SVM 모델을 이용한 절 경계 인식

이현주 김상수 박성배 이상조
경북대학교 대학원 컴퓨터공학과 언어정보연구실
hjlee@sejong.knu.ac.kr

Clause Boundary Identification Using Support Vector Machines

Hyun-Ju Lee Sang-Soo Kim Seong-Bae Park Sang-Jo Lee
Dept. of Computer Engineering, Kyungpook National University

요 약

여러 개의 절로 이루어진 긴 문장에서 절 단위를 인식해냄으로써 구문분석의 복잡도를 크게 줄일 수 있다. 본 논문에서는 SVM 모델을 이용하여 한국어 문장에서 절의 경계를 인식하는 방법을 제안하였다. 첫 번째 단계로 중심어가 후행하는 한국어 문장의 특성을 고려하여 절의 끝점을 먼저 찾고, 첫 번째 단계의 결과인 절의 끝점 정보와 절의 끝점 인식을 위한 정보보다 더 전역적인 정보를 이용해 절의 시작점을 인식하는 두 번째 단계로 나누어 진행하였다. 구문구조 부착 말뭉치를 이용하여 학습하고 실험한 결과, F-score 86.87%와 단어 단위의 정확도 96.63%의 성능을 나타내었다.

1. 서 론

정보검색이나 문서 요약, 기계번역과 같은 자연언어처리 응용 기술의 발전을 위해서는 근간이 되는 구문분석 기술이 필수적이다. 그러나 현재 구문 분석 기술은 아직 실용적인 단계에 있지 못하다. 단어를 기본 단위로 하여 전체 문장의 구조를 파악해 내는 완전 구문분석(full parsing)은 문장이 길어지게 되면 복잡도가 현저히 커지게 되어 정확한 분석에 많은 어려움이 따르기 때문이다. 이로 인해 문장의 기본적인 구성 단위인 단어를 더 큰 단위로 묶어 구문분석의 복잡도를 줄여 나가는 부분 구문분석(partial parsing)에 관한 연구들이 활발히 진행되고 있다.

한국어의 경우 단어를 구로 묶어 주는 구 단위화(chunking)에 대한 연구들이 활발하게 이루어졌으며 어느 정도 안정된 성능을 보여 주고 있다. 따라서 이를 바탕으로 구 단위를 더 큰 단위인 절 단위로 묶어주는 절 인식(clause identification)은 자연언어처리 응용 시스템들에 더 많은 구문적인 정보를 제공하는 향상된 부분 구문 분석 결과를 보여줄 수 있다.

본 논문에서는 Support Vector Machines(이하 SVM) 모델을 이용하여 한국어 문장에서 절의 경계를 인식하는 방법을 제안한다. 절 인식은 절의 끝점을 인식하는 단계와 절의 시작점을 인식하는 단계인 두 단계로 이루어진다. 한국어 문장은 서술어가 절이나 문장의 맨 마지막에 오는 중심어 후행 문장이므로 절의 끝점의 경계가 명확하다. 반면, 절의 시작점을 인식하는 데에는 많은 모호성이 발생하기 때문에 보다 넓은 범위의 정보가 필요하다. 그러므로 한국어 절 인식에서는 절의 시작점보다 절의 끝점을 먼저 찾는 것이 효율적이며 절의 끝점을 먼저 찾음으로써 절의 시작점을 찾는데 절의 끝점에 대한 정보까지 이용할 수 있다.

논문의 구성은 다음과 같다. 2장에서는 절 인식에 관한 기존 연구를 살펴보고, 3장에서는 SVM 모델을 이용하여 절의 끝점과 시작점을 인식하는 알고리듬에 대해 설명한다. 4장에서는 본 연구에 쓰인 말뭉치에 대해 설명하고, 한국어 절 경계 인식에 대한 실험 결과를 보인다. 마지막으로 5장에서 결론을 내리면서 향후 연구방향에 대해 이야기한다.

2. 기존연구

구문분석의 단위가 되는 문장(sentence)은 어떤 사태(事態)를 표현하는 완결된 문법단위로 볼 수 있다. 하나의 사태는 하나의 서술과 그것에 대한 서술 대상으로 표현된다. 그러나 한 문장이 되기 위해서는 형태적으로 종결 표현이 하나만 있으면 되므로 한 문장 안에는 여

러 개의 서술 대상과 서술 표현이 나타날 수 있다. 그리고 실제로 우리가 구사하는 문장은 하나의 사태를 표현하는 단문보다는 둘 이상의 사태를 표현하는 복문의 비율이 훨씬 높다[1]. 이것이 구문분석을 어렵게 하는 큰 이유가 된다. 그러므로 여러 개의 서술 표현과 서술 대상이 얽혀 있는 문장에서 하나의 사태를 표현하는 단위인 절(clause) 단위를 인식해낼 수 있다면 구문분석의 어려움을 한층 줄일 수 있다. 또한 자연언어처리 응용 시스템들에 훨씬 깊은 문법적 정보를 제공할 수 있다.

이런 요구에 따라 영어권에서는 절 인식 또는 절 분할에 관한 연구들이 활발하게 진행되어 왔다. 초기의 연구들에서는 주로 인간이 규칙을 추출하는 방식이었다. Ejerhed(1988)는 음성 합성(text-to-speech) 시스템을 개선시키기 위한 목적으로, 저자가 만든 절 문법 규칙을 이용한 방식과 확률적인 접근 방식 두 가지로 절 분할에 대한 실험을 하였다. Leffa(1998)는 영어 문법책과 신문에서 무작위로 뽑은 1000개의 문장에서 인간이 추출한 규칙으로 절을 분할하고 이 절을 명사절이나 부사절로 분류하였다. 그러나 이러한 규칙 기반 방법들은 사람이 손으로 많은 규칙들을 추출해야 하는 부담이 따른다. 이러한 부담을 줄이기 위해 Orsan(2000)은 기계학습과 규칙 기반을 혼합하는 방식을 채택하였다. 먼저 기억 기반 학습(Memory based learning)으로 절의 경계를 결정하고 그 결과를 더 높이기 위해서 두 단계의 규칙 모듈을 적용하였다. 이후에 CoNLL(Conference on Computational Language Learning)에서는 2001년의 shared task로 절 인식에 대해 여러 가지 기계 학습 방법을 적용한 연구를 진행했으며, Penn Treebank의 Wall Street Journal 부분의 일부 데이터를 대상으로 6개 시스템의 성능에 대한 평가가 이루어져 있다[2].

한국어를 대상으로 한 연구는 대부분 규칙 기반 방법이다. 김광진 외(1993)에서는 내포문을 대상으로 의미표지 테이블과 용언의 하위범주 정보를 이용하여 단문으로 분리하였다. 그리고 규칙을 이용해 생략된 성분을 복원하고 대용어를 처리하였다. 그러나 이 연구는 내포문만을 대상으로 하고 있으며 접속문은 고려하지 않고 있다. 윤승(2001)에서도 단어의 태깅 정보만을 가지고 분할점을 찾는 규칙을 이용하여 복합문을 단문으로 분리하는 알고리듬을 제안하였으며, 김미진(2003)에서는 한국어 복합문에서 영대용어를 처리하기 위한 이전 단계로 문장 유형에 따른 규칙을 이용해 복합문을 분해하고 있다. 이들은 모두 많은 수의 규칙과 용언의 하위범주화 정보 같은 대용량의 지식을 손으로 구축해야 하는 어려움이 따른다.

위에서 살펴 본 것처럼 한국어를 대상으로 하는 절 인식에 관한 연구는 거의 규칙과 사전 정보를 이용한 것이고 기계 학습을 이용한 것은 찾아 보기 힘들다. 이렇게 한국어를 대상으로 한 연구에서 기계 학습을 이용한 연구가 미진한 중요한 이유 중의 하나는 학습과 실험에 이용할 표준화된 말뭉치가 없다는 것이다. 본 연구에서는 구문구조 부착 말뭉치를 변형하여 절 인식을 위한 학습 말뭉치로 이용하였다. 말뭉치에 대해서는 4장에서 자세하게 언급한다.

한국어를 대상으로 절보다 더 작은 단위인 구를 단위화하는 연구도 크게 규칙을 사용하는 방법과 기계학습 기법을 사용하는 방법으로 나뉠 수 있으며, 규칙 기반 방법과 기계 학습 방법을 결합한 방법도 있다. 한국어에 대한 구 단위화 시스템들이 어느 정도 안정된 성능을 보여 주고 있으므로 이 결과를 더 큰 단위인 절 인식 연구에 대한 발판으로 삼을 수 있을 것이다.

3. 한국어 절 인식

3.1 문제 정의

절(clause)은 주어와 서술어를 하나만 가지는 단어열이다. 예를 들어 아래의 문장을 살펴보자.

((남아메리카에서 1천 2백 킬로미터 이상 떨어져 있으며) (그 사이의 바다가 (세계에서 (험하기)로 유명한) 드레이크해협이다.))

위 예문에서 괄호는 문장의 절의 경계를 나타낼 뿐만 아니라 절의 층위를 표시하기도 한다. 위 예문을 살펴보면, 문장은 하나이지만 두 절이 대등적으로 연결되어 있고 대등적으로 연결된 뒷 절은 관형절을 안고 있으며 그 관형절은 다시 명사절을 안고 있는 복잡한 구성의 복합문이다. 본 연구의 목표는 이러한 복합문에서 SVM 모델을 이용하여 절의 경계를 인식하는 것이며, 절의 유형과 절의 층위는 고려하지 않는다. 즉 위의 문장에서 네 개의 절의 경계를 인식하는 것이 목표이며 그 절이 관형절이나 명사절이라는 등의 절의 유형과 관형절 안에 명사절이 내포되어 있다거나 하는 절의 포함 연결관계를 밝히는 것은 본 연구 대상에서 제외한다.

절 경계 인식은 절의 끝점(E)과 절의 시작점(S)을

1) 윤승(2003)에 따르면 절 하나가 문장이 된 단문의 비율은 전체 말뭉치의 16.8%에 지나지 않는다. 특히, 신문이나 잡지, 논문과 같은 실용문은 단문의 비율이 훨씬 낮다.

2) http : //cnts.uia.ac.be/conll2001/clauses

찾는 것으로 볼 수 있다. 그리고 이것은 문장 내의 한 단어 w_i를 기준으로 해서 본다면 이 단어 w_i를 절의 시작(S), 절의 끝(E), 절의 시작도 끝도 아님(X)의 세 개 클래스 중 하나로 분류하는 문제로 볼 수 있다. 본 논문에서는 이를 다시 끝점(E)에 대한 이진 분류와 시작점(S)에 대한 이진 분류로 나누어 접근한다.

3.2 Support Vector Machines

SVM은 Vladimir Vapnik(1995)에 의해 이원 패턴 인식 문제를 풀기 위해 제안된 것으로, 지금은 패턴 인식 문제 외에도 이진분류 문제를 해결하기 위한 학습 알고리듬으로 널리 쓰이고 있다.

$$(x_1, y_1), \ldots, (x_n, y_n) \quad x_i \in R^n, \quad y_i \in \{+1, -1\}$$

위와 같이 학습 데이터가 주어졌을 때, xi는 n차원 벡터로 표현된 i번째 데이터이고, yi는 i번째 데이터의 클래스(+1 또는 -1)를 표시하는 레이블이다.

SVM은 아래와 같이 이 학습 데이터를 두 클래스로 정확하게 분류하는 최적의 초평면(hyperplane)을 찾는다.

$$(w \cdot x) + b = 0, \; w \in R^n, \; b \in R^n$$

이를 위해 초평면과 가장 인접한 점과의 거리(margin)가 최대가 되도록 초평면을 학습한다. 이 거리(d)는 다음과 같이 나타낼 수 있다.

$$(w \cdot x) + b = \pm 1, \; d = 2/\|w\|$$

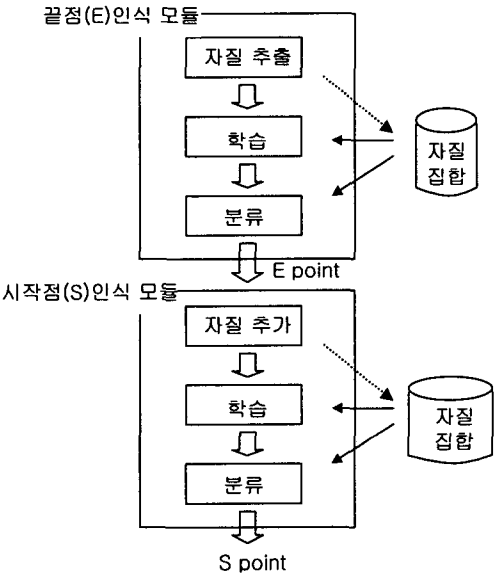
그러므로 SVM은 $y_i[(w \cdot x) + b] \geq 1$이라는 제약 하에서 $\|w\|$가 최소가 되도록 w와 b를 찾아낸다.

SVM은 얼굴인식이나 문자인식과 같은 여러 가지 패턴 인식 분야 외에 자연언어처리 분야에서 자동 문서분류, 구 단위화 문제로까지 적용 영역이 확대되어 좋은 성능을 나타내고 있다.

3.3 두 단계 절 인식

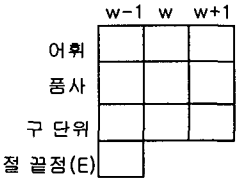
절은 주어와 서술어의 관계가 한 번 맺어진 것을 말하는데, 절에서 핵이 되는 성분은 서술어이다. 한국어의 경우에 주어는 생략이 될 수도 있고, 용언의 특성에 따라 두 번 나타날 수도 있으므로 절을 구분할 때는 서술어가 기준이 된다. 즉 서술어의 개수가 곧 절의 개수이다. 한국어 문장은 영어와 달리 서술어가 절이나 문장의 맨 마지막에 오는 중심어 후행 문장이므로 서술어가 절의 끝점이 된다. 반면 절의 시작점의 경우, 서술어를 포함하여 그 서술어의 앞에 오는 모든 성분이 후보가 될 수 있다. 그러므로 절의 끝점이 절의 시작점보다 더 명확하며, 절의 시작점을 찾는 데에는 절의 끝점을 찾는데 필요한 정보보다 더 전역적인 정보가 필요하다. 따라서 본 연구에서는 절 경계 인식 시스템을 아래 그림 1과 같이 두 단계로 나누어 구성하였다.

먼저 학습 데이터로부터 끝점 인식을 위한 자질을 추출하고 이 자질을 가지고 SVM 모델을 이용해 학습하고 그 단어가 절의 끝점인지 아닌지를 분류하게 된다.

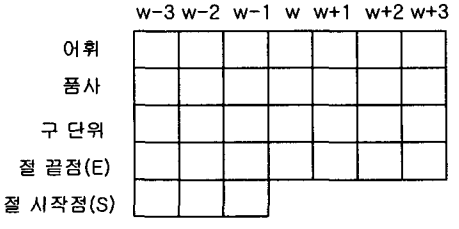


[그림 1] 한국어 절 경계 인식 모델의 구성도

이렇게 얻은 절의 끝점은 절의 시작점을 찾는 자질에 추가되어 절의 시작점을 찾는데 이용된다. 절의 끝점과 절의 시작점 인식을 위한 자질은 각각 그림 2, 그림 3과 같다.



[그림 2] 절의 끝점 인식을 위한 자질



[그림 3] 절의 시작점 인식을 위한 자질

위의 그림에서 보는 바와 같이 절의 끝점 인식을 위해서는 4가지 유형의 자질이, 절의 시작점 인식을 위해서는 1단계의 결과인 그 단어의 절의 끝점 정보와 앞

단어들의 시작점 정보를 추가하여 5가지 유형의 자질들이 사용되었다. 또한 절의 시작점을 인식하는 데에 절의 끝점을 인식하는 것보다 더 넓은 범위의 자질을 이용한다. 절의 끝점을 찾는 1단계에서는 후보 단어 자신과 좌우 한 개의 단어에 대해 어휘, 품사, 구단위화 정보를 이용하여 후보 단어가 절의 끝점인지 아닌지를 추정하고, 그리고 2단계에서는 후보 단어 자신과 좌우 세 개의 단어에 대해 어휘, 품사, 구 단위 정보를 이용하여 후보 단어가 절의 시작점인지 아닌지를 추정하게 된다. 어휘 자질로는 말뭉치에서 빈도 5이상인 단어만 사용한 결과, 전체 어휘 수 16,838개 중에 4,171개만 어휘 자질로 선택되었다.

4. 실 험

4.1 말뭉치

본 논문에서 사용한 말뭉치는 STEP 2000 과제의 결과물인 구문구조 부착 말뭉치를 변형하여 만들었다. 아래 그림 4는 STEP 2000 구문구조 부착 말뭉치의 모습이다.

```
(S
 (VP
  (VP
   (VP
    (VP (NP 남아메리카/nq )+에서/jca
     (VP
      (NP (NP 1천/nnc 2백/nnc+킬로미터/nbu ) 이
       상/ncn ) 떨/pvg ))+
      (AUXP 어/ecx+지/px ))+(AUXP 어/ecx 있
       /px ))+으며/ecs
  (VP
   (NP
    (NP (MODP 그/mmd ) 사이/ncn )+의/jcm 바다
     /ncn )+가/jcs
   (VP
    (NP
     (ADJP (NP 세계/ncn )+에서/jca
      (ADJP
       (NP (ADJP 험하/paa )+기/etn )+로/jca 유
        명/ncps+하/xsm ))+ ㄴ/etm 드레이크해협
        /nq )+이/jp )))+다/ef+./sf )
```

[그림 4] STEP 2000 구문구조 부착 말뭉치

이 구문구조 부착 말뭉치는 이진 트리 구조로 이루어져 있으며, 이로부터 절의 경계에 해당하는 정보를 자동으로 추출하였다. 이 말뭉치는 12,085개의 문장과 50,658개의 절, 111,658개의 구, 321,328개의 단어로 구성되어 있다. 절의 개수는 서술어의 개수를 기준으로 한 것으로 평균적으로 한 문장이 4개의 절로 구성되어 있음을 알 수 있다.

그림 5는 이로부터 추출된 절 인식 학습을 위해 사용된 데이터 형태의 예이다. 데이터 형태는 CoNLL-2001의 데이터 형태를 그대로 따랐다.

단어	품사	구단위화	시작	끝
남아메리카	nq	B-NP	S	X
에서	jca	I-NP	X	X
1천	nnc	B-NP	X	X
2백	nnc	I-NP	X	X
킬로미터	nbu	I-NP	X	X
이상	ncn	I-NP	X	X
떨	pvg	B-VP	X	X
어	ecx	I-VP	X	X
지	px	I-VP	X	X
어	ecs	I-VP	X	X
있	px	I-VP	X	X
으며	ecs	I-VP	X	E
그	mmd	B-NP	S	X
사이	ncn	I-NP	X	X
의	jcs	I-NP	X	X
바다	ncn	I-NP	X	X
가	jcs	I-NP	X	X
세계	ncn	B-NP	S	X
에서	jca	I-NP	X	X
험하	paa	B-VP	S	X
기	etn	I-VP	X	E
로	jca	B-NP	X	X
유명	ncps	B-VP	X	X
하	xsm	I-VP	X	X
ㄴ	etm	I-VP	X	E
드레이크해협	nq	B-NP	X	X
이	jp	B-VP	X	X
다	ef	I-VP	X	E
.	sf	O	X	X

[그림 5] 학습을 위해 변형한 말뭉치 형태의 예

두 번째 열의 품사 태그는 KAIST 태그셋[15]을 따른 것으로 52개의 품사로 표현되어 있다. 세 번째 열은 구의 단위 종류를 나타내는데 9가지 단위 종류 중 하나를 나타낸다. 이 구 단위 정보는 [11][3]에서 가져온 것이다. 네 번째와 다섯 번째 열이 절의 시작점과 끝점에 대한 정보를 나타내는 것으로 시작점인 경우 S로 나타내며, 끝점인 경우 E로 나타내고, 시작점도 아니고 끝점도 아닌 경우 X로 나타낸다.

3) http : //bi.snu.ac.kr/~sbpark/Step2000에서 가져온 것으로 STEP 2000 말뭉치에 대한 구 단위화 데이터이다.

4.2 실험 및 평가

실험의 평가 기준은 다음과 같다.

$$\text{Recall} = \frac{\text{절 끝(시작)점으로 바르게 인식된 개수}}{\text{절 끝(시작)점의개수}}$$

$$\text{Precision} = \frac{\text{끝(시작)점으로 바르게 인식된 개수}}{\text{끝(시작)점으로 인식된 개수}}$$

$$F_\beta - \text{score} = \frac{(\beta^2+1)\cdot \text{Recall}\cdot \text{Precision}}{\beta^2 \cdot \text{Recall} + \text{Precision}}$$

$$\text{Accuracy} = \frac{\text{정확하게 추정된 단어 수}}{\text{총 단어 수}}$$

위의 F-score에서 재현율과 정확률에 같은 가중치를 주기 위해 $\beta=1$로 설정하였다.

말뭉치 전체 문장 수는 12,085개이며, 전체 말뭉치의 90%인 10,877개의 문장을 학습 집합으로, 나머지 10%인 1,208개 문장을 테스트 집합으로 사용하였다. 실험에는 SVM^{light}[10]를 사용했다.

실험결과는 다음 표 1과 같다.

[표 1] 실험 결과

	precision	recall	F-score	accuracy
끝점(E)	92.75%	92.88%	92.81%	97.74%
시작점(S)	87.58%	75.21%	80.93%	95.52%
평균	90.17%	84.05%	86.87%	96.63%

기존의 연구들은 모두 영어를 대상으로 한 것이므로 객관적인 비교를 할 수가 없다. 그러나 CoNLL'01에서 HMM을 이용한 [6]의 경우 F-score가 시작점은 86.48%, 끝점은 78.38%의 결과가 나왔으며, AdaBoost를 사용하여 가장 좋은 성능을 나타낸 [8]의 경우 F-score가 시작점은 91.72%, 끝점은 89.22%의 결과를 보였다.

본 연구의 실험 결과는 모든 단어를 후보로 두고 실험한 것이다. 그러나 구 단위화 정보를 이용하게 되면 사실상 문장 내 모든 단어들을 후보로 고려할 필요가 없다. 단위화된 구(chunking)의 경계에 있는 단어들만 후보로 고려하고 구 단위의 내부에 있는 단어들은 모두 시작도 끝도 아닌(X) 것으로 판단하면 된다. 그 중에서도 절의 끝점인가 아닌가를 결정하기 위해서는 구의 끝에 있는 단어들에 대해서만 결정을 내리면 되고, 절의 시작점인가 아닌가를 결정하기 위해서는 구의 시작에 있는 단어들에 대해만 결정을 내리면 된다. 이러한 전처리를 하게 될 경우 결과는 더 향상될 수 있다.

5. 결론 및 향후 연구

복잡한 문장에서 절을 인식함으로써 향상된 구문분석 결과를 보여줄 수 있다. 본 연구에서는 SVM 모델을 이용하여 한국어 문장에서 절의 경계를 인식하였으며, 구문구조 부착 말뭉치를 이용하여 학습하고 실험한 결과, 정밀도 90.17%와 재현율 84.05%, 단어 단위의 정확도 96.63%의 성능을 나타내었다. 한국어의 경우 끝점 인식은 정밀도와 재현율이 모두 높게 나타났으나, 시작점은 재현율이 많이 떨어지는 것으로 나타났다. 끝점의 경우에는 끝점을 인식하기 위한 자질이 분명하나 시작점은 그렇지 않기 때문인 것으로 판단된다. 향후 연구로는

$$Accuracy = \frac{\textit{정확하게 추정된 단어 수}}{\textit{총 단어 수}}$$

시작점의 재현율 향상과 더 나아가 본 연구에서 인식한 절의 시작점과 끝점을 연결하여 완전한 절을 인식해 내는 것을 목표로 한다.

감사의 글

이 논문은 2004년도 경북대학교 학술진흥연구비에 의하여 연구되었음.

참고문헌

[1] Eva I. Ejerhed(1988) : Finding clauses in unrestricted text by finitary and stochastic methods, *Proceedings of the 2nd conference on applied natural language processing*, Austin, Texas.

[2] Vilson J.Leffa(1998) : Clause Processing in Complex Sentences, *Proceedings of LREC'98*, Granada, Espanha.

[3] Constantin Orasan(2000) : A hybrid Method for clause splitting in unrestricted English text, *Proceedings of ACIDCA'2000*, Monastir, Tunisia.

[4] T.Kudo and Y.Matsumoto(2000) : Use of support vector learning for chunk identification, *Proceedings of the 4th Conference on Computational Natural Language Learning*, Lisbon, Portugal.

[5] Erik F.Tjong Kim Sang and Hervé(2001) : Déjean, Introduction to the CoNLL-2001 Shard Task : Clause Identification, *Proceedings of CoNLL-2001*, Toulouse, France.

[6] Antonio Molina and Ferran Pla(2001) : Clause Detection using HMM, *Proceedings of CONLL-2001*, Toulouse, France.

[7] Xavier Carreras and Luís Márquez(2001) : Boosting Trees for Clause Splitting, *Proceedings of CoNLL-2001*, Toulouse, France.

[8] Xavier Carreras, Lluís Màrquez, Vasin Punyakanok and Dan Roth(2002) : Learning and Inference for Clause Identification, *Proceedings of the 13th European Conference on Machine Learning*, Helsinki, Finland.

[9] Erik F.Tjong Kim Sang(2002) : Memory-based shallow parsing, *Journal of Machine Learning Research 2*, pp559-594.

[10] T. Jochims(1998) : Making large-scale SVM learning practical, *Technical Report LS8*, Universitat Dortmund.

[11] 박성배, 장병탁(2004) : 한국어 구 단위화를 위한 규칙 기반 방법과 기억 기반 학습의 결합, *정보과학회논문지* 제31권 제3호, pp369-378.

[12] 윤승(2001), 한국어 복합문 분할 방안 연구, 연세대학교 대학원 국어정보학 협동과정 석사학위논문.

[13] 김미진(2003) : 한국어 복합문의 Zero Anaphpra 처리를 위한 분해 및 복원 알고리즘, 경북대학교 대학원 컴퓨터공학과 박사학위논문.

[14] 김광진, 송영훈, 이정현(1993) : 한국어 내포문을 단문으로 분리하는 시스템의 구현, *제5회 한글 및 한국어 정보처리 학술발표 논문집*, pp25-34.

[15] 최기선, 남영준, 김진규, 한영균, 박석문, 김진수, 이춘택, 김덕봉, 김재훈, 최병진(1996) : 한국어정보베이스를 위한 형태, 통사 태그 표준에 관한 연구, *인지과학*, 제7권 제4호, PP43-61.