부분 분석 결과를 공유하는 한국어 형태소 분석

이상호[○*], 김재훈[*], 조정미[*], 서정연[*]
한국과학기술원 전산학과[*]

Korean Morphological Analysis Sharing Partial Analyses

Sangho Lee[○*], Jae-Hoon Kim[*], Cho, Jeong Mi[*], Jungyun Seo[*]
Department of Computer Science, KAIST[*]

요 약

한국어 어절의 모든 가능한 형태소 분석 결과는 형태소 격자 구조로 대응된다. 즉, 형태소 분석 과정은 형태소 격자 구조를 만드는 과정과 동일하다고 말할 수 있다. 기존의 방법들은 여러 개의 가능한 분석 결과에 중복되는 형태소들을 그대로 저장하여 자료 관리의 비효율성이 있었다. 본 논문에서 설명하는 형태소 분석기는 형태소 분석의 중간 결과를 공유하여, 자료의 중복 저장을 피했고, 모든 가능한 형태소 분석 결과를 형태소 격자 구조의 가능한 모든 경로로 대응하였다. 한편, 형태소 배열 규칙은 품사 태깅된 말뭉치로부터 자동으로 추출되었다. 또한, 사전도 품사 태깅된 말뭉치로부터 자동으로 구축되었으며, 굴절된 형태소는 등록되지 않는다. 그러나 불규칙 및 축약 현상에 관한 정보는 수동으로 추가되었다. 불규칙 및 축약 현상의 발생 가능 위치는 한글 자소 패턴에 의해서 찾아지고, 이들 현상의 처리는 절차적인 방법에 의해 해결되었다.

1. 서론

언어는 여러가지 층위로 구성이 되어 있고, 각각의 층위는 나름의 언어 단위로 이루어져 있다. 그러한 언어 단위 중 의미 또는 기능을 지니고 있는 단위를 일반적으로 문법 단위라고 하고, 그 중 최소의 단위를 '형태소'라고 한다[이 희승 1991]. 한편, 주어진 어절을 이러한 '형태소' 단위로 나누는 것을 '형태소 분석'이라 하고, 한국어 처리 시스템을 위해 형태소 분석에 관한 연구가 많이 진행되어 왔다. '형태소 분석' 방법에는 음절 정보를 이용하는 방법[강 승식 1993], CYK 파싱 방법을 기초로 한 Tabular Parsing 방법[김 성용 1987, 이 은철 1992], 어절의 중의성 유형 분류에 근거한 방법[임 희석 1993], two-level 형태소 분석 방법[이 성진 1992] 등이 있다.

기존의 방법들은 형태소 분석 결과의 표현 구조보다는 분석 알고리즘의 효율에 치중하여 자료 관리에 비효율성이 있었다. 본 논문에서는 이러한 점을 해결하기 위해, 분석 결과를 공유하는 형태소 격자 구조를 형태소 분석 결과의 표현 구조로 삼았다. 예를 들어, [그림 1]은 어절 "나는"의 형태소 분석 결과로서의 격자 구조이다.

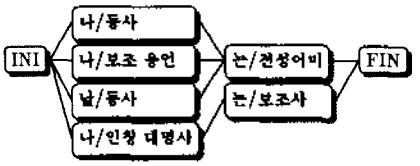

그림 1: "나는"의 형태소 격자 구조

"나는"에 대한 형태소 분석은 "나/동사 + 는/어미","나/보조용언 + 는/어미", "날/동사 + 는/어미","나/인칭대명사 + 는/보조사"이 가능하다. 이 결과를 살펴보면, "는/어미"는 앞의 세 가지 분석 결과에 모두 쓰인다. 본 논문에서는 이와 같은 점에 착안하여 [그림 1]과 같이, 부분적인 분석 결과가 같을 경우에 이를 공유하도록 하였다. 이와 같은 처리는 구문 분석에서 차트(chart)를 이용해서 부분 분석결과를 공유하는 것과 동일한 개념이다[1].

또한, 최종 형태소 분석 결과는 [그림 1]에서 보듯이 "INI" 노드에서 "FIN" 노드까지의 모든 경로에 대응하고[2], 최종 형태소 격자 구조를 형성한다. 이 구조는 통계적 품사 모호성 해소에 그대로 이용될 수 있다. 따라서 품사 태깅 시스템 구현시, 그대로 품사 모호성 해소 모듈의 입력으로 사용될 수 있다.

본 논문의 구성은 구현된 형태소 분석기가 사용하는 정보와 형태소 분석기에 대해 각각 2장, 3장에서 설명하고 4장에서 결론 및 토의를 한다.

[1] 본 연구는 한국 통신의 정기기초 과제 "자동 통역 전화 개발을 위한 대화체 기계번역에 관한 연구"의 부분 지원을 받은 것입니다.
[2] "INI","FIN" 노드는 어절의 처음과 어절의 마지막을 의미하는 가상의 노드이다.

2. 형태소 분석기의 이용 정보

본 논문에서 설명되는 형태소 분석기가 이용하는 정보는 [그림 2]에서 보는 바와 같이 사전, 형태소 분석 엔진, 형태소들의 배열 규칙에 관한 정보, 불규칙 추정 자소 테이블이다.

2.1 품사 접속표

형태소 배열 규칙을 처리하는 기존의 방법에는 형태소들의 접속 제약을 기술한 곽우 접속 정보 [김 성동 1987, 이 은철 1992], 어절 구조를 표현한 결정적인 유한 오토마타 [김 덕봉 1993] 등이 있다. 본 논문에서의 형태소 분석기는 태깅된 말뭉치(약 55,000어절)로부터 품사 접속표 C을 구한다. 이를 정형적으로 설명하면 다음과 같다[3].

$$\begin{aligned} T &= \{\text{ 대명사 , 보조사 , } \cdots \text{ , 부사 파생 접미사 }\} \\ T^E &= T \cup \{INI, FIN\} \\ C &\subset T^E \times T^E \\ |T| &= 52 \\ |T^E| &= 54 \end{aligned}$$

여기서 T는 품사 집합이고, T^E는 어절의 처음과 끝을 나타내는 특별한 품사, "INI", "FIN"이 포함된 품사 집합이며, $|\cdot|$는 집합의 원소 수를 의미한다.

한편, 말뭉치에서 얻은 전체 품사열 집합 TS^C와 확장된 품사열 집합 TS^{EC}는 다음과 같다[4].

$$\begin{aligned} TS^C &\subset T \cup T \times T \cup T \times T \times T \cup \ldots \\ TS^{EC} &= \{INI \oplus ts^C \oplus FIN | ts^C \in TS^C\} \end{aligned}$$

여기서 $\oplus$는 연접 연산자(concatenator)이다.

ts_i^{EC}는 ts^{EC}의 i번째 원소(품사)이며, C은 다음과 같은 방법으로 정의된다[5].

$$C = \{< ts_i^{EC}, ts_{i+1}^{EC} > | 1 \le i \le |ts^{EC}| - 1, ts^{EC} \in TS^{EC}\}$$

결국, 두 품사 $t_i \in T^E$와 $t_j \in T^E$가 접속 가능한지 여부를 $< t_i, t_j >$가 C에 속해 있는 지로 대신한다.

2.2 사전

본 형태소 분석기의 논리적 사전 구성은 형태소를 키(key)로 하는 {(형태소, 품사, 불규칙-코드)} 집합이고, 물리적인 사전 구성은 확장 해싱(extendible hashing)이다. 사전은 품사 접속표와 마찬

[3] T : Tag Set , T^E : Extended Tag Set , C : Connectability Matrix
[4] TS^C : Tag Sequence (Corpus)
TS^{EC} : Extended Tag Sequence (Corpus)
[5] C를 구하는 과정에서 한국어 어절에 관한 정보의 손실이 생겨 실제 형태소 분석을 할 때 과분석(over analysis)이 발생되는 것을 발견할 수 있었다.

자소 패턴				불규칙	예		
(러)	ㄹ	ㅣ	ㅓ	'이+어→여' 축약	해아렸다	→	헤아리 + 었 + 다
(나ㄴ)	ㄴ	ㅏ	ㄴ	'ㅅ' 탈락	나는	→	낫 + 는
(는)	ㄴ	_	ㄴ	'ㅅ' 탈락	가는	→	가는 + ㄴ
(만)	ㅁ	ㅏ	ㄴ	'ㅎ' 불규칙	까만	→	까맣 + ㄴ
(말)	ㅁ	ㅏ	ㄹ	'ㅎ' 불규칙	까말	→	까맣 + ㅅ
(맘)	ㅁ	ㅏ	ㅁ	'ㅎ' 불규칙	까맘	→	까맣 + ㅁ
(맙)	ㅁ	ㅏ	ㅂ	'ㅎ' 불규칙	까맙나다	→	까맣 + ㅂ니다
(아우)	ㅂ	ㅏ	ㅜ	'ㅂ' 불규칙	고마운	→	고맙 + ㄴ

표 1: 불규칙 추정 자소 테이블의 일부분

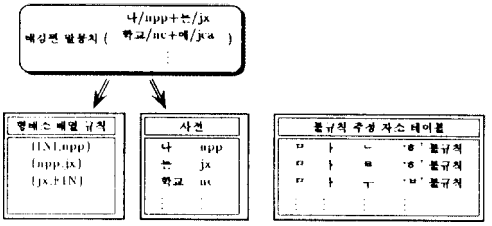

그림 2: 형태소 분석기 이용 정보

가지로 태깅된 말뭉치에서 추출하고, 말뭉치에서 구할 수 없는 불규칙 정보는 수동으로 첨가하였다. 또한, 준말 사전을 두어 준말을 본디말로 확장하여, 처리할 수 있도록 하였다. 예를 들면, "가야겠다"와 옳은 분석은 "가/동사 + 아야/연결어미 + 하/보조용언 + 겠/선어말어미 + 다/종결어미"이다. 이러한 분석을 위해 실질적으로 "아야겠"은 형태소가 아니지만, 준말 사전에 분석결과인 "아야/연결어미 + 하/보조용언 + 겠/선어말어미"를 등록한다.

2.3 불규칙 추정 자소 테이블

일반적으로 불규칙 현상을 처리하기 위해서는 주어진 어절의 임의의 위치에서 불규칙이 발생할 수 있다고 추정하고, 그것을 검증하는 방법과 어간 혹은 어미의 이형태를 사전에 등록시켜 분석이 끝난 후 원형을 복구하는 방법으로 크게 나뉜다. 특히 후자의 경우, 예를 들면, '아름다운'의 분석을 위해 '아름다우'를 사전에 등록시킬 때, '아름다우'가 이형태임을 표시해야 하므로 품사 이외에 또 다른 정보를 추가시키는 결과가 된다.

본 형태소 분석기에서 사용한 불규칙 현상 처리 방법은 전자의 경우이다. 즉, 사전에는 이형태소가 등록되지 않고 형태소 분석 과정에서 불규칙 현상 가능 위치를 추정한 후 그것을 검증한다. 불규칙 현상 가능 위치를 추정하기 위해서, 불규칙 추정 자소 테이블을 이용한다.이 테이블에는 불규칙, 탈락, 축약 현상이 발생되는 한글 자소 패턴을 가지고 있다. [표 1]은 본 논문에서 사용하는 불규칙 추정 자소 테이블의 일부분이다.

3. 형태소 분석기

3.1 형태소 분석기의 구성 모듈

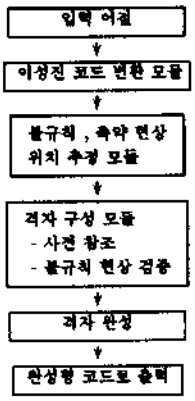

그림 3: 형태소 분석기 자료 흐름도

한글표기	ㅇ	ㅓ	ㅗ	ㅜ	ㅔ	ㅐ	ㅡ	ㅣ	ㅟ	ㅘ	·					
내부코드	a	e	o	u	E	9	-	i	wu	wi						
한글표기	ㅇㅑ	ㅇㅕ	ㅇㅛ	ㅇㅠ	ㅇㅖ	ㅇㅒ	ㅟㅓ	ㅟㅔ	ㅘㅐ	ㅙㅔ	ㅚ					
내부코드	ya	ye	yO	yu	yE	y9	we	wE	wa	we	yi					
한글표기	ㄱ	ㄴ	ㄷ	ㄹ	ㅁ	ㅂ	ㅅ	ㅇ	ㅈ	ㅊ	ㅋ	ㅌ	ㅍ	ㅎ		
내부코드(초성)	g	n	d	l	m	b	s		J	c	k	t	p	h		
내부코드(종성)	G	N	D	L	M	B	S	*	J	C	K	T	P	H		
한글표기	ㄲ	ㅆ	ㅃ	ㅉ	ㅽ											
내부코드(초성)	q	f	r	v	z											
내부코드(종성)	Q			V												

표 2: 이성진 코드

본 형태소 분석기의 구성 모듈은 [그림 3]에 보여진다. 주어진 어절에 대해 먼저 이성진 코드로 변환을 한 후, 불규칙 추정 자소 패턴을 참조하여, 어절의 어느 위치에서 어느 불규칙, 축약 현상이 발생할 수 있는가를 추정한다. 물론, 이때는 실제 발생한 것보다 많은 추정을 하게 된다. 하지만 한국어에서 불규칙 축약 현상은 어간 어미 사이에서 발생하므로 추정된 위치의 오른쪽에 어미가 있지 않다면 불규칙 축약 현상 추정 자체를 무시할 수 있다.

한편, 형태소 분석은 [그림 3]에서 보듯이 변환된 어절을 갖고 형태소 격자 구성 알고리즘을 이용하여 처리한다. 형태소 격자 구성 알고리즘은 주어진 어절의 오른쪽에서 왼쪽으로 형태소를 찾는 방법이고, 사전을 참조할 때는 캐쉬(cache)를 두어 이미 찾은 형태소의 사전 참조를 방지했다.

3.2 내부 코드로의 변환

형태소 분석기의 내부 한글 코드로 이성진 코드를 이용한다[이 성진 1992]. 이성진 코드는 음가가 없는 'ㅇ'은 코드가 부여되지 않고 반모음에 대해 코드가 부여되어 이중모음은 연속된 모음 코드를 이용한다[표 2]. 따라서, 본 논문에서는 "아름다워"를 "al_Mdawe"로 변환 후 처리된다. 이성진 코드는 어절의 오른쪽에서 왼쪽으로 형태소를 찾아갈 때 음운 변화를 쉽게 처리할 수 있게 해 주는 특징이 있다. 예를 들어 '고마워(gomawe)'에 대해 '어(e)'를 먼저 찾아 '어/어미'의 정보를 사전에서 찾고 반모음인 'w'를 'ㅂ(B)'로 대체하면 '고맙(gomaB)'이 되는 것을 볼 수 있다.

번호	불규칙,축약 현상		번호	불규칙,축약 현상	
1	'으'	탈락	11	'너라'	불규칙
2	'ㅂ'	탈락	12	'ㅎ'(1) 예)빨간	불규칙
3	'ㅅ'	탈락	13	'ㅎ'(2) 예)빨개서	불규칙
4	'ㄷ'	불규칙	14	'예,에'	축약
5	'ㅂ'	불규칙	15	'해'	축약
6	'르'	불규칙	16	'이+어=어'(체언,용언)	축약
7	'러'	불규칙	17	'오+아=와' '우+어=워'	축약
8	'우'	불규칙	18	'아+아=아' '어+어=어'	축약
9	'여'	불규칙	19	'외+어=왜'	축약
10	'거라'	불규칙	20	'이(서술격 조사)'	탈락

표 3: 한국어 불규칙, 탈락, 축약 현상

코드		불규칙 추정	추정 예
n	ㄴ		
a	ㅏ	'ㄹ' 탈락	naL(날)+n_N(는)
n	ㄴ		
-	ㅡ	'ㄹ' 탈락	nan_L(나날)+N(ㄴ)
N	ㄴ		

표 4: 나는(nan_N)의 불규칙 추정 위치

다.

3.3 불규칙 및 축약 현상 위치 추정

2.3장에서 설명하였듯이 본 형태소 분석기는 주어진 어절에 대한 불규칙 추정을 먼저 한다. 본 형태소 분석기에서는 한국어의 불규칙, 탈락, 축약 현상을 총 20개로 나누어 처리한다[표 3].

예를 들어, [표 1]의 불규칙 추정 자소 테이블을 이용하여, "나는(nan_N)"의 불규칙 추정을 해 보면, 추정 위치는 [표 4]와 같고, 이를 불규칙 위치 테이블이라고 부른다. [표 4]의 경우는 추정된 부분에 대해 형태소 분석 알고리즘에서 모두 검증을 하게 된다. 다시 말해 'ㄴ(N)'과 '는(n_N)'의 종사 중 어미가 있기 때문에 '나날(nan_L)'과 '날(naL)'이 용언이 될 수 있는 조건을 만족하게 된다[6]. 또한, "나날(nan_L)"은 추정되었으나, 사전에 이와 같은 형태소가 없기 때문에 "나날"에 대한 추정은 최종 형태소 격자에는 포함되지 않는다[그림 1].

3.4 형태소 분석 알고리즘

지금까지 우리는 형태소 분석에 필요한 정보, 이성진 코드로의 변환 등을 알아보았다. 지금부터는 본 형태소 분석기에서 사용한 형태소 격자 구성 알고리즘에 대해 알아보자. 먼저, 이성진 코드로 변환된 어절의 각 코드 사이에 서로 다른 정수값을 넣는다[7]. 우리는 이것을 격자 초기화라고 부른다. 예를 들어, '나는(nan_N)'의 격자 초기화는 [그림 4]와 같다. 이렇게 구성된 초기 격자를 가지고 형태소 격자를 구성하기 시작하는데, 이 때 필요한 변수와 함수

[6]사전을 참조할 때 먼저는 'ㄹ' 불규칙 처럼 불규칙 코드를 보아야 하는 경우도 있다.
[7]구현된 형태소 분석기는 정수값이 6씩 증가하는 방법을 택하였는데 이러한 값은 단지 형태소와 형태소의 경계점을 쉽게 표현하기 위해 사용된 심벌(symbol)일 뿐이다.

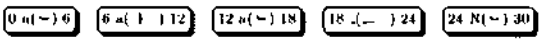

그림 4: 나는(nan_N)의 격자 초기화

Left_Pos , *Right_Pos*	형태소를 만들 때 필요한 변수들. *Left_Pos*와 *Right_Pos*는 형태소의 각각 왼쪽, 오른쪽 번호이다.
Make_Morpheme()	*Left_Pos*와 *Right_Pos*를 갖고 형태소를 만드는 함수. 예를 들어 [그림 4]에서 *Left_Pos*와 *Right_Pos*가 각각 12, 30일 경우 Make_Morpheme()의 결과는 'n_N(는)'이다
Lookup_Dict()	형태소를 갖고 사전을 참조하는 함수. 형태소에 관한 품사, 불규칙-코드 등 정보를 내준다.
Put_Into_Lattice()	형태소와 정보를 격자에 넣는 함수. 여기서 품사 접속표 C을 이용하여 새로운 격자를 구성한다.
Irr_Handle()	불규칙 현상 추정의 검증을 위한 함수. 불규칙 현상 추정 위치와 불규칙 코드를 갖고 사전을 참조하여 검증한다.
Update_Next_Pos()	새로운 *Left_Pos*, *Right_Pos*를 구하는 함수. *Left_Pos*, *Right_Pos*는 다음 번 Make_Morpheme()을 수행할 때 사용된다.

표 5: 격자 구성 알고리즘에 사용된 변수와 함수들

들의 기능을 보면 [표 5]과 같다.

한편, 형태소 분석 알고리즘은 [그림 5]과 같고 이 알고리즘을 이용한 "나는"의 분석 과정이 [그림 6]에 보여진다.

형태소 분석 알고리즘은 [그림 5]에 기술된 것처럼 먼저 어절의 마지막을 뜻하는 'FIN' 노드를 격자에 넣는 것으로 시작한다(line 1). 그 후, 초기 격자에서 *Left_Pos*와 *Right_Pos* 사이에 있는 코드를 연결하여 형태소를 만들고(line 3), 사전을 참조한 후(line 4), 정보가 있을 경우 격자에 넣는다(line 5). 한편, 사전을 참조한 형태소의 *Left_Pos*에 불규칙 추정 번호가 등록되어 있고(line 6), *Right_Pos*가 'FIN' 노드의 끝 번호와 같을 때, 불규칙 처리를 하는 Irr_Handle()을 실행한다(line 6)[8]. 예를 들어, [그림 6]의 단계 2가 끝났을 때 *Left_Pos*는 12인데 불규칙 추정 테이블에 'ㄹ'탈락 현상이 등록되어 해당 불규칙 처리를 한 것이다[9].

Update_Pos()은 *Left_Pos*와 *Right_Pos*를 새롭게 구하는 함수로 *Left_Pos*는 'FIN' 노드의 시작 번호로부터 'INI' 노드의 끝 번호로 감소하고, *Right_Pos*는 현재의 *Left_Pos*로부터 'FIN' 노드의 시작 번호로 증가한다. 물론, 이때 몇 가지 휴리스틱을 사용

[8]이것은 3.3장에서 설명했듯이 어미 구한 불규칙 위치 테이블을 참조하는 것이다.

[9]'ㄹ'을 넣고 (*Left_Pos* = 12,*Right_Pos* = 30)을 스택에 저장한 후, *Right_Pos*를 12로 하고 *Left_Pos*를 감소시키며 형태소를 만들어 사전을 참조한다. 사전 등록에 성공하면 Put_Into_Lattice()을 실행하고 *Left_Pos*,*Right_Pos*를 스택에 저장했던 값으로 복구한다.

```
1 Put_Into_Lattice(NULL,{ 'FIN' }) ;
2 do {
3   morpheme = Make_Morpheme() ;
4   information = Lookup_Dict(morpheme) ;
5   if (∃ information)
      Put_Into_Lattice(morpheme,information) ;
6   if (Left_Pos has Irr_Code_Flag)
          Irr_Handle(Irr_Code,Left_Pos) ;
7 } while (!Update_Pos()) ;
8 Put_Into_Lattice(NULL,{ 'INI' }) ;
```

그림 5: 형태소 격자 구성 알고리즘

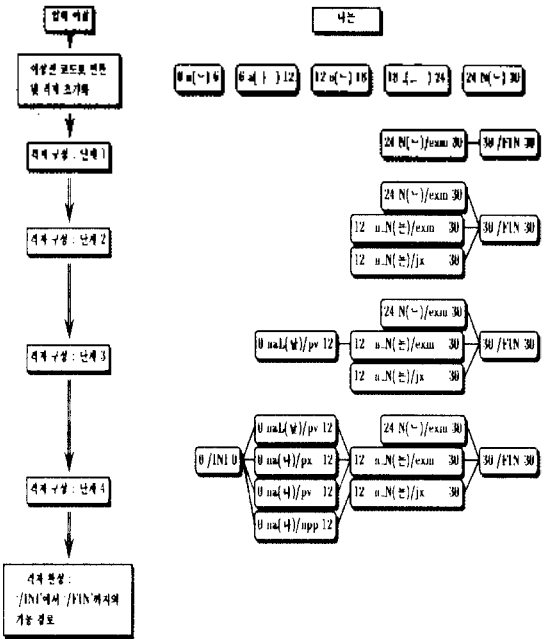

그림 6: "나는"의 격자 구성 과정

하는데, 먼저 형태소의 시작 코드는 'ㄴ,ㅌ,ㅁ,ㅂ' 이외의 종성으로 시작할 수 없다는 것과 형태소의 마지막 코드는 초성으로 끝날 수 없다는 것이다. 또한 격자를 구성해 가면서 이미 만들어진 노드와 *Left_Pos*만이 현재 조사할 형태소의 *Right_Pos*가 될 수 있어서 [그림 6]의 단계 4에서 *Left_Pos*가 0일 때 가능한 *Right_Pos*는 이미 격자내에 있는 'n_N(는)','N(ㄴ)','FIN' 노드의 *Left_Pos*인 12,24,30이 된다.

한편, 현재의 *Left_Pos*가 'INI' 노드의 끝 번호와 같고 *Right_Pos*는 'FIN' 노드의 시작 번호와 같을 때 루프(loop)문을 탈출하게 된다.

끝으로 'INI' 노드를 격자에 넣어 알고리즘은 끝나고, 'INI' 노드에서 'FIN' 노드까지의 경로가 우리가 원하는 모든 가능한 형태소 분석 결과가 되는 것이다.

4. 결론 및 토의

본 논문에서는 격자 구성 알고리즘을 이용하는 형태소 분석기를 소개하였다. 형태소 분석 결과를 격자 구조로 대응하고 최종 격자 구조는 또한 일반적인 통계적 품사 모호성 해소 모듈의 입력으로 그대로 사용될 수 있는 자료구조의 장점을 갖고 있다. 형태소 분석기에서 사용된 정보를 약 55,000어절의 태깅된 말뭉치에서 추출하여, 형태소 배열 규칙을 위한 품사 접속표와 약 8,000개의 표제어를 갖는 사전을 만들었다. 준말사전은 태깅되지 않은 말뭉치를 형태소 분석한 후, 실패한 어절을 수동으로 분석하여 작성되었다. 물론, 이 때 불규칙 추정 자소 테이블을 튜닝(tuning)할 수 있었다.

한편, 구현된 형태소 분석기는 한국어 어절을 선형적인 구조로 분석을 하지만 실제 말뭉치에서는 그렇지 않은 어절을 찾을 수 있었다. 예를 들어 "그가 뛰기 시작한 시간은 10시가 넘어서였다."라는 문장에서 '넘어서였다.'는 '넘/동사 + 어서/연결 어미'가 다시 명사 처럼 이용되어, 옳은 형태소 분석은 '(넘/동사 + 어서/연결 어미)/명사 + 이/서술격 조사 + 었/선어말 어미 + 다/종결 어미'라고 말할 수 있다. 사실 이와 같은 경우는 말뭉치에서 많이 발견되지는 않았지만 옳은 구문 분석을 위해서도 이러한 어절에 대한 처리가 필요하다고 생각된다.

참고 문헌

[이 희승 1991] 이 희승, 안 병희, 한글 맞춤법 강의, 신구문화사, 1991.

[강 승식 1993] 강 승식, 음절 정보와 복수어 단위 정보를 이용한 한국어 형태소 분석, 서울대학교 컴퓨터 공학과 박사 학위 논문, 1993.

[이 은철 1992] 이 은철, CYK법에 기반한 한국어 형태소 분석에서의 개선기법, 포항공과대학 대학원 전자계산학과 석사 학위 논문, 1992.

[김 성용 1987] 김 성용, Tabular Parsing 방법과 접속정보를 이용한 한국어 형태소 해석기, 한국과학기술원 전산학과 석사 학위 논문, 1987.

[임 희석 1993] 임 희석, 어절의 중의성 유형 분류에 근거한 한국어 형태소 분석기, 고려대학교 대학원 전산과학과 석사 학위 논문, 1993.

[김 재훈 1994a] 김 재훈, 서 정연, 자연언어 처리를 위한 한국어 품사 태그, 한국과학기술원, 인공지능연구센타, CAIR-TR-94-55, 1994.

[김 재훈 1994b] 김 재훈, 서 정연, 실용적인 한국어 형태소 해석, 한국과학기술원 전산학과 컴퓨터 시스템 실험실 내부 메모, 1994.

[이 성진 1992] 이 성진, Two-Level 한국어 형태소 해석, 한국과학기술원 석사학위 논문, 1992.

[김 덕봉 1993] 김 덕봉, "DDAG: 효율적인 한국어 형태소 해석 방법,", 제 5회 한글 및 한국어 정보처리 학술발표 논문집, pp. 341 - 353. 1993.