음성합성 플랫폼을 위한 언어처리부의 설계 및 구현

이 상 호
(주)첫눈 정보마이닝팀

Design and Implementation of the Language Processor for Educational TTS Platform

Sangho Lee
Information Mining Team, 1noon.com
sangholee@1nooncorp.com

요약

본 논문에서는 한국어 TTS 시스템을 위한 언어처리부의 설계 및 구현 과정을 설명한다. 구현된 언어처리부는 형태소 분석, 품사 태깅, 발음 변환 과정을 거쳐, 주어진 문장의 가장 적절한 발음열과 각 음소의 해당 품사를 출력한다. 프로그램은 표준 C언어로 구현되어 있고, Windows와 Linux에서 모두 동작되는 것을 확인하였다. 수동으로 품사가 할당된 4.5만 어절의 코퍼스로부터 형태소 사전을 구축하였으며, 모든 단어가 사전에 등록되어 있다고 가정할 경우, 488 문장의 실험 자료에 대해 어절 단위 오류율이 3.25%이었다.

1 서론

현재 국내 대학에서 음성합성분야의 교육 및 연구를 새로이 시작하고자 할 경우, 참고로 할 수 있는 한국어 TTS 프로그램과 데이타, 프로그램과 관련된 상세한 문서, 원시코드가 있다면 큰 도움이 될 것이다. 그러나 기존에 공개된 한국어 음성합성기 플랫폼이 거의 없는 실정이며 이를 접할 수 있다해도 사용자 API만을 제공하므로 독자적인 알고리즘을 이식할 수 있는 방법이 없어서 알고리즘에 대한 검증이 불가능하다. 그러므로 TTS와 관련된 특정 알고리즘만 개발, 검증하고자 할 경우라도 TTS에 필요한 모든 프로그램과 데이터를 개발해야만 되는 실정이다.

그러나 이를 위한 작업량이 방대하므로 음성합성 연구개발 분야의 신규 진입에 큰 장애가 되고 있다. 특히 TTS 시스템을 연구하는 학생들에게 있어서, TTS 시스템을 구성하는 모듈 중 특정 모듈에 대한 연구를 하고 싶어도, 다른 나머지 모듈들의 준비도 함께해야 한다는 어려움이 있다. 이에, 본 논문에서는 학생들이 합성기 개발에 쉽게 다가갈 수 있도록, TTS 구성 모듈 중 하나인 언어처리부를 개발하고 이의 소스코드를 공개하고자 한다.

본 논문의 구성은 다음과 같다. 다음 장에서는 언어처리부의 구성과 각 모듈들의 알고리즘을 설명한다. 3장에서는 구현된 시스템의 성능에 대해서 설명하고, 4장에서는 시스템의 사용 방법에 대해 간략히 소개한다. 끝으로 5장에서 결론을 맺는다.

2 언어처리부의 구성

본 논문에서 구현한 언어처리부는 크게 형태소 분석 모듈, 품사 태깅 모듈, 발음열 생성 모듈로 이루어져 있으며, 각 모듈들은 다음 절부터 차례로 자세히 설명되어 있다.

2.1 형태소 분석

형태소 분석 모듈은 입력 어절들의 모든 가능한 형태소 분석 결과를 구한다. 하나의 형태소 분석 결과는 그림 1처럼 형태소 격자로 이루어져 있으며, 격자의 가장 왼쪽에 있는 INI에서 가장 오른쪽에 있는 FIN까지의 가능한 경로가 그 어절의 해당 형태소 분석 결과가 된다. 예를 들어, "나는"이라는 어절을 보면, 그림에서 총 4개의 경로가 가능하므로, 4개의 가능한 형태소 분석 후보들을 얻을 수 있다.

한편, 형태소 분석 모듈에서 구한 결과는 다음 모듈인 품사 태깅 모듈의 입력으로 생각할 수 있는데, 현재 구현된 시스템에서는 형태소 분석 모듈과 품사 태깅 모듈이 서로 독립적이지 않고, 태깅 모듈이 형태소 분석 모듈을 포함하는 관계로 이루어져있다.

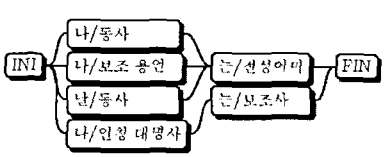

그림 1: "나는"의 형태소 격자

구현된 형태소 분석 모듈은 입력 어절을 한글 알파벳 코드로 변환한후, 입력 어절의 불규칙, 축약 현상 위치를 미리 추정한다. 본 시스템에서 사용하는 한글 알파벳 코드의 특징은 초성 'ㅇ'에 대한 코드가 존재하지 않고, 이중 모음과 복자음 종성에 대해 두개의 알파벳을 이용한다. 이와 같은 표기법은 불규칙 용언을 포함하는 어절을 분석할 때 효과적이다. 예를 들어, "도와"라는 어절은 "dowa"로 표기될 수 있으며, 형태소 분석 결과는 "돕/용언+아/어미"이

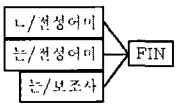

그림 2: "흡로그래피는"의 분석 실패된 격자

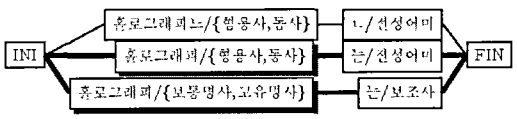

그림 3: "흡로그래피는"의 복구된 격자

다. 이는 "dowa"의 'w'를 'B'으로 바꾸고 "doB"을 사전에서 참조하여 원형을 유추할 수 있다.

본 분석기는 주어진 어절의 불규칙 혹은 축약 위치를 추정할 수 있도록, 약 500개의 음소 패턴을 저장하고 있다. 이 음소 패턴을 이용하여 추정된 위치에서의 음운 현상들을 복원하게 된다. 한편, 형태소 분석 도중, 완전한 형태의 형태소 격자를 얻지 못할 경우에는 어절의 앞부분에 사전에 등록되지 않은 단어가 사용되었다고 판단하고, 그 시점까지 구축된 형태소 격자를 바탕으로 미등록어를 추정하게 된다.

2.2 미등록어 추정

본 논문에서 구현된 형태소 분석 모듈은 사전에 등록되어 있지 않은 형태소가 어절에 존재할 때, 그 형태소를 추정하는 미등록어 추정 모듈을 포함하고 있다. 예를 들어 다음의 문장을 분석한다고 하자.

"삼차원 영상인 흡로그래피는 구현이 어렵다."

위의 문장에서 "흡로그래피"가 사전에 등록되어 있지 않다고 가정하면, "흡로그래피는"에 대한 형태소 격자는 그림 2에서처럼 단지 "는/{전성어미, 보조사}"와 "ㄴ/전성어미"의 정보만 얻을 수 있고 하나의 경로도 존재하지 않는다. 이 때, 미등록어 추정 모듈은 형태소 분석 모듈이 남긴 부분 격자를 가지고 품사 접속표를 이용하여 추정을 하게 되는데, 예를 들면, "는/전성어미" 앞에 올 수 있는 품사는 형용사와 동사이므로 "흡로그래피"는 이 두 가지 품사 중 하나일 가능성이 있는 것이다. 본 시스템에서는 가능한 미등록어로 동작성 보통명사 (na), 상태성 보통명사 (ns), 보통 명사 (nc), 고유 명사 (nr), 동사 (vb), 형용사 (vj)를 가정하고 있다. 한편, 부분 격자로부터 어절의 좌측에 있게 될 미등록어를 추정하다가 모든 추정 후보들이 품사 접속표에 의해 추정이 안될 경우에는 전체 어절을 고유 명사로 가정한다.

이와 같은 방법으로 어절 "흡로그래피는"에 대해 미등록어를 추정하게 되면 그림 3에서 보듯이 등록 어절의 형태소 격자와 전혀 다름이 없고 단지 경로가 많다는 특징이 있을 뿐이다. 그러나, 미등록어를 포함하고 있는 형태소 격자는 정상적인 격자보다 많은 경로를 가지고 있고, 이는 곧 격자가 크다는 것을 뜻하므로 결과적으로 언어 처리 모듈의 성능을 저하시키는 요인이 된다.

개발된 언어 처리 모듈은 시스템의 성능을 높이기 위해 두 가지 휴리스틱을 이용하여 미등록어 후보의 개수를 줄

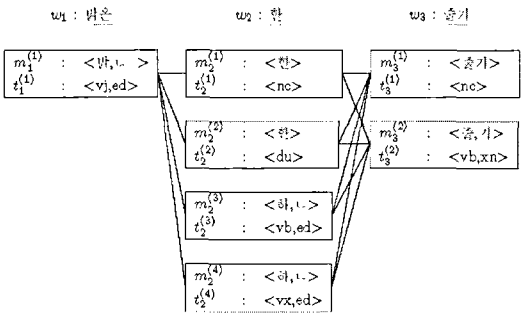

그림 4: "밝은 한 줄기"의 어절 단위 형태소 분석 격자

이다. 먼저 그림 3의 경우 음절 '느'는 형용사와 동사의 마지막 음절로 절대 사용되지 않는다고 가정하여 결과적으로 "흡로그래피느/{형용사, 동사}" 노드를 제거한 최종 4개의 경로만을 가지는 격자를 생성한다. 형태소들의 마지막 음소 패턴으로 사용될 수 없는 것들로는 "께", "은", "을", "었", "앴" "으"이다. 즉 임의의 미등록어 접미부가 앞의 6개 음소 패턴일 경우 미등록어 후보에서 제거한다.

이외에도 후보를 여과하는 두번째 방법으로 단서 (clue) 형태소를 이용하는 방법이 있다. 이는 미등록어들이 추정된 격자내에 그 어절의 구성을 추정하기에 충분한 단서 형태소가 발견되면 그 형태소를 지나지 않는 경로를 격자에서 모두 제거하는 방법이다. 사용된 단서 형태소로는 "하", "되"와 같은 동사 파생접미사와 "하", "답", "스럽"과 같은 형용사 파생접미사들이다.

2.3 품사 태깅

품사 태깅 모듈은 형태소 분석 모듈의 결과를 바탕으로 하나의 어절당 최적의 형태소 분석 결과를 찾는다. 품사 태깅 모듈이 필요한 이유는 그 다음 모듈인 발음열 생성 모듈에서 유용하게 사용되기 때문이다.

품사 태깅을 하는 방법은 하나의 형태소 분석 결과들을 노드로 표현하고, 형태소 분석 결과들 사이에 에지(edge)를 두어 모든 가능한 경로중 최적의 경로를 찾는 문제로 바꾸어 해결하는 것이 일반적이다. 예를 들어, "밝은 한 줄기"라는 문장에 대해 그림 4의 격자를 만들고, 하나의 경로는 전체 문장에 대한 품사 태깅 결과가 되므로, 이 중 최적의 경로를 찾으면 된다. 이 그림에서 w_i는 i번째 어절을 뜻하고, m_i, t_i는 각각 i번째 형태소 열과 품사 열을 뜻한다. 다시 말해, 품사 태깅이란 i번째 어절에 대해 m_i, t_i를 선택하는 문제로 볼 수 있다.

본 논문에서 사용하는 확률 품사 태깅 방법은 다음과 같다. 주어진 어절열 $w_{1..n}$에 대해 최적의 $(m_{1..n}, t_{1..n})$을 찾는 것이 품사 태깅을 하는 과정이며, 여기서 n은 주어진 문장의 어절 개수를 의미하고, (m_i, t_i)는 i번째 어절의 형태소 분석 결과를 의미한다.

$$\phi : w_{1..n} \rightarrow (m_{1..n}, t_{1..n}) \quad (1)$$

식 1을 만족하는 $\phi(w_{1..n})$은 다음과 같이 진개가 가능하다.

$$\phi(w_{1..n}) \stackrel{\text{def}}{=} \arg \max_{m_{1..n}, t_{1..n}} P(m_{1..n}, t_{1..n} | w_{1..n}) \quad (2)$$

$$\cong \arg \max_{m_{1..n}, t_{1..n}} \prod_{i=1}^{n} P(m_i|t_i)P(t_i|t_{i-1}) \quad (3)$$

식 3의 $P(t_i|t_{i-1})$에서 t_i는 하나의 어절을 이루는 형태소 태그 열이다. 그러므로 이것은 다음과 같이 전개될 수 있다.

$$P(t_i|t_{i-1}) \stackrel{\text{def}}{=} P(t_{i^1}..t_{iN_i}|t_{i-1^1}..t_{i-1N_{i-1}}) \quad (4)$$

$$\cong P(t_{i^1}..t_{iN_i}|t_{i-1N_{i-1}}) \quad (5)$$

$$= P(t_{i^1}|t_{i-1N_{i-1}})P(t_{i^2}|t_{i^1}t_{i-1N_{i-1}}) \cdots P(t_{iN_i}|t_{iN_i-1}..t_{i^1}t_{i-1N_{i-1}}) \quad (6)$$

$$\cong P(t_{i^1}|t_{i-1N_{i-1}}) \prod_{j=2}^{N_i} P(t_{ij}|t_{ij-1}) \quad (7)$$

식 4에서 현재 처리 중인 어절의 형태소 태그 열은 이전 어절의 마지막 형태소 태그에만 의존한다는 가정에 의해 식 5를 얻었고, 연쇄 규칙(chain rule)에 의해 식 6으로 전개되며, 1차 마르코프 가정을 적용하여 식 7을 얻었다.

식 7은 이전 어절의 마지막 형태소 태그에서 현재 처리 중인 어절의 첫 형태소 태그로 천이하는 확률과 어절 내에서의 형태소 태그 천이 확률들을 곱한 것이다. 한편, $P(m_i|t_i)$는 형태소 태그 열이 주어졌을 때 형태소 열이 발생할 확률을 의미하는데, 식 11와 같이 전개될 수 있다.

$$P(m_i|t_i) \stackrel{\text{def}}{=} P(m_{i^1}..m_{iN_i}|t_{i^1}..t_{iN_i}) \quad (8)$$

$$= P(m_{i^1}..m_{iN_i}|t_{i^1}..t_{iN_i}) \quad (9)$$

$$= P(m_{i^1}|t_{i^1}..t_{iN})P(m_{i^2}|t_{i^1}..t_{iN_i}m_{i^1}) \cdots P(m_{iN_i}|t_{i^1}..t_{iN_i}m_{i^1}..m_{iN_i-1}) \quad (10)$$

$$\cong P(m_{i^1}|t_{i^1}) \prod_{j=2}^{N_i} P(m_{ij}|t_{ij}) \quad (11)$$

위 식에서 i번째 어절의 가장 앞에 오는 형태소 m_{i^1}의 발생 확률 $P(m_{i^1}|t_{i^1})$을 나머지 식과 다르게 둔 이유는 품사 태깅시에 미등록어에 대한 고려를 하기 위해서이다. 즉, 앞 절에서 가정한 것처럼 입력 어절에서 미등록어는 항상 어절의 가장 앞에 오게 된다. 형태소 m_{i^1}이 미등록어일 경우 발생 확률을 구할 수 없게 되는데 이 때 그 값을 1.0으로 가정한다.

지금까지 설명한 품사 태깅 방법은 어절 단위의 품사 태깅 방법으로, 임의의 어절에서 가능한 모든 품사열을 하나의 단위로 보았다. 이러한 방법은 실제 프로그램으로 구현할 때 프로그램의 효율성이 떨어지므로, 본 연구에서는 형태소 단위의 품사 태깅을 시도하였다. 다시 말해, 2.2 절에서 구한 형태소 격자들을 그대로 연결하여 전체 문장의 형태소 수준 격자를 만들고, 그 위에서 품사 태깅을 시도하였다. 앞 절의 품사 태깅 수식에서 $P(m_{ij}|t_{ij})$는 형태소 격자의 노드에 할당하고, $P(t_{ij}|t_{ij-1})$은 간선 (edge)에 할당하여 최적 경로를 찾는다. 이와 같이 구현하면, 형태소 분석과 품사 태깅이 바로 하나의 틀에서 모두 이루어지므로 프로그램 속도 향상을 기대할 수 있다. 그림 4의 어절 단위 격자를 형태소 단위의 격자로 만들면 그림 5와 같이 되고, 실제 프로그램은 형태소 단위의 격자 위에서 최적의 경로를 찾는다.

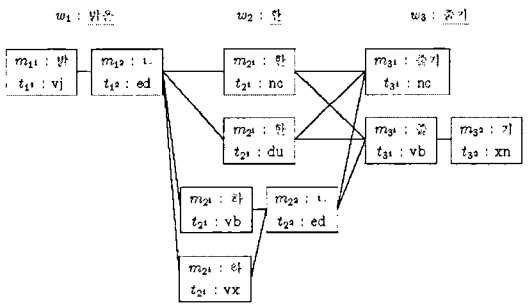

그림 5: "밝은 한 줄기"의 형태소 단위 형태소 분석 격자

2.4 발음열 생성

발음열 생성 모듈은 품사 태깅 모듈에서 구한 어절의 품사열을 이용하여 그 어절의 발음열을 구한다. 발음열을 구한기 위해 어절의 품사 열이 필요한 이유는 한국어의 발음 규칙이 품사에 의존하기 때문이다. 예를 들면, "신과장은 신을 신고 신고하러 갔습니다."의 발음열는 "신과장은 시늘 신꼬 신고하러 갇씀니다."인데, 동일한 "신고"에 대해 전자의 경우는 "신꼬"로, 후자의 경우는 "신고"로 발음되는 것을 볼 수 있다. 이는 전자의 "신고"는 품사 태깅 결과가 "신/용언+고/어미"로, 연결어미인 '고'때문에 경음화 현상이 발생되기 때문이다. 물론 후자의 '신고'는 단순히 체언이므로 아무런 현상도 발생하지 않는다.

이와 같은 처리를 위해서 본 언어 처리 모듈에서 발음열 생성 모듈은 입력 어절의 각 자소에 품사를 할당한 후, 발음 변환 규칙을 적용하여, 그 문장에 맞는 정확한 발음을 찾는다. 자소에 품사를 할당하는 과정은 형태소 분석 단계에서 자연스럽게 얻을 수 있는 것이며, 불규칙이나 축약 현상이 있는 위치에서는 초성과 중성이 서로 다른 품사를 가질 수 있다. 불규칙의 경우를 예로 들면, "고마와"의 경우, 형태소 분석 결과는 "고맙/용언+아/어미"이고 발음은 "고마와"이다. 이 경우 "고마"를 이루는 'ㄱ', 'ㅗ', 'ㅁ', 'ㅏ'는 용언의 품사를 가지며 '아'는 어미의 품사를 가지게 된다. 축약의 대표적인 예인 "어떻게 해"에서 '해'의 경우, 형태소 분석 결과는 "하/용언+어/어미"이고 발음은 '해'이다. 이 경우 'ㅎ'은 용언의 품사를 가지며 'ㅐ'는 어미의 품사를 가진다. 이러한 결과가 나오는 이유는 형태소 분석 모듈에서 사용된 불규칙/축약 현상의 처리 방법 특징 때문이다.

구현된 발음열 생성 모듈은 문교부 고시, 제 88-2호 표준어 규정집에 나와 있는 "표준 발음법"을 그대로 따른 것으로, 총 30항이 있는데, 그 중 1항부터 28항까지의 내용을 규칙에 의해 변환하고, 28항부터 30항까지는 예외어 발음 사전을 이용하여 처리한다.

3 실험 및 성능

구현된 언어처리부는 수동으로 품사가 할당된 45,854 어절의 코퍼스로부터 7,725개의 형태소를 추출하여 사용하고 있다. 이외에 약 1만개의 체언 사전을 추가하였으며, 1,600개의 예외 발음 단어를 포함시켰다. 구현된 시스템의 성능을 알아보기 위해, 488문장 (4729 어절, 9980 형태소)에 대해 품사 태깅 실험을 하였다. 그 결과, 어절 단위 오류율이

표 1: 발음 변환 예제

입력문장	나는 신을 신고 신고하러 간다
형태소열	나/np+는/pt 신/nc+을/po 신/vb+고/ec 신고/na+하/xv+러/ec 가/vb+ㄴ다/ef
발음열	나는 신을 신꼬 신고하러 간다 (nanXN sinXL siNqo siNgohalv gaNda)
입력문장	나는 감기에 걸렸다
형태소열	나/np+는/pt 감기/nc+에/pa 걸리/vb+었/ep+다/ef
발음열	나는 감기에 걸렸다 (nanXN gaMgie gvLlVDfa)
입력문장	머리를 감기에 힘들다
형태소열	머리/nc+를/po 감/vb+기/en+에/pa 힘들/vb+다/ef
발음열	머리를 감끼에 힘들다 (mvlilXL gaMqie hiMdXLda)

표 2: 한국어 음소 알파벳 코드 표

초성	b	ㅂ	d	ㄷ	g	ㄱ	j	ㅈ
	r	ㅃ	f	ㄸ	q	ㄲ	z	ㅉ
	p	ㅍ	t	ㅌ	k	ㅋ	c	ㅊ
	s	ㅅ	x	ㅆ	h	ㅎ	l	ㄹ
	n	ㄴ	m	ㅁ				
중성	a	아	v	어	o	오	u	우
	X	으	i	이	e	에	R	애
	A	아	V	어	O	요	U	유
	E	예	y	외	w	와	W	워
	Y	위	I	의				
종성	G	ㅇ	N	ㄴ	D	ㄷ	L	ㄹ
	M	ㅁ	B	ㅂ	@	ㅇ		

18%이었으며 형태소 단위 오류율이 14%로, 좋지 않은 성능을 나타내었다.

이는 대부분 미등록어의 추정이 어려웠기 때문이었다. 즉, "럭비공/명사"이라는 미등록어에 대해, 사전에 "공/명사"만 등록되어 있는 경우, 형태소 분석 길과는 "럭비/명사+공/명사"가 되며, 이러한 경우에도 평가과정에서 오류로 간주하기 때문이었다. 그리하여 모든 미등록어를 사전에 등록시킨 후, 다시 실험해 보았으며, 그 결과 어절 단위 3.25%, 형태소 단위 1.76%의 낮은 오류율을 보였다.

지금까지 설명한 언어처리부를 가지고 테스트 문장들의 분석 결과를 표 1에 보인다. 표에서 보는 바와 같이, 품사에 의해 발음이 결정되는 경우에 대해 올바른 발음을 구하는 것을 알 수 있으며, 시스템에서 정의한 음소 기호는 표 2와 같다.

4 사용 방법

구현된 시스템은 Windows와 Linux에서 모두 동작하며, 두 경우 모두 라이브러리의 형태로 사용자가 쉽게 이용할 수 있게 하였다. 라이브러리를 사용하는데 필요한 파일은 `libkts.a`와 `ktslib.h`이며 `ktslib.h`의 내용은 다음과 같다.

```
#include "ktsds.h"

extern int      LoadDictionary(char *str) ;
extern void     FreeDictionary()          ;
extern ErrCode SentenceAnalysis (char *inputStr,
                Sentence *sent, AnalysisMode aMode) ;
extern ErrCode MemoryAllocSentence(Sentence *sent) ;
extern void MemoryFreeSentence (Sentence *sent) ;
```

위 내용과 같이, 5개의 함수만을 이용해서 언어처리기를 구동시킬 수 있다. 위 함수들을 이용하여 "나는 학교에 간다."라는 문장을 처리하는 프로그램은 다음과 같다. 아래 코드에서 PrintMorphemeSequence()와 PrintPhonemeSequence()는 문장 분석 결과를 가지고 있는 자료구조의 내용을 출력하는 함수로, 라이브러리와 함께 제공되고 있다.

```
#include <stdio.h>
#include "ktslib.h"
void main() {
  char     *example = "나는 학교에 간다." ;
  Sentence  sent    ;
  ErrCode   errcode ;

  LoadDictionary("dicttbl.bin") ;
  errcode = MemoryAllocSentence(&sent) ;
  errcode = SentenceAnalysis(example,&sent,
                    MORANRSLT|MORANHNGL) ;
  PrintMorphemeSequence(stdout, &sent) ;
  PrintPhonemeSequence (stdout, &sent) ;
  MemoryFreeSentence(&sent) ;
  FreeDictionary() ;
}
```

위 프로그램의 실행 결과는 다음과 같다.

```
[speech:bin/381] example
나/np+는/pt
학교/nc+에/pa
가/vb+ㄴ다/ef+./se
nanXN np np pt pt pt
haGqOe nc nc nc nc nc pa
gaNda vb vb ef ef ef
```

위에서 보는 바와 같이, 형태소 분석 결과와 발음열, 그리고 각 음소의 해당 품사를 함께 출력하고 있다*.

5 결론

본 논문에서는 한국어 TTS 시스템을 위한 언어처리부의 설계 및 구현 과정을 설명하였다. 이 프로그램은 TTS 시스템을 연구하는 학생들을 위해 구현되었으며, 현재 음성정보산업기술지원센터가 소유권을 가지고 있다.

참고 문헌

[1] 이 상호, 미등록어를 고려한 한국어 품사 태깅 시스템 구현, 석사학위논문, 한국과학기술원 전산학과, 1995.

*np: 인칭대명사, pt: 주제격조사, nc: 보통명사, pa: 부사격조사, vb: 동사, ef: 어말어미