S-절 분할을 통한 구문 분석
(Syntactic Analysis based on Subject-Clause Segmentation)

김 미 영 † 이 종 혁 ††
(Mi-Young Kim) (Jong-Hyeok Lee)

요 약 한국어 장문에서는 하나의 주어를 여러 용언이 공유하는 경우가 흔하고 주어의 생략 또한 빈번하다. 따라서 주어를 공유하는 용언들의 구간을 파악하는 것이 어렵고 의존문법을 이용한 구문분석시 주어의 의존관계를 찾는데 많은 오류가 생긴다. 이러한 주어의 의존관계의 애매성을 해소하기 위하여 우리는 S(ubject)-절이라는 개념을 제안한다. S-절은 한 개의 주어와 이 주어를 공유하는 단어그룹의 집합으로 정의되고, 본 논문에서는 결정트리를 이용하여 S-절을 자동적으로 분할하는 방법을 제안한다. S-절을 사용한 결과 의존문법에 기반한 구문분석 시스템의 성능이 5% 향상되었고 주어의 지배소를 찾는 정확률이 32% 증가했다.

키워드 : S-절, 구문분석, 장문분할, 결정트리

Abstract **In dependency parsing** of long sentences with fewer subjects than predicates, it is difficult to recognize which predicate governs which subject. To handle such syntactic ambiguity between subjects and predicates, this paper proposes an "S-clause" segmentation method, where an S(ubject)-clause is defined as a group of words containing several predicates and their common subject. We propose an automatic S-clause segmentation method using decision trees. The S-clause information was shown to be very effective in analyzing long sentences, with an improved parsing performance of 5 percent. In addition, the performance in detecting the governor of subjects was improved by 32%.

Key words : S-clause segmentation, Decision trees, Syntactic analysis, Long sentence segmentation

1. 서 론

1.1 S-절이 왜 필요한가

입력 문장이 길수록 구문분석의 애매성이 급격히 증가하기 때문에 구문분석 성능은 떨어진다. 장문을 대상으로 구문분석한 후 본 연구실의 의존트리 구문분석기[1]의 오류를 6가지로 나누어 분석해 보면, 주어의 지배소를 판단하는 오류가 24.26%로서 두 번째로 큰 비중을 차지한다(표 1 참조). 실제로 가장 큰 비중을 차지한 오류는 명사구 내에서의 의존관계 오류이지만, 구문분석기를 이용한 많은 응용시스템들(예. 기계번역 시스템)이 명사구 전체를 하나의 단위로 취급하여 명사구 내부의 구문분석을 하지 않으므로 이 오류는 크게 중요하지 않다. 그러므로 이 논문은 주어의 지배소를 판단하는 오류를 해결하기 위한 방법을 제안한다.

대부분의 장문에서 용언의 개수보다 주어의 개수가 더 적다. 여러 용언이 하나의 주어를 공유하는 경우가 많고, 용언의 주어가 생략되어 문장에서 드러나지 않는 경우도 많기 때문이다. 보통 논항의 의존관계를 설정할 때, 지배소후보들의 하위범주화 및 선택제약 정보를 이용하는데, 다른 논항정보와는 달리 주어정보는 대부분 선택제약정보가 비슷하기 때문에 하위범주화 및 선택제약 정보를 이용하여 주어의 실제 지배소를 판단하기가 어렵다. 이러한 이유로 장문에서 주어가 포함되지 않은 절의 정확한 주어를 가려내기가 힘들다.

그림 1에서 제시한 예제문장을 살펴보자. 이 그림에서 사각형은 주어역할을 하는 단어들을 표시하고 있고, 밑줄은 용언을 표시한다. 이 예제문장은 4개의 주어와 14개의 용언을 포함하고 있다. 구문분석을 위해 그림 1의 예제문장에서 절을 인식한 후의 결과가 그림 2에 나타나 있다. 절 인식 후 주된 문제는, 여러 개의 절들이 하나의 주어를 공유하는 경우가 많은데 그 주어는 단 하나의 절 내에만 속해 있다는 것이다. 그림 2에서, 주어

† 학생회원 : 포항공과대학교 컴퓨터공학과
colorful@postech.ac.kr
†† 종신회원 : 포항공과대학교 컴퓨터공학과 교수
jhlee@postech.ac.kr
논문접수 : 2005년 1월 31일
심사완료 : 2005년 7월 22일

표 1 50,000개의 학습문장의 구문분석 오류 (14.42어절/문장)

구문트리 오류	주어의 지배소 판단 오류	용언의 지배소 판단 오류	부가어의 지배소 판단 오류	필수어(논항)의 지배소 판단 오류	명사구 내에서의 의존관계 오류	품사태깅 오류로 인한 구문분석 오류
오류(%)	24.26%	14.18%	17.49%	8.93%	27.37%	7.77%

둘은 모두 늙어서 시집을 갔는데, 언니는 운 좋게도 부잣집에 시집을 가게 되어 넓은 기와집에 편안히 앉아 비단옷에 고깃국을 마음껏 먹으면서 호강을 하였지만, 동생은 가난한 농가에 시집을 가서 고생을 하며 살게 되었습니다.

그림 1 예제문장

1. [둘은 모두 늙어서]
2. [시집을 갔는데,]
3. [언니는 부잣집에 시집을 가게 되어]
4. [운 좋게도]
5. [넓은]
6. [기와집에 편안히 앉아]
7. [비단옷에 고깃국을 마음껏 먹으면서]
8. [호강을 하였지만,]
9. [동생은 농가에 시집을 가서]
10. [가난한]
11. [고생을 하며]
12. [살게 되었습니다.]

그림 2 예제문장의 절 인식 결과

'둘은'은 첫 번째 절에 포함되어 있지만, 두 번째 절의 주어 또한 '둘은'이다. 한국어는 지배소 후위 언어이기 때문에 여러 개의 용언이 하나의 주어를 공유하는 경우 그 주어의 지배소는 마지막 용언이 된다. 따라서 '둘은'의 지배소는 '갔는데'이어야 한다. 이와 같은 경우로, 또 다른 주어 '언니는'은 세 번째 절에 포함되지만, 6번째, 7번째, 8번째 절들이 '언니는'을 그들의 주어로 요구하고 있다. 그러므로 '언니는'은 8번째 절의 용언인 '하였지만'에 의존하게 된다.

하나의 절 내에 주어가 생략되어 있을 때, Leffa[2]는 앞쪽에 가장 가까이 위치한 주어를 공유하고 있다고 가정하고 그 주어의 자질을 절 앞의 접속사에 표시하였다. 하지만 절의 생략된 주어가 항상 바로 앞 절의 주어와 일치하는 것은 아니다. 그러므로 절의 정확한 주어를 판단하는 새로운 방법이 필요하다.

이 논문은 "S(ubject)-절(clause)"이라는 새로운 개념을 소개하고, S-절을 하나의 주어와 그 주어를 공유하는 단어들 그룹으로 정의한다. 절(clause)은 용언의 하위범주화 정보에 기반하여 용언을 중심으로 용언의 의존소들을 묶어놓은 그룹이지만, S-절은 주어를 중심으로 S-절의 구간을 분할한다. 우리는 S-절 정보를 이용하여 구문분석에 있어서 주어의 지배소를 찾는 데 중의성을 해소할 수 있다.

1.2 S-절의 예

그림 1에서 제시한 예제문장을 대상으로 S-절을 인식해 보자. 그러기 위해서는 우선 각 용언의 실제 주어가 무엇인지를 알아야 한다. 예제문장에 등장한 4개의 주어 중 각 용언의 실제 주어를 선택하여 표시한 결과는 그림 3과 같다. '넓은', '가난한'과 같은 관형형 용언의 실제 주어역할을 하는 명사들은 예제문장 내에 존재하는 4개의 주어에 속하지 않기 때문에, 이들의 주어는 그림 3에서 따로 표시하지 않았다.

둘은 모두 늙어서 시집을 갔는데, 언니는 운 좋게도 부잣집에 시집을
주어:둘은 주어:둘은 주어:운
가게 되어 넓은 기와집에 편안히 앉아 비단옷에
주어:언니는 주어:언니는 주어: 언니는
고깃국을 마음껏 먹으면서 호강을 하였지만, 동생은 가난한 농가에
주어: 언니는 주어: 언니는
시집을 가서 고생을 하며 살게 되었습니다.
주어:동생은 주어:동생은 주어:동생은 주어:동생은

그림 3 각 용언의 실제 주어 표시

공통된 주어를 공유하는 단어의 그룹을 만들어 보면, 그림 4에서 표시한 것과 같은 4개의 S-절을 획득할 수 있다. 여기에서 두 번째 S-절은 세 번째 S-절을 내포하고 있고, 두 번째 S-절과 네 번째 S-절은 중문을 형성한다. 본 논문은 이와 같은 S-절의 분할을 자동적으로 하는 방법을 제안하고 구문분석에 있어서 S-절의 효과를 실험을 통하여 보여준다.

이 논문은 다음과 같이 구성되어 있다. 2장에서는 문장 분할과 절 인식에 관련된 기존 연구를 설명하고, 3장에서는 자동적으로 S-절을 인식하는 방법을 설명한다.

1. [둘은 모두 늙어서 시집을 갔는데,]
2. [언니는 부잣집에 시집을 가게 되어 넓은 기와집에 편안히 앉아 비단옷에 고깃국을 마음껏 먹으면서 호강을 하였지만,]
3. [은 좋게도]
4. [동생은 가난한 농가에 시집을 가서 고생을 하며 살게 되었습니다.]

그림 4 예제문장의 S-절 분할 결과

4장에서는 S-절을 사용한 구문분석 방법을 논한다. 그리고, 5장에서는 실험결과를 통하여 제안된 S-절 분할 방법이 구문분석에 있어서 효과적임을 보인다. 마지막으로 결론이 이어진다.

2. 기존 연구

장문의 구문분석에 관련된 많은 연구가 기존에 이루어졌다. 첫 번째로, 장문에 주로 등장하는 대등구조문의 인식 방법이 제안되었다[3-6]. 이들은 대등문은 구조적으로 병렬적이고 어휘적으로 유사하다는 점을 이용하여 대등문의 구간 인식을 시도하였다. 이 논문들은 대등문의 구간을 정하는 데는 좋은 성능을 보이나 하나의 주어를 공유하는 용언들의 구간을 판단하는 데는 도움이 되지 못한다.

장문의 분석에 관련된 두 번째 연구들로서 절의 분할을 들 수 있다[7-9]. 절의 분할은 용언의 하위범주화 정보에 기반하여 용언과 용언의 의존소를 묶음으로써 이루어진다. 절의 분할의 주된 문제점은 여러 절들이 같은 주어를 공유하는 경우 주어는 하나의 절 내에만 속하게 되므로, 이 주어를 공유하는 절이 어디까지인지 파악하기 어려울 뿐더러 표시할 방법도 없다는 것이다[2]. 앞에서 언급했듯이 절 안에 주어가 없는 경우, Leffa[2]는 앞쪽에 등장한 가장 가까운 주어를 절 앞의 접속사에 자질로 덧붙여서 표시하는 방법을 제안했다. 그러나, 절의 주어가 항상 앞쪽의 가장 가까운 주어일 수가 없으므로 절의 정확한 주어를 판단하기 위한 새로운 방법이 요구된다.

이외에도 장문의 분할에 관련된 많은 연구들이 있다. 어떤 논문들은 패턴과 규칙을 사용하여 장문분할을 한 후, 분할된 각 단위를 독립적으로 분석하는 방법을 제안한다[10-13]. 유사한 방법으로, 기계학습을 이용한 문장내 분할 방법이 제안되었다[14]. 이러한 방법들은 장문을 분할함으로써 구문분석의 복잡성을 줄일 수는 있으나 주어의 지배소를 결정하는 애매성은 여전히 해결하지 못하고 있다. 더욱이 Lyon and Dickerson[15]은 평서문은 거의 항상 세 개의 인접한 부분(주어 앞부분, 주어부, 서술부)으로 나뉘어진다는 점을 이용하여 세 개의 부분으로 분할한 후 구문분석하여 영어문장의 구문분석 복잡도를 줄였다. 이러한 방법은 하나의 주어부와 하나의 서술부를 포함하는 단문에서는 유용하겠으나, 일반적으로 하나 이상의 주어부와 서술부를 포함하는 장문을 구문분석하는 데 있어서는 비효과적이다. 위에서 살펴본 바와 같이, 장문분할에 관련된 기존 연구들은 많았으나 공통된 주어를 공유하는 용언들의 구간을 인식하는 데는 거의 관심이 없었다.

주어를 포함하지 않은 절의 주어를 정확히 판단하기 위해서, 우리는 S(ubject)-절(clause)이라는 개념을 도입하고, 자동적으로 S-절을 분할하는 방법을 제안한다. 기존 연구에서 절(clause)은 하나의 용언을 중심으로 인식되었다. 이와 대조적으로, 본 논문에서는 문장을 주어를 중심으로 분할한다. 이전 연구에서의 절 개념과 구분하기 위하여, 본 논문에서 제안하는 방법으로 분할된 단위를 S(ubject)-절이라고 명명한다.

3. S-절 분할

3.1 전체적인 구문분석 과정

이번 장에서는 전체적인 구문분석 과정을 개괄적으로 설명하겠다.

(1) 명사구 구묶음(Chunking)
(2) 절 분할 (분할된 절의 정보를 이용하여, 논항(주어 제외)의 지배소를 결정한다.)
(3) 2단계 S-절 분할
  (3.1) S-절 경계의 후보를 줄임
  (3.2.) 결정트리를 이용한 자동적인 S-절 분할
(4) S-절에 기반한 구문분석

우선, 김미영 외[16]의 방법을 이용하여 명사구의 구묶음을 수행한다. 그 다음으로 하위범주화와 선택제약정보를 이용하여 주어를 제외한 논항의 지배소를 결정한다. 이 과정은 절을 인식하는 과정과 유사하므로 절 분할 과정이라고 기술한다.

절 분할 과정에서 각 용언과 그들의 의존소를 선택제약정보에 의거하여 묶는다. 이 때, 격조사가 붙은 체언에 대하여 먼저 지배소 용언을 결정하고, 그 후 용언들의 논항슬롯 중 아직 채워지지 않은 격들을 대상으로 격조사가 붙지 않은 미지격 체언의 지배소 용언 결정 및 격할당을 하게 된다. 미지격 체언의 격할당 과정은 아래와 같이 이루어진다.

격조사가 붙어있지 않은 체언(미지격 체언)의 격을 알기 위해서는, 미지격 체언의 지배소 용언을 찾아내서 그 용언이 어떤 격을 요구하고 있는지를 알아야 한다. 격조사가 붙은 체언에 대한 지배소를 먼저 설정하여, 지배소가 요구하는 논항들 중 이미 채워진 논항슬롯은 미지격 체언의 논항후보로 제외된다. 미지격 체언의 지배소 용언을 찾는 방법은 다음과 같다. 지배소 후보가 되는 용언들의 선택제약정보를 이용하여, 용언들이 요구하는 논항들 중 아직 채워지지 않은 논항슬롯들의 의미와 미지

격체언의 의미와의 유사도를 측정하여, 가장 유사도가 큰 개념을 논항으로 요구하는 용언을 이 체언의 지배소로 설정하고, 그 때의 논항격을 이 미지격체언의 격으로 설정한다.

지배소후보 용언들이 요구하는 논항들의 의미와 미지격체언의 의미와의 개념유사도가 모두 0일 때에는, 이 입력체언을 필수어(논항)가 아닌 부가어로 간주하고 뒤의 부가어-용언 의존관계 처리하는 과정에서 지배소를 설정한다. 이를 정리하면 그림 5와 같다.

의미를 나타내는 방법으로는 카도가와 개념코드를 사용하며, 개념유사도 계산 방법은 Kim93[17]이 제안한 그림 6과 같은 방법으로 계산한다.

아래는 미지격 체언의 격할당의 예문이고, 예문 내의 용언과 체언의 사전정보는 아래와 같이 기술되어 있다.

```
미지격체언 z의 논항격 및 지배소 구하기

SIM(논항 x, 용언 y, 체언 z)  : 용언 y가 요구하는 논항 x의 의미부류와 입력체언 z의
                               의미부류 사이의 개념유사도

for each(지배소후보용언 y)
  for each(y의 논항들 중 빈 논항슬롯 x)
{
     SIM(x,y,z)를 계산한다.
}
     MAX := SIM(x,y,z)값들 중  최대값
     MAX_Y := 최대값을 가질 때의 SIM(x,y,z)의 용언 y
     MAX_Y := 최대값을 가질 때의 SIM(x,y,z)의 논항 x

// 같은 max값을 가진 용언이 여러 개인 경우, 체언 z와 가장 가까운 용언을 선택
// 선택된 지배소용언 y에서 같은 max값을 도출하는 논항 x가 여러 개인 경우,
// 주격논항> 목적격논항> 부사격논항 순으로 우선순위

If(MAX> 0.0)
  체언 z의 격 := MAX_X 으로 설정  //← 이때, 미지격의 격할당이 이루어짐
  if(체언 z의 격이 주격이 아니면)  // 주어일 때는 지배소설정의 오류를 줄이기 위해,
                                  //지배소를 S-절 분할 후에 결정한다.
     지배소 := MAX_Y
Otherwise
  체언 z를 부가어로 간주
```

그림 5 미지격체언의 격할당 과정

$$sim(C_1,C_2)=\frac{2\times level(MSCA(C_1,C_2))}{level(C_1)+level(C_2)}\times weight$$

※ *MSCA* : 가장 가까운 공통된 조상(ancestor)노드
※ $sim(C_1,C_2)=0$ if $MSCA(C_1,C_2)$ is level 1

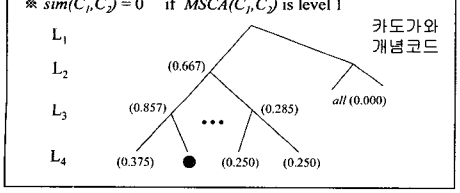

그림 6 개념유사도 계산 방법

실제로 의미정보 기술은 카도가와 개념코드인 3자리 숫자로 이루어져 있으나, 여기서는 이해를 돕기 위하여 한글어휘로 표현했다.

예문) 그는 밥도 제대로 못 먹고 학교에 갔다.

```
<그> :  의미부류정보 - 사람
<밥> :  의미부류정보-- 음식
<학교> : 의미부류정보 - 건물
<먹다> : 선택제약정보 - 주어 (생물)
                      - 목적어 (음식물)
<가다> : 선택제약정보 - 주어 (생물)
                      - 부사어[~에]  (장소,건물)
```

<사전정보>

1) 우선, 격조사가 붙은 논항을 대상으로 우선 지배소를 설정하면, '학교에 ← 갔다'라는 의존관계가 '가다'에 대한 사전의 선택제약정보에 의해 먼저 결정된다.
2) 문장 내 미지격체언인 '그는'과 '밥도'의 격을 결정한다. 우선, '그는'을 대상으로 개념유사도 값을 계산하여 비교하면 다음과 같다.

A. <그>가 <먹다>의 '주어'역할을 할 경우
   SIM(그, 주격, 먹다) = <사람>과 <생물>의 개념유사도 = 0.857
B. <그>가 <먹다>의 '목적어'역할을 할 경우
   SIM(그, 목적격, 먹다) = <사람>과 <음식>의 개념유사도 = 0.0
C. <그>가 <가다>의 '주어' 역할을 할 경우
   SIM(그, 주격, 가다) = <사람>과 <생물>의 개념유사도 = 0.857
D. <그>가 <가다>의 '부사어' 역할을 할 경우
   이미 <가다>의 '부사어' 논항슬롯이 '학교'로 채워졌으므로 불가

따라서 A의 경우가 가장 유사도 값이 컸으므로, <그는>은 '주격'으로 결정된다. 하지만 주어의 지배소를 절분할 과정에서 결정시에 오류가 많았으므로, S-절 분할이 이루어지기까지 지배소결정은 미루어둔다.

위와 같은 방법으로, <밥도>의 지배소 및 격을 결정하기 위해 각각 개념유사도를 계산해보면 다음과 같다.

A. <밥>이 <먹다>의 '주어'역할을 할 경우
   SIM(밥, 주격, 먹다) = <음식>과 <생물>의 개념유사도 = 0.0
B. <밥>이 <먹다>의 '목적어'역할을 할 경우
   SIM(밥, 목적격, 먹다) = <음식물>과 <음식물>의 개념유사도 = 1.0
C. <밥>이 <가다>의 '주어' 역할을 할 경우
   SIM(밥, 주격, 가다) = <음식물>과 <생물>의 개념유사도 = 0.0
D. <밥>이 <가다>의 '부사어' 역할을 할 경우
   이미 <가다>의 '부사어' 논항슬롯이 '학교'로 채워졌으므로 불가

B의 경우의 값이 가장 컸으므로, '밥도'의 지배소는 '먹다'가 되고, 이 때의 논항격은 목적어가 할당된다.

위의 절 분할 과정이 끝나면, 주어의 지배소를 찾아주기 위해서 2단계를 통한 S-절 분할 방법이 적용된다. 마지막으로 S-절을 이용하여 구문분석을 수행한다. 아래의 장에서, S-절 인식을 위한 두 단계의 방법을 설명하고, 4장에서 S-절을 기반으로 구문분석을 어떻게 수행하는지를 자세히 설명한다.'

3.2 두 단계를 통한 S-절 분할

3.2.1 단계 1: S-절 경계 후보 줄이기

S-절 분할을 하기 위해서 좌우 구간을 결정해야 한다. 그러기 위해서, 우리는 S-절에 포함되어야 할 절들을 선택해야 한다. 절의 중심어는 용언이므로 S-절을 인식하기 위해 용언의 위치를 S-절 구간 경계의 후보로 삼는다.

장문 속에는 많은 용언이 있기 때문에(50,000개 문장의 학습데이타 내에 평균적으로, 6.60용언/문장, 14.42단어/문장, 2.09주어/문장이 존재한다. 데이타 선정은 5장에 기술되어 있다.), S-절 경계가 되는 용언 하나를 선택하는 데 애매성이 존재한다. 그래서, 우리는 경계 후보를 줄이기 위해서 아래의 두 과정을 수행한다.

(1) 동사구 구묶음

(2) 관계절 틈(gap) 인식

우선, 우리는 동사구 구묶음을 실시한다[16]. 만약 여러 개의 용언이 인접하여 위치해 있는 경우, 동사구 구묶음 규칙에 의해 이것들을 하나의 그룹으로 묶는다. 그리하여, 동사구를 하나의 단위로 취급함으로써 용언의 수가 줄어들게 되므로 S-절 경계의 후보를 줄일 수 있다.

두 번째로는 관계절의 틈을 인식한다. 관계절의 틈은 용언의 선택제약정보와 '절 하나에 한 종류의 논항'(one argument per clause) 규칙에 의해 인식된다. 뒤따르는 명사를 수식하는 관형형 용언의 경우, 만약 뒤따르던 명사가 용언의 주어 역할을 하면, 관형형 용언은 오른쪽 경계의 후보에서 제외된다. 그 이유는 다음과 같다. 만약 어떤 용언이 오른쪽 경계가 되려면, 그 용언을 앞서는 단어 중 하나를 주어로 요구해야 한다. 따라서 만약 관형형 용언이 뒤따르는 명사를 주어로 요구한다면, 이 용언은 앞서 존재한 주어의 오른쪽 경계가 될 수가 없다. 이와 같이, 동사구 묶음과 관계절의 틈 인식을 통해 S-절의 경계의 후보를 줄일 수 있다.

3.2.2 단계 2: 결정트리를 이용한 S-절 분할

S-절 분할을 위해 어떤 기계학습 방법이라도 적용이 가능하지만, 본 논문에서는 결정트리를 사용하여 실험한다. 결정트리 학습은 결과가 트리로 표현된다. 트리는 또한 if-then 규칙으로 표현될 수 있으므로 학습된 결과를 사람이 쉽게 이해할 수 있고 학습에 영향을 미치는 중요한 자질이 무엇인지를 쉽게 알 수 있는 장점이 있다. 결정트리 알고리즘은 자연언어처리에 있어서 성공적으로 적용되어 왔던 알고리즘으로서, 구문분석[18], 담화분석[19], 단어 분할[20] 등에 쓰여왔다. 우리는 S-절 분할을 위한 학습 알고리즘으로서 C4.5[21] 결정트리 프로그램을 사용한다.

S-절 분할을 위하여 주어(지금부터, S-절에 포함되는 한 개의 주어를 '목표주어'라고 명명한다.), 왼쪽 경계 후보, 오른쪽 경계 후보를 고려한다. S-절 내부에는 주어가 하나 들어있다고 가정하므로, 목표주어와 왼쪽의 인접한 주어 사이의 용언들이 왼쪽 경계의 후보가 되고, 목표주어와 오른쪽 인접한 주어 사이의 용언들이 오른쪽 경계의 후보가 된다.

그림 7은 목표주어를 기준으로, S-절의 좌우 경계가 될 수 있는 용언들의 위치표시를 한 것이다. 왼쪽 경계의 후보 위치인 -1인 경우에는, -1 위치에 있는 용언을 중심으로 하는 절이 S-절의 왼쪽 경계가 됨을 의미하고, 이와 마찬가지로 -2인 경우에는, -2 위치에 있는 용언을 중심으로 하는 절까지가 S-절의 왼쪽 경계가 되는 것을 의미한다. 이러한 위치정보는 실제 기계학습시 할당될 클래스의 값으로도 사용된다. 하지만 예외적으로 S-절의 왼쪽 경계의 경우에는 어떤 용언도 포함되지 않을 수 있기 때문에, 이런 경우에는 클래스의 값을 0을 부여한다. 이러한 경우에는 S-절 구간 경계인 '용언의 위치'를 표시할 수가 없다. 따라서 그림 7에서는 목표주어의 자리에 클래스 값 0을 표시하고, 이 때 S-절의 왼쪽 경계는 -1 위치의 용언의 바로 뒤가 된다.

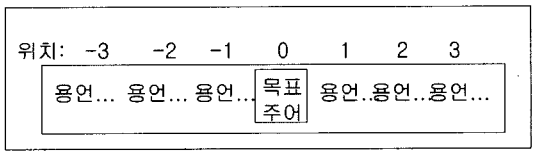

그림 7 목표주어 주변의 용언들의 위치 표시

표 2는 50,000개 문장의 학습데이타에서 S-절 경계의 위치분포를 나타낸 것이다. 이 표는 목표주어 가까이에서 왼쪽으로 2개, 오른쪽으로 7개의 용언들의 범위 내에 경계의 99%가 존재함을 보여준다. 따라서, 왼쪽경계의 후보로는 목표주어 왼쪽에 인접한 2개의 용언까지를 대상으로 하고, 오른쪽 경계의 후보로는 목표주어의 오른쪽에 인접한 7개의 용언들을 대상으로 한다.

우리는 우에서 좌로의 순서에 의해 입력 문장에서 S-절을 인식한다. 다시 말하면, 입력 문장의 마지막 주어를 포함하는 S-절을 먼저 인식한다. 그 이유는 다음과

표 2 S-절 경계가 되는 용언 위치의 분포 (50,000 학습데이타 문장)

용언의 위치	0	-1	-2	-3	-4	-5	-6	-7	-8	
왼쪽 경계 (%)	82.67%	15.21%	1.39%	0.41%	0.16%	0.04%	0.0%	0.0%	0.0%	…
용언의 위치	0	1	2	3	4	5	6	7	8	
오른쪽 경계 (%)	0.0%	60.34%	19.17%	9.30%	5.33%	2.71%	1.52%	0.83%	0.10%	…

같다: 내포된 S-절의 위치에 따라 내포문 S-절은 두 가지 유형으로 나눌 수 있다(그림 8 참조). 첫 번째 유형 (a)는 A(내포하는 S-절)의 주어 A'가 B(내포된 S-절)의 주어 B'보다 앞쪽에 위치하는 경우이고, 두 번째 유형인 (b)는 A(내포하는 S-절)의 주어인 A'가 B(내포된 S-절)의 주어인 B'보다 뒤에 존재하는 경우이다. 한국어에서는 내포문 S-절의 구조는 대부분 (a)의 경우이고, (b)의 경우는 보기가 어렵다. 실제로 50,000문장의 학습데이타에서도 모든 내포문 S-절이 (a)의 경우에 속해 있었다. 따라서, 만약 좌에서 우로 스캔하면서 등장하는 주어를 대상으로 S-절을 인식한다면 내포되어 있는 S-절을 인식 못할 수가 있으므로 우에서 좌로, 즉 제일 마지막 주어가 포함된 S-절을 먼저 인식하도록 한다. 하나의 S-절이 인식되면, 이미 인식된 S-절에 포함된 단어들을 입력 단어 스트링에서 제외한다. 그리하여 줄어든 입력 단어 스트링을 대상으로 다시 가장 마지막 주어를 찾아내어 반복적으로 S-절을 인식한다. 이러한 과정은, 계속 수정되어 입력되는 단어 스트링에 주어가 하나만 존재할 때까지 반복된다.

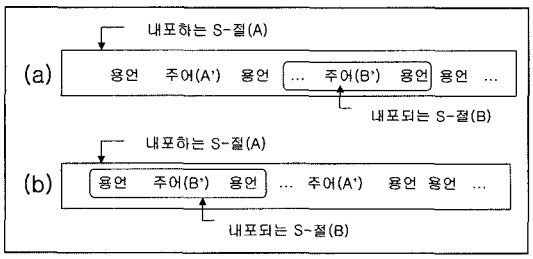

그림 8 내포하는 S-절의 구조

3.3 S-절 인식에 필요한 자질들

이미 언급한 바와 같이 S-절을 인식하기 위해서는 하나의 S-절에 포함되어야 할 절들을 선택해야 한다. 절들의 중심어는 용언이므로 용언의 정보를 자질로 사용한다.

표 3에서 보여지는 바와 같이, 용언에서 3가지 종류의 자질을 사용하게 된다. 첫 번째 자질은 용언의 어말형태와 관련된 것이다. 두 번째 자질은 용언의 어미의 어휘 정보를 이용한다. 한국어는 교착어이고, 용언의 어미는 다음 동사구와의 연결적 기능을 나타낸다(예를 들어, '으므로'는 다음 동사구를 위한 이유의 기능을 나타낸다.). 세 번째 자질로서 용언 뒤의 쉼표가 사용된다. 문장에서 쉬기 위해 추가되는 쉼표의 사용은 S-절의 경계를 판단하기 위한 유용한 특징이 된다. 각 자질에 대한 상세한 값들은 표 4에 나타나 있다.

표 3 학습에 사용되는 자질들

첫 번째 자질	용언의 어말형태
두 번째 자질	용언 어미의 어휘
세 번째 자질	쉼표

왼쪽 경계를 판단하기 위해 표 3에서 살펴본 3개의 자질을 목표주어 왼쪽의 2개의 용언에 대하여 추출한다. 각 학습데이타에서 왼쪽 경계를 표시하기 위한 클래스 값은 3가지가 존재한다(0~2). 만약 클래스 값이 0이면, S-절 내에 목표주어의 앞쪽에 위치하는 용언이 하나도 포함되어 있지 않다는 것을 의미한다. 이와 유사하게 만약 클래스 값이 1이면, 목표주어의 앞쪽에 가장 가까운 용언 한 개가 S-절에 포함됨을 의미한다.

표 4 각 자질에 대한 값들

자질 종류	자질값
첫 번째	관형형, 연결형, 인용형, 명사형, 종결형
두 번째	ㄴ, ㅁ, 기, 음, ㄴ데, ㄴ즉, ㄴ지, ㄹ지, ㄹ지니, 거나, 게, 고, 나, 는데, 는지, 니, 다가, 도록, 든지, 듯이, 라, 려고, 며, 면, 면서, 므로, 아, 아서, 어, 어도, 어서, 어야, 으나, 으니, 으려고, 으며, 으면, 으면서, 으므로, 자, 지, 지마는,… (모든 연결어미)
세 번째	1, 0

3.4 결정트리를 이용한 S-절 인식의 예

그림 1의 예제문장을 대상으로, S-절 분할까지의 구문분석 과정을 예로 들어 보이도록 한다.

예제문장 :

둘은 모두 늙어서 시집을 갔는데, 언니는 운 좋게도 부잣집에 시집을 가게 되어 넓은 기와집에 편안히 앉아 비단옷에 고깃국을 마음껏 먹으면서 호강을 하였지만, 동생은 가난한 농가에 시집을 가서 고생을 하며 살게 되었습니다.

(1) 명사구 구묶음

김미영 외[16]의 방법을 이용하여, 명사구 구묶음을 수행한 결과는 다음과 같다.

넓은+기와집에

가난한+농가에

(2) 절 분할

절 분할 결과는 아래와 같다.

[둘은 모두 늙어서]
[시집을 갔는데,]
[언니는 부잣집에 시집을 가게_되어]
[운 좋게도]
[넓은]
[기와집에 편안히 앉아]

<단계 1: S-절 경계 후보 줄이기>

우선, 동사구 구묶음 후에 구묶음된 동사구 2개가 다음과 같이 만들어진다.

구묶음된 동사들 : 가게+되어
살게+되었습니다.

두 번째로, 관계절 틈(gap) 인식 후에 S-절의 오른쪽 경계후보에서 제외되는 관형형 용언은 아래의 두 개이다.

S-절의 오른쪽 경계에서 제외되는 관형형 용언
: 넓은
가난한

<단계 2: 결정트리를 이용한 S-절 분할>

1> 첫 번째 S-절 인식 과정

S-절 인식은 우에서 좌로 수행되므로, 문장에서 가장 오른쪽에 위치한 주어인 '동생은'이 첫 번째 목표주어로 설정된다.

첫 번째 목표주어: 동생은

목표주어를 포함하는 S-절의 좌우 경계후보 용언들의 위치는 다음과 같이 나타낼 수 있다.

운 좋게도 가게_되어 넓은 앉아 먹으면서 하였지만
-5 -4 (제외) -3 -2 -1
동생은 가난한 가서 하며 살게_되었습니다.
0 (제외) +1 +2 +3

결정트리에 의한 경계 인식 결과가 다음과 같이 정확하게 도출되었다고 가정하자.

왼쪽 경계 인식 결과: 0

오른쪽 경계 인식 결과: +3

그렇다면 얻어지는 S-절은 다음과 같다.

첫 번째 인식된 S-절 :
[동생은 가난한 농가에 가서 시집을 가서 고생을 하며 살게_되었습니다.]

획득된 S-절에 포함된 단어들을 예제문장에서 제외한 후, 수정된 문자열을 대상으로 반복적으로 위의 과정을 수행하여 두 번째 S-절을 인식한다.

2> 두 번째 S-절 인식 과정

수정된 입력문자열은 다음과 같다.

수정된 입력문자열:
둘은 모두 늙어서 시집을 갔는데, 언니는 운 좋게도 부잣집에 시집을 가게 되어 넓은 기와집에 편안히 앉아 비단옷에 고깃국을 마음껏 먹으면서 호강을 하였지만,

두 번째 목표주어를 찾으면, 가장 마지막 주어인 '운'으로 설정된다.

두 번째 목표주어: 운

목표주어를 포함하는 S-절의 좌우 경계후보 용언들의 위치는 다음과 같이 나타낼 수 있다.

동생은 운 좋게도 가게_되어 넓은 앉아 먹으면서
0 +1 +2 (제외) +3 +4
하였지만
+5

결정트리에 의한 경계 인식 결과가 다음과 같이 정확하게 도출되었다고 가정하자.

왼쪽 경계 인식 결과: 0

오른쪽 경계 인식 결과: +1

그렇다면 얻어지는 S-절은 다음과 같다.

두 번째 인식된 S-절 :
[운 좋게도]

획득된 S-절에 포함된 단어들을 제외한 문자열들을 대상으로, 반복적으로 위의 과정을 수행하여 세 번째 S-절을 인식한다.

3> 세 번째 S-절 인식 과정

수정된 입력문자열은 다음과 같다.

수정된 입력문자열:
둘은 모두 늙어서 시집을 갔는데, 언니는 부잣집에 시집을 가게 되어 넓은 기와집에 편안히 앉아 비단옷에 고깃국을 마음껏 먹으면서 호강을 하였지만,

세 번째 목표주어를 찾으면, 가장 마지막 주어인 '언니는'으로 설정된다.

세 번째 목표주어: 언니는

목표주어를 포함하는 S-절의 좌우 경계후보 용언들의 위치는 다음과 같이 나타낼 수 있다.

둘은 늙어서 갔는데 언니는 가게_되어 넓은 앉아
-2 -1 0 +1 (제외) +2
먹으면서 하였지만,
+3 +4

결정트리에 의한 경계 인식 결과가 다음과 같이 정확하게 도출되었다고 가정하자.

왼쪽 경계 인식 결과: 0

오른쪽 경계 인식 결과: +4

그렇다면 얻어지는 S-절은 다음과 같다.

세 번째 인식된 S-절 : [언니는 부잣집에 시집을 가게 되어 넓은 기와집에 편안히 앉아 비단옷에 고깃국을 마음껏 먹으면서 호강을 하였지만,]

획득된 S-절에 포함된 단어들을 제외한 문자열들을 대상으로, 반복적으로 위의 과정을 수행하여 네 번째 S-절을 인식한다.

4> 네 번째 S-절 인식 과정

수정된 입력문자열은 다음과 같다.

수정된 입력문자열: 둘은 모두 늙어서 시집을 갔는데,

남은 입력문자열에는 주어성분이 하나만 존재하기 때문에, 더 이상의 S-절 인식알고리즘은 수행되지 않고, 남은 입력문자열을 하나의 S-절로 인식하고, S-절 분할 과정은 종료된다. 따라서 자동적으로 아래의 S-절이 획득된다.

네 번째 인식된 S-절 : [둘은 모두 늙어서 시집을 갔는데,]

4. S-절에 기반한 구문분석

S-절 분할 후에 구문분석 시스템은 각 S-절 단위로 의존관계를 분석한다. 의존관계를 결정하는 과정은 다음과 같다.

(1) S-절 내의 의존관계 설정 (1.1) 주어-용언간의 의존관계 설정 (1.2) 부가어-용언간의 의존관계 설정 (2) S-절 간의 의존관계 설정 (2.1) 각 S-절 내의 가장 마지막 용언들 간의 의존관계 설정

전체적인 구문분석 과정은 위와 같이 두 단계로 나뉘어진다. 우선, S-절 내의 의존관계를 설정한다. 그 다음, S-절 간의 의존관계를 설정한다.

먼저, S-절 내의 의존관계를 설정하는 첫 단계에서 주어-용언간의 의존관계 설정에 대해 알아보자. 3.1에서 언급한 바와 같이, S-절 분할 이전에 절 인식 단계에서 이미 논항(주어 제외)-용언간의 의존관계를 설정하였다. 따라서 S-절 분할 이후에는 주어-용언간의 의존관계 설정을 하면 된다. 주어-용언 의존관계에서는, 하나의 S-절 내의 주어는 S-절 내의 가장 오른쪽 용언을 지배소로 가지도록 의존관계를 설정한다. 왜냐하면 S-절 내의 모든 용언들이 '목표주어'를 그들의 주어로 요구하고, 한국어는 지배소 후위 언어이기 때문이다. 부가어-용언의 의존관계 설정시, 부가어는 링크가 겹쳐지지 않도록 하는 가장 가까운 용언에 의존하도록 설정한다.

다음으로, 아직 의존관계를 설정하지 못한 단어들로서 S-절 내의 가장 마지막 용언이 남아있다. 이 용언의 지배소를 설정하기 위해서는 S-절 간의 의존관계를 고려해야 한다. 따라서 S-절 간의 의존관계 설정 단계로 용언과 용언간의 의존관계를 설정하도록 한다. 이 관계의 설정시, 우리는 일본어 KN 구문분석기에서 사용된 Minami 규칙을 사용한다[5]. Minami는 절들 사이의 의존관계를 설정하기 위하여, 그림 9와 같은 규칙을 사용하였다. Minami는 다음과 같이 제안했다. 만약 절1이 절2에 내포되어 있으면, 절1의 마지막 용언인 용언1은 절2의 마지막 용언인 용언2에 의존한다. 우리는 이와 같은 Minami의 규칙을 S-절에 있어서의 용언의 의존 관계를 설정하는데 사용하되, 서로 대등하게 연결되어 있는 S-절들에 대해서는 새로운 규칙을 추가한다. 만약 S-절이 대등하게 연결되어 있으면, 이전 S-절의 마지막 용언인 용언1이 바로 뒤 S-절의 마지막 용언인 용언2에 의존한다. 이것은 그림 10에 나와있다.

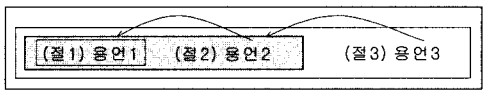

그림 9 내포절을 위한 우선규칙 (Minami)

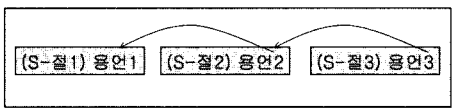

그림 10 일렬로 인접한 S-절 간의 우선규칙

5. 실 험

5.1 S-절 분할 성능과 이에 따른 구문분석 성능

우리는 제안된 S-절 분할 방법을 Matec99(제 1회 형태소 분석기 평가 대회)[22] 말뭉치의 설명문들과 KIBS (국어정보베이스)[1] 말뭉치의 역사, 건강 그리고 예술에 관련된 문장들을 뽑아서 실험한다. 학습데이타 사이즈는 5,000문장부터 50,000문장까지 다양하게 이루어져 있고, 테스트용 데이타로는 학습데이타와 분리하여 5,000문장(평균 18.82어절/문장)을 따로 추출하여 사용하였다.

구문분석기의 정확률은 (정확하게 분석된 구문분석 링크 수)/(모든 인식된 구문분석 링크 수)이고, 재현율은 (정확하게 분석된 구문분석 링크 수)/(모든 정확한 구문분석 링크 수)이다. 우리는 S-절 분할 프로그램의 아래

1) http://kibs.kaist.ac.kr

3가지 성질을 분석했다.

1. 경계 후보를 줄인 후 S-절 분할 성능 vs. 경계 후보를 줄이지 않은 S-절 분할 성능
2. S-절 분할 성능 vs. 구문분석 성능
3. S-절을 사용한 후 구문분석 성능 vs. S-절을 사용하지 않은 구문분석 성능

실험에서, 우리는 아래의 4가지 결과를 얻었다.

1. 경계의 후보를 줄인 후 S-절 분할 성능이 6% 향상되었다(표 5 참조).
2. S-절 분할 성능이 더 좋을수록, 구문분석 성능이 더 좋다(표 6 참조).
3. S-절 분할의 정확률이 83.28%일 때, 구문분석 정확률은 87.38%이다. 이 때 학습데이타 사이즈는 50,000 문장이고, 테스트데이타 사이즈는 5,000문장이다.
4. S-절 분할 결과는 약 5.1%의 구문분석 성능향상을 보였다(표 7 참조).

표 5에서, 베이스라인은 S-절 경계가 가장 많이 생기는 위치를 S-절 경계로 선택했을 때의 S-절 분할 성능을 의미한다. 다시 말하면, 오른쪽 경계는 '목표주어' 오른쪽의 첫 번째 용언으로 선택하고, 왼쪽 경계는 '목표주어' 왼쪽의 0번째 용언(즉, '목표주어'의 왼쪽으로 어떤 용언도 포함되지 않는다.)을 선택하여 S-절 분할을 했을 경우를 의미한다. 이것은 표 2에서 보여진 바와 같다.

표 5 S-절 구간 후보를 줄인 후의 성능 변화

S-절 분할 정확률	결정트리	베이스라인
구간 후보를 줄이지 않은 경우	77.01%	60.28%
구간 후보를 줄인 경우	83.28%	62.32%

주어의 지배소를 정할 때, S-절 분할을 사용하지 않는 경우의 구문분석기는 주어의 지배소는 링크의 교차를 허용하지 않는 가장 가까운 용언이라고 판단한다. 그 결과, S-절 분할을 했을 때 주어의 지배소를 찾는 데 있어서의 정확률이 S-절을 사용하지 않을 때보다 더 좋음을 표 7을 통해서 알 수 있다. 또한 용언 간의 의존관계의 성능을 S-절을 사용했을 때와 그렇지 않을 때로 비교해 본다. S-절을 사용하지 않은 용언 간 의존관계에 있어서 용언의 지배소는 링크의 교차를 허용하지 않는 가장 가까운 용언으로 설정한다. 표 7은 S-절에 기반한 용언간의 의존관계가 S-절을 사용하지 않았을 때보다 더 좋은 성능을 보임을 증명한다. 그 결과 구문분석 성능은 87.38%로 S-절 분할을 하지 않았을 때보다 5%의 성능향상을 보였다.

표 7 S-절 분할 후 구문분석 성능 변화

	S-절 분할을 하지 않은 파서의 성능	S-절 분할한 후 파서의 성능
주어의 지배소를 인식하는 정확률	69.03%	83.28%
용언의 지배소를 인식하는 정확률	58.42%	65.44%
구문분석 정확률	82.23%	87.38%

5.2 자질의 중요성

다음으로, 우리는 각 자질의 중요도를 실험해 본다. 표 8과 표 9는 각 자질을 제외했을 때 S-절의 성능변화를 보여준다. 표 8은 왼쪽 경계를 위한 가장 중요한 자질은 왼쪽의 첫번째 용언의 어말형태임을 보인다. 표 9는 오른쪽 경계의 가장 중요한 자질은 각 용언의 어말형태와 쉼표 정보임을 보인다. 왜냐하면 6번째와 7번째 용언을 제외한 모든 용언의 어말형태와 쉼표정보가 정확률에 영향을 주기 때문이다. 이 결과는 부분적으로 몇 가지 휴리스틱 규칙을 보여준다. 첫째, 왼쪽 경계가 가능한 용언의 어말형태는 관형형이었다. 왜냐하면 단지 관형형 용언만이 주어의 앞쪽에 등장이 가능하기 때문이다. (다른 어말형태의 용언들은 주어의 뒤쪽에 등장한다.) 이는 학습된 결정트리에서도 알 수 있었다. 두 번째로, 쉼표 뒤에 경계가 잘 나타난다. 그 외에도, 더 좋은 성능을 보이기 위해서 용언들의 어말형태 정보에 좀 더 관심을 기울일 필요가 있다. 표 8과 표 9를 토대로 다시 정리하자면, 대부분의 자질들이 구문분석의 성능에 이바지함을 보인다.

5.3 S-절 오류 분석

표 10에서 보이는 바와 같이, S-절 분할의 오류의 원인을 분석한다. 표 10은 대부분의 S-절 오류가 한국어의 특징 때문임을 보여준다.

S-절의 오류들 중에서, 주어를 인식하는 오류가 26.01%로 큰 비중을 차지한다. 비록 주격조사가 주어의 역할을 하는 단어를 표시해 주기는 하나, 대부분의 명사들은 격조사 없이 단독으로 쓰이는 경우도 많기 때문에

표 6 학습데이타 사이즈 vs. S-절 분할 성능 vs. 구문분석 성능

학습 데이타 사이즈	5000	10000	20000	30000	40000	50000
S-절 분할 정확률	79.74%	80.23%	81.90%	83.18%	83.26%	83.28%
S-절 분할 재현율	80.38%	80.54%	82.31%	83.23%	83.27%	83.31%
구문분석 정확률	85.69%	85.90%	86.94%	87.29%	87.32%	87.38%
구문분석 재현율	85.93%	86.18%	87.30%	87.55%	87.58%	87.60%

표 8 각 자질을 제외했을 경우, S-절의 왼쪽 경계 인식 정확률의 변화

자질	정확률 변화	자질	정확률 변화
첫 번째 용언의 어말형태	7.34 %	두 번째 용언의 어말형태	-0.22 %
첫 번째 용언의 어미	0.12 %	두 번째 용언의 어미	-0.09 %
첫 번째 용언의 쉼표	0.03 %	두 번째 용언의 쉼표	-0.02 %

표 9 각 자질을 제외했을 경우, S-절의 오른쪽 경계 인식 정확률의 변화

자질	정확률 변화	자질	정확률 변화
첫 번째 용언의 어말형태	-9.71 %	다섯 번째 용언의 어말형태	-0.00 %
첫 번째 용언의 어미	-2.36 %	다섯 번째 용언의 어미	+0.79 %
첫 번째 용언의 쉼표	-0.78 %	다섯 번째 용언의 쉼표	-0.79 %
두 번째 용언의 어말형태	-0.26 %	여섯 번째 용언의 어말형태	-0.08 %
두 번째 용언의 어미	-0.78 %	여섯 번째 용언의 어미	+0.79 %
두 번째 용언의 쉼표	-0.52 %	여섯 번째 용언의 쉼표	+0.26 %
세 번째 용언의 어말형태	-0.10 %	일곱 번째 용언의 어말형태	+0.79 %
세 번째 용언의 어미	+1.05 %	일곱 번째 용언의 어미	+0.53 %
세 번째 용언의 쉼표	-1.83 %	일곱 번째 용언의 쉼표	-0.03 %
네 번째 용언의 어말형태	-0.00 %		
네 번째 용언의 어미	+1.05 %		
네 번째 용언의 쉼표	-1.83 %		

표 10 S-절 분할 오류 분석

S-절 분할 오류	주어 인식 오류	품사태깅 오류	이중주어에 의한 오류	S-절의 왼쪽 경계인식 오류	S-절의 오른쪽 경계인식 오류	부사어의 서술어 역할에 의한 오류	그 외
오류	26.01%	20.66%	11.06%	8.10%	29.12%	4.00%	1.05%

이런 경우 격을 결정하는 데 애매성이 존재한다. 다음으로, 품사태깅 오류가 S-절 분할 오류의 20.66%를 차지한다. 이러한 두 문제들은 S-절 분할 전에 해결되어야 한다. 그래서, 이것들은 또다른 이슈로 남겨둔다.

또한, 이중주어관련 오류가 11.06%이다. 몇몇 한국어 용언들은 이중주어문을 형성할 수 있다. 이것은 주어를 한 개 포함한다는 S-절의 가정에 위배되는 것으로, 이러한 이중주어문의 경우 S-절 분할이 실패한다. 그림 11과 그림 12에서 이중주어문에 의한 S-절 분할 오류 예를 보여준다. 그림 12의 두 번째 S-절을 살펴보면, '성분이', '효과가' 등 2개의 주어를 포함하고 있음을 알 수 있다. 이는 '있음을'이라는 용언이 이중주어문을 가질 수 있기 때문에 생긴 현상이다. 11.06%는 전체 오류 중 무시할 수 없는 큰 숫자이므로 우리는 이중주어문을 고려하여 이중주어문을 형성하는 용언들의 특징들을 파악한 후 따로 처리가 필요하다. 표 10은 또한 오른쪽 경계를 인식할 때가 왼쪽 경계 인식보다 더 오류가 많음을 보여준다. 즉, 오른쪽 경계 파악이 더 어렵다는 말이다. 마지막으로 용언이 아닌 몇몇 부사어들 또한 주어의 서술어로 기능할 수 있다. 우리는 단지 용언에 기반하여 경계를 인식하기 때문에 이러한 부사어는 S-절 경계 후보에서 제외된다. 그림 13과 그림 14에서 부사어의 서술어 기능 때문에 생긴 S-절 분할 오류 예를 보여준다.

1920년 맥켈럼과 그의 동료들은 구루병을 치료하는 것으로 알려졌던 대구 간유의 성분이 안구 건조증과 구루병에 모두 효과가 있음을 밝혔다.

그림 11 예제문장 <2>

1. [1920년 맥켈럼과 그의 동료들은 밝혔다.]
2. [구루병을 치료하는 것으로 알려졌던 대구 간유의 성분이 안구 건조증과 구루병에 모두 효과가 있음을]

그림 12 예제문장 <2>의 S-절 분할 결과

이 소설의 주인공은 어린 소녀로, 그녀가 바라보는 한국전쟁이 상세히 기술된다.

그림 13 예제문장 <3>

1. [이 소설의 주인공은 어린 소녀로,]
2. [그녀가 바라보는]
3. [한국전쟁이 상세히 기술된다.]

그림 14 예제문장 <3>의 S-절 분할 결과

그림 13의 첫 번째 S-절에서 서술어 역할을 하는 것은 '소녀로,'라는 부사어이다. 현재 S-절 분할 시스템에서는

'소녀로'라는 부사어를 구간 경계 후보에 포함시키지 않았기 때문에, '이 소설의 주인공은 어린' 까지가 하나의 S-절로 분할되어 잘못된 결과를 생성하게 된다. 따라서 우리는 이와 같이 서술어 기능이 가능한 부사어들을 S-절 경계후보에 포함시킬 필요가 있다.

6. 결 론

이 논문은 장문에 있어서 구문적 애매성을 줄이기 위해 S-절이라는 새로운 개념을 제안하고, 2단계 S-절 분할 방법을 제안하였다. S-절은 여러 개의 용언과 하나의 공통된 주어를 포함하는 단어그룹으로 정의된다. 우리는 S-절 분할을 위한 두 단계 방법을 제안한다. 첫번째 단계로서, 동사구 묶음과 관계절의 틈을 인식하여 S-절 경계의 후보를 줄인다. 두 번째 단계로, 결정트리를 이용한 S-절 분할 방법을 수행한다. 실험결과는 S-절이 주어와 용언 각각의 지배소를 결정하는데 유용함을 보이고 있다. S-절 분할 후에, S-절을 사용한 구문분석기가 S-절을 사용하지 않은 것보다 5%의 성능향상을 보이고 있다. 향후 작업으로 S-절의 정확률을 높이기 위해, 이중주어문 구조와 서술어로 기능하는 부사어의 특성을 파악할 필요가 있다.

앞으로 이 연구를 두 가지 방향으로 계속해 나갈 생각이다. 첫번째로, S-절 분할 방법을 다른 언어에도 적용해본다. 왜냐하면 대부분의 언어들은 장문에서 하나의 주어를 공유하는 경향이 있고, 이러한 용언들의 구간을 파악하는 데 있어서 애매성이 존재하기 때문이다. 두 번째로, 우리는 기계번역 시스템에 S-절을 도입하여, S-절 단위로 문장을 번역하는 것에 대해 연구해 볼 예정이다.

참 고 문 헌

[1] M. Kim, S.J. Kang, and J.H. Lee, "Resolving Ambiguity in Inter-chunk Dependency Parsing," Proc. 6th Natural Language Processing Pacific Rim Symposium, Tokyo, Japan, pp. 263-270, 2001.

[2] V. J. Leffa, "Clause processing in complex sentences," Proc. 1st International Conference on Language Resources and Evaluation, Granada, Spain, pp.937-943, 1998.

[3] R. Agarwal, L. Boggess, "A simple but useful approach to conjunct identification," Proc. 30th Annual Meeting of the Association for Computational Linguistics, Nantes, France, pp.15-21, 1992.

[4] 장재철, 박의규, 나동렬, "구간분할 기반 한국어 대등접속 구문분석 기법", 제 14회 한글 및 한국어 정보처리 학술대회, 청주, pp.139-146, 2002.

[5] S. Kurohashi and M. Nagao, "A syntactic analysis method of long japanese sentences based on the detection of conjunctive structures," Computational Linguistics, vol.20, no.4, pp.507-534, 1994.

[6] 윤준태, 송만석, "한국어의 대등접속구문 분석", 정보과학회논문지, 24:326-336, 1997.

[7] X. Carreras, L. Marquez, V. Punyakanok, and D. Roth, "Learning and inference for clause identification," Proc. 13th European Conference on Machine Learning, Helsinki, Finland, pp.35-47, 2002.

[8] A. Molina and F. Pla, "Clause detection using HMM," Proc. 5th Conference on Computational Natural Language Learning, Toulouse, France, pp.70-72, 2001.

[9] E. F. T. K. Sang and H. Dejean. "Introduction to the CoNLL-2001 shared task: clause identification," Proc. CoNLL-2001, pp. 53-57, 2001.

[10] S. Doi, K. Muraki, S. Kamei and K. Yamabana, "Long sentence analysis by domain-specific pattern grammar," Proc. 6th Conference on the European Chapter of the Association of Computational Linguistics, p.466, OTS, The Netherlands, 1993.

[11] 김광백, 박의규, 나동렬, 윤준태, "구간 분할 기반 한국어 구문분석", 제 14회 한글 및 한국어 정보처리 학술대회, 청주, pp.163-168, 2002.

[12] W. C. Li, T. Pei, B. H. Lee and C. F. Chiou. "Parsing long English sentences with pattern rules," Proc. 13th International Conference on Computational Linguistics, Helsinki, Finland, pp.410-412, 1990.

[13] D. D. Palmer, M. A. Hearst, "Adaptive multilingual sentence boundary disambiguation," Computational Linguistics, vol.27, pp.241-261, 1997.

[14] S. Kim, B. Zhang and Y. Kim, "Learning-based intrasentence segmentation for efficient translation of long sentences," Machine Translation, vol.16, no.3, pp.151-174, 2001.

[15] C. Lyon, and B. Dickerson, "Reducing the complexity of parsing by a method of decomposition," Proc. 6th International Workshop on Parsing Technology, Boston, USA, pp.215-222, 1997.

[16] 김미영, 강신재, 이종혁, "규칙과 어휘정보를 이용한 한국어 문장의 구묶음", 제 12회 한글 및 한국어 정보처리 학술대회, 전주. pp.103-109. 2000.

[17] E. J. Kim and J. H. Lee, "A collocation-based transfer model for Japanese-to-Korean machine translation," Proc. NLPRS1993, Fukuoka, Japan, pp.223-231, 1993.

[18] M. Haruno, S. Shirai, and Y. Ooyama, "Using decision trees to construct a practical parser," Proc. 36th Annual Meeting of the Association for Computational Linguistics, Monteal, Quebec, Canada, pp.505-511, 1998.

[19] T. Nomoto and Y. Matsumoto, "Discourse parsing: a decision tree approach," Proc. 6th Workshop on Very Large Corpora, Montreal, Quebec, Canada, pp.216-224, 1998.

[20] V. Sornertlamvanich, T. Potipiti and T. Charoenporn, "Automatic corpus-based Thai word extraction with the C4.5 learning algorithm," Proc. 18th International Conference on Computational Linguistics, Saarbrucken, Germany, pp.802-807, 2000.
[21] J. R. Quinlan. C4.5 Programs for Machine Learning. Morgan Kaufmann Publishers. 1993.
[22] 이재성, 박재득, 차건희, 박세영, "형태소분석기 및 품사 태거 평가대회(MATEC99) 개요.", 제 11회 한글 및 한국어 정보처리 학술대회, 1999.

김 미 영
1999년 포항공과대학교 컴퓨터공학과 학사. 2005년 포항공과대학교 컴퓨터공학과 박사. 관심분야는 자연언어처리, 구문분석, 기계번역, 기계학습 등

이 종 혁
1980년 서울대학교 수학교육학과 학사 1982년 한국과학기술원 전산학과 석사 1988년 한국과학기술원 전산학과 박사 1989년~1991년 일본전기(NEC) 중앙연구소 초청연구원. 1991년~현재 포항공과대학교 컴퓨터공학과 교수. 1998년~1999년 미국 CRL/NMSU(뉴멕시코주립대학) 방문교수. 관심분야는 자연언어처리, 기계번역, 정보검색 등