(제 10회 한글 및 한국어 정보처리 학술대회)

연속음성인식 후처리를 위한 음절 복원 rule-base시스템

박미성*, 김미진*, 이문희*, 최재혁**, 이상조*
* : 경북대학교 컴퓨터공학과, ** : 신라대학교 컴퓨터교육과

The syllable recovery rule-base system for the post-processing of a continuous speech recognition

* : Dept. of Computer Engeering, Kyungpook National University
** : Dept. of Computer Education, Silla University
mspark@bh.kyungpook.ac.kr

요 약

한국어가 연속적으로 발음될 때 여러 가지 음운변동현상이 일어난다. 이것은 한국어 연속음성 인식을 어렵게 하는 주요 요인 중의 한가지이다. 본 논문은 음운변동현상이 반영된 음성 인식 문자열을 규칙에 의거하여 text 기반 문자열로 다시 복원시키고 복원 결과 후보들을 형태소 분석하여 유용한 문자열만을 최종 결과로 생성하게 하는 시스템을 구성하였다. 복원은 4가지 rule 즉, 음절 경계 종성 초성 복원 rule, 모음처리 복원 rule, 끝음절 종성 복원 rule, 한음절처리 rule에 따라 이루어진다. 규칙 적용 과정 중에 효과적인 복원을 위해 x-clustering정보를 정의하여 사용하고, 형태소 분석기에 입력될 복원 후보수를 제한하기 위해 postfix음절 빈도정보를 구하여 사용한다.

1. 서론

음성 인식은 보통 고립단어 인식과 연속음성 인식으로 분류하는데, 고립단어 인식은 단어간에 쉼(Pause)이 존재하여 각 단어의 발음이 서로 다른 단어에게 영향을 미치지 않는다. 그러나 연속음성 인식에서는 단어를 연속적으로 발음하기 때문에 단어간에 쉼이 존재하지 않아 각각의 음소가 주변 음소에 영향을 끼쳐 음성인식을 어렵게 한다. 한국어를 연속적으로 발음할 때 나타나는 이러한 현상을 한국어 음운변동현상이라 한다[1,2,3,4,5]. 그러므로 연속음성 인식을 위해서는 연속음성 문자열에 대한 한국어의 음운변동현상에 대한 처리가 우선적으로 이루어져야 하고 다음은 형태소분석, 구문분석, 의미분석까지 가능하게 되어야 한다[6,7,8]. 음운변동현상 처리에 관한 기존 연구로는 읽기 규칙을 역으로 적용한 방식[9], 음성인식의 결과로 나온 음소의 열로부터 원래의 음절로 복원하기 위해 음소열 사전을 이용한 방식[10], 발음열 사전을 이용한 방식[7,11], 자소 단위 사전을 이용하여 형태소 단계에서 음운변동현상을 처리한 방식[12] 등이 있다. [9]는 규칙 적용 순서가 복잡하고 예외가 많으며 알고리즘이 복잡하다. [10]은 음성 인식의 결과로 나온 음소열을 표제어로 하여 그 음소열이 발음될 수 있는 모든 단어를 사전에 수록했는데, 이 방법은 확장할 경우 사전량이 매우 커지고 구축하는데 시간도 많이 걸린다. [7,11]도 [10]과 마찬가지로 발음규칙을 기반으로하여 발음열 사전을 미리 만들어 놓고 형태소분석에 이용하므로 사전량이 커지는 단점이 있다. [12]는 사잇소리, 경음화, 끝음절 대표음 처리를 제외한 일부 음운변동현상을 규칙으로 정의하고, 자소 단위 사전 검색을 통해 형태소분석 단계에서 복원 후보를 결정하는 방식으로 음운변동규칙 모두를 수용하지 못했고 사전 검색 횟수가 많다. 본 논문에서는 한국어가 연속적으로 발음될 때 일어나는 음운변동현상이 반영된 음성 문자열을 규칙들에 의거하여 text 기반 문자열로 다시 복원시키고, 복원 결과 후보를 형태소 분석하여 유용한 문자열만을 최종결과로 생성하게 하는 시스템을 구성하였다. 본 논문에서 제안한 방법은 기존의 방법들과는 달리 음소 변이 규칙기반 시스템이므로 대량의 발음열사전이나 음소열사전이 필요 없고, 읽기 규칙을 여러 단계로 역적용할 필요도 없다. 또한 이 시스템은 기존의 문서 기반 형태소 분석기를 그대로 이용할 수 있다는 이점이 있다.

1) 본 연구는 1997년 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었습니다.

2. 음절복원 rule-base시스템

한국어 음성인식 결과를 자연어처리기법과 접목시키기 위해 몇 가지 어려운 문제점이 있다[7]. 첫째는 말하는 단위와 문서의 띄어 쓰기 단위가 불일치 한다는 것이다. 둘째는 각 형태소들이 형태소 내부에서 뿐 만 아니라 형태소와 형태소 사이에서 음운변동 현상 일어난다는 것이다. 이 논문에서 첫 번째 문제점은 어절생성기를 통해 해결할 수 있다고 가정하였다. 그래서 본 시스템의 입력은 음성문자열의 어절로 가정한다. 두 번째 문제점은 이 논문의 주제로 음절 복원rule_base시스템을 제안하여 구현한다.

2.1 전체 시스템 구성도

입력된 어절은 앞으로 설명하는 각종 규칙에 의거하여 복원된 후에 형태소 분석기로 넘겨져 유용한 문자열만 생성되도록 하였다. 전체 시스템 구성도는 그림1과 같다.

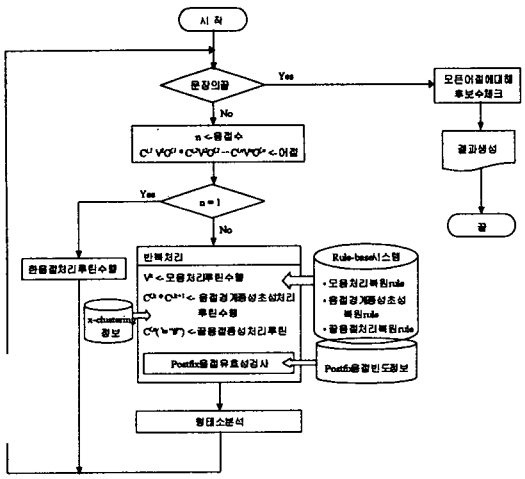

그림 1. 시스템 구성도

2.2 음절복원 Rule_Base 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,n 은 한 어절내의 각 음절 위치 즉 첫번째, 두번째, k번째, n번째 음절을 의미한다. 그러므로 $C^{i,1}$는 첫음절 초성을, V^1는 첫음절 중성을, $C^{f,1}$는 첫음절 종성을 가리킨다. 음절 복원 rule은 크게 4가지 종류 즉 음절경계종성초성 복원 rule, 끝음절종성 복원 rule, 모음처리 복원 rule, 한음절처리 rule로 나누어진다.

2.2.1 음절경계 종성 초성 복원rule

앞음절 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 83개의 main rule과 221개의 sub path이다. 여기서 sub path란 한 rule에서 "->" 의 오른쪽에 정의된 음소 쌍의 개수로 아래 [Rule001]은 4개의 sub path {ㄱ}과 {ㅇ}, {ㄲ}과 {ㅇ}, {ㄹ}과 {ㅇ}, {ㅋ}과 {ㅇ}을 가진다. 모든 rule의 정의에서 자신으로 변환되는 path는 생략했다.

1. fill-code(무종성)에 관한 복원rule

19개의 main rule과 54개의 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] $C^{f,k}\{(\#)\}{*}C^{i,k+1}\{ㄱ\}$->$C^{f,k}\{ㄱ,ㄱ,ㅅ\}{*}C^{i,k+1}\{ㄱ,ㅇ,ㄱ\}$
:
[Rule008] $C^{f,k}\{(\#)\}{*}C^{i,k+1}\{ㅂ\}$->$C^{f,k}\{ㅂ,ㅃ,ㅍ^*\}$ * $C^{i,k+1}\{ㅇ,ㅇ,ㅇ^*\}$
:
[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] $C^{f,k}\{ㄱ\}{*}C^{i,k+1}\{ㄱ\}$->$C^{f,k}\{ㄱ,ㄲ,ㄺ,ㅋ\}{*}C^{i,k+1}\{ㄱ,ㄱ,ㄱ,ㄱ\}$
[Rule021] $C^{f,k}\{ㄱ\}{*}C^{i,k+1}\{ㄸ\}$->$C^{f,k}\{ㄱ,ㄱ,ㄲ,ㄺ\}{*}C^{i,k+1}\{ㄷ,ㄷ,ㄷ,ㄷ\}$
[Rule022] $C^{f,k}\{ㄱ\}{*}C^{i,k+1}\{ㅃ\}$->$C^{f,k}\{ㄱ,ㄲ,ㄺ\}{*}C^{i,k+1}\{ㅂ,ㅂ,ㅂ\}$
[Rule023] $C^{f,k}\{ㄱ\}{*}C^{i,k+1}\{ㅆ\}$->$C^{f,k}\{ㄱ,ㄲ\}{*}C^{i,k+1}\{ㅅ,ㅇ\}$
[Rule024] $C^{f,k}\{ㄱ\}{*}C^{i,k+1}\{ㅉ\}$->$C^{f,k}\{ㄱ,ㄺ\}{*}C^{i,k+1}\{ㅈ,ㅈ\}$

3. ㄴ-종성에 관한 복원rule

12개의 main rule과 29개의 sub path가 있다.

[Rule026] $C^{f,k}\{ㄴ\}{*}C^{i,k+1}\{ㄱ\}$->$C^{f,k}\{ㄴ\}{*}C^{i,k+1}\{ㄱ\}$
[Rule027] $C^{f,k}\{ㄴ\}{*}C^{i,k+1}\{ㄲ\}$->$C^{f,k}\{ㄴ,ㄳ\}{*}C^{i,k+1}\{ㄱ,ㄱ\}$
:
[Rule036] $C^{f,k}\{ㄴ\}{*}C^{i,k+1}\{ㅋ\}$->$C^{f,k}\{ㄶ\}{*}C^{i,k+1}\{ㄱ\}$
[Rule037] $C^{f,k}\{ㄴ\}{*}C^{i,k+1}\{ㅌ\}$->$C^{f,k}\{ㄶ\}{*}C^{i,k+1}\{ㄷ\}$

4. ㄷ-종성에 관한 복원rule

8개의 main rule과 32개의 sub path가 있다.

[Rule038] $C^{f,k}\{ㄷ\}{*}C^{i,k+1}\{ㄲ\}$->$C^{f,k}\{ㄷ,ㅅ,ㅆ,ㅈ,ㅊ,ㅌ\}$
$*C^{i,k+1}\{ㄱ,ㄱ,ㄱ,ㄱ,ㄱ,ㄱ\}$
[Rule039] $C^{f,k}\{ㄷ\}{*}C^{i,k+1}\{ㄸ\}$->$C^{f,k}\{ㄷ,ㅅ,ㅅ,ㅆ,ㅈ,ㅊ,ㅌ\}$
$*C^{i,k+1}\{ㄷ,ㄷ,ㄸ,ㄷ,ㄷ,ㄷ,ㄷ\}$
:
[Rule044] $C^{f,k}\{ㄷ\}{*}C^{i,k+1}\{ㅊ\}$->$C^{f,k}\{ㅅ\}{*}C^{i,k+1}\{ㅊ\}$
[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 30개의 sub path가 있다.

[Rule046] $C^{f,k}${ㄹ}*$C^{i,k+1}${ㄱ}->$C^{f,k}${ㄺ}*$C^{i,k+1}${ㅇ}
[Rule047] $C^{f,k}${ㄹ}*$C^{i,k+1}${ㄲ}->$C^{f,k}${ㄹ,ㄺ,ㄻ}*$C^{i,k+1}${ㄱ,ㄱ,ㄱ}
:
[Rule060] $C^{f,k}${ㄹ}*$C^{i,k+1}${ㅌ}->$C^{f,k}${ㅀ}*$C^{i,k+1}${ㅇ}
[Rule061] $C^{f,k}${ㄹ}*$C^{i,k+1}${ㅍ}->$C^{f,k}${ㄻ,ㅄ}*$C^{i,k+1}${ㅇ,ㅇ}

6. ㅁ-종성에 관한 복원rule

8개의 main rule과 21개의 sub path가 있다.

[Rule063] $C^{f,k}${ㅁ}*$C^{i,k+1}${ㄱ}->$C^{f,k}${ㄻ,ㅁ}*$C^{i,k+1}${ㄱ,ㄱ}
[Rule064] $C^{f,k}${ㅁ}*$C^{i,k+1}${ㄲ}->$C^{f,k}${ㄻ,ㅁ}*$C^{i,k+1}${ㄱ,ㄱ}
:
[Rule069] $C^{f,k}${ㅁ}*$C^{i,k+1}${ㅇ}->$C^{f,k}${ㅂ}*$C^{i,k+1}${ㅇ}
[Rule070] $C^{f,k}${ㅁ}*$C^{i,k+1}${ㅉ}->$C^{f,k}${ㄻ,ㅁ}*$C^{i,k+1}${ㅈ,ㅈ}

7. ㅂ-종성에 관한 복원rule

7개의 main rule과 22개의 sub path가 있다.

[Rule072] $C^{f,k}${ㅂ}*$C^{i,k+1}${ㄲ}->$C^{f,k}${ㅃ,ㄼ,ㅄ,ㅍ}*$C^{i,k+1}${ㄱ,ㄱ,ㄱ,ㄱ}
[Rule073] $C^{f,k}${ㅂ}*$C^{i,k+1}${ㄴ}->$C^{f,k}${ㅃ}*$C^{i,k+1}${ㄴ}
:
[Rule077] $C^{f,k}${ㅂ}*$C^{i,k+1}${ㅉ}->$C^{f,k}${ㄼ,ㅄ,ㅂ,ㅃ,ㅍ}
*$C^{i,k+1}${ㅈ,ㅈ,ㅈ,ㅈ,ㅈ}
[Rule078] $C^{f,k}${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.

[Rule079] $C^{f,k}${ㅇ}*$C^{i,k+1}${ㄲ}->$C^{f,k}${ㅇ}*$C^{i,k+1}${ㄱ}
[Rule080] $C^{f,k}${ㅇ}*$C^{i,k+1}${ㄴ}->$C^{f,k}${ㄱ,ㄱ,ㄱ,ㄲ,ㄳ,ㅇ,ㅇ}
*$C^{i,k+1}${ㄴ,ㄹ,ㅇ,ㄴ,ㅇ,ㄴ,ㄹ,ㅇ}
:
[Rule085] $C^{f,k}${ㅇ}*$C^{i,k+1}${ㅇ}->$C^{f,k}${ㅇ}*$C^{i,k+1}${ㅇ}
[Rule086] $C^{f,k}${ㅇ} * $C^{i,k+1}${ㅉ}->$C^{f,k}${ㅇ}*$C^{i,k+1}${ㅈ}

2.2.2 모음처리 복원 rule

모음처리 복원 rule은 아래와 같이 4개의 main rule로 정의하였다. 모음의 경우에는 대부분이 [Rule101]과 같이 그대로 복원이 된다. 특별한 경우가 [Rule102], [Rule103], [Rule104]의 경우인데 [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]은 'ㅖ'는 'ㅔ', 'ㅖ', 'ㅢ' 세가지 경우로 복원 될 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 계집[계집], 시계[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', 'ㅢ' 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 발음난 경우의 복원을 나타낸 것이다.

[Rule101] V^{k}{ㅏ,ㅐ,ㅗ,ㅚ,ㅜ,ㅟ,ㅡ,ㅑ,ㅒ,ㅕ,ㅖ,ㅘ,ㅙ,ㅛ,ㅝ,ㅞ,ㅠ,ㅢ}
-> V^{k}{ㅏ,ㅐ,ㅗ,ㅚ,ㅜ,ㅟ,ㅡ,ㅑ,ㅒ,ㅕ,ㅖ,ㅘ,ㅙ,ㅛ,ㅝ,ㅞ,ㅠ,ㅢ}
[Rule102] if $C^{i,k-1}$ = {ㅈ,ㅉ,ㅊ} then V^{k}{ㅓ} -> V^{k}{ㅓ} | V^{k}{ㅕ}
else V^{k}{ㅓ} -> V^{k}{ㅓ}
[Rule103] V^{k}{ㅖ} -> V^{k}{ㅖ} | V^{k}{ㅔ} | V^{k}{ㅢ}
[Rule104] if $C^{i,1}$ = {ㅇ} then V^{k}{ㅣ} -> V^{k}{ㅣ}
else V^{k}{ㅣ} -> V^{k}{ㅣ} | V^{k}{ㅢ}

2.2.3 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,ㅀ}
[Rule204] $C^{f,k}${ㅁ} -> $C^{f,k}${ㄻ}
[Rule205] $C^{f,k}${ㅂ} -> $C^{f,k}${ㅄ,ㅍ}

2.2.4 한음절 처리 rule

한음절 처리 rule은 한 어절이 한음절인 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

Rule을 적용할 때 올바른 음절만을 생성하도록 불필요한 복원 과정(path)을 미리 자르기 위하여 x-clustering 정보를 x 받침을 가질 수 있는 {초성, 중성}쌍들의 집합으로 정의하여 사용하였다. 28가지 종류로 나누어 표 1에 일부만 보인다.

표 1. x-clustering의 종류

종류(갯수)	{초성,중성}쌍들의 집합
ㄱ-clustering(201)	{가개갸거겨고과괴구궈귀그 ..... 홰회후훼휘휴호히}
ㄲ-clustering(13)	{겨꺼껴나다더바무보서소여파}
ㄳ-clustering(6)	{너모바사써싸}
:	:
ㅂ-clustering(207)	{가개거게겨계고과괘괴교구귀 ..... 표푸프하해허헤혀혜호회효화호회히}
ㅄ-clustering(3)	{가어여}
:	:
ㅍ-clustering(20)	{가거기노느니더라르부서수시아어여으이지토}
ㅎ-clustering(20)	{가거나너오다따떠라러마머무빠싸야여으조쪄}

2.4 postfix 음절 빈도 정보

규칙에 의해 복원된 후보는 한개부터 여러개까지 나타났다. 후보들을 조사해 보니 한음절 한음절은 문자로 만들어질 수는 있지만 그 음절들이 모여서는 의미를 갖지 못하는 후보가 많이 나타났다. 예를 들어 음성문자열 [하나뿌니야]의 복원후보들은 '한아뿌니야', '하낫부니야', '한아부니야', '한앗부니아', '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 입력후보수를 제한하는 것이 필요하다. 그래서 단어로 생성될 수 없는 후보를 미리 제한하기 위한 방안으로 임의의 한 음절 뒤에 바로 나타날 수 있는 문자의 빈도값을 말뭉치로부터 구해서 이용한다. 말뭉치로 사용된 것은 신문사설, 초등학교 교과서, 소설로 약 60만 어절이다. 빈도값은 상용조합 2350자를 기준으로 하여 말뭉치로부터 획득하였는데 2350자중 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 일부이다. 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 빈도 정보

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 비 문법적 후보 어절을 필터링 해준다. 예를들어 "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 [13]을 이용한다.

3. 처리 알고리즘 및 적용예

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

```
Algoritm  음절_복원_rule()
/* 입력 : buffer[0..n]
          --> 3바이트 조합형 코드로 저장된 배열
   출력 : 복원 rule 적용 결과 후보들의 list */
c = n;
begin
  if  c == 2   then  call  한_음절_처리_rtn()
     else  for ( i = 0; i <= c ; i++ )
             if (i == 0)  then result[0] = buffer[0]
             else if ( 1 == i%3 )  then
                  call 모음_처리_rtn()
             else if ( 2 == i%3)  then
                  call  음절_경계_종성_초성_처리_rtn()
             else if  ( i == n )
                  call 끝_음절_종성_처리_rtn();
             call  postfix_음절_check();
  call  morphem_analysis(buffer, result);
  end /* begin-end */
```

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) $C^{f,1}*C^{i,2} \in C^{f,k}\{\#\}*C^{i,k+1}\{ㅂ\}$->$C^{f,k}\{ㅂ,ㅶ,ㅍ\}*C^{i,k+1}\{ㅇ,ㅇ,ㅇ\}$

(a.2) $V^1, V^2 \in V^k\{ㅏ,ㅐ,ㅗ,ㅚ,ㅜ,ㅟ,ㅡ,ㅑ,ㅒ,ㅕ,ㅖ,ㅘ,ㅙ,ㅛ,ㅝ,ㅞ,ㅠ,ㅢ,ㅣ\}$->$V^k\{ㅏ,ㅐ,ㅗ,ㅚ,ㅜ,ㅟ,ㅡ,ㅑ,ㅒ,ㅕ,ㅖ,ㅘ,ㅙ,ㅛ,ㅝ,ㅞ,ㅠ,ㅢ,ㅣ\}$

(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)

ㄾ -> 바븗(x)

ㅀ -> 바븕(x)

(a.2)|(a.3)

-> path1> ㅂ+ㅇ -> 밥을(o) -> self -> **밥을(o)**

ㄾ -> 밥읇(x)

ㅀ -> 밥읋(x)

-> path2> ㅶ+ㅇ -> 밦을(x)

-> path3> ㅍ+ㅇ -> 뱦을(x)

(a.2)|(a.1)는 (a.2)를 먼저 적용하고 (a.1)을 적용함을 의미하고, 밑줄이 있는 부분은 각각 ㄾ-clustering, ㅀ-clustering, ㅶ-clustering, ㅍ-clustering을 참조하므로 생성되지 않는 부분이다. 그러나 이해를 돕기 위해 기술하였다. 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 -> 꾺(x)

-> path2> ㄳ -> 꾺(x)

-> path3> ㄺ -> 꿁(x)

-> path4> ㅋ -> 꿀(x)

(b)는 한음절로 이루어진 어절로 종성 "ㄱ"이 있으므로 이에 대한 rule (b.1)이 적용될때 각각의 clustering 정보를 참조하면 "꼭"만 결과로 생성된다. 이를 형태소분석하면 "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 $V^k\{ㅓ\}$->
$V^k\{ㅓ\}$ | $V^k\{ㅕ\}$ 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) $V^3 \in V^k\{ㅏ,ㅐ,ㅗ,ㅚ,ㅜ,ㅟ,ㅡ,ㅑ,ㅒ,ㅕ,ㅖ,ㅘ,ㅙ,ㅛ,ㅝ,ㅞ,ㅠ,ㅢ,ㅣ\}$
->$V^k\{ㅏ,ㅐ,ㅗ,ㅚ,ㅜ,ㅟ,ㅡ,ㅑ,ㅒ,ㅕ,ㅖ,ㅘ,ㅙ,ㅛ,ㅝ,ㅞ,ㅠ,ㅢ,ㅣ\}$

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

-> self -> 머거 -> self -> 머거라(o)

-> path1> ㄹ+ㅇ -> 머^걸아(o)

-> path2> ㄾ+ㅇ -> 머걿아(x)

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

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

-> path1> ㄹ+ㅇ-> 먹을아(o)

-> path2> ㄾ+ㅇ-> 머읋아(x)

-> path2> ㄳ+ㅇ -> 먻어(x)

-> path3> ㄺ+ㅇ -> 멁어(x)

-> path4> ㅋ+ㅇ -> 멐어(x)

(c)의 경우에도 4개의 후보에서 "머^걸아"는 postfix음절 빈도정보 이용으로 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 유용한 문자열은 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.

4. 결과 분석

4.1 Rule 적용 횟수 분석

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

1) $1 < k < n$ 일 때, $C^{f,k}*C^{i,k+1}$에 대해서는 음절경계종성초성 복원rule을 적용한다.

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

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

4) $k = n = 1$ 일 때, $C^{i,k}V^kC^{f,k}$ 에 대해서는 한 음절 처리 rule을 적용한다.

이상과 같은 조건에 맞게 복원 rule들이 적용되는데, 한 어절에 대해 rule 적용횟수를 한 번 분석해 보자. 먼저 한 어절이 한 음절로 구성되어 있다면 1회의 한 음절 처리루틴으로 수행이 끝난다. 그리고 한 음절이 n개의 음절로 구성되어 있고, 한 Rule내에 c개의 가능한 path가 존재한다면 $c*(n-1)$회의 음절경계종성초성 복원rule 적용과 n번의 모음처리rule 적용 그리고 한 어절의 마지막 음절의 종성이 있다면 c번의 끝음절 복원 rule적용으로 모든 수행이 끝난다. 그러므로 한 어절에 대한 rule 적용횟수를 계산해 보면 다음과 같다.

1) n=1 일 때, 한 어절에 대한 수행시간 = 1회의

한 음절처리 rule 적용

2) n>1 일 때, 한 어절에 대한 수행시간 = c*(n-1)회의 음절경계종성초성 복원rule+n회 모음처리rule 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 한문장의 복원시간은 O(nm)에 수행이 완료된다. 여기에 양방향 최장일치 형태소 분석[13] 시간 O(n)을 합하면 전체 수행시간이 된다.

4.2 실험 결과 분석

본 논문에서는 연속 음성 문자열 14,930어절에 대해 실험을 하였다. 실험은 두가지 방법으로 행해 보았다. 먼저 복원 규칙만 적용해 결과를 생성한 방법을 A라 하고, 복원규칙 적용후 그 결과를 형태소 분석한 방법을 B라 하였다. A와 B에서 얻어진 결과로 어절의 후보수를 비교 분석해 보고, 후보수 전이율을 분석해 보았다. 그 결과는 각각 표2, 표3과 같다.

표 2. A와 B 후보수 비교

후보수 / 분석방법	1개	2개	3개	4개	5개	6개
복원rule적용(A)	53.6%	29.9%	9.3%	4.1%	2.1%	1%
형태소분석(B)	83.16%	12.37%	3.44%	1.03%	0%	0%

표 3. 후보수 전이율 분석표

방법A	방법B	백분율	방법A	방법B	백분율
2	1	62.89%	5	1	5.15%
3	1	11.34%	5	2	1.03
3	2	8.25%	6	1	1.03
4	1	6.19%	6	2	2.06
4	2	2.06%			

이상의 결과를 보면, 음운변화현상이 반영된 음성문자열을 복원시켜서 이를 자연어처리 기법인 형태소 분석을 행하여 줌으로 결과가 단일 후보로 생성될 확률이 30% 이상 높아졌다는 것을 알 수 있다. 그리고 후보수 전이 상태는 형태소 분석을 하므로 후보수 2개가 1개로 되는 경우가 62.89%로 가장 높았다. 실험에서 모든 어절중 83.16%는 단일 후보를 생성했지만 16.8%는 복수개 후보가 생성되었다. 복수개 생성된 후보들 중에는 원하는 text 문자열은 포함하고 있지만 문맥에 맞는 정확한 한 후보를 찾아 주는 것은 구문분석, 의미분석이 이루어져야 하므로 향후 과제로 남기고 복수개 생성된 이유만 분석해 보았다. 복원후보가 복수개 나타난 주요인은 다음과 같은 세 가지 이유였다. 첫 번째 이유는 실제 형태소의 자소가 다르지만 음운현상에 의해 동일한 음가로 생성되는 동음이형어로 생긴 경우로 표4와 같은 예들이다.

표 4. 동음이형어의 모호성

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

두 번째 이유는 모음의 발음에서 "ㅖ", "ㅞ", "ㅢ"가 모두 "ㅔ"로 발음날 수 있고, "ㅣ"와 "ㅢ" 둘 다 "ㅣ"로 발음 날 수 있으므로 이를 역으로 복원하면서 후보가 복수개 발생한 경우로 표5와 같다.

표 5. 모음 발음의 모호성

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

세 번째 이유는 rule의 정의에서 발음법의 예외 부분들을 규칙으로 일반화시킴으로 부수적으로 생성된 후보들이다. 예를들면 음성문자열 "인는"을 복원한 결과 {인은|있는|잇는|읮는} 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 일반화시킴으로 합성어가 아닌 "인는"의 경우에 영향을 미친 것이다. 이처럼 세가지 이유로 복원 결과가 복수개 나타났다. 첫 번째, 두번째 이유는 생길 수밖에 없는 문제이며 동시에 해결 방안이 강구되어야 하는 주요 연구과제이고, 세 번째 경우는 규칙을 좀 더 정비하므로 해결될 수 있는 문제라 생각한다. 이 실험의 결과로 연속음성 인식을 위해서 우선적으로 음운변화 현상에 대한 복원이 이루어져야 하고, 그 다음은 복원결과 문자열을 형태소 분석하고, 다음은 형태소 분석된 어절들을 대상으로 1개에서 N개의 문장을 생성해 이를 구문분석, 의미분석까지 되어야 완전한 연속음성 인식시스템이라고 하겠다.

5. 결론

한국어가 연속적으로 발음될 때 여러 가지 음운변동현상이 일어나는데 이러한 음운변동현상은 한국어 음성 인식을 어렵게 하는 주요 요인이 된다. 본 논문에서는 이러한 음운변동현상이 반영된 음성 인식 문자열을 text 기반 문자열로 다시 복원시키는 Rule_Base시스템을 제안했다. 이 시스템은 기존의 시스템들과 달리 대량의 발음열 사전이나 음소열 사

전이 필요 없는 규칙기반시스템이다. 이 시스템에 정의된 rule은 음절경계 종성 초성 복원rule로 83개 main rule과 221개 sub path, 모음처리 복원rule로 4개 main rule과 25개 sub path, 끝음절 종성 복원 rule로 5개의 main rule과 13개의 sub path를 정의했다. 한음절처리 rule은 종성이 있을때만 끝음절 종성 복원rule을 함께 사용한다. 그리고 효율적인 복원을 위해 x-clustering정보를 정의하여 사용하고, 형태소 분석기에 입력될 복원 후보 수를 제한하기 위해 postfix음절 빈도정보를 구하여 사용했는데 이 정보들은 복원 중에 유용하게 사용되었다. 그리고 복원결과는 형태소 분석되어 유용한 문자열만 최종결과로 생성되는데, 실험 대상 중 83.16%가 단일 text문자열로 생성되었고, 16.84%는 IV.2에 언급한 세가지 이유로 결과가 다수개 복원 후보가 나타났다. 16.8%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 인식결과를 신속히 문서화 하는 응용분야에 기초 시스템으로 유용하게 사용될 수 있으리라고 생각한다. 향후 방향은 형태소 분석단계에서 생성된 결과중 후보가 복수개 생성된 경우에 이를 구문 분석, 의미 분석을 하여 의미있는 하나의 문장을 생성해내는 연구를 계속하는 것이다.

참고문헌

[1] 이호영, 국어음성학, 태학사, 1996.

[2] 문화체육부, 국어 어문 규정집, 대한교과서주식회사, 1997.

[3] 이계영, "한국어 음운변동 처리 규칙의 설계 및 구현", 한국어정보처리학회 논문지, 제5권 제 3호, p851-861, 1998년.

[4] 김병수, "음절단위를 이용한 한국어 음성합성에 관한 연구", 한양대학교, 석사학위논문, 1989.

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

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

[7] 정민화, "한국어 연속음성인식을 위한 자연언어처리 기술의 적용방법", '98 지능기술 튜토리얼, pp27-55, 1998.

[8] 김병창, 이원일, 이근배, 이종혁, 이영직, "형태소 그래프를 이용한 한국어 연속음성인식과 형태소분석의 통합", 한국정보과학회 가을 학술발표논문집, pp549-552, 1996.

[9] 서상현, "한글 음운 규칙에 기반한 음절 복원기 구현", 경북대학교, 석사학위논문, 1997.

[10] 이원일, "신경망과 CYK-table을 이용한 음성 언어의 분석", 포항공대, 석사학위논문, 1993.

[11] 전재훈, 차선희, 정민화, "형태음운론적 분석에 기반한 한국어 발음 생성", 한국어 정보과학회 가을 학술발표논문집, pp247-250, 1997.

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

[13] 최재혁, "양방향 최장일치법에 의한 한국어 형태소 분석기의 구현", 경북대학교, 박사학위논문, 1993.