트리뱅크를 사용한 TAG 문법 자동 구축

박정열
파리 7대학, 언어학과 - 형식 언어학 연구소
jungyeul.park@linguist.jussieu.fr

LTAG Extraction from Treebank for Korean

Jungyeul Park
Universté Paris VII, UFR Linguistique - Laboratoire de Linguistique Formelle

요 약

문법 구축은 NLP 작업에서 중요한 역할을 한다. 이 논문에서는 트리뱅크 코퍼스에서 자동으로 어휘화 문법을 추출하는 시스템을 소개한다. 문법 자동 추출 시스템에서 자동으로 추출한 어휘화 TAG 문법, CFG 문법, 의존관계 등 여러 정보는 이후 한국어 파서 구현 및 다양한 NLP 연구에 사용된다.

1. 들어가는 글

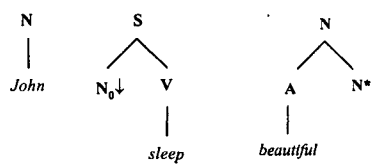
CFG 문법을 코퍼스에서 자동 추출하는 방법은 이미 많이 소개되었다[1][2][3]. 하지만 어휘화 TAG (Lexicalized Tree-Adjoining Grammar, 이하 LTAG)와 같이 특정 형식 통사론에 따른 문법을 자동으로 추출하는 연구는 아직까지는 소수의 연구 결과만 있다[4]. LTAG 문법 자동 추출이 CFG 문법 추출보다 활발하지 못한 가장 큰 이유는 LTAG 문법 특성에 기인한다. LTAG 문법의 특징은 어휘화된 트리 구조이기 때문에 트리 구조상으로 깊이 1 의 CFG 규칙과는 달리 1 이상의 깊이를 가진다. 따라서 LTAG 문법 추출 알고리즘은 구구조 전체를 탐색하여 기본 트리를 추출하기 때문에 복잡한 알고리즘을 가진다. 또한, CFG 에는 존재하지 않는 트리 내부 치환 연산을 사용하기 때문에 더욱 문법을 추출하기 힘들다.

이 논문에서는 트리뱅크 코퍼스를 사용하여 한국어 LTAG 문법에서 술어-논항 구조를 자동으로 추출한다. 이렇게 자동으로 추출하여 구축된 LTAG 문법은 다양한 NLP 작업에서 사용할 수 있으며 무엇보다도 한국어 TAG 파서 구현에 기반이 된다.

2. TAG 문법 개요

TAG 문법은 트리를 사용해 다시 쓰기 규칙을 구현한 시스템이다[5][6][7]. TAG 형식론은 기본 트리 (elementary tree)로 구성되며 기본 트리는 다시 초기 트리(initial tree)와 보조 트리(auxiliary tree)로 구분된다.

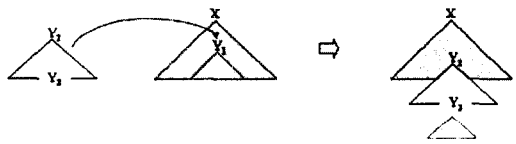
다음은 초기 트리와 보조 트리의 간단한 예이다. 아래 그림 1의 초기 트리 α에서 *sleep*의 내부 노드인 V는 비종단 기호이고 경계 노드는 *John*과 *sleep*과 같이 종단 기호이거나 $N_0\downarrow$와 같이 트리 외부 치환 (substitution)이 표시된 비종단 기호이다. 보조 트리 β에는 트리 내부 치환 (adjunction)이 가능한 경계 노드 N^*은 루트 노드 N과 동일한 타입이다.



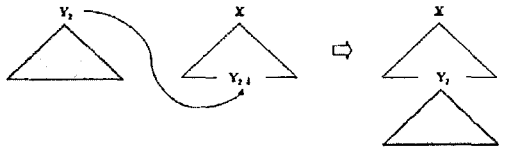
(a) 초기 트리 α (b) 보조 트리 β

그림 1 : 초기 트리 및 보조 트리 예

일반적으로 TAG 에서는 트리 내부 치환 및 트리 외부 치환 두 가지 연산을 사용한다. 트리 외부 치환은 초기 트리의 루트 노드가 트리 외부 치환이 표시되어 있는 다른 초기 트리의 비종단 경계 노드로 치환되어 새로운 트리를 생성하는 연산이다. 루트 노드와 비종단 경계 노드는 동일한 타입이어야 한다. 트리 내부 치환은 보조 트리가 초기 트리의 해당 비종단 노드로 치환된다. 위의 그림에서 처럼 보조 트리의 루트 노드와 경계 노드는 동일한 타입이어야 한다. 그림은 트리 외부 치환과 트리 내부 치환을 설명한다.



(a) 트리 내부 치환 연산



(b) 트리 외부 치환 연산

그림 2 : TAG 문법에서 사용하는 연산

3. 시스템 개요

이 장에서는 *Grex*(*G*rammar *Ex*tractor)라는 시스템을 설명한다[1]. Grex는 트리뱅크 코퍼스에서 다양한 형태의 문법 및 정보를 추출할 수 있다. 이 장에서는 Grex 시스템을 사용하여 위에서 설명한 TAG 문법 및 NLP 작업에서 필요한 정보를 트리뱅크 코퍼스에서 추출한다. 이 논문에서 사용하는 트리뱅크 코퍼스는 21세기 세종계획의 일환으로 구축된 구문분석 말뭉치 (이하 SJTree)[8]이며, SJTree의 분석 결과를 논문의 목적에 맞게 수정하기 위해 [9]를 참조했다.

3.1 추출 대상 문법의 형태

Grex는 트리뱅크 코퍼스에서 LTAG 문법의 구성

[1] 부록 참조.

형태인 기본 트리를 자동으로 추출한다. 일반적으로 기본 트리는 다양한 형태로 표시된다. 따라서 Grex가 추출하는 기본 트리의 형태는 [4]에서 제안한 술어-논항 관계, 부가항 관계, 등위 관계에서 술어-논항 관계와 부가항 관계만을 기본 형태로 제한하며 Grex가 추출하는 기본 트리는 두가지 기본 형태중 하나에 부합해야 한다[2].

술어-논항 관계는 TAG의 기본 트리 중에서 초기 트리이고, 부가항 관계는 보조 트리이다.

3.2 트리뱅크 코퍼스

SJTree의 구조 및 구축 방법에 대해서는 [8]에 잘 기술되어 있다. 그림 3에서와 같이 SJTree에는 어절별 형태분석 결과와 함께 구문 표지 및 기능 표지, 그리고 구구조 정보를 지니며, 총 어절수는 30,853개, 총 문장수는 2,512개이다.

```
; 막일을 기피하는 풍조는 건설 공사장뿐만이 아니다.
(S- (S (NP_SBJ (VP_MOD (NP_OBJ 막일/NNG+을/JKO)
                        (VP_MOD 기피/NNG+하/XSV+는/ETM))
               (NP_SBJ 풍조/NNG+는/JX))
    (VP (NP_COMP (NP 건설/NNG)
                 (NP_COMP 공사장/NNG+뿐/JX+만/JX+이/JKC))
        (VP 아니/VCN+다/EF)))
    (S-+./SF))
```

그림 3 : 세종계획 트리뱅크 코퍼스 예

3.3 추출 알고리즘

트리뱅크 코퍼스에서 Grex가 TAG 문법을 추출하는 알고리즘은 크게 두 단계로 구성되는데 먼저, TAG 문법에 맞게 트리뱅크 코퍼스를 변환하고, 트리를 트래버스하면서 기본 트리를 구축한다.

3.3.1 트리뱅크 코퍼스 변환

TAG 문법에서 필요한 술어-논항 정보 및 부가항 정보를 얻기 위해서는 SJTree의 구조를 일부 변경하여 기본 트리를 추출한다. 특히, SJTree는 생략된 논항이나 논항 이동에 따른 흔적 정보를 위한 공범주는 일체 표기하지 않았다. 하지만 이 논문의 목표가 TAG 문법을 추출하여 용언의 논항구조에 대한 틀을 마련하자는데 있기 때문에 흔적 정보의 경우에는 VP_MOD 노드 하단에 실현된 논항들로 다시 논항 정보를 복원한다. 생략된 논항은 트리뱅크 코퍼스에서 그 이상의 정보를 얻을 수 없기 때문에 이 논문에서는 고려하지 않는다.

또한 이 논문에서는 조사를 부가항으로 처리하는 반면 어미에 대해서는 용언과 결합 형태를 그대로 표시한다. 이는 어미에서 얻을 수 있는 시제와 양상 정보는 이후 자질기반 LTAG (FB-LTAG) 문법 구성에 사용한다[3].

그림 4은 변환된 트리뱅크 코퍼스이다. 변환된 부분과 새로 추가된 부분은 굵게 표시했다.

```
; 막일을 기피하는 풍조는 건설 공사장뿐만이 아니다.
(S- (S (NP_SBJ
   (S (NP_SBJ pro)
      (VP_MOD (NP_OBJ (NP (막일/NNG))
                      (을/JKO))
              (VP_MOD (기피/NNG+하/XSV+는/ETM))))
      (NP_SBJ (NP (풍조/NNG)))
              (는/JX))
      (VP (NP_COMP (NP (건설/NNG))
                   (공사장/NNG))
                   (뿐/JX+만/JX+이/JKC))
          (VP (아니/VCN+다/EF))))
   (S-+./SF))
```

그림 4 : 변환된 트리뱅크 코퍼스 예

3.3.2 기본 트리 구축

변환된 트리뱅크 코퍼스에서 Grex는 기호를 제외한 루트 노드 S에서 중심어까지 경로를 확인하고, 이들 경로에서 부가항 및 논항을 먼저 추출한 다음, 남은 트리 부분들을 결합하여 술어-논항 정보를 추출한다. 논항을 추출하기 위해 Grex는 중심어 정보 및 논항 정보를 사용한다. 예를 들어, ADJP와 같은 형용사구는 VJ나 ADJP가 중심어가 될 수 있고, NP-COMP와 같은 논항을 가질 수 있다. 다른 문제점으로 SJTree에서는 예를 들어, VP가 확장될 때 논항 NP_OBJ와 VP로 확장된다는 점이다. 이는 X바 이론에 위배되는 것으로 기본 트리를 추출할 때 확장된 VP를 VV(동사)로 변환한다.

두 개 이상의 명사 연쇄로 구성된 명사구의 경우에는 중심어 선택이나 다른 제약 사항들 때문에 이 논문에서는 명사구 내에서 평면적 구조로 구성된다고 가정하며 [4]에서와 달리 명사 명사 연쇄에서 명사 부가항은 존재하지 않는다[4].

---

[2] 이 논문에서는 등위관계를 설정하지 않는다. 등위관계는 문장을 분석하는 과정에서 파싱 전처리 단계에 임시로 생성하는 걸로 간주한다.

[3] 한국어 FB-LTAG에 대해서는 [10]를 참조하라. 조사의 처리에 있어서 이 논문에서는 부가항으로 처리하지만 [10]에서는 명사-조사 결합 형태를 명사구로 설정하고 다른 설명을 보이지 않고 있다. 이런 설정을 가정할 때 조사, 어미를 같은 형태로 처리한다는 일관성의 장점은 있지만 명사 명사 연쇄와 같은 명사구에 대한 명확한 논의가 없는 상황에서 명사-조사 결합 형태를 하나의 명사구로 처리하는것은 무리가 따른다. 이러한 문제는 이후 '-이다'와 같은 명사-긍정지정사 구성에서도 논의될 수 있다. 명사-조사 결합 형태에 대해 조사를 부가항으로 처리한 경우는 일본어에 대한 LFG 분석에서도 볼 수 있다[11].

[4] 명사 명사 연쇄(또는 그 이상의 연쇄)로 구성된 명사구 구조를 [4]에서 제안한 내용을 그대로 한국어에 적용할 경우에는 논란을 가지고 올 수 있다. 위의 예에서 '건설 공사장'에 대해 [4]를 그대로 적용하여 문법을 추출하는 경우에 부가항 (NP (NNP 건설) NP*)와 중심어 (NP (NNG 공사장))이라는 두 개의 문법을 추출한다. 이는 앞의 명사를 부가항으로 설정하는 것 자체가 문제가 될 수 있고, 이런 식으로 구구조를 설정하는 경우에 모든 복합 명사에 대해서 추출할 수 있는 부가항으로 설정되는 명사가 중심어 위치에 올 수 있기 때문에 부가항과 중심어에 대한 LTAG 문법을 이중으로 구성하는 잉여성의 문제가 발생한다. 현재 한국어 TAG 파서에서는 이러한 복합 명사 처리를 명사 명사 병렬적 관계로 구성되는 구조로 가정하고 파싱 전처리 과정에서 명사 명사에 해

트리뱅크에서 가능한 모든 기본 트리를 추출할 경우에 n을 노드 갯수라 할 때 2^n개의 트리를 추출할 수 있다. 트리뱅크 코퍼스에서 추출되는 기본 트리들은 LTAG 문법을 적용하면 최종적으로 그림 5의 기본 트리만 남게 된다.

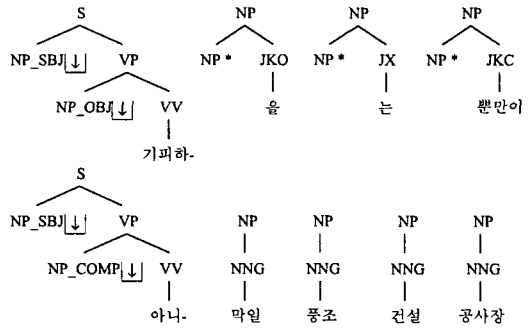


그림 5 : 자동으로 추출된 기본 트리

최종적으로 추출된 문법에 대한 결과는 다음 표1로 정리된다.

표 1 : 추출 결과

기본트리	초기트리	보조트리	CFG 규칙
7227	6563	664	479

4. 결론 및 향후 과제

이 논문에서는 트리뱅크 코퍼스를 사용하여 한국어 LTAG 문법을 자동으로 추출했다. 이후 연구 방향으로는 보다 많은 구문 분석 코퍼스를 사용하여 Grex 시스템을 적용하는 것이고, 무엇보다도 추출된 문법을 평가할 수 있는 공개된 다른 문법이 없어 문법의 적용범위를 평가할 수 없었는데, 이를 평가할 수 있는 방법을 모색해야 한다.

참고문헌

[1] Kiyoaki Shirai, Takenobu Tokkunaga and Hozumi Tanaka. Automatic Extraction of Japanese Grammar from a Bracketed Corpus. In *Proceeding of Natural Language Processing Pacific Rim Symposium*. 1995.

[2] Eugen Charniak. Treebank Grammars. In *Proceeding of AAAI-1996*. 1996.

[3] Alexander Krotov, Mark Hepple, Robert Gaizauskas, and Yorick Wilks. Compacting the Penn Treebank Grammar. In *Proceeding of ACL-COLING*. 1998.

[4] Fei Xia. *Automatic Grammar Generation from Two Different Perspectives*, PhD Dissertation. University of Pennsylvania. 2001.

[5] Aravind Joshi, L. Levy and M. takahashi. Tree Adjoining Grammar. In *Journal of Computer and System Science*. 1975.

[6] Aravind Joshi, and Yves Schabes. Tree Adjoining Grammar. In A. Salomma and G. Rogenberg, editors, *Handbook of Formal Languages and Automata*. Springer-Verlag, Herdelberg. 1994.

[7] The XTAG Research Group. *A Lexicalized Tree Adjoining Grammar for English*, Technical Report, University of Pennsylvania. 1999.

[8] 21 세기 세종계획. 국어 기초자료 구축 분과 - 결과 보고서. 2003.

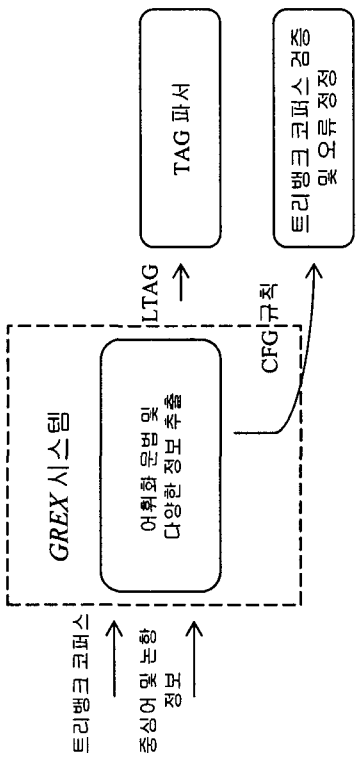
[9] Chung-hye Han, Na-rae Han and Eon-suk Ko. *Bracketing Guidelines for Penn Korean Treebank*, IRCS Report 01-09, Institute for Research in Cognitive Science, University of Pennsylvania. 2003.

[10] Chung-hye Han, Juntae Yoon, Nari Kim and Martha Palmer. *A Feature-Based Lexicalized Tree Adjoining Grammar for Korean*. IRCS Report 00-04, Institute for Research in Cognitive Science, University of Pennsylvania. 2000.

[11] Hiroshi Masuichi, and Tomoko Okuma. Japanese Parser on the basis of the Lexical-Functional Grammar Formalism and its Evaluation. In *Journal of Natural Language Processing*, Vol.10, No.2, pp.77-109 (in Japanese). 2003.

[12] 박정열. 한국어 TAG 파싱 연구, 파리 7 대학 기술 문서 TR-UFRL-04-010bis. 2004.

부록: Grex 시스템 개념도



당하는 명사구 내용을 임시로 생성하여 실제 파싱하는 동안에는 명사구 전체만을 고려한다. 그외 한국어 TAG 파싱 과정에 대한 자세한 사항은 [12]을 참조하기 바란다.