단위(Chunks)분석과 의존문법에 기반한 한국어 구문분석

김미영★, 강신재, 이종혁,

포항공과대학교 컴퓨터공학과

Dependency Parsing by Chunks

Miyoung Kim★, Sinjae Kang, Jong-Hyeok Lee
Dept. of Computer Science & Engineering, POSTECH

요약

기존의 구문분석 방법은 구구조문법과 의존문법에 기반한 것이 대부분이다. 이러한 구문분석은 다양한 분석 결과들이 분석되는 동안 많은 시간이 소요되며, 잘못된 분석 결과를 찾아내어 삭제하기(pruning)도 어렵다. 본 논문은 구문분석에 필요한 의존문법을 적용하기 이전에, 단위화(Chunking) 방법을 사용하는 것을 제안한다. 이렇게 함으로써, 의존문법에 적용하는 차트의 수를 줄이게 되고, 의존관계의 설정 범위(scope)도 제한을 가할 수 있으며, 구문분석 속도 또한 빨라지게 된다

1. 서론

한국어는 교착어이며, 서술어가 문말에 위치한다. 또한 어순이 자유롭고, 문장 구성 성분의 생략이 흔하기 때문에 구구조문법보다는 의존문법에 의한 구문분석이 더 큰 효과를 가져올 수 있다.

의존문법은 문장을 구성성분 단위로 나누어 분석을 하려는 구구조문법과는 달리, 한 구성요소와 다른 구성요소와의 문법적 관계를 밝혀서 문장을 분석한다.[2]. 즉, 의존노드와 지배노드의 쌍을 찾아내고 그 관계를 정의함으로써 문장을 분석하는 것이다.

의존노드와 지배노드가 문장에서 차지하는 위치는 정해져 있지 않다. 다만, 한국어는 서술어가 문말에 위치하는 특성 때문에, 지배노드가 의존노드보다 문장의 후미에 위치하는 것이 보통이나, 지배노드와 의존노드 사이에 다양한 문장성분의 어절들이 위치할 수 있기 때문에, 정확한 쌍을 찾아내는 것이 어렵다. 그렇다고 해서, 하나의 의존노드에 대해서 가능한 지배노드를 모두 형성하게 되면, 무수히 많은 결과가 생성되므로 메모리와 시간이 많이 소비된다.

따라서 본 논문에서는 위와 같은 문제점을 최소화하기 위해, 단위화(Chunking) 방법을 제안한다.

아래와 같은 예문을 살펴보자.

- **작은 세 마리의 곰이 마당에서 놀다가 잠든 듯하다.**

문장에서 지배노드로 가능한 것은 용언이다. 따라서 위의 문장에서 지배노드로 가능한 어절들은 '작은', '놀다가', '잠든', '듯하다' 로서 총 4 개이다. "놀다가 잠든 듯하다" 는 주체가 곰으로서, 이 세 어절 각각을 지배노드로 하는 의존노드들이 같다고 할 수 있으므로, 하나의 차트로 묶는 것이 편리하다.

또한, "작은 세 마리의 곰"을 살펴보면, "작+은(관형형 전성어미)"와 "세 마리+의(관형격 조사)"가 둘 다 곰을 수식하는 관형어이므로, "작은 세 마리의 곰"을 하나의 차트로 묶어서 명사구 하나로 취급한 후, 의존관계를 설정하는 것이 편리하다.

위와 같이, 하나의 노드로 간주될 수 있는 명사구 어절들과 동사구의 연속된 어절들을 단일한 차트로 묶은 후, 의존문법을 적용시킨다. 단일화된 차트 내부에 속한 어절들은, 단일화에 속하지 않은 어절들과 의존 관계가 이루어질 수 없으므로, 의존관계 설정에 있어서 지배가능 범위가 제한이 된다. 따라서, 잘못된 분석 결과의 수를 줄일 수 있고, 단일화된 차트는 그 내부에서의 의존관계만 설정해 주면 되므로, 좀더 간단하고 빠르게 구문 분석을 할 수 있다.

여러 개의 구문 분석 결과가 나올 수 있는 경우는 세 가지이다. 첫째, 단일화된 명사구가 세 개 이상의 어절로 구성되어 있는 경우와, 둘째, 문장 내에 산재되어 있는 용언들 사이의 의존관계가 불확실할 때, 마지막으로 부사구의 지배노드가 불확실할 때다.

두 개 이상의 구문 분석 결과 중에서, 가장 그럴 듯한 한 가지를 선택하는 것은, 많은 실험과 분석이 필요하겠으나, 단일화를 통해 실제 적용하는 차트의 수를 많이 줄였고, 또한 의존노드와 지배노드 사이의 위치에 대한 범위도 한정을 지을 수 있었기 때문에, 의미정보를 추가하면, 높은 성능의 구문분석기의 구현이 가능하다.

따라서 본 논문에서는 단일화의 방법과, 이를 통해 간단해진 차트와 의존트리를 제시함으로써, 보다 빠르고 정확한 구문분석기가 생성될 수 있음을 보인다.

2. 단위화(Chunking)

문장에서 단위(chunks)의 존재성은 Gee &Grosjean (1983)의 성능구조(performance structure)에서 알 수 있다[3]. Gee & Grosjean 은 문장을 읽을 때 끊어 읽는 곳을 기점으로 하여, 어절들을 단위화하였다. Gee & Grosjean 의 경우처럼 운율에 따라 문장을 쉬어 읽는(pause) 방법은 아니지만, 구문분석에 적절한 단위화(chunking) 방법을 한국어에 적용해 보도록 한다.

2.1 명사구 단위화

구문분석시 가장 먼저 행해지는 것은 명사구 단위화(noun phrase chunking)이다. 명사구 단위화는 몇 가지 규칙을 근간으로 이루어진다[1].

[신효필 1999]에서 제안된 규칙을 참조하였으나, 포항공대 KLE 연구실의 형태소분석기 KOMA 에 맞게 변형시켜, 다음의 6 가지 규칙만을 사용한다.

<명사구>

- 관형격 조사 * 명사(구)
- 관형형 전성어미 * 명사(구)
- 관형사파생접사 * 명사(구)
- 관형사 * 명사(구)
- 양수사 * 명사(구)
- 체언 * 명사

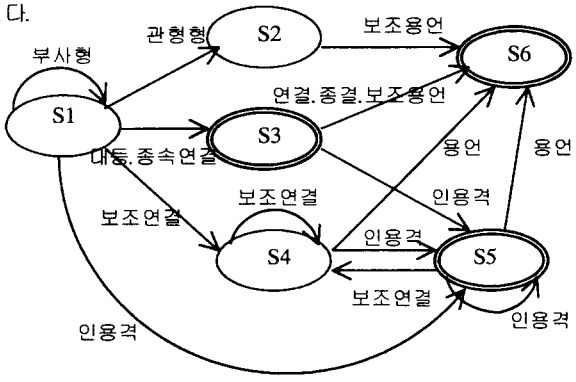
위에서, 관형사파생접사로는 '-적' 만을 취급하고 있으며,

6 번째의 규칙만이 명사구와의 결합은 불가능하도록 되어 있다. 예를 들어, [국어 + 사전]의 경우 단위화가 가능하지만, [국어+[사전의 색깔]]과 같이 체언 * 명사(구)인 경우에는 단위화가 되지 않아야 하기 때문이다.

2.2 동사구 단위화

명사구 단위화 과정을 거치면, 문장에서 남는 성분은 부사, 동사, 형용사 등이다. 이 성분들을 묶기 위해 동사구 단위화 과정을 거친다.

동사구 단위화는 아래와 같은 오토마타에 의해 이루어진다.



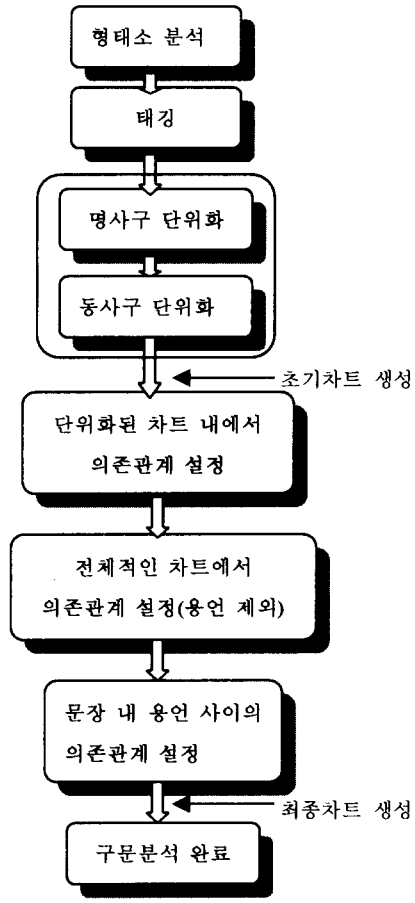
<그림 1>. 동사구 단위화를 위한 오토마타

위에서 S3, S5 와 S6 은 수락상태(final state)이다.

위와 같은 유한오토마타(finite state automata)에서, 패턴이 일치하는 상태(state)가 여러 가지 존재할 수가 있는데, 영어에서는 가장 길게 일치하는 경우를 택한다[4]. 동사구 단위화에서도, 더 길게 일치하는 것이 보다 정확한 단위화이므로, 가장 길게 일치하는 경우를 택하기로 한다.

3. 의존 관계의 설정

전체적인 시스템의 구조를 살펴보면 다음과 같다.



<그림 2> 전체적인 구문분석 시스템 구조

위에서도 이미 설명했듯이, 단일화된 명사구가 세 개 이상의 어절로 구성되어 있는 경우, 여러 경우의 구문분석 결과가 생길 수 있다.

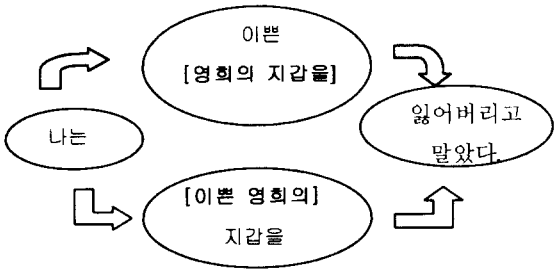
예를 들어, "나는 이쁜 영희의 지갑을 잃어버리고 말았다" 와 같은 예문의 경우, 아래와 같이 명사구 단위화에서 두 개의 결과가 생성된다. 이 때, 각각의 경우에 대해 새로운 차트를 형성하여 구문분석을 하게 되면, 시간과

{ 잃어버리고—말았다 <동사구 단위화>
  [이쁜 [영희의 지갑을]] <명사구 단위화>
  나는 }

{ 잃어버리고—말았다 <동사구 단위화>
  [[이쁜 영희의] 지갑을] <명사구 단위화>
  나는 }

< 표 1> 명사구 단위화의 애매성의 예

함께 메모리의 소비도 급격하게 증가하게 되므로, 단위화된 명사구의 앞뒤에 존재하는 똑같은 어절들은 공유하도록 한다. 이것을 그림으로 나타내면 아래와 같다.



<그림 3> 단위화 과정 후 차트의 구성 예

하나의 예를 들어, 아래의 문장에 대한 구문분석 결과를 의존트리로 나타내 보기로 한다.

- 이 역설은 러셀이 발견한 것으로 흔히 <이발사의 역설>이라 불린다.

<단위화 결과>

이_역설은 ---------- [명사구 단위화]
러셀이
발견한_것으로 ---------- [명사구 단위화]
흔히
<이발사의_역설>이라 -------- [명사구 단위화]
역설>이라_불린다. ---------- [동사구 단위화]

위에서 중복되는 어절은, 의존트리를 그릴 때 제거된다. 위의 문장에 대한 의존 트리는 <그림 4>와 같다.

4. 시스템 처리 결과

실험에 사용한 테스트 문장은, 1999 년 ETRI 주최로 개최된 Matec99 에서 제공받았던 말뭉치를 사용했다.
전체 205 문장에서 명사구 및 동사구 단위화의 정확률을 표로 나타내면 <표 2>와 같다.

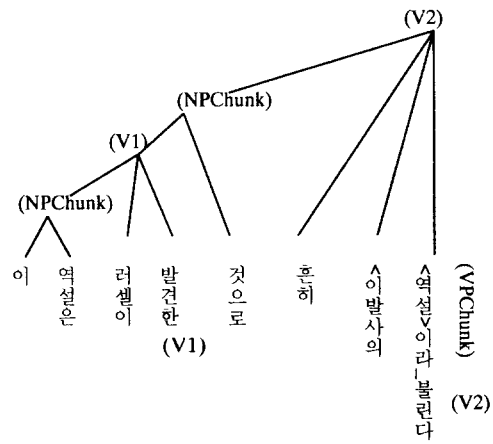
<표 2>의 정확률을 보면 짐작할 수 있듯이, 명사구 단위화 결과 중 24 개, 동사구 단위화 결과 중 27 개가 오류이다.

명사구 단위화		동사구 단위화	
단위화 수	정확률	단위화 수	정확률
556	95.68%	179	84.91%

<표 2> 단위화의 정확률

이 오류들 중, 태깅 오류에 기인한 것은, 명사구 단위화는 14 개, 동사구 단위화는 6 개이다.

나머지 오류들은 비슷한 패턴을 보인다. 오류들의 원인을 크게 살펴보면, 병렬 명사구의 정확한 범위 인식 부족, 대등연결어미 뒤의 관형어가 단위화되는 경우와 되지 않는 경우의 불명확한 구분에 있다고 하겠다. 이와 같이 단위화 과정을 거치고 나면, 그렇지 않은 경우보다 파생되는 차트도 많이 줄어들고, 의존관계 설정시 지배노드와 의존노드 사이의 범위도 제한이 되므로, 시간과 메모리가 많이 줄어들 뿐더러, 최종 구문분석 결과도 향상될 수 있겠다.



<그림 4> 의존 트리의 예

참고문헌

[1] 신효필, " 최소자원 최대효과의 구문분석" , 제 11 회 한글 및 한국어 정보 처리 학술 대회, pp.242-248, 1999

[2] Convington, M. A., " A Dependency Parser for Variable-Word-Order Languages" , Research Report AI-1990-01, Artificial Intelligence Programs, Univ.of Georgia, 1990.

[3] James Paul Gee & Francois Grosjean. Performance Structures: A Psycholinguistic and Linguistic Appraisal. Cognitive Psychology 15, pp.411-458. 1983. Steven Abney, " Chunks and Dependencies: Bringing Processing Evidence To Bear on Syntax" , CSLI, 1995

[4] Steven Abney, " Parsing by chunks " , Kluwer Academic Publishers, 1991