표층 구문 타입을 사용한 조건부 연산 모델의 일반화 LR 파서

(Generalized LR Parser with Conditional Action Model(CAM) using Surface Phrasal Types)

곽 용 재 † 박 소 영 †† 황 영 숙 ††
(Yong-Jae Kwak) (So-Young Park) (Young-Sook Hwang)
정 후 중 ††† 이 상 주 †††† 임 해 창 †††††
(Hoojung Chung) (Sang-Zoo Lee) (Hae-Chang Rim)

요 약 일반화 LR(Generalized LR, 이하 GLR) 파싱은 선형 스택을 사용하는 전통적인 LR 파싱 방식의 한계를 극복하도록 만들어진 LR 파싱 기법의 하나로서, LR 기법에 여러 가지 매커니즘을 통합하여 자연어 파싱에 응용하는 작업의 토대가 되어 왔다. 본 논문에서는 기존의 확률적 LR 파싱 기법이 가지고 있는 문제를 개선한 조건부 연산 모델(Conditional Action Model)을 제안한다. 기존의 확률적 LR 파싱 기법은 그래프 구조 스택의 복잡성으로 인해 상대적으로 제한된 문맥 정보만을 사용하여 왔다. 제안된 모델은 부분 생성 파스의 표현을 위하여 표층 구문 타입(Surface Phrasal Type)을 사용하여 그래프 구조 스택에 들어 있는 구문 구조를 기술함으로써 좀 더 세분된 구조적 선호도를 파서에 반영시킬 수 있다. 실험 결과, 어휘를 고려하지 않고 학습한 조건부 연산 모델로 구현된 본 GLR 파서는 기존의 방식보다 약 6-7%의 정확도 향상을 보였으며, 본 모델을 통해 풍부한 스택 정보를 확률적 LR 파서의 구조적 중의성 해결에 효과적으로 사용할 수 있음을 보였다.

키워드 : 구문분석, 구문 중의성 해소, 확률적 LR 파싱, 연산 확률 학습

Abstract Generalized LR parsing is one of the enhanced LR parsing methods so that it overcome the limit of one-way linear stack of the traditional LR parser using graph-structured stack, and it has been playing an important role of a firm starting point to generate other variations for NL parsing equipped with various mechanisms. In this paper, we propose a Conditional Action Model that can solve the problems of conventional probabilistic GLR methods. Previous probabilistic GLR parsers have used relatively limited contextual information for disambiguation due to the high complexity of internal GLR stack. Our proposed model uses Surface Phrasal Types representing the structural characteristics of the parse for its additional contextual information, so that more specified structural preferences can be reflected into the parser. Experimental results show that our GLR parser with the proposed Conditional Action Model outperforms the previous methods by about 6~7% without any lexical information, and our model can utilize the rich stack information for syntactic disambiguation of probabilistic LR parser.

Key words : syntactic parsing, syntactic ambiguity resolution, probabilistic GLR parsing, action probability learning

· 이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2000-V01452-E00305)

† 학생회원 : 고려대학교 정보통신공동연구소
yjkwak@nlp.korea.ac.kr
†† 학생회원 : 고려대학교 컴퓨터학과
ssoya@nlp.korea.ac.kr
yshwang@nlp.korea.ac.kr
††† 비 회 원 : 고려대학교 정보통신공동연구소
hjchung@nlp.korea.ac.kr
†††† 종신회원 : (주)엔엘피솔루션
zoo@nlpsolution.com
††††† 종신회원 : 고려대학교 컴퓨터학과 교수
rim@nlp.korea.ac.kr

논문접수 : 2002년 4월 30일
심사완료 : 2002년 10월 25일

1. 서 론

구구조 문법을 사용한 자연어 파싱 분야에서는 차트 등의 WFST(Well Formed Substring Table) 기반의 탐색 방법이 주로 활용되어 왔다. 한편 프로그래밍 언어에서 주로 사용되어 오던 LR 파싱 기법[1]이 자연어 처리 연구에도 도입되기 시작하였는데, Left-to-right scan, Rightmost derivation의 약어인 이 LR 파싱 기법은 구현이 까다로우나 가장 효율적인 기법 중 하나로 알려져 있다. 그 이유는 문법 규칙 집합을 해석하여 만들어 놓은 파싱 테이블(parsing table)로 인해 원하는 문법 규칙을 일치시킬 때 소요되는 오버헤드가 없고, WFST 대신 사용되는 스택으로 인해 이전에 생성된 부분 파스 구조를 탐색할 필요가 없기 때문이다. 1980년대 후반에는 M. Tomita에 의해 전통적인 LR 파서가 가지고 있는 단방향 선형 스택의 한계를 극복한 방법이 만들어졌다[2]. 일반화(Generalized) LR, 소위 GLR 파싱이라 불리는 이 기법은 이후 LR 기법에 기반한 응용 시스템을 구축하는 작업의 기본이 되어 왔으며, 현재는 구구조 생성을 수행하는 효율적인 파서를 요구하는 기계 번역 시스템이나 음성 인식 시스템[3, 4, 5] 등에서 다양하게 응용, 변형되어 쓰이고 있다.

LR 기법을 사용하여 구조적 중의성을 해결하는 자연어 파서는 중복되는 연산 중 하나를 바로 선택해 가면서 최종적으로 하나의 파스만을 생성하는 결정적 LR 파서와, 중의적인 구조를 모두 생성한 후에 확률적인 방법을 통해 가장 신뢰도가 높은 파스를 선택하는 확률적 LR(GLR) 파서의 두 가지로 구현되는 것이 일반적이다. 기존의 확률적 LR 파서는 구조적 중의성을 모두 고려하여 연산을 수행하여야 하는 동작 원리의 복잡성 때문에 상당히 제한적인 문맥 정보만을 사용하여 왔기 때문에, 그 결과로 구조적 중의성에 많은 도움을 주지 못하였다. 본 논문에서는 파서의 동작 중에 부분적으로 생성된 파스 구조를 조건으로 하여 쉬프트(shift) 및 리듀스(reduce) 연산의 확률을 얻고, 그에 따라 그래프 구조 스택에 대해 연산을 수행하는 새로운 GLR 파싱 모델을 소개한다. 제안된 모델에서, 부분적으로 생성된 파스의 정보는 그래프 구조 스택의 리듀스 경로(reduce path)로부터 얻고, 부분 생성 파스의 구문 구조는 표층 구문 타입(Surface Phrasal Type)이란 기술 방식을 사용하여 효과적으로 표현한다.

본 논문은 다음과 같이 구성된다. 2절에서 LR 기법을 사용한 기존의 자연어 파싱 기법에 대해 전반적으로 소개하며, 3절에서 GLR 파싱의 소개와 함께 본 논문에서 제안하는 조건부 연산 모델과 부분 파스 구조의 표현 방식, 그리고 확률 모델에 대해 설명한다. 4절에서 실험을 통하여 제안된 모델을 부착한 GLR 파서의 성능을 평가하고 고찰하며, 마지막으로 향후 작업 계획과 결론을 제시하는 5절로서 끝을 맺는다.

2. 관련 연구

문장의 구조적 중의성을 LR 기법으로 구현한 파서를 통해 해결하는 방법은 지금까지 여러 가지가 발표되었는데, 이들은 서론에서 기술한 바와 같이 크게 두 개의 부류로 나눌 수 있다. 하나는 규칙 기반의 결정적(deterministic) 방법, 또 하나는 확률적(probabilistic) 방법이다.

2.1 결정적(Deterministic) LR 파싱

이 방법은 자연어 파서가 가지고 있는 치명적인 약점 중 하나인 높은 계산 복잡도의 개선에 초점을 맞춘 것으로서, 학습 집합에서 추출한 연산 결정 규칙(action selection rule)을 사용하여 쉬프트/리듀스 연산을 선택한다. 쉬프트와 리듀스 연산을 동시에 수행해야 하는 시점에서 하나의 연산을 선택하여 바로 중의성을 해결하는 것이다. 이 방법은 단방향 선형 스택(one-way linear stack)으로 구현이 가능하며, 내부 자료구조가 비교적 간단하므로 수행 효율이 상당히 높다. 그러나 이 방법은 이전에 수행한 연산의 결과가 다음의 연산 결정에 영향을 주기 때문에, 이전에 잘못 선택된 연산으로 인해 파싱이 실패하거나 틀린 파싱 결과가 나올 확률이 높다는 약점을 가지고 있다. Simmons와 Yu는 스택과 입력열의 품사 및 문법 태그로 이루어지는 길이제한 문맥(windowed context)을 연산 선택 규칙으로 사용하여 결정적 LR 파싱을 수행하였고[6], 곽용재는 이 방법을 한국어 파싱에 적용하였다[7]. Hermjakob은 연산을 결정하는 데 기여하는 자질을 품사, 어휘 및 의미 정보로 확장하고 기계학습 기법을 통해 연산 결정 규칙을 학습하여 사용하였으며[8], Wong과 Wu는 구구조 문법을 사용하지 않고 결정트리를 사용하여 스택 내의 구성 요소에 구구조 표지를 할당하는 쉬프트/리듀스 파서를 제안하였다[9].

2.2 확률적(Probabilistic) LR 파싱

이 방법은 GLR 파싱 기법의 골격에 확률적인 구조적 중의성 해결 기법을 결합한 것이다. GLR 파싱에 확률 기법을 도입한 첫 시도는 Wright와 Wrigley의 방법이다. 이 방법은 PCFG 확률을 파싱 테이블 내의 리듀스 연산에 그대로 할당한 것이다[10]. 이 파서는 PCFG를 사용한 기존의 WFST 기반 파서에 비해 빠른 수행 효율을 보여주었으나, 문맥을 고려하지 않고 동작하므로

최종적으로 만들어지는 분석 결과는 PCFG 파싱을 수행했을 때와 정확히 똑같다. Su 외 3인은 생성된 파스 트리를 구 단위(phrase unit)의 집합으로 보고 두 개의 쉬프트 연산 사이에서 스택 내부의 내용이 변경되는 것으로 파싱 과정을 재정의한 후, 수행할 연산을 결정해야 하는 부분의 주변에 있는 좌우 기호를 문맥으로 사용하여 전체 파스에 대한 점수(score)를 부여하였다[11]. 이 방법은 GLR 파싱에 문맥 의존적인 확률 기법을 사용한 최초의 시도였으나, 기본적으로 GLR 파싱 기법에 맞추어 만들어진 방법이 아니기 때문에 시스템 통합이 어렵다는 단점이 있었다.

본격적으로 GLR 파싱 자체의 특징을 살린 확률적 기법은 Briscoe와 Carroll의 방법[12, 13]을 시작으로 하여 [14, 15, 5]에 이른다. 이들 파서는 스택 최상부의 상태(state) 번호와 미리보기 기호(lookahead)에서 문맥 정보를 뽑아 파싱 테이블 상의 연산에 확률을 부여하여 확률적으로 가장 신뢰도가 높은 파스를 선택한다. 스택 최상부의 상태 번호는 연산이 일어나는 시점에서의 왼쪽 문맥 정보를 제공하며, 미리보기 기호는 오른쪽 문맥 정보를 제공한다. Briscoe과 Carroll은 연산이 일어나기 전의 상태와 연산이 일어난 후의 상태를 모두 고려한 상태 전이 모델(state transition model)을 제시하였고[12], Inui 외 3인은 연산이 일어나기 전의 상태만을 고려하는 스택 전이 모델(stack transition model)을 제시하여 [12]에서 제시한 확률 모델의 결함을 보완하였다[14, 15]. Ruland는 짧은 길이의 대화체 문장의 다국어 번역 시스템을 위한 확률 LR 파서를 구현하였는데, 스택 최상부의 상태 번호, 스택 최상부 기호의 구 중심어(phrase head), 미리보기 기호의 부분 집합을 부분 문맥(sub-context)으로 학습하고 각 부분 문맥 사이의 선형 보간법(linear interpolation)을 적용한 모델을 사용하였다[5]. 이들 방법은 구조적 중의성을 모두 유지하며 성장하는 그래프 구조 스택의 복잡성 때문에 스택 정보를 지극히 간략화된 형태로 정의함으로써, 스택 정보를 구조적 중의성 해결에 효과적으로 사용하지 못하고 있다.

3. 조건부 연산 모델을 사용한 GLR 파싱

3.1 GLR 파싱의 기본 개념

일반화(Generalized) LR 파싱의 요점은 하나의 선형 스택(linear stack)을 쓰는 전통적인 LR 파서를, 그래프 구조의 다중 스택을 쓰도록 확장하여 자연어의 구조적 중의성을 나타낼 수 있도록 만든 것이다. 그림 1의 상단은 전형적인 LR파서의 스택을, 하단은 GLR 파서의 스택을 나타내며, 오른쪽편의 두 그림은 쉬프트와 리듀스 연산을 각각 취했을 때의 부분 결과이다.

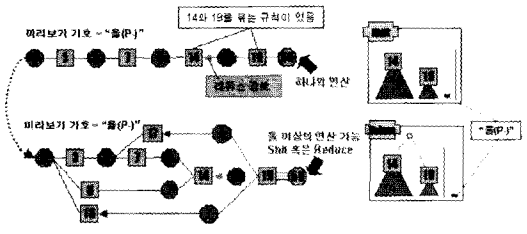

그림 1 LR 파서와 GLR 파서의 스택 구조

기본적으로 GLR 파서는 **LR 파싱 테이블**과 **그래프 구조 스택**으로 구성된다. LR 파싱 테이블(LR parsing table)은 특정한 상태에서 수행할 수 있는 연산을 알려주는 자료 구조이며 문맥 자유 문법을 해석하여 만든다. 자연어 파싱에 사용하는 문맥 자유 문법은 중의성을 가지고 있기 때문에, 주어진 상황에서 수행할 수 있는 연산을 나타내는 테이블 엔트리가 두 개 이상이 될 수도 있다. 따라서, 프로그래밍 언어 등에서 사용하는 전통적인 LR 파서의 선형 스택(linear stack)이 아닌 여러 개의 선형 스택이 연결된 그래프 구조의 스택이 사용되는 것이다.

그래프 구조 스택(Graph-Structured Stack)은 GLR 파서가 동작하는 동안 생성되는 중간 파스 상태를 저장해 놓는 역할을 한다. LR 파싱 테이블에서 얻는 연산의 개수가 2개 이상일 수 있으므로, 전통적인 LR 파서의 선형 스택과 달리 여러 갈래의 경로가 존재하는 그래프 형태를 띤다. 이 자료구조는 차트 파서 등에서 사용하는 전통적인 WFST와 기능적으로 비슷하며, 파서의 동작 과정에서 스택 전체를 탐색할 필요 없이 가장 마지막에 수행한 연산에 의해 도달한 상태 노드들만을 점검하면 되기 때문에 기존의 WFST 기반의 구구조 파서와 비교할 때 빠른 수행속도를 보여 준다.

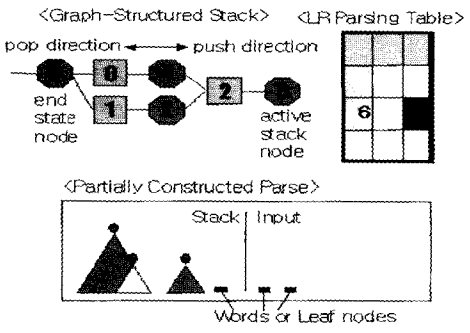

그림 2 GLR 파서의 기본적인 구성요소

그림 2는 GLR 파서의 기본 구조를 나타낸 것이다. GLR 파서는 입력(input)열에 있는 입력기호를 스택에 넣으면서 입력기호를 모두 소비할 때까지 동작한다. 스택, 즉 그래프 구조 스택은 스택에 노드를 첨가하는 푸시 방향(push direction)인 좌-우 방향으로 커지며(쉬프트 연산 시), 스택에서 노드를 꺼내는 팝 방향(pop direction)인 우-좌 방향으로 작아진다(리듀스 연산 시). 네모진 모양의 노드와 그 안의 번호는 생성된 구문 트리 노드 및 노드의 생성 순서 번호를 나타내며, 둥근 모양의 노드와 번호는 파서 연산의 결과로 파서가 도달했거나 거쳐간 상태를 나타낸다. 그리고 가장 마지막에 수행된 연산의 결과로 만들어진 상태 노드(스택의 가장 오른쪽에 있다)를 **활성 스택 노드**(active stack node)라고 한다. GLR 파서는 이 노드의 번호와 현재의 입력기호를 가지고 LR 파싱 테이블을 접근하여, 해당 테이블 엔트리가 지시하는 쉬프트 혹은 리듀스 연산을 활성 스택 노드에 대해 수행하면서 파스 구조를 부분적으로 구축해 간다. 이때 그래프 구조 스택은 중간에 생성된 부분 파스를 반영한다. 동일한 단말 기호의 범위(span)에 대해 만들어지는 구문 구조가 두 개 이상 생길 수 있는데, GLR 파서는 **지역적 구조 중의성**(local ambiguity)라 불리는 이러한 중복된 파스 포리스트(forest)를 압축(packing)이란 과정을 통해 하나로 묶는다. 그림 3은 GLR 파서가 생성하는 압축된 파스 포리스트의 예를 보여준다.

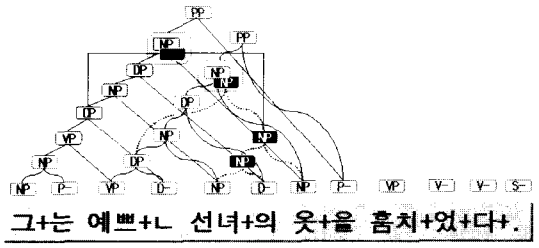

그림 3 GLR 파서가 생성하는 압축된 파스 포리스트. '훔치'를 미리보기 기호로 읽은 시점에서 일어난 모든 연산 결과이다. 두 개의 비단말 노드를 묶은 팔각형은 압축(packing)된 것임을 나타낸다.

3.2 GLR 파싱의 조건화와 조건부 연산 모델

확률적 GLR 파싱은 기술한 바와 같이 LR 파싱 테이블이나 스택 및 입력열의 조건을 확률 추정을 위한 문맥 정보로 사용하는 방법이다. 따라서, 어떤 정보를 조건으로 삼았는지가 가장 중요한 관건이 되는데, 본 논문에서는 기존의 방법보다 더 많은 스택 정보를 조건 정보로 사용하기 위해 GLR의 기본 골격에서 조건화가 가능한 시점을 아래와 같은 두 가지의 경우로 정의하였다(그림 4).

중의적 연산(Ambiguous Action) : 활성 상태 노드의 번호와 입력 기호가 주어졌을 때 LR 파싱 테이블에서 지시하는 연산이 2개 이상인 경우를 가리킨다. 예를 들어 어떤 상태 노드에 대해 쉬프트 연산과 리듀스 연산을 동시에 수행해야 한다면, 스택은 리듀스 연산에 의한 노드 생성과 쉬프트 연산에 의한 입력 기호의 푸시가 동시에 일어난 결과를 가지게 된다.

다중 파스 생성(Multiple Parse) : GLR 파서의 스택은 이전에 수행된 중의적 연산의 결과로 경로가 분할되는데, 이렇게 분할된 경로들은 쉬프트 연산 시점에서 하나로 묶인다. 결국 활성 스택 노드는 분할된 리듀스 경로들과 연결된 상태에 있게 되며, 이 활성 스택 노드에 리듀스 연산이 수행되어야 한다면 연결된 경로의 개수만큼 리듀스 연산이 수행되어야 한다.

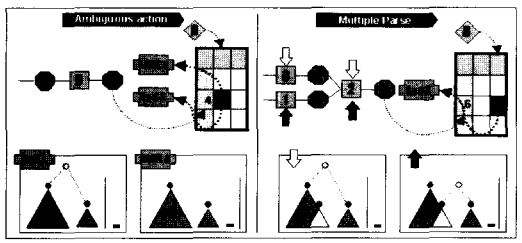

그림 4 GLR 파서에서 발생할 수 있는 중의적 연산과 다중 파스 생성 상황. 그래프 구조 스택은 설명을 위해 임의로 만들었다. 마름모 B 기호는 스택 외부에 있는 첫째 입력 기호(미리보기 기호)를 나타낸다. "Shift"와 "Red"는 각각 쉬프트(Shift) 연산과 리듀스(Reduce) 연산을 나타낸다.

본 연구에서 제안하는 조건부 연산 모델(Conditional Action Model, 이하 CAM)은 GLR 파서의 구문 중의성을 해소하기 위한 확률 모델로서, 앞에서 제시한 두 가지의 경우를 고려하여 만들어졌다. 즉 그래프 구조 스택이 나타내는 부분 생성 파스의 구조적 특징과 입력 기호를 조건 정보로 사용하여, 주어진 스택에 대해 취할 연산의 확률을 부여하는 것이다. CAM이 이전의 확률적 LR 파싱 모델에 대해 가지는 차이점은 그래프 구조 스택이 반영하는 부분 파스의 구조를 적극적으로 문맥 정

보로 사용한다는 것이다. 기본적으로 확률적 LR 파싱 모델은 입력 기호열의 첫째 기호(미리보기 기호)를 오른쪽 문맥, 그래프 구조 스택을 왼쪽 문맥으로 사용하여 동작하는데, 기존의 확률적 LR 파싱 기법은 그래프 구조 스택이 가지는 왼쪽 문맥 정보가 매우 유용함에도 불구하고 스택 자체의 복잡한 구조 때문에 활성 상태 노드의 상태 번호, 혹은 스택 최상부의 구문 중심어 하나로만 스택 정보를 한정하였다. 그러나, CAM은 스택 내부의 리듀스 경로를 고려하여 각 리듀스 경로에 있는 구성 요소들을 문맥 정보에 추가하고, 이것을 조건 정보로 사용하여 구한 확률을 해당 연산에 부여한다.

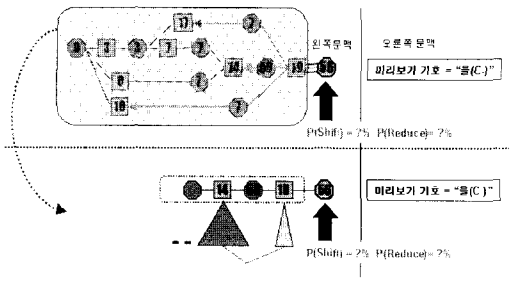

그림 5 기존의 확률적 LR 파싱 모델과 CAM의 차이점. 점선의 윗부분은 기존의 모델을, 아랫부분은 CAM을 나타낸다.

LR 파싱 기법에 스택 내부의 정보를 문맥으로 사용하는 방법은 이미 [5, 7, 15] 등에서 도입된 바 있다. 하지만 이들 방법은 단순 구조의 선형 스택을 기반으로 만들어진 것이며, 스택 내의 서브 트리 정보를 적절히 활용하지 않았다. 반면, CAM은 그래프 구조 스택 내에 부분 생성된 서브 트리 자체를 문맥 정보로 사용하여 구조적 중의성 해소를 수행한다. 이를 위해서는 결국 서브 트리의 구조를 어떻게 효과적으로 기술하느냐가 가장 중요한 요소라고 할 수 있다.

3.3 표층 구문 타입(Surface Phrasal Type)

스택에 만들어 진 부분 파스는 이전까지 수행되어 온 파스 연산들의 결과이고, 이것들은 하나의 이력 정보(history)가 되어 그 다음의 파스에 영향을 준다[16]. 이것은 쉬프트/리듀스 연산을 수행하는 GLR 파서에서도 해당되기 때문에, 파서가 수행할 연산에 대해 더욱 정확한 확률을 할당하기 위해서는 이전까지의 부분 파스를 충분히 고려할 필요가 있다. 따라서 효과적으로 부분 파스를 기술하는 방법이 매우 중요하다. 그래프 구조 스택 내의 부분 서브 트리를 효과적으로 표현할 수 있는 방법에 대해 지금까지 발표된 사례는 없으며, Kwak은 서브 트리의 단말노드 개수와 루트 노드의 문법 태그, 그리고 앞부분에 있는 단말노드의 부분 집합을 사용하여 서브 트리의 구조를 기술한 바 있었다[17].

표층 구문 타입(Surface Phrasal Type, 이후 SPT)은 본 논문에서 제안하는 서브 트리의 구조 기술 방식으로, 주어진 서브 트리의 구조를 나타내는 단위코드(mnemonic)들을 조합한 개념이다. 이때 조합되는 단위코드로는 주어진 서브 트리의 단말 기호 수준에서 내용어(content word)가 아닌 기능어(functional word)를 사용하도록 한다. 이는 기능 범주가 구와 절(문장)의 구조를 결정한다는 생성 문법적인 언어학 이론[18]에 기반한 것으로서, 한국어는 영어보다도 기능어(형식 형태소)가 잘 발달한 언어이기 때문에 단말 기호 수준의 기능어열은 서브 트리의 구조를 기술하는데 있어서 충분한 효과를 가질 수 있다. 주어진 서브 트리(sub-tree)들에 대해 어떤 리듀스 연산이 일어나 부모 노드를 생성할 필요가 있을 때, 각각의 서브 트리에 대해 주어진 어구의 단말 기호열을 뒤에서 앞으로 스캔해 가면서 미리 정의된 단위코드를 찾는다. 이렇게 하여 만들어진 단위코드열을 부분-표층 구문 타입(sub-SPT)이라고 부른다. 리듀스 연산으로 만들어지는 부모 노드의 표층 구문 타입은 각 서브 트리의 부분 표층 구문 타입이 합쳐져 만들어진다. 표층 구문 타입을 구성하는 단위코드는 단말 기호 수준에서 가장 효과적으로 서브 트리의 구조를 나타낼 수 있고 하나의 단위코드의 역할이 다른 단위코드의 역할과 겹치지 않도록 문법 전문가의 문법적 지식을 바탕으로 정의하였다. 본 연구에서는 명사구(NP)와 동사구(VP)를 생성할 수 있는 서브 트리에 대하여 각각의 단위코드 집합을 설정하였으며, 각 단위코드 집합의 요소들은 동일한 문법(구조)적 기능을 가지는 품사들을 대표하는 것으로 정하였다. 표 1과 표 2는 명사구와 동사구의 생성에 대하여 정의한 표층 구문 타입용 단위코드 목록이다.[1]

예를 들어, "그는 예쁜 선녀의 옷을 훔쳤다."라는 문장이 있을 때 GLR 파서가 "을(목적격 조사: PO)"을 미리보기 기호로 읽은 상태에서 만들어지는 스택은 그림 6과 같은데, 활성 스택 노드(56)부터 세 개로 갈라지는 리듀스 경로(Red 1, Red 2, Red 3)를 따라 동시에 생성될 수 있는 명사구(NP)들에 대한 표층 구문 타입은 다음과 같다.

1) 각 표에 나와 있는 PX 단위코드는 주어진 어구가 격구조(postpositional phrase)를 포함하고 있는지의 여부를 나타내는 불린(boolean) 플래그로 처리되며, 반복 출현은 무시된다.

표 1 명사구 생성에 대한 표층 구문 타입 단위코드

단위코드	생성되는 명사구의 특성	문법 구조	형식 형태소 (품사)	예
ED	관형사절에 의해 수식되는 명사구	동사구+ED+명사구	EFD	예쁘+ㄴ 선녀
EN	명사형 전성어미에 의해 바뀐 명사구	동사구+EN	EFN	선녀의 예쁘+ㅁ
PD	속격 명사	명사구+PD+명사구	PD	선녀+의 옷
DX	관형사에 의해 수식된 명사구	DX+명사구	DI, DU, DA	그+ 선녀
XN	나열형 명사구	명사구+XN+명사구+XN+...	AC, PN, SS,	선녀+와 예쁜 옷
PX	필수격구조의 포함 여부를 나타냄	명사구+PX+...	모든 필수격조사	그+가 훔친 옷

표 2 동사구 생성에 대한 표층 구문 타입 단위코드

단위코드	생성되는 동사구의 특성	문법 구조	형식 형태소 (품사)	예
AX	부사에 의해 수식되는 동사구	AX+동사구	AA, AP	빨리+ 훔쳤다
PA	부사격 조사로 만들어진 부사어에 의해 수식되는 동사구	명사구+PA+술부	PA	선녀+에게서 훔쳤다
EA	부사형 전성어미에 의해 바뀐 부사어에 의해 수식되는 동사구	동사구+EFA+동사구	EFA	선녀가 모르+게 훔쳤다
EC	연결형 동사구	동사구+EC+동사구+EC+...	AC, EFC, SS	훔치+어 달아나+다가 넘어졌다
PX	필수격구조의 포함 여부를 나타냄	...+PX+...+동사구	PS, PX, PO	그+가 옷+을 훔쳤다

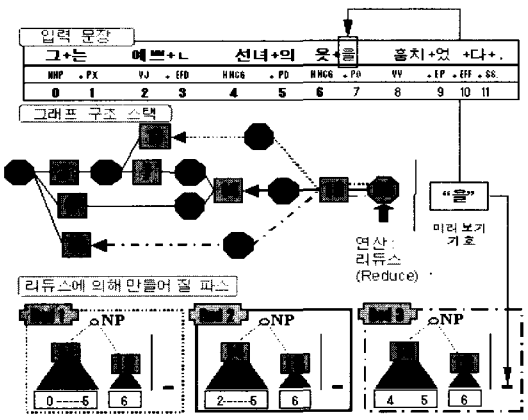

그림 6 GLR 파서의 동작예. 미리보기 기호는 "을"이며, 문장과 서브트리 아래의 번호는 단말 기호의 순서를 나타낸다.

- Red 1 : NP→DP(17) NP(19) ((그는 예쁜 선녀의) 옷)
  - * 17의 부분 표층 구문 타입: {**PD**(의_PD), **ED**(ㄴ_EFD), **PX**(는_PX)}
  - * 19의 부분 표층 구문 타입: {}
  - * 최종 명사구의 표층 구문 타입:[{**PD**, **ED**, **PX**} - {}]
- Red 2 : NP→DP(14) NP(19) ((예쁜 선녀의) 옷)
  - * 14의 부분 표층 구문 타입: {**PD**(의_PD), **ED**(ㄴ_EFD)}
  - * 19의 부분 표층 구문 타입: {}
  - * 최종 명사구의 표층 구문 타입:[{**PD**, **ED**} - {}]
- Red 3 : NP→DP(18) NP(19) ((선녀의) 옷)
  - * 18의 부분 표층 구문 타입: {**PD**(의_PD)}
  - * 19의 부분 표층 구문 타입: {}
  - * 최종 명사구의 표층 구문 타입:[{**PD**} - {}]

이러한 표층 구문 타입을 구조적 문맥 자질로 포함시킴으로써, GLR 파서에 세분화된 통계적 구문 선호도를 쉽게 포함시킬 수 있다. 또한 표층 구문 타입은 주어진 문법 규칙에 의해 생성되는 어구를 기준으로 정의하기 때문에 중의성을 가지고 있는 구 생성을 기술하는 데에도 적합하다. 예를 들어, 아래와 같이

1) ((예쁜 선녀의) 옷)

2) (예쁜 (선녀의 옷))

구조적 중의성을 가지고 있는 위의 두 어구는 단말 기호 수준에서 보면 관형형 전성어미-관형격조사의 동일한 순서이지만, 위의 기준에 따라 단위코드열을 정의하면 서로 다른 구조로 기술된다.

1) ((예쁜 선녀의) 옷) : [{**PD ED**} [}]

2) (예쁜 (선녀의 옷)) : [{**ED**} {**PD**}]

제안된 방법을 부분 파스의 구조 표현에 사용할 때의 장점은 ① 서브 트리의 내부 구조를 모두 고려할 경우

압축된 상태에 있는 서브 트리들의 내부 구조를 일일이 고려해야 하는 오버헤드를 피할 수 있다는 점과, ② 단말노드 수준에서의 단위코드 조합만으로도 내부 구조를 효과적으로 반영할 수 있는 점이다. 그 대신, 대상 언어의 특성을 분석한 결과를 가지고 단위코드를 만들기 때문에 해당 언어의 문법 전문가의 기반 지식이 요구되며, 언어 의존적이라는 단점을 가지고 있다.

3.4 확률적 구조 중의성 해소

입력 문장에 대한 전체 파스는 GLR 파서가 억셉트(accept) 연산에 의해 종료 상태에 왔을 때까지 수행한 연산들의 열(action sequence)로 정의한다. 각 연산은 CAM에 의해 부여된 확률 값을 가지고 수행되며, 각 연산의 확률은 본 논문에서 제안한 조건부 연산 모델에 의해 부여된다. 확률 모델로 보면 이것은 Inui 외 3인이 제안한 스택 전이 모델(stack transition model)[14,15]을 변형시킨 개념이다. 스택 전이 모델은 Briscoe와 Carroll이 제안한 상태 전이 모델(state transition model)[12, 13]의 확률적 결함을 보완한 것으로, i번째에 수행될 연산(a_i)은 i-1번째의 스택(σ_{i-1}), i번째의 미리보기 기호(l_i)에 의해 결정된다고 보고 전체의 파스 확률을 식 (1)과 같이 스택 전이 확률의 곱으로 정의한다.

$$P(T) \approx \prod_{i=1}^{n} P(l_i a_i \sigma_i | \sigma_{i-1}) \tag{1}$$

즉, 스택 전이 확률 $P(l_i a_i \sigma_i | \sigma_{i-1})$는 연산이 일어나기 이전의 스택을 문맥 정보로 하여 미리보기 기호와 연산이 발생하는 사건을 취하여 정의된 것으로, 스택 σ_{i-1}은 활성 스택 노드의 상태 번호(s_{i-1})로 간략화한다. 한편, 확률값은 이전에 취해진 연산이 쉬프트이냐 리듀스이냐에 따라 다르게 계산되는데, 이는 LR 파싱의 특성상 리듀스가 수행된 후에는 미리보기 기호의 변화가 없기 때문이다.

$$P(l_i a_i \sigma_i | \sigma_{i-1}) = \begin{cases} P(l_i a_i | s_{i-1}), (\text{for } s_{i-1} \in S_s) & \sum_{l \in L} \sum_{a \in Act(s,l)} P(a) = 1 \\ P(a_i | s_{i-1}, l_i), (\text{for } s_{i-1} \in S_r) & \sum_{a \in Act(s,l)} P(a) = 1 \end{cases} \tag{2}$$

S_s와 S_r은 각각 쉬프트와 리듀스가 수행된 후에 도달하는 상태의 집합을, $Act(s, l)$은 상태와 미리보기 기호에 의해 결정되는 연산의 집합을 나타낸다.

한편, 표층 구문 타입을 사용한 CAM의 파스 확률은 스택 전이 모델과 달리 주어진 문맥 정보에 대해 쉬프트나 리듀스 연산이 수행되는 사건을 취하여 정의한다. 이때 미리보기 기호와 이전 상태 번호 및 서브 트리의 표층 구문 타입이 문맥 정보로 사용된다. i번째에 수행될 연산의 확률은 다음과 같이 정의된다.

$$P(action_i) = \begin{cases} P(a_i | TY_0, \ldots, TY_{n-1}, s_{i-1}, l_i) & (\text{명사구, 동사구 생성}) \\ P(a_i | , s_{i-1}, l_i) & (\text{이외의 구 생성}) \end{cases} \tag{3}$$

여기서, $TY_0, \ldots TY_{n-1}$은 리듀스 경로에 있는 n개의 서브 트리(어떤 구문규칙에 의해 부모 노드를 생성할 수 있는 서브 트리들)의 부분 표층 구문 타입의 열이다. 파스 확률은 이 연산 확률의 곱이며, CAM을 사용한 GLR 파싱은 가장 높은 파스 확률을 보이는 연산열 T_j을 찾는 과정으로 정의한다.

$$P(T_j|S) \approx \prod_{i=1}^{n} P(action_i^j) \tag{4}$$

$$P(T_{best}|S) = \arg\max_j P(T_j | S) \tag{5}$$

제안된 모델에서 발생할 수 있는 자료 부족 문제(sparse-data problem)을 처리하기 위하여, Collins의 방법[19]과 유사하게 백오프(back-off) 레벨에 딜리티드(deleted) 보간법을 적용하였다. 우선 다음과 같이 네 개의 MLE 추정값을 준비한다.

$$\begin{aligned} E_1 &= \frac{c(a, TY_0, TY_1, s_{i-1}, l_i)}{c(TY_0, TY_1, s_{i-1}, l_i)} \\ E_2 &= \frac{c(a, TY_0, s_{i-1}, l_i) + c(a, TY_1, s_{i-1}, l_i)}{c(TY_0, s_{i-1}, l_i) + c(TY_1, s_{i-1}, l_i)} \\ E_3 &= \frac{c(a, s_{i-1}, l_i)}{c(s_{i-1}, l_i)} \\ E_4 &= \frac{c(a, s_{i-1})}{c(s_{i-1})} \end{aligned} \tag{6}$$

연산 확률은 필요한 표층 구문 타입의 존재 여부에 따라 다음과 같이 계층적으로 계산한다. 우선 명사구와 동사구를 생성하는 경우의 연산 확률은 식(7)과 같이 계산한다.

$$\begin{aligned} &*If\ E_1\text{이 존재함 } (c(TY_0, TY_1, s_{i-1}, l_i) > 0) \\ &\quad p(action) = \lambda_1 \times E_1 + (1 - \lambda_1) \times E_2 \\ &*Else\ \text{if } E_2\text{가 존재함 } (c(TY_0, s_{i-1}, l_i) \\ &\quad + c(TY_1, s_{i-1}, l_i) > 0) \\ &\quad p(action) = \lambda_2 \times E_2 + (1 - \lambda_2) \times E_3 \\ &*Else \\ &\quad p(action) = E_3 \end{aligned} \tag{7}$$

그리고 MLE 추정값 E_3는 다시 E_4를 사용하여 보간하는데, 이 값은 명사구와 동사구 이외의 서브 트리를 생성하는 경우를 고려할 때의 연산 확률값으로도 사용한다.

$$\begin{aligned} &*If\ E_3\text{가 존재함 } (c(s_{i-1}, l_i) > 0) \\ &\quad p(action) = \lambda_3 \times E_3 + (1 - \lambda_3) \times E_4 \\ &*Else \\ &\quad p(action) = E_4 \end{aligned} \tag{8}$$

한편, 각 수식에서 추정값의 가중치를 주기 위한 λ 값

은 Collins가 취한 것과 동일한 방법[19]을 사용하여, 0에 근접하게 적은 회수로 출현한 사건에 낮은 가중치(최소 0.5)를 주었다.

$$\lambda_1 = \frac{c(TY_0, TY_1, s_{i-1}, l_i)}{c(TY_0, TY_1, s_{i-1}, l_i)+1}$$
$$\lambda_2 = \frac{c(TY_0, s_{i-1}, l_i)+c(TY_1, s_{i-1}, l_i)}{c(TY_0, s_{i-1}, l_i)+c(TY_1, s_{i-1}, l_i)+1} \quad (9)$$
$$\lambda_3 = \frac{c(s_{i-1}, l_i)}{c(s_{i-1}, l_i)+1}$$

그림 7은 그림 6의 입력 문장에 대해 발생되는 두 개의 중의적인 파스에 파스 확률을 할당하는 예를 보이고 있다. GLR 파서는 "을(목적격 조사: PO)"을 미리보기 기호로 읽은 시점에서, ((예쁜 선녀의) 옷)과 (예쁜 (선녀의 옷))의 두 개의 파스를 가진 지역적 구조 중의성(local ambiguity)을 생성할 수 있는데, 각각의 파스 생성에는 그림 6에서 보았던 그래프 구조 스택의 활성 스택 노드(상태 56)에 대해 수행되는 두 개의 리듀스 액션 Red 2, Red 3이 가담한다.

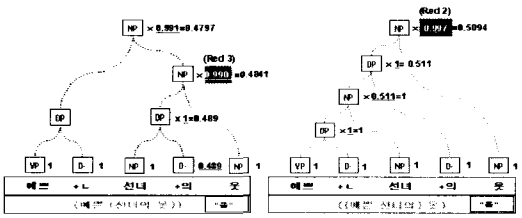

그림 7 구조적 중의성을 가진 부분 파스에 대해 CAM에 의해 할당되는 파스 확률. 밑줄이 있는 부분은 연산 확률을 나타내며, 이 확률과 서브 트리의 파스 확률들과 곱해져서 해당 노드의 파스 확률이 계산된다.

((예쁜 선녀의) 옷)을 만드는 Red 2 리듀스 연산(reduce 35)에 대한 연산 확률은 다음과 같이 계산되며,

$$P(reduce35 \mid \{PD, ED\}, \{\}, 56, PO) = \lambda_1 \times E_1 + (1-\lambda_1) \times E_2 = 0.997$$
$$\lambda_1 = \frac{c(\{PD, ED\}, \{\}, 56, PO)}{c(\{PD, ED\}, \{\}, 56, PO)+1} = \frac{174}{174+1}$$
$$E_1 = \frac{c(reduce35, \{PD, ED\}, \{\}, 56, PO)}{c(\{PD, ED\}, \{\}, 56, PO)+1} = \frac{173}{174}$$
$$E_2 = \frac{c(reduce35, \{PD, ED\}, 56, PO)+c(reduce35, \{\}, 56, PO)}{c(\{PD, ED\}, 56, PO)+c(\{\}, 56, PO)+1} = \frac{240+4540}{241+4555}$$

(예쁜 (선녀의 옷))을 만드는 Red 3 리듀스 연산(reduce 35)에 대한 연산 확률은 다음과 같이 계산된다.

$$P(reduce35 \mid \{ED\}, \{\}, 56, PO) = \lambda_1 \times E_1 + (1-\lambda_1) \times E_2 = 0.990$$
$$\lambda_1 = \frac{c(\{ED\}, \{\}, 56, PO)}{c(\{ED\}, \{\}, 56, PO)+1} = \frac{1826}{1826+1}$$
$$E_1 = \frac{c(reduce35, \{ED\}, \{\}, 56, PO)}{c(\{ED\}, \{\}, 56, PO)} = \frac{1810}{1826}$$
$$E_2 = \frac{c(reduce35, \{ED\}, 56, PO)+c(reduce35, \{\}, 56, PO)}{c(\{ED\}, 56, PO)+c(\{\}, 56, PO)} = \frac{1720+4540}{1745+4555}$$

4. 실험 및 평가

본 논문에서 제안하는 모델의 평가를 위하여, 총 12,084 문장의 한국어 구문 트리 뱅크를 작업 말뭉치로 사용하였다. 이 트리 뱅크는 56개의 CFG 규칙으로 이루어진 한국어 자질 기반 문법[20]으로 구문 태깅되어 있으며, 전체적인 문장의 길이 분포는 표 3과 같다. 작업 말뭉치에 대해 10,906 문장을 모델 학습에, 1,178 문장을 모델 평가에 사용하였다.

표 3 작업 말뭉치의 문장 길이 분포. 평균 형태소 길이는 22.51이다

형태소 길이	문장 개수	형태소 길이	문장 개수
1-10	573	31-40	2241
11-19	4244	41-50	350
20-29	4650	51-59	26

제안하는 조건부 연산 모델의 학습을 위하여, 선형 스택을 사용한 LR 파서를 구현한 후 이것을 사용하여 트리 뱅크로부터 리듀스 연산과 쉬프트 연산을 역으로 해독하고 연산 시의 문맥 정보 및 부분 문맥을 추출하였다. 표층 구문 타입을 제한 없이 추출했을 때 심한 자료 부족 문제가 있었기 때문에, 동일한 단위코드가 연속으로 나오는 회수를 1로 정하고, 표층 구문 타입의 최대 길이(단위코드 개수)를 3으로 정하였다. 이 수치는 실험적으로 가장 좋은 성능을 보인 수치이다. 모델 학습 결과의 예는 그림 8과 같다.

GLR 파서는 한국어 자질 기반 문법[20]에서 제공되는 이진(binary) 문맥 자유 문법 규칙으로부터 만든 68개 상태의 정준형(canonical) SLR 파싱 테이블을 사용하고, 이상주에 의해 정의된 품사집합[21]의 품사열을 입력으로 받아 동작한다. 한편, 통계적으로 거의 만들어지지 않는 구문 구조로의 리듀스 연산을 막기 위해, 언어 현상에 대해 구분된 속성 집합으로부터 학습한 28,347개의 문법 제약[22]을 사용한다. GLR 파서는 이것을 사용하여 리듀스 연산을 수행하기 전에 각 서브 트리들의 속성을 점검하고, 점검 결과에 따라 리듀스 연

```
(1) 연산 : 상태 : 미리보기 문법태그 : 미리보기 품사 : [TYO-TY1]          @빈도)
(2) 상태 : 미리보기 문법태그 : 미리보기 품사 - [TYO-TY1]                 @빈도)

1) RE35 56 11 PO [ 31- ]                @173
2) 56 11 PO  [ 31- ]                    @174
...
1) RE35 56 11 PO [ 31-X ]               @240
2) 56 11 PO [ 31-X ]                    @241
...
1) RE35 56 11 PO [ -X ]                 @4540
2) 56 11 PO [ -X ]                                @4555
```

그림 8 학습된 CAM 모델의 예. 표층어구 타입의 단위코드는 숫자로 대표하였다. 3=PD, 1=ED 이고, X는 고려하지 않음을 나타내며, 빈 부분은 발견된 단위코드가 없다는 뜻이다.

산을 제어한다. ANSI C++ 및 STL로 구현한 본 GLR 파서는 1.2Ghz의 Athlon CPU와 256Mbyte의 메모리를 가진 Windows 2000 시스템에서 평균 0.1998초에 한 문장에 대한 모든 분석 후보를 출력하고(12,084 문장의 구문 분석에 총 2,414.61초 소요), 확률적 구조 중의성 해소를 함께 수행시켰을 때 문장 당 0.3316초의 수행 속도를 보인다. 모델 평가 시, 입력 원시 문장을 모든 단어에 대해서 100%의 정확도로 수동으로 품사 태깅하고 휴리스틱과 기정의 테이블에 의한 복합명사 및 복합용언구 인식단계를 거치게 한 후 파서에 입력하였다.

제안하는 모델과 기존의 확률적 GLR 파싱 모델을 비교하기 위하여, Briscoe와 Caroll의 상태 전이 모델[12, 13], Inui 외 3인의 스택 전이 모델[14, 15], 그리고 단말노드의 개수와 앞쪽의 형태소로써 서브 트리의 구조를 기술한 조건부 연산 모델[17]을 참조 구현하였다. 이전의 모델들과 제안된 모델을 GLR 파서에 부착하여 학습 코퍼스와 실험 코퍼스에 적용한 결과는 표 4와 같다. 각각의 모델을 평가하기 위하여 Labeled Recall(LR)과 Exact Matching(EM) 기준을 사용하였다[23].[2] 각 기준은 다음과 같이 계산된다.

$$LR(\%) = \frac{\text{정답과 맞는 시스템 파스의 개수}}{\text{트리뱅크에 있는 정답 노드의 개수}} \times 100$$
$$EM(\%) = \frac{\text{정답과 정확히 맞는 문장의 개수}}{\text{트리뱅크에 있는 문장의 개수}} \times 100 \quad (10)$$

실험 결과에서 볼 수 있듯이 제안하는 모델은 이전 모델과 비교하여 약 6-7% 정도의 성능 향상을 보였다. 이는 문맥에 대해 연산이 발생하도록 재정의한 확률 모델과, 명사구와 동사구 생성 시점에서 문맥 정보로 추가한 표층 구문 타입이 성능 향상에 도움을 주었기 때문이다. 이를 통해 본 논문에서 제시한 표층 구문 타입이

표 4 제안된 모델과 이전 모델의 구문 분석 성능 비교. LR과 EM 열 각각에서 윗부분은 학습 데이타에 대한 모델 성능을, 아랫부분은 실험 데이타에 대한 모델 성능을 나타낸다

	Briscoe & Carroll	Inui et al.	Kwak 2001	제안된 모델
LR	72.016%	74.792%	77.231%	83.636%
	71.223%	74.271%	76.010%	82.181%
EM	2.131%	3.810%	6.991%	12.937%
	1.702%	3.771%	6.037%	10.357%

구문 구조를 기술하는 방법으로 효과적이며, 확률적 GLR 파싱에 문맥 정보로 사용될 수 있음을 알 수 있다. 또한, 제안하는 모델은 아직 어휘화가 되지 않았기 때문에 적절한 어휘화를 거친다면 더 좋은 성능을 보일 수 있다. 표층 구문 타입을 문맥 정보에 추가한 효과는 형태소 길이 10-19 사이의 짧은 길이의 문장에서 나타났으며, 특히 EM(exact matching) 값이 눈에 띄게 높아졌다. 이 효과는 명사구와 동사구에 대한 표층 구문 타입이 동시에 쓰일 때 더욱 잘 드러났는데, 동사구에 대한 표층 구문 타입이 없을 때에 명사구 표층 구문 타입에 의해 맞는 구조로 선택된 명사구 서브 트리가 최종 구문 분석 결과에 포함되지 않는 경우가 많았기 때문이다. 그림 9는 그러한 양상을 보이는 문장 분석 예이다. "소각로 및 차도 등"에 대해 만들어지는 두 개의 중의적 명사구에서, (((소각로 및) 차고) 등)은 ((소각로 및) (차고 등))보다 높은 부분 파스 확률을 갖는다. 이는 "및(접속 부사: AC)"에 의해 나타나는 명사구 표층 구문 타입의 단위코드 XN(나열형 명사구)에 의해 리듀스 Re_1의 빈도값과 쉬프트 Sh_1의 빈도값이 확실히 구분되어 P(Re_1)이 P(Sh_1)보다 크기 때문이다. 한편, "소각로 및 차도 등이 있다"에 대해 만들어지는 두 개의 중의적 동사구에서는 ((소각로 및 차도 등이) (있다))는 ((소각로 및) (차도 등이 있다))보다 더 높은 파

2) LR 파싱 테이블을 만드는 데 사용된 CFG 규칙들이 모두 이진 형식(binary branching)이기 때문에 Labeled Recall과 Labeled Precision의 값이 동일하다. 따라서 Labeled Recall만을 사용하였다.

스 확률을 갖는다. 이는 동사구 표층 구문 타입의 단위 코드 PX (필수격)가 앞쪽 서브 트리에 들어 있는 경우의 리듀스 연산 확률(P(Re_2))이 PX가 뒤쪽 서브 트리에 들어 있는 경우의 리듀스 연산 확률(P(Re_3))보다 더 높기 때문이다.

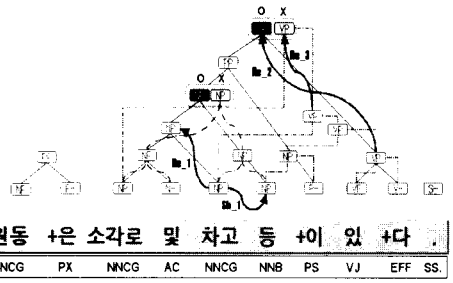

그림 9 명사구와 동사구에 대한 표층 구문 타입이 구문 분석에 영향을 주는 예. 압축된 파스 구조 내에서 역상으로 나타난 노드는 GLR 파서에 의해 맞는 파스로 골라진 것을 뜻한다. 실제의 정답 및 오답은 O와 X로 나타내었다.

제안된 모델의 문제점은 다음과 같다. 첫째, 제안된 표층 구문 타입으로도 다른 구문 생성 경로에 대해 다른 확률을 부여해 주지 못할 수 있다. 그림 9의 분석 결과도 그러한 예를 보여주는데, (있다)를 중심으로 만들어 질 수 있는 세 개의 파스 ((소각로 및 차도 등이) 있다), ((차도 등이) 있다), 그리고 ((등이) 있다)는 필수격이 포함된 구와 동사구가 결합되는 똑같은 표층 구문 타입에 의해서 여전히 똑같은 리듀스 연산 확률을 부여받는다. 둘째, 자료 부족 문제로 인해 표층 어구 타입의 단위코드 길이를 최대 세 개로 제한했기 때문에 반복되지 않는 단위코드의 개수가 세 개를 초과하는 구문 구조가 고유의 통계값을 가지지 못하는 문제도 있다. 긴 단말 기호열을 가진 구문 구조도 적은 정보로 효과적으로 표현할 수 있도록 하기 위해, 구문 구조의 지배를 받는 단말 기호열 전체에 대해 서브 트리의 구조를 나타내는 결정적인 기능어를 추출하는 일반화 과정이 필요할 것으로 생각된다.

5. 결 론

본 논문에서는 기존의 확률적 LR(GLR) 파싱 기법이 가지고 있는 문제를 개선한 조건부 연산 모델(CAM)을 소개하였다. 기존의 기법은 내부의 그래프 구조 스택이 가진 복잡성 때문에 상당히 제한된 문맥 정보만을 사용하여 왔으나, 제안하는 기법은 파싱 도중에 동적으로 생성되는 그래프 구조 스택에 포함된 부분 생성 파스를 조금 더 깊게 고려하여 쉬프트 및 리듀스 연산의 확률을 추정하여 파싱 동작을 전개한다. 그래프 구조 스택에 들어 있는 부분 생성 파스의 구조를 표현하기 위하여 표층 구문 타입(Surface Phrasal Type)이라고 불리는 기술 방식을 사용하였으며, 실험을 통해 표층 구문 타입이 파스 구조의 기술 방법으로 효과적이며, 표층 구문 타입을 확률적 LR 파싱의 문맥 정보로 활용할 수 있음을 보였다.

추후작업으로는 표층 구문 타입을 더욱 GLR 파싱에 효과적으로 만들기 위한 일반화를 수행할 필요가 있다. 이를 통하여 단말 기호열 전체에 대해 서브 트리의 구조를 나타내는 결정적인 기능어만을 추출하여 단말 기호열을 많이 가지는 서브 트리의 구조도 적은 정보로 효과적으로 표현할 수 있도록 해야 한다. 또한, 모델의 어휘화를 통해 더욱 정확한 구문분석을 수행할 수 있도록 할 예정이다.

참 고 문 헌

[1] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman, Compilers : Principles, Techniques, and Tools, Addison-Wesley, 1986.

[2] Masaru Tomita, *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*. Kluwer Academic Publishers, 1986.

[3] Ulf Hermjakob, Rapid Parser Development: A Machine Learning for Korean. In Proceedings of the North American chapter of the Association for Computational Linguistics (NAACL-2000), pages 118-123, 2000.

[4] Alon Lavie. GLR*: A Robust Parser For Spontaneously Spoken Language. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, 1996.

[5] Tobias Ruland, A Context-Sensitive Model for Probabilistic LR Parsing of Spoken Language with Transformation-Based Postprocessing. In Proceedings of the 18th International Conference on Computational Linguistics, pages 677-683, 2000.

[6] Robert F. Simmons and Yeong-Ho Yu, The Acquisition and Use of Context-Dependent Grammar for English. In Proceedings of the 29th annual Meeting of the Association for Computational Linguistics, pages 122-129, 1991.

[7] 곽용재, 한국어의 결정적 구문 분석을 위한 문맥 의존 문법 규칙의 획득과 적용, 고려대학교 컴퓨터학과 석사 학위 논문, 1999.

[8] Ulf Hermjakob, *Learning Parse And Translation Decisions From Examples with Rich Context*. PH.D thesis, University of Texas at Austin, 1997.
[9] Aboy Wong and Dekai Wu, Are Phrase Structured Grammars Useful in Statistical Parsing?. In Proceeding of 5th Natural Language Processing Pacific Rim Symposium, pages 120-124, 1999.
[10] J. H. Wright and E. N. Wrigley, GLR Parsing with Probability. In *Generalized LR Parsing*. Kluwer Academic Publishers, 1991.
[11] Keh-Yi Su, Jong-Nae Wang, Mei-Hui Su, and Jing-Shin Chang, GLR Parsing with Scoring. In *Generalized LR Parsing*, Kluwer Academic Publishers, pages 93-112, 1991.
[12] Ted Briscoe and John Carroll, Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars. In Computational Linguistics,19(1):pages 25-59, 1993.
[13] Ted Briscoe and John Carroll, Developing and evaluating a probabilistic LR parser of part-of-speech and punctuation labels. In Proceedings of the 4th ACL/SIGPARSE International Workshop on Parsing Technologies, Prague, Czech Republic, pages 48-58, 1995.
[14] Kentaro Inui, Virach Sornlertlamvanich, Hozumi Tanaka and Takenobu Tokunaga, A New Formalization of Probabilistic GLR Parsing. In Proceedings of the 5th International Workshop on Parsing Technologies, 1997.
[15] Kentaro Inui, Virach Sornlertlamvanich, Hozumi Tanaka and Takenobu Tokunaga, Probabilistic GLR parsing: a new formalization and its impact on parsing performance. In Journal of Natural Language Processing, Vol. 5, No. 3, pages 33-52, 1998.
[16] E. Black, F. Jelinek, J. Lafferty, D. M. Magerman, R. Mercer, and S. Roukos. Toward history-based grammars: Using richer models for probabilistic parsing. Proceedings of the February 1992 DARPA Speech and Natural Language Workshop, 1992.
[17] Yong-Jae Kwak. So-Young Park, Hoojung Chung, Young-Sook Hwang, Sang-Zoo Lee, Hae-Chang Rim, GLR Parser with Conditional Action Model (CAM). In proceedings of 6th Natural Language Processing Pacific Rim Symposium, pages 359-366, 2001.
[18] Noam Chomsky *Lectures on Government and Binding*, Foris, 1981.
[19] Michael Collins, Head-Driven Models for Natural Language Parsing. PH.D thesis, Dept. of Computer and Information Science, University of Pennsylvania, 1999.
[20] So-Young Park, Young-Sook Hwang, Hoojung Chung, Yong-Jae Kwak, and Hae-Chang Rim, A Feature-based Grammar for Korean Parsing. In proceedings of 5th Natural Language Processing Pacific Rim Symposium, pages 167-171, 1999.
[21] 이상주, 자동 품사 부착을 위한 새로운 통계적 모형, 고려대학교 컴퓨터학과 박사 학위 논문, 1999.
[22] So-Young Park, Yong-Jae Kwak, Hoojung Chung, Young-Sook Hwang, Sang-Zoo Lee, Hae-Chang Rim, A Feature-based Korean Grammar with Unification Constraints, In proceedings of International Conference on Speech Processing 2001, pages 995-999, 2001
[23] Joshua Goodman, Parsing Algorithms and Metrics. In Proceedings of the 34th Annual Meeting of the ACL, pages 177-183, 1996.

곽 용 재
1997년 8월 고려대학교 컴퓨터학과 학사. 1999년 8월 고려대학교 컴퓨터학과 석사. 2001년 9월 ~ 현재 고려대학교 정보통신공동연구소 연구원. 관심분야는 자연어처리, 구문분석, 정보검색, 기계학습

박 소 영
1997년 2월 상명대학교 전자계산학과 학사. 1999년 8월 고려대학교 컴퓨터학과 석사. 1999년 9월 ~ 현재 고려대학교 컴퓨터학과 박사과정. 관심분야는 자연어처리, 구문분석, 정보검색, 기계학습

황 영 숙
1991년 2월 고려대학교 전산과학과 학사. 1991년 ~ 1995년 쌍용정보통신 근무. 1998년 2월 고려대학교 컴퓨터학과 석사. 1998년 ~ 현재 고려대학교 컴퓨터학과 박사과정. 관심분야는 자연어처리, 기계학습, 정보추출

정 후 중

1997년 2월 고려대학교 컴퓨터학과 학사. 1999년 8월 고려대학교 컴퓨터학과 석사. 2001년 3월 ~ 현재 고려대학교 정보통신공동연구소 연구원. 관심분야는 자연어처리, 정보 검색

이 상 주

1992년 2월 고려대학교 컴퓨터학과 학사. 1995년 2월 고려대학교 컴퓨터학과 석사. 1999년 8월 고려대학교 컴퓨터학과 박사. 1999년 11월 ~ 2001년 3월 일본 동경학교 정보과학과 연구원(일본학술진흥회 지원). 1997년 3월 ~ 2002년 2월 고려대학교 기초과학연구소 연구원. 2002년 3월 ~ 현재 (주)엔엘피솔루션 대표이사. 관심분야는 자연어처리, HCI, 기계학습, 정보검색

임 해 창

1991년 ~ 현재 고려대학교 컴퓨터학과 교수. 1993년 인지 과학회 이사. 1994년 ~ 1998년 한국 정보과학회 편집위원. 1998년 5월 ~ 2000년 5월 한국정보과학회 한국어정보처리연구회 운영위원장. 1999년 3월 ~ 2000년 8월 고려대학교 컴퓨터과학기술연구소 연구소장. 관심분야는 자연어처리, 구문분석, 정보검색, 기계학습