 알고리즘은 크게 4가지 규칙 루틴에 의해 수행되고, 그 복원 결과를 형태소 분석하여 최종 결과가 생성된다.

```
syllable_recovery_rule() {
/* 입력 : buffer [ 0..n ]  --> 3바이트 조합형 코드로 저장된 배열
   출력 : 복원 rule 적용 결과 후보들의 list */
  c = n;
  if ( c == 2 ) mono_syllable_rtn() /* 한음절 처리 루틴 실행 */
  else {
```

```
        for ( i = 0; i < c ; i++ ) {
            swich ( i%3 ) {
              case 0 : result [ 0 ]  = buffer [ 0 ] ;
                       break;
              case 1 : vowel_rtn(); /* 모음 처리 루틴
                       수행 */
                       break;
              case 2 : if ( i ≠ c )
                         syllable_boundary_rtn();
                       /* 음절 경계 종성 초성 처리 루틴
                       수행 */
                       else
                         final_syllable_rtn();
                       /* 끝음절 종성 처리 루틴 수행 */
                       break;
            } /* swich-end */
          } /* for-end */
  postfix_syllable_check(); /* postfix 음절 빈도 정보 체
  킹 */
  morpheme_analysis(buffer, result); /* 형태소 분석 */
  }
}
```

다음은 간단한 예제를 통하여 처리과정을 보인다. 음성문자열 "바블 꼭 머거라"의 처리과정을 살펴보겠다.

예제) 바블 꼭 머거라.

$C^{i,1}$	V^1	$C^{f,1}$	$C^{i,2}$	V^2	$C^{f,2}$
ㅂ	ㅏ	#	ㅂ	ㅡ	ㄹ

(a) 바블

$C^{i,1}$	V^1	$C^{f,1}$
ㄲ	ㅗ	ㄱ

(b) 꼭

$C^{i,1}$	V^1	$C^{f,1}$	$C^{i,2}$	V^2	$C^{f,2}$	$C^{i,3}$	V^3	$C^{f,3}$
ㅁ	ㅓ	#	ㄱ	ㅓ	#	ㄹ	ㅏ	#

(c) 머거라

위의 (a)(b)(c) 항에 적용되는 rule과 처리과정은 다음과 같다.

(a) [바블] 의 적용rule과 처리과정

(a.1) $C^{f,1}*C^{i,2} \in C^{f,k}\{\#\}*C^{i,k+1}\{ㅂ\}$->$C^{f,k}\{ㅂ,ㅃ,ㅍ\}*C^{i,k-1}\{ㅇ,ㅇ,ㅇ\}$
(a.2) $V^1, V^2 \in V^k\{ㅏ,ㅐ,ㅑ,ㅒ,ㅓ,ㅔ,ㅕ,ㅖ,ㅗ,ㅘ,ㅙ,ㅚ,ㅛ,ㅜ,ㅝ,ㅞ,ㅟ,ㅠ,ㅡ,ㅢ,ㅣ\}$-> $V^k\{ㅏ,ㅐ,ㅑ,ㅒ,ㅓ,ㅔ,ㅕ,ㅖ,ㅗ,ㅘ,ㅙ,ㅚ,ㅛ,ㅜ,ㅝ,ㅞ,ㅟ,ㅠ,ㅡ,ㅢ,ㅣ\}$
(a.3) $C^{f,2} \in C^{f,k}\{ㄹ\}$->$C^{f,k}\{ㄾ,ㅀ\}$

(a.2)|(a.1) **(a.2)|(a.3)**

-> self -> 바블(ㅇ) -> self -> 바블(o)
래 -> 바븗(x)
라 -> 바븗(x)

(a.2)|(a.3)

-> path1> ㅂ+ㅇ -> 밥을(o) -> self -> **밥을(o)**
래 -> 밥읊(x)
라 -> 밥읊(x)
-> path2> ㅃ+ㅇ -> 빱을(x)
-> path3> ㅍ+ㅇ -> 팝을(x)

(a.2)|(a.1)는 (a.2)를 먼저 적용하고 (a.1)을 적용함을 의미하고, 밑줄이 있는 부분은 각각 래-clustering, 라-clustering, ㅃ-clustering, ㅍ-clustering을 참조하므로 생성되지 않는 부분이다. 아래 (b), (c)의 경우에도 마찬가지이다. 결과로는 후보 "바블"과 "밥을"이 생성되고, 이 경우는 postfix음절 빈도정보를 이용해도 "바블"과 "밥을"의 두 후보가 모두 유효하여 형태소 분석기로 넘겨진다. 그러나 "바블"은 형태소 분석에 실패하게 되고, "밥을"은 밥(명사)+을(조사)로 분석되어 형태소 분석에 성공하게 된다.

(b) [꼭] 의 적용 rule과 처리과정

(b.1) $C^{f,1} \in C^{f,k}\{ㄱ\}$ -> $C^{f,k}\{ㄲ,ㄳ,ㄺ,ㅋ\}$

(b.1)

-> self -> **꼭(o)**
-> path1> ㄲ -> 꾝(x)
-> path2> ㄳ -> 꽋(x)
-> path3> ㄺ -> 꽑(x)
-> path4> ㅋ -> 꽄(x)

(b)는 한 음절로 이루어진 어절로 종성 "ㄱ"이 있으므로 이에 대한 rule (b.1)이 적용될 때 각각의 clustering 정보를 참조하면 "꼭"만 결과로 생성된다. 이를 형태소 분석하면 "꼭(부사)"로 성공한다.

(c) [머거라] 의 적용rule과 처리과정

(c.1) $C^{f,1}*C^{i,2} \in C^{f,k}\{\#\}*C^{i,k+1}\{ㄱ\}$->$C^{f,k}\{ㄱ,ㄳ,ㄺ,ㅋ\}*C^{i,k-1}\{ㅇ,ㅇ,ㅇ,ㅇ\}$
(c.2) $V^1, V^2 \in$ if $C^{i,k-1}=\{ㅈ, ㅉ, ㅊ\}$ then $V^k\{ㅓ\}$-> $V^k\{ㅓ\} \mid V^k\{ㅕ\}$ else $V^k\{ㅓ\}$->$V^k\{ㅓ\}$
(c.3) $C^{f,2}*C^{i,3} \in C^{f,k}\{\#\}*C^{i,k+1}\{ㄹ\}$->$C^{f,k}\{ㄹ,ㄾ\}*C^{i,k-1}\{ㅇ,ㅇ\}$
(c.4) $V^3 \in V^k\{ㅏ,ㅐ,ㅑ,ㅒ,ㅓ,ㅔ,ㅕ,ㅖ,ㅗ,ㅘ,ㅙ,ㅚ,ㅛ,ㅜ,ㅝ,ㅞ,ㅟ,ㅠ,ㅡ,ㅢ,ㅣ\}$
->$V^k\{ㅏ,ㅐ,ㅑ,ㅒ,ㅓ,ㅔ,ㅕ,ㅖ,ㅗ,ㅘ,ㅙ,ㅚ,ㅛ,ㅜ,ㅝ,ㅞ,ㅟ,ㅠ,ㅡ,ㅢ,ㅣ\}$

(c.2)|(c.1) **(c.2)|(c.3)|(c.4)**

-> self -> 머거 -> self -> 머거라(o)
-> path1> ㄹ+ㅇ -> 머^걸아(o)
-> path2> ㄾ+ㅇ -> 머겋아(x)

(c.2)|(c.3)|(c.4)

-> path1> ㄱ+ㅇ ->먹어 -> self -> **먹어라(o)**
-> path1> ㄹ+ㅇ-> 먹을아(o)

-> path2> 러+ㅇ-> 머을아(x)
-> path2> ㄱ+ㅇ -> 멐어(x)
-> path3> 러+ㅇ -> 멐어(x)
-> path4> ㅋ+ㅇ -> 멐어(x)

(c)의 경우에도 4개의 후보에서 "머^걸아"는 postfix음절 빈도정보를 이용하므로 제거되고, "머거라", "먹어라", "먹을아"가 형태소분석기로 넘겨진다. 형태소분석기에서 형태소 분석을 하면 "머거라"와 "먹을아"는 실패하게 되고 "먹어라"만 "먹(어간)+어라(어미)"로 형태소분석에 성공하게 된다.

그러므로 최종 결과로는 각각 형태소분석에 성공한 후보 (a)의 "밥(명사)+을(조사)", (b)의 "꼭(부사)", (c)의 "먹(어간)+어라(어미)"가 선택되어 "밥을 꼭 먹어라"라는 문서기반 문자열이 생성된다.

Ⅳ. 결과 분석

4.1 Rule 적용 횟수 분석

다음의 어절단위 음성문자열 $C^{i,1}V^{1}C^{f,1}*C^{i,2}V^{2}C^{f,2}*\cdots*C^{i,k}V^{k}C^{f,k}*\cdots*C^{i,n}V^{n}C^{f,n}$ 이 입력되었을 때, 복원 rule이 적용되는 조건은 다음과 같다.

1) $1 < k < n$ 일 때, $C^{f,k}*C^{i,k+1}$에 대해서는 음절 경계 종성 초성 복원 rule을 적용한다.

2) $C^{f,k} \neq$ fill-code(#) and k=n 일 때, $C^{f,k}$ 에 대해서는 끝음절 복원 rule을 적용한다.

3) $k = \{1,2,3 \cdots n\}$ 일 때, V^{k} 에 대해서는 모음처리 복원 rule을 적용한다.

4) $C^{f,k} \neq$ fill-code(#) and k=n=1 일 때, $C^{i,k}V^{k}C^{f,k}$에 대해서는 한 음절 처리 rule을 적용한다.

이상과 같은 조건에 맞게 복원 rule들이 적용되는데, 한 어절에 대해 rule 적용횟수를 한번 분석해 보자. 먼저 한 어절이 한 음절로 구성되어 있다면 1회의 한 음절 처리루틴으로 수행이 끝난다. 그리고 한 음절이 n개의 음절로 구성되어 있고, 한 rule 내에 c개의 가능한 path가 존재한다면 c*(n-1)회의 음절 경계 종성 초성 복원 rule 적용과 n번의 모음처리 복원 rule 적용 그리고 한 어절의 마지막 음절의 종성이 있다면 c번의 끝음절 복원 rule의 적용으로 모든 수행이 끝난다. 그러므로 한 어절에 대한 rule의 적용횟수를 계산해 보면 다음과 같다.

1) n=1 일 때, 한 어절에 대한 수행시간 = 1회의 한 음절 처리 rule 적용

2) n>1 일 때, 한 어절에 대한 수행시간 = c*(n-1)회의 음절 경계 종성 초성 복원 rule + n회 모음처리 복원 rule 적용 + c회의 끝음절 복원 rule적용 (단, 끝음절의 종성이 무종성일 경우는 3번째 항목은 비적용)

그러므로 수행시간은 O(1) 또는 c*(n-1)+n+c= O(n)이 된다. 그리고 한 문장이 m개의 어절로 구성되어 있다면 한 문장의 복원시간은 O(nm)에 수행이 완료된다. 여기에 양방향 최장일치 형태소 분석[14] 시간O(n)을 합하면 전체 수행시간이 된다.

4.2 postfix 음절 빈도 정보 사용 효과

본 논문에서 제안한 postfix 음절 빈도 정보는 규칙을 적용한 복원 후보들을 모두 형태소 분석하지 않고 복원된 후보 중에서 가능성 있는 후보들만 추출하여 분석하기 위한 정보이다. 이 정보의 사용으로 형태소 분석기의 부담을 많이 줄여 줄 수가 있다. 이 정보의 사용 효과를 검증하기 위해 복원 규칙을 적용해 나온 후보의 수에 postfix 음절 빈도 정보를 사용하여 줄어든 후보의 수를 비교 분석하여 일부를 나타내면 표 2와 같다.

표 2. postfix음절 빈도 정보 사용 효과
Table 2. The using effect of a postfix-syllable frequency information.

1 -> 1 : 231개	2 -> 1 : 198개 2 -> 2 : 79개	3 -> 1 : 56개 3 -> 2 : 22개 3 -> 3 : 22개
4 -> 1 : 84개 4 -> 2 : 42개 4 -> 3 : 43개 4 -> 4 : 7 개	5 -> 1 : 18개 5 -> 2 : 3개 5 -> 5 : 1개	6 -> 1 : 38개 6 -> 2 : 37개 6 -> 3 : 17개 6 -> 4 : 9개 6 -> 5 : 1개
7 -> 1 : 10개 7 -> 2 : 5개 7 -> 3 : 1개 7 -> 4 : 11개 7 -> 5 : 2개	8 -> 1 : 27개 8 -> 2 : 8개 8 -> 3 : 18개 8 -> 4 : 8개 8 -> 5 : 3개 8 -> 6 : 1개	9 -> 1 : 5개 9 -> 2 : 5개 9 -> 3 : 12개 9 -> 4 : 1개 9 -> 5 : 1개 9 -> 6 : 2개
10 -> 1 : 8개 10 -> 2 : 6개 10 -> 3 : 7개 10 -> 4 : 5개	11 -> 1 : 1개 11 -> 2 : 1개 11 -> 3 : 1개	12 -> 1 : 19개 12 -> 2 : 12개 12 -> 3 : 7개 12 -> 4 : 10개 12 -> 5 : 3개 12 -> 6 : 1개 12 -> 10 : 1개
:	:	:

표 2에서 "->"의 좌측은 postfix 음절 빈도 정보를 사용하기 이전의 후보 수이고, 우측은 postfix 음절 빈도 정보를 사용하므로 줄어든 후보의 수를 나타낸다. 그리고 ":" 다음의 숫자는 각각의 개수를 나타낸다. 표

2에서 복원 후보수가 6개 나온 경우는 총 102개인데, 그 중 postfix음절 정보를 사용하므로 후보수가 1개로 줄어든 경우가 38개, 2개로 줄어든 경우가 37개, 3개로 줄어든 경우가 17개, 4개로 줄어든 경우가 9개, 5개로 줄어든 경우가 1개임을 나타낸다. 위 표에 의하면 postfix 음절 빈도 정보를 사용하므로 여러 개 후보가 1개 또는 2개, 3개 … 로 후보수가 감소하는데 큰 영향을 끼침을 알 수 있다. 한 어절의 길이가 길수록 복원 후보의 수가 많이 나타났다. 주로 문장에서 동사의 경우가 후보수가 많았고, 이 정보를 이용하므로 많은 후보수가 단일 후보 또는 소수 개 후보로 전이된 경우도 동사가 많은 비중을 차지했다.

4.3 형태소분석 적용 결과와 비교 분석

본 논문에서는 연속 음성 문자열 1,405어절을 대상으로 실험을 하였다. 실험은 두 가지 방법으로 행했는데, 먼저 복원 규칙을 적용한 후 postfix음절 빈도 정보를 체크해서 얻어진 결과를 A라 하고, A의 결과에 대해 자연어 처리 기법인 양방향 최장 일치 형태소 분석 기법을 적용[14]하여 나온 결과를 B라 하였다. 먼저 A와 B 실험의 복원 후보수를 비교 분석한 결과는 그림 2와 같다.

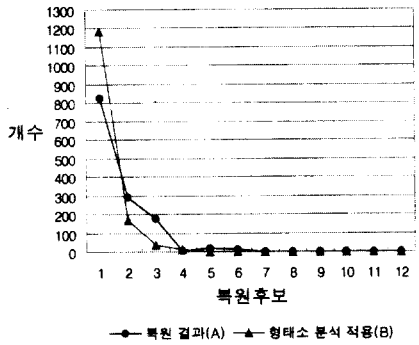


그림 2. A와 B 후보수 비교
Fig. 2. Candidate number comparison of A and B.

그림 2에서 나타난 결과를 보면, 음운변동이 반영된 음성문자열을 복원시켜 형태소 분석을 행하여 줌으로써 결과가 단일 후보로 생성되는 어절의 수가 829에서 1,187개로 357개 즉 25.48%가 증가되었고, 나머지 5~12개의 후보수를 가지는 경우는 모두 없어졌음을 알 수 있다. 그러므로 이 결과를 통해 음성 인식된 결과는 후처리되어 자연어 처리 기법과 접목되어 처리되는 것이 훨씬 효과적임을 명백히 알 수가 있다.

다음은 A와 B 실험의 후보수 전이율을 분석한 결과로 다음의 표 3과 같다.

표 3. 후보수 전이율 분석표
Table 3. Analysis table of candidate number tansition rate.

방법A	방법B	백분율	방법A	방법B	백분율
2	1	47.05%	6	2	0.22%
3	1	19.39%	6	3	0.65%
3	2	12.85%	6	4	0.22%
4	1	5.44%	7	1	0.22%
4	2	6.12%	8	1	0.22%
4	3	0.65%	9	2	0.22%
5	1	3.92%	10	2	0.43%
5	2	0.65%	12	1	0.22%
6	1	1.31%	12	4	0.22%

표 3에 나타난 후보수 전이 상태는 형태소 분석을 하므로 후보수 2개가 1개로 결정되는 경우가 47.05%로 가장 높았다.

위 그림 2의 형태소 분석 적용 결과에서 실험 어절 1,405어절 중 1,187 어절 즉 84.48%가 단일 후보를 생성했고, 나머지 15.52%에 해당하는 218어절은 여러 개 후보가 생성되었다. 15.52%에 해당하는 어절의 여러 개 복원 후보들 중에는 원하는 text 문자열은 포함하고 있었다. 하지만 어느 후보가 문맥에 맞는 정확한 후보인지는 형태소 분석만으로는 알 수가 없었다. 그래서 다음은 후보가 여러 개 생성된 요인을 분석해 보았다. 복원후보가 여러 개 생성된 주 요인은 다음과 같은 세 가지 이유였다. 첫 번째 이유는 실제 형태소의 자소가 다르지만 음운현상에 의해 동일한 음가로 생성되는 동음이형어로 생긴 경우로 표 4와 같은 예들이다.

표 4. 동음이형어의 모호성
Table 4. Ambiguous of the same-sound different-form word.

음성문자열	text기반문자열	복원후보
읻게	있게	{잇게|있게|잊게}
말꼬	맑고	{말꼬|맑고}
안는다면	않는다면	{안는다면|않는다면}

두 번째 이유는 모음의 발음에서 "ㅐ", "ㅔ", "ㅙ"가 모두 "ㅐ"로 발음될 수 있고, "ㅣ"와 "ㅢ" 둘 다 "ㅣ"로 발음 날 수 있으므로 이를 역으로 복원하면서 후보가 여러 개 발생한 경우로 표 5와 같다.

표 5. 모음 발음의 모호성
Table 5. Ambiguous of the vowel pronunciation.

음성문자열	text기반문자열	복원후보
집게	집게	{집게|집계}
마으리	마을이	{마을이|마을의}
마으레	마을에	{마을에|마을의}

세 번째 이유는 rule의 정의에서 발음법의 예외 부분들을 규칙으로 일반화시킴으로써 부수적으로 생성된 후보들이다. 예를 들면 음성문자열 "인는"을 복원한 결과 {인은|있는|잇는|잊는} 4개의 후보가 나타났는데 두 번째, 세 번째, 네 번째 후보는 동음이형어로 나타난 경우이고 첫 번째 후보는 합성어에서 [눈뇨기] 눈-요기, [남존녀비] 남존-여비의 경우처럼 "ㄴ"종성과 "ㅇ"초성이 만나 "ㄴ"과 "ㄴ"으로 발음이 바뀌는 경우를 규칙화하여 일반화시킴으로써 합성어가 아닌 "인는"의 경우에 영향을 미친 것이다. 이처럼 세 가지 이유로 복원 결과가 여러 개 나타났다. 첫 번째, 두 번째 이유는 생길 수밖에 없는 문제이며 동시에 해결 방안이 강구되어야 하는 주요 연구과제이고, 세 번째 경우는 규칙을 좀 더 정비하면 해결될 수 있는 문제라 생각한다.

4.4 기존 방식과의 비교

다음의 표 6은 서론에서 언급한 기존의 연구 방식들과 본 논문의 연구 방식을 몇가지 관점에서 비교한 것이다.

표 6. 기존 방식과의 비교
Table 6. Comparison of the existing methods.

방식 / 비교 관점	본 시스템	읽기 규칙 역적용 방식	음운변이 처리	음소열사전을 이용한 방식
rule사용	간결한 rule적용	여러차례의 복잡한 rule적용	모든 음운현상에 대해서 rule을 적용못함.	rule 사용하지 않음
사전사용 (형태소분석 사전 제외)	사전 없음 x-clustering postfix음절 빈도정보	조사어미사전 선어말어미사전, 예외사전, 음절사전	자소단위 사전	음운변동 고려한 사전
형태소분석	음운 복원후 형태소분석	음운 복원후 형태소분석	형태소분석시 음운복원	형태소분석시 음운복원
사전검색횟수	없음	적음	빈번	많음
사전 구축 시간	불필요	조금 필요	조금 필요	많이 필요
도메인에 따른 안전성	안정적	조금 안정적	조금 안정적	불안정
알고리즘	간단	복잡	간단	복잡
초점	하나의 입력이 하나의 출력 결과를 내는 것	올바른 출력을 가지는 것이 목표	올바른 출력을 가지는 것이 목표	사전검색으로 빠른 처리

V. 결 론

한국어가 연속적으로 발음될 때 여러 가지 음운변동이 일어나는데 이러한 음운변동은 한국어 음성 인식을 어렵게 하는 주요 요인이 된다. 본 논문에서는 이러한 음운변동이 반영된 음성 인식 문자열을 text 기반 문자열로 다시 복원시키는 음절 복원 rule-based 시스템을 제안했다. 이 시스템은 기존의 시스템들과 달리 대량의 발음열 사전이나 음소열 사전이 필요 없는 규칙기반 시스템이다. 이 시스템에 정의된 rule은 음절 경계 종성 초성 복원 rule로 85개 main rule과 227개 sub path, 모음처리 복원 rule로 4개 main rule과 25개 sub path, 끝음절 종성 복원 rule로 5개의 main rule과 13개의 sub path 이다. 한 음절처리 rule은 종성이 있을 때만 끝음절 종성 복원 rule을 함께 사용한다. 그리고 좀 더 효과적인 복원을 위해 x-clustering정보를 정의하여 사용하고, 형태소 분석기에 입력될 복원 후보 수를 제한하기 위해 postfix음절 빈도정보를 구하여 사용했는데 이 정보들은 복원중에 유용하게 사용되었다. 그리고 복원결과는 형태소 분석되어 유용한 문자열만 최종결과로 생성되는데, 실험 대상 중 84.48%가 단일 text문자열로 생성되었고, 15.52%는 4.3에 언급한 세 가지 이유로 결과가 여러 개 복원 후보가 나타났다. 15.52%에 해당하는 복원후보들은 원하는 결과 문자열은 포함하고 있지만 이들 중 문맥에 맞는 정확한 하나의 결과를 찾아내는 것은 구문분석 또는 의미분석이 되어야 하므로 향후 과제로 남겼다. 보통 음성을 이용하는 시스템의 경우 대부분이 음성 그 자체로서의 의미보다는 음성이 문자의 형태로 변환되어야 의미를 가지는 경우가 많으므로 이 시스템은 대용량 연속 음성 인식 결과를 신속히 문서화하는 응용분야에 기초 시스템으로 유용하게 사용될 수 있으리라고 생각한다. 향후 방향은 형태소 분석단계에서 생성된 결과 중 후보가 여러 개 생성된 경우에 형태소 분석된 어절들을 대상으로 1개에서 N개의 문장을 생성해 이를 구문 분석, 의미 분석을 하여 의미있는 하나의 문장을 생성해내는 연구를 계속하는 것이다.

참 고 문 헌

[1] 김영길, 김한우, 최병욱, "견고한 대화시스템을

위한 한국어 대화체의 음운론적, 구문론적 오류 분석 및 복구", *전자공학회논문지*, 제34권 C편 제5호, 1997

[2] 이호영, *국어음성학*, 태학사, 1996

[3] 문화체육부, *국어 어문 규정집*, 대한교과서주식회사, 1997

[4] 이계영, "한국어 음운변동 처리 규칙의 설계 및 구현", *한국어정보처리학회 논문지*, 제5권 제 3호, pp. 851-861, 1998년

[5] 김병수, "*음절단위를 이용한 한국어 음성합성에 관한 연구*", 한양대학교, 석사학위논문, 1989

[6] 이계영, 이상범, "한국어 음운 변동 처리를 위한 효율적인 Rule Base System의 구성", *전자공학회논문지*, 제 28권 B편 제 12호, 1991

[7] 김경희, 이근배, 이종혁, "한국어 음성언어 처리를 위한 음소 단위 인식과 형태소 분석의 결합", *정보과학회 논문지(B)*, 제22권 제10호, 1995년

[8] 정민화, "한국어 연속음성인식을 위한 자연언어처리 기술의 적용방법", *'98 지능기술 튜토리얼*, pp. 27-55, 1998

[9] 김병창, 이원일, 이근배, 이종혁, 이영직, "형태소 그래프를 이용한 한국어 연속음성인식과 형태소분석의 통합", *한국정보과학회 가을 학술발표논문집*, pp. 549-552, 1996

[10] 서상현, "*한글 음운 규칙에 기반한 음절 복원기 구현*", 경북대학교, 석사학위논문, 1997

[11] 이원일, "*신경망과 CYK-table을 이용한 음성언어의 분석*", 포항공대, 석사학위논문, 1993

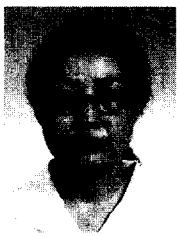
[12] 전재훈, 차선희, 정민화, "형태음운론적 분석에 기반한 한국어 발음 생성", *한국어 정보과학회 가을 학술발표논문집*, pp. 247-250, 1997

[13] 이근용, 이기오, 안동언, 이용석, "한국어 음성인식 후처리를 위한 음운변이 처리", *한국정보과학회 봄 학술발표논문집*, pp 927-930, 1996

[14] 최재혁, "*양방향 최장일치법에 의한 한국어 형태소 분석기의 구현*", 경북대학교, 박사학위논문, 1993

[15] 박미성, 김미진, 이문희, 이상조, 최재혁, "연속음성인식 후처리를 위한 음절복원 rule-base 시스템", *제10회 한글 및 한국어 정보처리 학술대회*, pp. 379-385, 1998

저 자 소 개



朴 美 星(正會員)

1967년 5월 20일생. 1990년 2월 창원대학교 전자계산학과 이학사. 1992년 2월 경북대학교 컴퓨터공학과 대학원 공학석사. 1997년 2월 경북대학교 컴퓨터공학과 대학원 박사과정 수료. 1992년 ~ 현재 경북대학교 도서관 전산실 근무. 주관심분야는 자연어처리, 음성인식, 전자도서관, 데이타베이스

金 桂 成(正會員) 第 34卷 C編 第 5號 參照

李 相 祖(正會員) 第 33卷 B編 第 4號 參照

金 美 辰(正會員)

1954년 7월 24일생. 1978년 2월 한양대 영어영문학과 문학사. 1991년 2월 경북대 대학원 컴퓨터공학과 공학석사. 1999년 현재 경북대 대학원 컴퓨터공학과 박사과정 수료. 주관심분야는 자연어처리, 정보검색, 음성인식 후처리

崔 宰 赫(正會員)

1984년 2월 경북대학교 전자공학과 공학사. 1986년 2월 경북대학교 전자공학과 공학석사. 1994년 2월 경북대학교 전자공학과 공학박사. 1989년 ~ 1994년 부산여자대학교 자연과학대학 전자계산학과 재직. 1995년 ~ 현재 신라대학교(구 부산여자대학교) 사범대학 컴퓨터교육과 부교수로 재직중. 주관심분야는 자연어처리, 기계번역, 컴파일러