확률적 문법규칙에 기반한 국어사전의 뜻풀이말 구문분석기

(A Parser of Definitions in Korean Dictionary based on Probabilistic Grammar Rules)

이 수 광 † 옥 철 영 ††

(Soo-Kwang Lee) (Cheol-Young Ock)

요 약 국어사전의 뜻풀이말은 표제어의 의미를 기술할 뿐만 아니라, 상위/하위개념, 부분-전체개념, 다의어, 동형이의어, 동의어, 반의어, 의미속성 등의 많은 의미정보를 내재하고 있다. 본 연구는 뜻풀이말에서 다양한 의미정보를 획득을 위한 기본적인 도구로서 국어사전의 뜻풀이말 구문분석기를 구현하는 것을 목적으로 한다. 이를 위해서 우선 국어사전의 뜻풀이말을 대상으로 일정한 수준의 품사 및 구문 부착 말뭉치를 구축하고, 이 말뭉치들로부터 품사 태그 중의성 어절의 빈도 정보와 통계적 방법에 기반한 문법규칙과 확률정보를 자동으로 추출한다. 본 연구의 뜻풀이말 구문분석기는 이를 이용한 확률적 차트파서이다. 품사 태그 중의성 어절의 빈도 정보와 문법규칙 및 확률정보는 파싱 과정의 명사구 중의성을 해소한다. 또한, 파싱 과정에서 생성되는 노드의 수를 줄이고 수행 속도를 높이기 위한 방법으로 문법 Factoring, Best-First 탐색 그리고 Viterbi 탐색의 방법을 이용한다. 문법규칙의 확률과 왼쪽 우선 파싱 그리고 왼쪽 우선 탐색 방법을 사용하여 실험한 결과, 왼쪽 우선 탐색 방식과 문법확률을 혼용하는 방식이 가장 정확한 결과를 보였으며 비학습 문장에 대해 51.74%의 재현률과 87.47%의 정확률을 보였다.

Abstract The definitions in Korean dictionary not only describe meanings of title, but also include various semantic information such as hypernymy/hyponymy, meronymy/holonymy, polysemy, homonymy, synonymy, antonymy, and semantic features. This paper purposes to implement a parser as the basic tool to acquire automatically the semantic information from the definitions in Korean dictionary. For this purpose, first we constructed the part-of-speech tagged corpus and the tree tagged corpus from the definitions in Korean dictionary. And then we automatically extracted from the corpora the frequency of words which are ambiguous in part-of-speech tag and the grammar rules and their probability based on the statistical method. The parser is a kind of the probabilistic chart parser that uses the extracted data. The frequency of words which are ambiguous in part-of-speech tag and the grammar rules and their probability resolve the noun phrase's structural ambiguity during parsing. The parser uses a grammar factoring, Best-First search, and Viterbi search in order to reduce the number of nodes during parsing and to increase the performance. We experiment with grammar rule's probability, left-to-right parsing, and left-first search. By the experiments, when the parser uses grammar rule's probability and left-first search simultaneously, the result of parsing is most accurate and the recall is 51.74% and the precision is 87.47% on raw corpus.

1. 서 론

국어사전은 한국어정보처리를 위한 기본적인 형태정보(품사, 활용형, 준말)와 구문정보(용법, 관용구) 등의 정보를 포함하고 있다. 또한 국어사전은 표제어의 뜻풀이말, 전문어, 용례 등과 같은 의미정보들을 포함하고

*이 논문은 1999년 한국학술진흥재단(1998-0001-E01184) 및 2000년 정보통신 우수시범학교 지원사업의 지원에 의하여 연구되었음.

† 비 회 원 : 울산대학교 컴퓨터정보통신공학부
skl@future.co.kr

†† 종신회원 : 울산대학교 컴퓨터정보통신공학부 교수
okcy@mail.ulsan.ac.kr

논문접수 : 2000년 3월 8일
심사완료 : 2001년 3월 28일

있다. 뜻풀이말은 표제어의 의미를 기술할 뿐만 아니라, 상위/하위개념[11], 부분-전체개념, 다의어, 동형이의어, 관련어(동의어, 반의어), 의미속성[10] 등의 많은 의미정보를 내재하고 있다. 예를 들어, 표제어 '**가마터**'의 뜻풀이말 "질그릇이나 사기 그릇을 굽는 가마가 있는 옛터. 요지(窯址)."에서 '옛터'는 '가마터'의 상위개념이며 '요지'는 동의어, '굽다', '질그릇', '사기그릇' 등은 '가마터'의 의미를 설명하는 관련 정보이다. 이러한 의미정보는 의미분석이나, Word Sense Disambiguation(WSD), 정보검색 시에 활용된다[21-25].

[11]에서는 사전의 뜻풀이말에서 표제어('가마터')의 상위개념(뜻풀이말에서의 의미적 중심어, '터')을 추출하고, 상위개념('터')의 뜻풀이말에서 상위개념('자리')를 추출하는 방식으로 명사의미계층구조를 구축하였다. 이러한 과정에서 뜻풀이말의 기술형태를 11가지로 분류하고 각 형태에 따른 상위개념을 추출하였다. 또한, [10]에서는 [11]에서 구축된 명사의미계층구조의 문제점을 해결하기 위해 뜻풀이말에서 상위개념을 수식하는 수식어구('있다', '굽다')를 의미속성으로 정의하고 의미속성에 기반한 명사의미계층구조를 재구축하였다.

[10, 11]의 경우 특정 MRD(Machine Readable Dictionary)로부터 필요한 상위개념과 의미속성을 추출하여 명사 계층구조를 구축하고 명사의 의미 TAG[1]를 설정하였다. 그러나 뜻풀이말 기술형태에 따른 상위개념을 선택하는 데 사람이 개입함으로써 구축 비용이 많이 들며, 의미속성을 단순히 상위개념의 바로 앞 수식어구만을 고려함으로써 대량으로 양질의 의미속성('가마터'의 예에서 '굽다')을 획득할 수 없다. 또한 [10,11]의 방법은 뜻풀이말 기술방법이 다른 사전에 적용할 때는 동일한 수작업 과정을 반복해야 하는 문제점이 있다.

사전의 뜻풀이말과 용례에는 다양한 언어 지식이 내재되어 있어, 그것을 통해 어휘의 의미를 파악할 수 있으며 필요한 의미속성을 추출할 수 있다. 특히 사전의 뜻풀이말은, 자연어처리과정의 의미분석에서 반드시 필요한 의미정보를 구축하기 위한 가장 기본적인 정보원이며 WSD을 위한 자체적으로도 훌륭한 말뭉치(Corpus)이다. 본 논문은 [10,11]의 방법을 다른 MRD에도 쉽게 적용하고 다양한 의미정보를 자동으로 추출하기 위한 기본 도구로서, 뜻풀이말 구문분석기(DPAS : Definition PArSer)를 개발하는 것을 목적으로 한다.

2. 관련 연구

구문 분석은 문장이 통사적으로 어떤 구조를 가지고 있는지를 밝히고 그것에 따라 문장의 구조를 생성해 내는 작업으로서, 이를 위한 여러 가지 문법이론들이 제시되었다. 그리고 이것을 컴퓨터에서 효율적으로 처리하기 위해 여러 가지 구문 분석 기법들이 등장했다. 그러나 현재의 어떤 문법 이론도 자연언어의 모든 문장 구조를 인식, 생성하기에는 역부족인 것으로 판단되고 있다. 또한, 문법 이론이 아무리 좋다고 하여도 그것을 구현하고 실제로 문장을 분석하는 시스템의 효율이 나쁘다면 실용적인 시스템으로 사용할 수 없을 것이다. 그래서 여러 가지 변형된 구문 분석 방법들이 연구되고 적용할 언어에 맞게 개량되고 있다.

한국어는 자유로운 어순과 불완전한 문장을 허용하는 구문적 특성 때문에 한국어의 구문 분석에는 단일화에 기반한(Unification-based) 구문 분석과 의존 문법(Dependency Grammar)에 기반한 구문 분석이 주목을 받아 왔다. 단일화에 기반한 구문 분석[2]은 문장의 구조적인 구문 규칙보다는 구문 구성 요소가 갖는 자질 정보(Feature Structure)에 따라 문장을 분석한다. 자질 정보에는 통사 정보 뿐만 아니라 어휘 정보, 의미정보가 통합적으로 표현되며 단일화라는 연산으로 문법규칙들을 선언적으로 기술한다. 이 문법 체계는 구조적인 정보보다는 자질 정보에 의존하기 때문에 자질 구조의 원천이 되는 어휘 사전의 중요성이 크게 강조되며, 이러한 면에서 통계적인 접근이 어렵다. 한편 의존 문법에서는 부적당한 의존 관계를 제거하는 것이 중요한 문제 중의 하나이다. 부적당한 의존 관계를 줄이기 위해서는 의존 관계가 정의되는 범주(Category)들을 세분하고 그것들 사이의 의존 관계를 정교하게 설정하여야 한다. 또한, 여러 가지 정보들을 이용하여 구문 단위들 간에 형성되는 의존 관계를 제약하여야 한다[4].

앞서의 구문 분석 방법론에서는 어휘의 자질 정보나 범주와 같은 언어 정보가 구문 분석에서 대단히 중요한 역할을 한다. 그러나 자질 정보의 명세, 격틀 정보의 구축, 어휘 범주의 설정, 의존 관계의 제약 등의 언어 지식을 견고하고 정확하게 구축하기 위해 실세계의 모든 언어 현상을 고려하기는 어렵고, 지속적으로 변화하는 언어 현상을 반영하기 위해서는 계속해서 언어 지식을

1) 의미 TAG란, 하나의 어휘가 나타내고자 하는 의미와 관련된 정보로서 의미 중의성을 지닌 다의어(Polysemy)나 동형이의어(Homonym)의 의미 해석시에 해당 어휘의 의미를 구분하는 역할을 한다[10].

2) 중심어 주도형 구구조 문법(HPSG, Head-driven Phrase Structure Grammar)이나 어휘 기능 문법(LFG, Lexical Functional Grammar) 및 PATR-II 등 [3, 7, 8]이 이 범주에 속한다.

획득해야 한다. 이와 같은 이유로 구문 분석 방법론은 언어의 변화에 능동적으로 대처할 수 없다고 할 수 있다.

반면에, 최근의 통계적 자연언어처리는 대용량 말뭉치로부터 기계학습 방법에 의해 획득된 통계적 언어 지식을 기반으로 언어 모델을 구축하여 자연언어처리 각 단계에서 발생하는 여러 중의성을 해결하는 연구에서 성공적인 결과를 보이고 있다[6,9,15,16, 19,20,27-29].

초기의 연구로는 [27]에서 Mutual Information을 사용하여 구의 경계를 결정하려 하였다. 단어간의 예상도에 의한 지역 최적해가 구의 경계와 잘 일치한다는 것이 주된 가정이었다. 즉, 두 단어를 중심으로 왼쪽 단어를 포함하는 왼쪽 단어열과 오른쪽 단어를 포함하는 오른쪽 단어열이 비교적 낮은 Mutual Information을 갖는다면 이 두 단어가 두 구의 경계 단어가 되는 것이다.

[28]에서는 Simulated Annealing 기법을 이용하여 문장을 분석하였다. 우선, 임의의 구문 구조를 입력으로 하여 이 구조의 질을 평가할 수 있는 평가함수가 정의된다. 그리고 비단말 노드의 이름을 변경하거나 트리를 재구성하는 등의 일련의 변경 사항들이 정의된다. 마지막으로 Simmulated Annealing 기법을 이용하여 탐색공간을 이동하며 구문 분석을 수행한다.

[29]에서는 확률문법으로 TSG(Tree Substitution Grammar)를 이용하였다. TAG(Tree Adjoining Grammar)와 유사하게 기본 규칙의 단위는 트리가 되나, 트리의 결합 연산은 대체만이 가능하다. [29]는 구문 구조가 부착된 말뭉치로부터 모든 가능한 문법규칙(부분 트리)을 추출하며, TSG에서의 확률 매개변수는 문맥 자유 구구조 문법과 마찬가지로 부분트리의 발생 확률로 정의된다. TAG에서는 하나의 구문트리를 생성해 내는 유도트리(Derivation Tree)가 하나 이상 존재할 수 있으며, 따라서 가장 높은 확률을 갖는 최적의 구문 트리를 찾기 위해서는 지수승의 시간이 소요된다. 실험에 사용된 말뭉치는 Penn Treebank 중의 Air Travel Information System(ATIS) 말뭉치를 수작업으로 수정한 750 문장으로서, 문장 유형이 질문과 명령형 뿐인 비교적 단순한 문장 구조의 구문 구조 부착 말뭉치이다. 학습용으로 사용된 675개 문장의 구문 구조로부터 모두 40만개의 규칙(부분 구문 트리)이 대치 확률과 함께 추출되었으며, 75개의 문장에 대해 실험되었다. 평가 기준으로 완전 일치(exact match)를 사용하였음에도 상당히 높은 정확률을 보였다. 그러나, 제한된 영역의 단순한 구문 구조를 갖는 말뭉치에 대한 실험이었음에도 규칙의 수가 상당히 많았던 점을 감안하면 좀 더 복잡한 구문 구조를 갖는 영역의 말뭉치에 대해서는 정확률에 대한 결과를 예측할 수 없다. 학습용 말뭉치가 대상 영역의 구문 구조를 대표할 정도가 되어야 위와 같은 실험 결과를 얻을 수 있으며, 이를 위해서는 많은 학습용 말뭉치가 필요하고, 또한 여기서 추출되는 구문규칙 또한 상당히 많아 질 것이다. 그러나, 비교적 간단한 구문 구조를 갖는 제한된 영역에서의 통계를 이용한 성공적 구문 분석은 주목할 만하다.

본 논문에서 구현하고자 하는 뜻풀이말 구문분석기는 통계적 방법론에 입각하여 국어사전의 뜻풀이말에 내재된 확률적 문법규칙을 추출하여 이를 이용한다. 또한, 기존의 차트파싱방법에서 수 많은 문법규칙을 다루는 데에 어려움을 보완하기 위한 방법으로 문법 Factoring, Viterbi, Best-First 탐색방법을 적용하였다.

국어사전의 뜻풀이말은 어휘의 개념을 설명하기 위해 간단 명료하고 쉬운 표현을 사용하며 어휘와 구문 표현에 있어서도 한정된 패턴으로 이루어져 있다[11,26]. 이렇게 사전의 뜻풀이말에서 나타나는 구문은 일반적인 문장에 비해 폐쇄적이고 어느 정도의 규칙성을 가지기 때문에 문법 체계의 대립항의 상대적 빈도가 바로 언어 체계에 대해 부분적 의미를 갖는 확률로 해석될 수 있는 이점을 가진다[15].

3. 어절 태그 사전과 트리 사전

DPAS에서는 어절 태그 사전(Word-Tag Dictionary)과 트리 사전(Tree Dictionary)이 품사 태깅과 파싱 과정에 각각 사용된다(그림 1). 어절 태그 사전은 품사태거를 이용하여 구축된 품사 부착 말뭉치에서, 트리 사전은 수동으로 구축된 구문 부착 말뭉치에서 자동으로 구축된다.

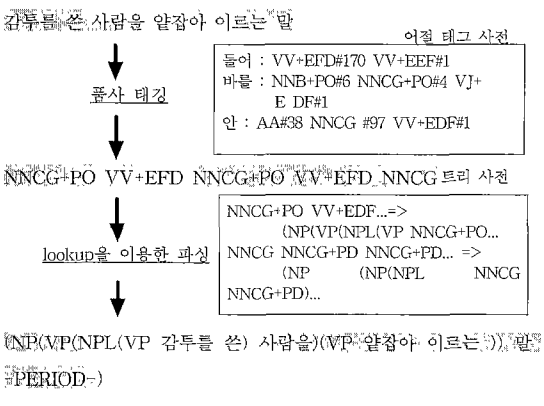

그림 1 어절 태크 사전과 트리 사전을 이용한 파싱

3.1 어절 태그 사전

어절 태그 사전은 표 1과 같이 어절, 어절의 품사 태그열[9]과 빈도수로 구성된다. 빈도수는 품사 태그열의 출현 횟수로 4.2절에서 설명될 사전확률(Dictionary Probability)의 계산에 이용된다.

표 1 어절 태그 사전의 예

들어	VV+EFD#170 VV+EFF#1
바를	NNB+PO#6 NNCG+PO#4 VJ+EFD#1 VV+EFD#2
수도	NNB+PX#13 NNCG#147 NNCG+PX#1
안	AA#38 NNCG#97 VV+EFD#1
줄	NNB#151 NNBU#2 NNCG#16 VV+EFD#4 VX+EFD#9
진	NNB+I+EFD#1 NNCG#2 VV+EFD#52 VX+EFD#11
하나	NNCG#10 NU#18 VV+EFC#1 VV+EFF#1998 VX+EFF#25

어절 태그 사전을 구축하기 위하여 MRD에서 76,975개의 명사의 뜻풀이말을 추출한 다음, 지능형형태소분석기[3]를 이용하여 자동 태깅한 후 태깅오류를 수작업으로 수정한 품사 부착 뜻풀이말을 구축하였다. 어절 태그 사전의 태그열 및 빈도는 품사 부착 뜻풀이말에서 수집하였으며, 수집된 총어절은 81,084개이고 어절 태그열의 종류는 총 1,563개 였으며 어절당 평균 분석 결과는 1.03개 였다.

3.2 트리 사전

어절의 품사 태그열과 구별하기 위해 문장내 전체 어절의 품사 태그열을 '문장 태그열' 이라고 부르기로 한다. 총 76,975개 문장을 품사 태깅하였는데 32,547가지의 문장 태그열이 존재하였다. 이 중 다빈도 문장 태

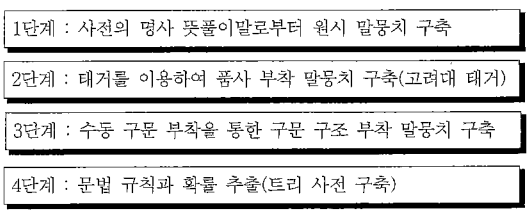

그림 2 트리 사전의 구축 과정

3) 1999년 문화관광부의 21세기 세종계획 프로젝트 산출물.

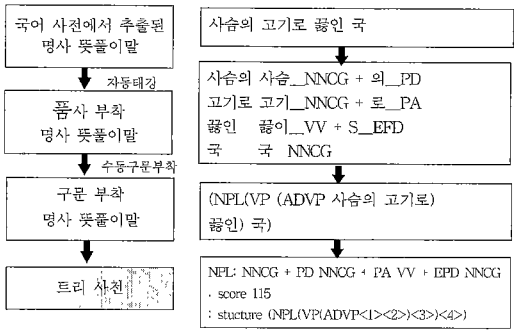

그림 3 트리사전 구축 과정의 예

그열에 해당되는 10,025개의 문장들에 대해 반자동으로 구문 부착하였고, 이 구문 부착 말뭉치로부터 411개의 문법규칙과 문법확률을 추출하여 트리 사전을 구축하였다. 트리 사전은 문법규칙, 스코어, 구문 구조로 구성되며 그림 2, 그림 3와 같은 과정을 거쳐 생성된다.

본 논문에서는 확률적 문맥 자유 문법(PCFG, Probabilistic Context Free Grammar)을 기반으로 하여 문법규칙을 기술한다. 구문 부착에 사용되는 구문 구조의 표현은 괄호식을 사용하며, 이 구문 부착 말뭉치로부터 문법규칙과 확률이 자동으로 추출된다.

문법규칙에 사용하는 터미널 노드는 각 어절의 품사 태그열이며 그 외에 사용된 심볼은 문법 심볼(Grammatical Node)이라 부른다. 문법 심볼 중 루트 노드의 심볼은 NP(명사구, Noun Phrase)나 NPL(단말 명사구, Lower Noun Phrase)[4]로 넌터미날 노드라 한다. 사용된 문법 심볼은 표 2와 같다.

표 2 문법 심볼

NP	명사구	(NP(NPL 초승달 모양의) 눈썹)
NPL	단말 명사구	(NPL 가물치과의 민물고기)
VP	동사구	(NP(VP(NPL 한 교파를) 개창한) 사람)
ADJP	형용사구	(NPL(ADJP 행실이 (VP 바르지 못한)) 계집)
ADVP	부사구	(NPL(ADVP 학원에서) 강의하는 사람)

4) NPL(Lower Noun Phrase)은 또 다른 NP를 가지지 않는다는 점에서 NP와 구별하기 위해 사용된 문법 심볼(Grammatical node)이다. 명사구들 중에서는 단말에 해당하므로 단말명사구라고 지칭한다. 그러나, 파서 트리에서는 넌터미날 노드가 된다. 이를테면 다음과 같은 문법규칙과 같이 사용된다.
"(NP(ADVP<1>)(NPL(ADJP<2><3>)<4>))"

구문 부착 말뭉치로부터 문법규칙을 자동으로 추출하는 것은 단순하다. 문맥 자유 구구조 문법 형식을 이용하여 말뭉치에 구문 구조가 부착되어 있다면, 각 문장의 부모-자식 관계를 이루는 모든 관계를 구구조 규칙으로 추출한 후, 추출된 모든 규칙들에 대해 중복성을 제거[5] 하면 되는 것이다. 다른 문법 체계를 이용하더라도 이와 유사한 과정을 거치게 된다.

다음은 추출된 문법규칙의 하나이다.

```
NP : NPL VV+EFC VJ+EFA VV+EFD NNCG
   : struct "(NP (VP <1> <2> (ADJP <3>) <4>) <5>)" ;
```

NP와 NPL은 넌터미날이고 그 외의 심볼들은 어절 태그열로서 터미날 노드이다. 이 규칙에 의해서 NP는 NPL VV+EFC VJ+EFA VV+EFD NNCG의 문장 태그열로 대치되며, VP와 ADJP의 문법 노드를 가진 NP 트리를 생성한다.

표 3 NP와 NPL 구문 구조에 대한 통계

목록	NP	NPL
전체 문장 수	3,937	6,088
서로 구별되는#1 문장 수	286	125
50%의 문장을 커버하는 구조의 수	4	7
2번 이상의 출현 빈도수를 갖는 구조에 의해 커버되는 문장의 비율	37.92%	63.86%
상위 10 개 구문 구조에 의해 커버되는 문장의 비율	24.16% (61.54%)#2	53.90% (88.76%)

#1 서로 구별된다 함은 서로 다른 문장 태그열을 가진다는 의미이다.

#2 괄호안의 수치는 해당 구문 구조(NP, NPL)의 문장에 대한 비율이다.

표 3과 그림 4는 그림 2의 과정을 통하여 구축된 문법규칙에 대한 통계이다. 총 411개의 문법규칙 중 50개(약 12.16%)의 규칙이 총 10,025개의 문장 중 8,150(약 81.29%)개의 문장에 적용된다. 표 3와 그림 4로부터 학습 말뭉치에는 동일한 구조를 가지는 많은 문장이 있음을 알 수 있고, 또한 아주 적은 수의 구문 구조가 학습 말뭉치의 많은 문장을 커버함을 알 수 있다.

5) 문법규칙에 중복성을 제거하지 않고 완전 매칭을 이용하여 파싱할 경우, 정확률은 매우 높지만 재현률은 다소 떨어진다. 본 연구에서 비학습 문장 38,803개에 대하여 중복성을 제거하지 않은 문법규칙의 경우(완전매칭) 32.05%, 중복성을 제거한 문법규칙의 경우 38.81%의 재현률을 보였다.

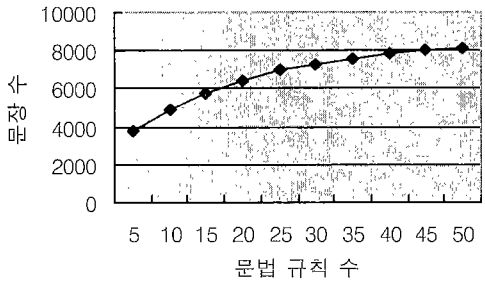

그림 4 문법규칙의 커버리지(COVERAGE)

가장 빈도 높은 NP와 NPL에 대한 구문 구조는 표 4와 같다. 표 4에서 왼쪽 열은 구문 부착 말뭉치에서의 구문 구조의 출현 빈도를 나타내며, 오른쪽 열은 해당 구문 구조와 예를 보였다.

표 4 다빈도 NP와 NPL 구문구조

출현빈도	NP 구문구조
258	(NP (NPL NNCG NNCV+PD) NNCG) 집안 조상의 기제사
250	(NP (NPL NNCG NNCG+PD (NP NNCG NNCG)) 집안의 사림 수효
199	(NP (NPL NNCG NNCG+PD) NNCG) 외교 문서의 하나
175	(NP NNCG+PD (NPL DU NNBU)) 물시계의 한 가지
149	(NP (VP (NPL NNCG+PD NNCG+PO) VV+EFD) NNCG) 염화은의 유체를 바른 인화지
출현빈도	NP 구문구조
1,558	(NPL (ADVP NNCG+PA) NNCV+XSVV+EFD NNCG) 규장각에서 심부름하던 사람
827	(NPL NNCG+PN NNP) 공자와 맹자
300	(NPL (VP NNCG+PO (VP VV+EFC VX+EFD)) NNCG) 솔가리를 긁어 모은 땔나무
220	(NPL (ADVP NNCG+PA) VJ+EFD NNCG) 가에 가까운 부분
201	(NPL (VP NNCG+PO VP(VV+EFC VJ+EFD)) NNCG) 매를 맞아 아픈 느낌

많은 문장이 적은 수의 문법규칙에 의해 커버되지만, 동일한 문장 태그열로부터 생성되는 두 개 이상의 구조가 존재할 수 있다. 즉 명사구 해석의 중의성[6]이 발생

6) [1]에서는 명사구의 유형을 관형 성분 수식 명사구, 병렬형 명사구, 보문 명사구로 나누고 있다. 각 유형들이 확장을 거듭하

하는데, 수집된 문법규칙에서 많이 발생하는 명사구 중의성의 유형과 그 예를 보이면 표 5와 같다.

표 5 명사구 해석의 중의성과 예

(유형1) NNCG+PO VV+EFC VV+EFD NNCG+PD NNCG
ㄱ. (NP (VP 수를 (VP 놓아 만든)) (NPL 젖먹이의 버선)) ㄴ. (NP (NPL (VP 오라를 (VP 차고 나선)) 포졸의) 위풍)
(유형2) NNCG+PD NNCG+PN NNCG+PO VV+EFD NNCG
ㄱ. (NP (VP (NP 창포의 (NPL 잎과 뿌리를)) 우려낸) 물) ㄴ. (NP (NPL 글씨의 체와) (NPL 붓을 놀리는 법))
(유형3) NNCG NNCG NNCG+PA VV+EFD NNCG
ㄱ. (NP (VP (ADVP (NPL 나무 줄기) 속에) 박힌) 심) ㄴ. (NP (VP (ADVP 가을철) (ADVP 남쪽 하늘에) 보이는) 별자리)
(유형4) NNCG+PN NNCG+PD NNCG+PS VV+EFD NNCG
ㄱ. (NP (VP (NP (NPL 동쪽과 북동쪽의) 중간이) 되는) 방위) ㄴ. (NP (VP (NP 머리카락이나 (NPL 몸의 털이)) 빠지는) 증세)
(유형5) NNCG+PN VJ+EFD NNCG+PD NNCG
ㄱ. (NP (ADJP 하프와 비슷한) (NPL 동양의 현악기)) ㄴ. (NP (NP 산소와 (NPL 다른 원소의)) 화합물)
(유형6) NNCG+PD NNCG+PN NNCG+PD NNCG
ㄱ. (NP (NPL 관청의 위력이나) (NPL 관직의 권위)) ㄴ. (NP (NP 옷의 (NP (NPL 소매와 깃의) 사이))

본 논문의 DPAS에서는 표 5와 같이 중의성을 가지는 품사 시퀀스에 대한 구문 구조를 선택하기 위해 부분 트리에 대한 발생 확률을 이용한다(4.2절 참조).

이러한 통계 자료로부터 소수의 NP와 NPL 구조가 많은 수의 문장을 커버할 수 있음을 알 수 있다. 각 문장의 부분 트리는 트리 사전을 look-up함으로써 결정되지만 동일한 품사 시퀀스를 가지는 부분 트리에서 중의성이 발생하기 때문에, 전반적인 파싱 과정은 단순한 table look-up이라기 보다는 좀 더 파싱에 가깝다.

4. 파싱 알고리즘

DPAS에서는 상향식의 확률적 차트파싱 방법을 사용한다. 여기서 확률적이라 함은 사전확률과 문법확률을 이용함을 말한다. 사전확률은 하나의 어절이 가지는 분석 결과들의 상대적인 빈도수로서 어절 태그 사전에 저장되어 있고, 문법확률은 구문 부착 말뭉치에서 추출된 문법규칙의 상대적 출현 빈도수로서 트리 사전에 저장되어 있다. 파서는 이 두 개의 확률을 이용하여 각 노드의 스코어를 계산하며, 스코어가 가장 높은 노드를 먼저 선택하여(Best-First 탐색) 부분 트리를 구성하고, 이 부분 트리들 중에서 스코어가 가장 높은 부분 트리만을 대상으로(Viterbi 탐색) 유일한 최적의 파스 트리를 찾아낸다.

Zipf의 법칙[14]에 따르면, 30,000개의 문장에 대해 90%의 커버리지를 얻기 위해 필요한 문법규칙의 수는 대략 22,175개이며, 80%의 커버리지를 위해서는 대략 19,175개가 필요하다[7]. 일반적인 언어 영역에 대해 80%의 커버리지를 얻기 위해서는 대략 10^{70}~10^{140}개의 문장을 학습해야 한다. 이렇게 수만 개에 이르는 규칙을 가지는 문법을 다루는 것은 쉽지 않다. 일반적인 LR 파서는 거대한 테이블을 필요로 하며, 간단한 차트파서의 경우 가장 많이 출현하는 문법규칙에 대해서 많은 수의 Active 노드를 필요로 한다. 본 논문에서는 이러한 문제점을 극복하기 위하여 문법 Factoring, Best-First 탐색, Viterbi 탐색 알고리즘을 사용한다.

4.1 상향식 차트파싱

차트파싱은 다이나믹 프로그래밍 기법을 사용하여 수행되어 온 분석 과정을 차트라는 구조물에 저장(Bookkeeping)하여 동일한 작업의 반복적인 수행을 방지함으로써 보다 효율적으로 파싱을 수행할 수 있는 알고리즘이다. 이 알고리즘은 새로운 분석을 수행하기 전에 차트를 살펴보고 이미 동일한 작업이 수행되었는지를 검색한다. 만일 이미 동일한 작업이 수행되었다면 이미 수행된 과정과 결과를 이용한다. 따라서 동일한 분석을 반복하게 하는 백트래킹 기법의 비효율성이 제거된다.

차트파싱의 효율은 차트 내의 초과된 구조의 양에 반비례하고 사용하는 파싱 연산자에 대한 자료 구조의 적합성에 비례한다[12]. 파싱 연산자에 대한 자료 구조의 적합성이란 연산을 하기에 자료 구조가 어느 정도 편리하게 되어 있는가를 말한다. 차트파서가 사용되는 자료 구조의 영향을 받는다는 것은 파싱의 대분분의 시간이 차트와 문법으로부터 정보를 탐색하는데 사용되기 때문이다. 파서는 어떤 형태의 노드가 차트에 첨가될 때마

면서 현대국어 명사구의 가능한 구조에 대해 논의하고 있으며 동시에 명사구 해석의 중의성의 유형을 정리하였다.

7) 본 논문에서 사용된 대상 문장 77,791개에 대해 90%와 80%의 커버리지를 얻기 위해 필요한 문법규칙의 수는 각각 24,849개와 17,151개였는데, 일반 문장의 그것과 비교하면 상당히 적은 개수이다. 이 수치는 명사 뜻풀이말이 한정된 구문 구조를 가지며, 한정된 규칙 또는 언어 연산에 대해 어느 정도 닫힌 집합(closed-set)임을 나타낸다.

다, 전체 차트를 탐색하여 그 노드와 결합되는 다른 노드를 찾거나 전체 문법을 탐색하여 사용되는 규칙을 차트에 기록해야 한다. 그러므로 차트 내의 노드의 수가 많아질수록, 문법의 크기가 증가할수록 성능은 떨어진다. 그 해결책으로 노드와 문법은 인덱싱하여 탐색을 빠르게 하는 방법이 있다[12]. 이 방법에 의한 대표적인 파서의 예가 도미타 파서이다[13]. 이것은 문법에서 규칙들의 lookahead를 이용하여 파싱 테이블을 미리 작성하여 어떤 입력에 대해 적용할 규칙을 인덱싱한 것이라 볼 수 있다.

본 연구에서 사용되는 차트는 이차원 배열[X,Y]로서 X 좌표는 노드의 문장 위치를 나타내고, Y 좌표는 노드가 표현하는 현재까지 완성된 부분 구조의 문장 범위(Span)를 나타낸다. 알고리즘에서 사용하는 노드의 구조를 간략하게 아래와 같이 표기하기로 한다.

[X, Y, CAT]

위에서 X, Y는 노드가 기록되어 있는 차트의 X좌표와 Y좌표이며, CAT은 터미날 심볼 또는 문법 심볼을 나타낸다. 즉 [X, Y, CAT]은 문장 위치 X에서 시작하여 Y개의 어절에 이르는 범위를 커버하며, 그 문법 심볼은 CAT인 노드를 의미한다. 예를 들면, 예문 "아직 피지 아니한 어린 꽃봉오리" 를 파싱할 때 생성되는 노드를 표현하면 그림 5과 같다.

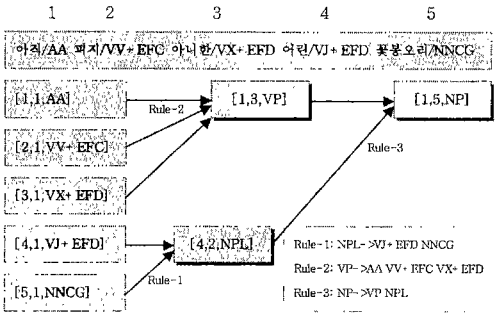

그림 5 "아직 피지 아니한 어린 꽃봉오리" 파스 트리에 대한 노드 기호의 표현 예

그림 5에서는 최적의 파스 트리의 구성에 직접 참여하는 노드들만 나타내었다. 문법규칙 Rule-1, Rule-2, Rule-3에 의해 단말 노드에서 중간 노드로, 그리고 최상위 루트 노드 [1,5,NP]로 축약(Reduction)이 일어난다. 각 어절이 가지는 분석 결과는 1개 이상이며 단말 노드 또한 각 어절의 총 개수 이상이 존재하게 된다. 그러나 파서가 유일하게 선택하는 것은 스코어가 가장 높은 단말 노드이며, 최상위 노드 또한 스코어가 가장 높은 중간 노드들로부터 생성된다.

4.2 태깅과 스코어 계산

태깅은 문장 파싱의 첫번째 단계로서, 학습 말뭉치 내의 동일한 어절에 할당된 어절 태그열에 기반하여 하나 이상의 어절 태그열을 입력 문장의 각 어절에 할당한다. 각 어절의 태그열은 스코어 계산에 사용되는 빈도값을 가지고 있고 아래 식(1)에 의해 어절 태그 사전확률(Word-Tag Dictionary Probability, $P_{tag}(t|w)$)이 계산된다. 특정 어절 w가 태그열 t를 가질 확률값($P_{tag}(t|w)$)은 어절 w의 총 출현 빈도수와 어절 w에 할당된 태그열 t의 상대적인 빈도수로 계산된다.

$$P_{tag}(t|w) = \frac{Frequency\ of\ word\ w\ with\ tag\ t}{Frequency\ of\ word\ w} \quad (1)$$

문법규칙의 확률공식과 파스 트리의 스코어는 식(2), 식(3)과 같이 계산된다. 규칙 X → Y의 확률($P_{rule}(X \rightarrow Y)$)은 트리 태깅 말뭉치에서 X가 Y를 유도하는 빈도와 넌터미날 X의 빈도값에 의해 계산된다. 파스 트리(T)의 스코어($S_{tree}(T)$)[8]는 트리를 구성하기 위해 사용된 규칙들의 확률값과 각 어절들의 태그 확률값의 제곱으로 계산된다. 어절의 태그 확률값에 제곱을 한 이유는 문법규칙의 확률값보다 어절 태그열의 확률값에 가중치를 좀 더 두기 위한 것으로, 동등한 가중치를 할당한 경우보다 더 좋은 결과[9]를 나타내었다. 최종 결과물인 파스 트리는 입력으로부터 유도되는 가능한 모든 파스 트리 중에서 발생 확률이 가장 높은 것이다.

$$P_{rule}(X \rightarrow Y) = \frac{Frequency\ with\ which\ X\ is\ expanded\ as\ Y}{Frequency\ of\ rules\ where\ LHS\ is\ X} \quad (2)$$

$$S_{tree}(T) = \prod_{R\ rules\ in\ T} P_{rule}(R) \times \prod_{t\ tags\ in\ T} (P_{tag}(t|w))^2 \quad (3)$$

각각의 규칙에 부여되는 확률은 문서로부터 자동으로 추출될 수 있는데, 학습 문서가 대상 언어를 대표한다고 가정하면 그 문서에서 규칙이 사용된 상대적 빈도값을 그 규칙의 확률로 볼 수 있다.

4.3 문법 Factoring

DPAS에서 사용하는 첫번째 기법은 문법 Factoring 이다. 문법규칙은 공통 접두사로 Factoring 되며 유한

8) 본 연구에서는 $S_{tree}(T)$에 log 연산을 취하여 log 확률값을 사용한다. 즉 log(A*B) = log(A) + log(B)의 성질을 이용하여 계산의 복잡도를 줄이고, 계산 속도를 빠르게 할 수 있다.

9) 어휘에 더 높은 가중치를 두어 더 나은 결과를 나타낸 것은, 뜻풀이말은 제한된 문장 패턴으로 기술되며 개별 어휘 또한 같은 위치에서 반복적으로 쓰이는 경우가 많기 때문이다.

상태 오토마톤[10]으로 표현될 수 있다. 그림 7과 같이 각 Arc는 문법규칙의 Right-Hand-Side(RHS)의 문법 심볼로 라벨링되어 있다. Left-Hand-Side(LHS) 문법 심볼, 스코어, 트리 구조와 같은 문법규칙 정보는 그 문법이 완성(complete)되는 노드에 저장된다. 동일한 심볼 시퀀스를 갖는 하나 이상의 규칙은 하나의 리스트 자료 구조로 표현된다.

NP → ABC : 30
NP → ABCD : 25
NP → ABF : 20
NPL → ABC : 40
NPL → AE : 25

그림 6 문법규칙의 예

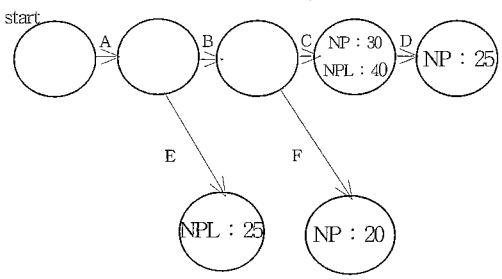

그림 7 문법규칙(그림 6)에 대한 오토마톤

예를 들어, 그림 7의 오토마톤은 그림 6의 문법규칙에 의해 생성된다. 각 규칙의 끝에 있는 숫자는 그 규칙의 스코어를 나타낸다. 그림 7에서 각 노드 내의 정보는 LHS 심볼과 각 규칙의 스코어이다. 예를 들어, 규칙 NP → A B C D는 그림 7의 오른쪽 끝 노드에 저장되어 있으며 A B C D 순의 전이로 도달할 수 있다. 심볼 A B C 시퀀스에 대해 두 개의 규칙 NP, NPL이 존재한다.

일반적으로 이러한 문법은 동일한 터미날로부터 시작하는 수천개의 규칙을 가지기 때문에, 동일한 수천개의 Active 에지(Edge)를 가져야 한다. 예를 들어 관형사의 경우 기존의 차트파서는 입력 시퀀스에서 관형사를 발견할 때 동일한 수의 Active 에지를 유지해야 한다. 명사 구문과 같은 짧은 문장들은 유사한 품사 시퀀스가 많으며 따라서 많은 수의 Active 에지를 저장할 공간이 필요하다. 그러나 Active 에지는 다음 순간에 확장될 수 있는 문법규칙을 나타내기 때문에, 문법 오토마톤 내의 대응되는 노드에 대한 하나의 포인터로써 수천개의 Active 에지를 대신할 수 있다. 오토마톤 내의 하나의 노드는 모든 가능한 다음 노드에 대한 Arc를 가지고 있기 때문에, 기존의 차트파서에서 다수의 Active 에지를 가지는 것과 동등하다.

문법 스코어는 Best-First 탐색 알고리즘을 적용하기 위하여 유한상태 오토마톤의 각 노드에 저장된다.

그림 8은 그림 6의 규칙들에 대한 수정된 유한상태 오토마톤을 나타낸다. 이 그림에서 노드의 윗 부분의 숫자는 문법 스코어 계산을 위한 부분 스코어이다. 아랫부분에 문법 정보를 저장하고 있는 노드는 문법규칙이 그 노드에서 완성되는 경우이다. 이 부분 스코어는 그 노드를 따르는 규칙에 대한 최소 가능 스코어(Minimum Possible Score)이다. 또한 규칙의 LHS 심볼은 그 노드에서 완성되는 문법규칙에 대한 부가의 스코어를 가지고 있다. 하나의 규칙 패스에 이르는 부분 스코어의 합이 그 규칙에 대한 스코어이다. 예를 들어, 규칙 NP → A B C 에 대한 스코어는 20 + 0 + 0 + 5 + 5 = 30 과 같이 계산된다. 각 노드의 부분 스코어는 Best-First 탐색에서 사용된다.

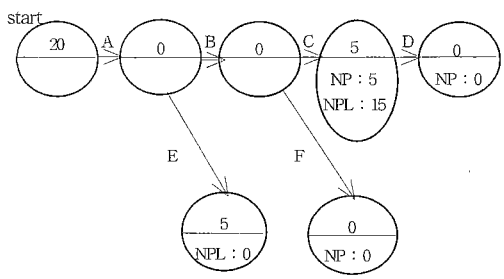

그림 8 문법규칙에 대한 수정된 오토마톤

4.4 Best-First 탐색

DPAS에서 사용하는 두번째 기법은 Best-First 탐색이다. 파서는 힙을 가지고 있는데, 이곳에 확장될 Active 노드들이 저장된다. 이 Active 노드들은 그들의 스코어에 따라 순서대로 힙에 저장된다. 따라서 힙의 Top은 베스트 스코어를 가진 Active 노드이며 이 노드가 다음에 확장될 노드이다. Top-NP 또는 Top-NPL[11]이 전체 문장에 대해서 생성되면 파싱은 더 높은 스코어를 가진 트리를 만들 가능성이 있는 Active 노드가

10) DPAS에서는 트라이(Trie) 구조로 구현하였다.

11) Top-NP, Top-NPL : 루트 노드에 해당하는 NP, NPL을 나타낸다.

있는지를 검사한 후 종료한다. 이러한 Best- First 탐색 기법은 일반적인 Left-to-Right 방식의 차트파서에 비교하면 많은 수의 Active 노드를 감소시킨다. 4.3절에서 소개된 문법 오토마톤 내의 부분 스코어는 각 Active 노드 스코어에서 가장 높은 추정해를 선택하는 데 사용된다.

4.5 Viterbi 탐색

DPAS에서 사용하는 세번째 기법은 Viterbi 탐색이다. 코스트 최소법은 해의 노드와 링크에 적당한 코스트를 두고, 코스트 최소의 패스를 우선해로 하여 선택하는 방법이다[5]. 격자해 중에서 코스트 최소해를 구하는 데는, 부분적 최적해를 유지해 간다고 하는 동적 프로그래밍의 일종으로 Viterbi 알고리즘이 있다.

DPAS에서는 입력 문장에 대해 하나의 최적의 파스 트리를 찾기 때문에 각 구문에 대해서 단지 하나의 완성된 품사 시퀀스를 필요로 한다. 그림 8의 문법 오토마톤과 같이 노드와 노드에 적당한 스코어를 계산해 놓고, 스코어 최대의 패스를 우선해로 하여 선택하는 방법을 사용한다. 다시 말해, 각 구문에 대해 가장 확률이 높은 품사 시퀀스 후보만이 유지되고 동일한 품사 시퀀스의 다른 후보들은 제거되는데, 이는 제거되는 후보들이 결코 최적해의 일부가 되지 못하기 때문이다. Viterbi 탐색 알고리즘은 표 6과 같다.

표 6 Viterbi 탐색 알고리듬

문법규칙의 임의의 위치에서 시작하여 끝나는 노드를 NE_1, NE_2, NE_3, … ,
시작하는 노드를 NS_1, NS_2, NS_3, … , 라 하자.
각 NS_i에 대해 다음으로 확장 가능한 NE_j를 구하고 NS_i와의 사이에 링크를 한다. 이때

- NE_j까지의 부분 최대 스코어,
- NS_i, NE_j간의 최소 가능 스코어,
- NS_i의 문법 스코어,

의 합이 최대인 NE_j를 구하고, 그 최대값을 NS_i까지의 부분 최대 스코어라하고, NE_j와 NS_i와의 링크에 특별 마크를 붙인다.

본 시스템과 같이 유일한 최적해를 찾는 경우, Viterbi 탐색은 매우 효율적인 알고리즘이다. 왜냐하면 파서는 보다 적은 수의 InActive 노드를 생성하기 때문에 결과적으로 보다 적은 수의 Active 노드가 생성되기 때문이다.

4.6 알고리즘

DPAS에서 힙(heap)은 Active 노드를 저장하는 데 사용하며, 한 노드의 스코어가 자식들의 스코어 보다 항상 작도록 이진 트리를 이용하여 구현하였다(Min Heap). 힙에 새로운 노드를 삭제 및 삽입하는 연산의 시간 복잡도는 $O(\log 2n)$(n: 노드수)이다(표 8 참조).

다음은 힙과 관련된 4가지의 연산이다.

- Extract_heap : 힙에서 베스트 스코어의 ANode를 얻음
- Store_heap(p) : 힙에 Anode p를 저장
- Not_empty_heap() : 힙이 비어 있지 않다면 리턴 1
- Best_score_in_heap() : 힙에서 베스트 스코어의 ANode를 리턴

알고리즘에서 사용하는 자료구조는 ANode와 Node의 두가지이며, ANode는 일반적인 차트파서의 Active 에지에 해당하고 Node는 InActive 에지에 해당한다. ANode와 Node는 단어 스코어와 문법 스코어로부터 계산된 자신의 스코어를 가지고 있다. ANode는 이미 지나온 문법규칙에 대응되는 문법 오토마톤에 대한 포인터를 저장한다. Node는 자신의 문법 심볼과 자식 노드들에 대한 포인터를 가지고 있다. 이미 발견된 전체 문장에 대한 베스트 스코어를 가지는 변수($best_score)가 있으며, ANode 구조에 대한 $p, $q 변수와 Node 구조에 대한 $n, $m 변수가 있다. 함수 score()는 아규먼트에 대한 스코어를 리턴한다.

그림 9는 차트 [X, Y]에 저장된 ANode와 Node의 구조를 나타낸 것이다. G_chart[X][Y]에는 문장의 X위치에서 Y개의 범위에 이르는 Node가 기록되어 있으며, Node는 문법규칙에 대한 정보(rinfo)와 자신을 구성하고 ANode의 리스트(sub_node) 그리고 다음 번에 확장될 ANode의 리스트(act_node)를 유지하고 있다.

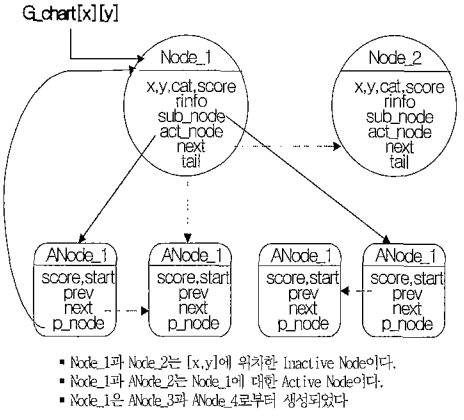

그림 9 차트에 저장된 노드와 활성호 노드의 구조

표 8 DPAS 알고리듬

```
int parse()
{
  $best_score = VERY_BIG;

  for(all word in the sentence)
    store_heap(ANode for each dictionary entry);

  while(not_empty_heap() and best_score_in_heap()
  < $best_score)
  {
        $p = extract_heap();
        if(there is a complete grammar rule at $p
        and
        score($p) + grammar score
             < score(Node at the same span and with
               the same  category))
        {
        create new Node $n for $p;

        if($n is a 'NP' or 'NPL' covering the entire
           sentence and score($n) < $best_score)
         $best_score = the score;
        else
         store_heap(ANode for $n);

        for(ANode $q which is finishing at the
              previous word to the span of $n
           and there is a rule which follows $n)
           store_heap(ANode combining $q and $n);
         }

        for(Node $m to which there is a grammar
        rule from $p)
        store_heap(ANode combining $p and $m);
  }
}
```

5. 실험 및 평가

이 장에서는 확률적 문법규칙의 유용성을 네 가지 실험을 통해 살펴본다. 실험1은 문법규칙의 정확률을 측정하기 위하여 학습 문장을 대상으로 실험하였다. 실험2은 사전확률과 문법확률을 이용한 실험이고, 실험3는 사전확률만을 이용하여 왼쪽 우선 파싱(Left-to-Right Parsing) 방법을 적용한 실험이다. 실험4는 실험2와 같은 방식으로 하되, 부분적으로 왼쪽 우선(Left-First) 탐색을 적용한 실험이다.

(실험1) 학습문장을 대상으로 확률적 문법규칙의 정확률 실험
(실험2) 비학습문장을 대상으로 확률적 문법규칙의 정확률 실험
(실험3) 왼쪽 우선 파싱 방법을 적용한 실험
(실험4) 실험3에 왼쪽 우선 탐색을 적용한 실험

각 실험에 사용된 문법규칙과 대상 문장 그리고 결과는 각각 표 9, 표 10, 표 11, 표 12와 같다.

5.1 학습문장을 대상으로 문법규칙의 정확률 실험

실험1을 위해서 총 77,791개의 뜻풀이말을 품사 태깅하여 말뭉치를 구성하고, 동일한 품사 태그열을 가지는 문장들끼리 분류하였다. 이 중에서 가장 많은 수의 문장을 가지는 상위 219개의 분류군(총 문장수: 9,776개)에서 문장을 하나씩 임의로 추출하여 219개의 대표 문장군을 구성하였다. 이 대표 문장군을 수동으로 구문 태깅하고 나머지 문장들에 자동으로 구문 구조를 부착한 후 문법규칙을 구축하였다. 상위 219개의 분류군에서 각각 하나씩 205(2.09%)[12]개의 실험 대상 문장을 선정하였다.

실험1의 결과 문법규칙 정확률은 90.19%로서 이는 학습 문장의 2.09%가 나머지 97.91%의 문장을 대표할 만하며, 이러한 다빈도 구문 구조를 가지는 문장들을 기반으로 구축된 확률적 문법규칙은 그 언어 영역 전체를 대표할 수 있음을 추정할 수 있다.

표 9 실험1의 파싱 결과

실험에 사용된 트리 사전		
문법규칙 수	219	
학습 문장 수	9,776	
총 어절 수	34,727	
평균 문장 길이	3.55	
문장 길이 2 인 문장 수	1,532	15.67%
문장 길이 3 인 문장 수	3,606	36.88%
문장 길이 4 인 문장 수	2,831	28.95%
문장 길이 5 인 문장 수	1,336	13.66%
문장 길이 6 인 문장 수	456	4.66%
문장 길이 7 인 문장 수	15	0.15%

실험에 사용된 문장의 분포		
총 문장 수	205	
총 어절 수	1,047	
평균 문장 길이	5.1	
문장 길이 2 인 문장 수		
문장 길이 3 인 문장 수	8	3.90%
문장 길이 4 인 문장 수	29	14.14%
문장 길이 5 인 문장 수	110	53.65%
문장 길이 6 인 문장 수	49	23.90%
문장 길이 7 인 문장 수	9	4.39%

12) 문장 길이 2인 문장들은 제외하였다.

표 10 실험1의 결과

실험 결과	
문법규칙의 정확률	90.19%
Node 수(InActive Edge)	1,944
Anode 수(Active Edge)	2,024

5.2 비학습문장을 대상으로 문법규칙의 정확률 실험

실험2에서는 사전확률과 문법확률을 사용하여 정확률을 실험하였다. 실험 결과, 문법규칙의 재현률은 51.74%, 정확률은 75.52%로 나타났다(표 12). 실험2의 분석 오류들 중 다수는 명사구 해석의 중의성 해소시 확률만을 이용함으로써 발생하는 것들이다.

실험3와 실험4는 실험2과 동일한 어절 태그 사전과 트리 사전을 이용하되, 정확률을 향상시키기 위해 분석 방법을 약간씩 바꾸어 실험을 수행하였다.

5.3 사전확률과 왼쪽 우선 파싱 방법을 적용한 실험

실험3는 문법확률은 이용하지 않고 사전확률과 왼쪽 우선 파싱(Left-to-Right Parsing) 방법을 이용하여 수행하였다. 이 실험은 뜻풀이말은 중심어가 문장의 맨끝에 위치하므로 중심어 후위의 원칙에 따라 수식 성분을 먼저 탐색하여 부분 트리를 찾는 것이 유용할 것이라는 가정을 기반으로 한다. 이는 속격조사 '의' 나 접속조사에 의해 명사구의 중의성이 발생할 때 바로 뒤 명사구가 수식을 받는 것으로 해석하는 경향이 강하다는 일반성을 또한 고려하기 위함이다.

실험3의 결과, 문법규칙의 정확률은 84.77%를 나타내었는데 실험2보다 9.25%의 향상을 보였다(표 11). 실험3의 분석 오류 중 다수는 명사구 해석의 중의성이 발생할 때 문법확률을 고려하지 않았기 때문에 생기는 것들이다. 실험2의 분석 오류들 중 30.4%가 실험3에서는 정확하게 분석되었는데, 그들의 대부분은 속격조사 '의' 나 접속조사에 의해 명사구의 중의성이 발생하는 것이었다. 즉 단순히 확률에 의존하는 분석 방법(실험2)보다 뜻풀이말의 특성을 고려한 왼쪽 우선 파싱 방법이 보다 유용함을 알 수 있다.

5.4 실험2에 왼쪽 우선 탐색을 적용한 실험

실험4는 실험2와 같은 방식으로 하되, 단말 노드 또는 부분 트리를 선택함에 있어서 Best-First 탐색이 아니라 왼쪽 우선(Left-First) 탐색을 적용하였다. 전체 파싱 방법은 사전확률과 문법확률이 최대가 되는 최적해를 찾는 방식이다.

실험4의 결과, 문법규칙의 정확률은 87.47%로 실험2에 비해 11.95%, 실험3에 비해 2.7% 향상되었다(표 12). 실험2의 분석 오류 중 30.4%, 실험3의 분석 오류 중 39.2%가 실험4에서는 정확하게 분석되었다. 실험 결과로부터, 중심어 후위의 원칙에 따라 왼쪽 우선 탐색 방식(실험3)과 명사구 해석의 중의성을 해소하기 위해 문법확률을 사용하는 방식(실험2)을 혼용하는 것이 가장 적합함을 알 수 있다.

표 11 실험2, 실험3, 실험4의 파싱 결과

실험에 사용된 트리 사전		
문법규칙 수	559	
학습 문장 수	10,175	
총 어절 수	35,831	
평균 문장 길이	3.52	
문장 길이 2 인 문장 수	1,577	15.50%
문장 길이 3 인 문장 수	4,040	39.71%
문장 길이 4 인 문장 수	2,881	28.31%
문장 길이 5 인 문장 수	1,354	13.31%
문장 길이 6 인 문장 수	99	0.97%
문장 길이 7 인 문장 수	171	0.68%
문장 길이 8 인 문장 수	21	0.21%
문장 길이 9 인 문장 수	16	0.16%
문장 길이 10인 문장 수	16	0.16%

실험에 사용된 문장의 분포		
총 문장 수	1,003	
총 어절 수	4,986	
평균 문장 길이	4.97	
문장 길이 2 인 문장 수		
문장 길이 3 인 문장 수	253	25.22%
문장 길이 4 인 문장 수	213	21.24%
문장 길이 5 인 문장 수	194	19.34%
문장 길이 6 인 문장 수	145	14.46%
문장 길이 7 인 문장 수	93	9.27%
문장 길이 8 인 문장 수	61	6.08%
문장 길이 9 인 문장 수	44	4.39%
문장 길이 10인 문장 수		

표 12에서 보는 바와 같이, 정확률은 실험4가 가장 높으며, 생성된 Node와 ANode의 수 또한 실험4가 가장 적다. 실험3의 경우, 문장 위치가 왼쪽에 있는 어절이나 부분 트리들을 차례로 선택하여 노드를 구성해 나

가기 때문에 실험3에 비해 더 많은 수의 Node와 ANode가 생성되며 실행 시간도 많이 걸린다.

표 12 실험2, 실험3, 실험4의 결과 비교

	실험2	실험3	실험4
재현률	51.74%	51.74%	51.74%
정확률	75.52%	84.77%	**87.47%**
Node 수(InActive Edge)	5,750	5,676	**5,516**
ANode 수(Active Edge)	9,329	8,388	**8,145**

6. 결론 및 향후 연구 과제

본 논문에서는 국어사전의 뜻풀이말에서 다양한 의미정보를 추출하기 위한 기본 도구로서 뜻풀이말 구문분석기를 구현하였다. 이를 위하여 사전 뜻풀이말을 대상으로 일정한 수준의 품사 및 구문 부착 말뭉치를 구축하고, 이 말뭉치들로부터 품사 태그 중의성 어절의 빈도정보(어절 태그 사전)와 통계적 방법에 기반한 문법규칙과 확률정보(트리 사전)를 자동으로 추출하였다. 뜻풀이말 구문분석기는 이들 사전을 이용하는 확률적 차트파서이다.

뜻풀이말 구문분석기는 품사 태그 중의성 어절의 빈도 정보와 문법규칙 및 확률정보를 이용하여 파싱 과정에서의 명사구 중의성을 해소한다. 또한, 파싱 과정에서 생성되는 노드의 수를 줄이고 수행 속도를 높이기 위한 방법으로 문법 Factoring, Best-First 탐색 그리고 Viterbi 탐색의 방법을 이용한다.

문법규칙의 확률과 왼쪽 우선 파싱 그리고 왼쪽 우선 탐색 방법을 사용하여 실험한 결과, 왼쪽 우선 탐색 방법과 문법규칙의 확률을 동시에 사용하는 방식이 가장 정확한 결과를 보였으며, 비학습 문장에 대해 51.74%의 재현률과 87.47%의 정확률을 보였다. 실험 결과로 어절 태그 사전과 트리 사전을 이용한 차트파서가 제한된 문장형식을 가진 사전의 뜻풀이말을 구문분석하는 데 유용하게 사용될 수 있음을 알 수 있다.

앞으로의 연구과제는 다음과 같다.

첫째, 현재의 구문분석 정확율(87.47%)을 향상시키기 위한 더 많은 구문 부착 말뭉치의 구축과 개선된 문법규칙 확률공식과 스코어 계산에 대한 연구가 필요하다.

둘째, 뜻풀이말 구문분석기의 개발 목적에 맞게 이를 이용하여 사전에서 다양한 의미속성을 추출하는 연구가 뒤따라야 할 것이다.

셋째, 사전은 그 용도에 따라 뜻풀이말의 구성 형식과 어절 수가 다르다. 따라서, 다른 사전에 이 파서를 적용하기 위해서는 15어절 가량의 문장을 처리할 수 있도록 개선되어야 한다. 이를 위한 문법규칙의 확장이 필요하고, 문법규칙 수의 증가에 따른 여러 문제점에 대한 연구가 진행되어야 한다.

참 고 문 헌

[1] 김철호, "병렬 명사구의 구문해석", 한국과학기술원 전산학과, 박사학위논문

[2] 김재한, "한국어 어휘 중의성 해소를 위한 태깅 시스템", 울산대학교 석사학위논문, 1994

[3] 이상국,김윤호,김재문,이상조, "용언의 하위범주화 정보를 이용한 특수문형의 처리방안", 정보과학회 추계학술발표논문집, 1993

[4] 서영훈, "의미정보를 이용하는 중심어 주도의 한국어 파싱", 서울대 컴퓨터공학과 박사학위논문, 1991

[5] Makoto Nago 저, "자연언어처리", 홍릉과학출판사 1996

[6] 이상주, "자동 품사 부착을 위한 새로운 통계적 모형", 고려대 컴퓨터학과 박사학위논문, 1999

[7] 장석진, "한국어 문법-NLP를 위한 HPSG/K", 한국과학기술원, 인공지능연구센터 기술보고서, CAIR-TR-92-33, 1992

[8] 윤덕호, "한국어의 문법적 특성과 LFG 분석기법", 정보과학회 인공지능연구회 소식지 11호, 1988

[9] 임희석,김진동,임해창, "어절 태그 변형 규칙을 이용한 한국어 품사 태거", 정보과학회논문지(B), 제26권 제6호, 1999

[10] 조평옥,옥철영, "의미속성에 기반한 한국어 명사 의미체계", 정보과학회논문지(B), 제26권 제4호, p.584-594, 1999

[11] 조평옥,안미정,옥철영,이수동 "사전 뜻풀이말에서 구축한 한국어 명사 의미계층구조", 한국인지과학회 논문지 제10권 제4호, p.1-10, 1999

[12] Gerald Gazdar, Chris Mellish, "Natural Language Processing in LISP : An Introduction to Computational Linguistics," Addison Wesley, 1989

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

[14] G. Zipf, The psycho-biology of language: An introduction to dynamic philogy , MIT Press, 1965

[15] Halliday, M. A. K.(1991), Corpus studies and probablistic grammar, in Aijmer , K. & Altenberg, B. (ed.)(1991)

[16] D.Magerman and M.Marcus, Pearl: A Probabilistic Chart Parser , In the Proceedings of European ACL, Berlin, 1991.

[17] D.Magerman and C.Weir, Efficiency, robustness and Accuracy in Picky Chart Parsing , In the Proceedings of European ACL, Newark, Delaware,

1992.
[18] E.Brill, M.Marcus, Automatically acquiring phrase structure using distributional analysis , In Darpa Workshop on Speech and Natural Language, Harriman, N.Y., 1992.
[19] R.Bob, Using an annotated corpus as a stochastic grmmar , In the Proceedings of European ACL, Utrecht, 1993.
[20] Satoshi Sekine, Corpus-based Parsing and Sublanguage Studies , New York University, 1998
[21] P.F.Brown, "Word Sense Disambiguation using Statistical Methods," Proc. of 29th Meeting of the ACL, 1991
[22] Gale, William, K.Church and D.Yarowsky, "A Method for Disambiguating Word Senses in a Large Copus," Computers and Humanities, 1992
[23] D.Yarowsky, "Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora, " Proc. of the 15th Int'l Conf. on Computaional Linguistics, 1992
[24] M.Sanderson, "Word Sense Disambiguation and Information Retrieval," Proc. of SIGIR, 1994
[25] E.M.Voorhees, "Using WordNet to disam biguate word sense for text retrival," Proc. of ACMSIGIR Conference, 1993
[26] 윤평현, 국어 명사의 의미관계에 대한 연구, 한국과학재단 연구결과보고서, 94-0100-11-01-1, 1995
[27] W. Stolz, A probabilistic Procedure for group ing words into phrases , Language and Speech, 8, 1965.
[28] R.Haigh, G..Sampson, E.Atwell, Project APRIL a progress report , In the Proceedings of the Annual Meeting of the Association for Computational Linguistics, Buffalo, N.Y., 1988.
[29] R.Bob, Using an annotated corpus as a stochastic grmmar , In the Proceedings of European ACL, Utrecht, 1993.

이 수 광
1998년 울산대학교 컴퓨터정보통신공학부 학사. 2000년 울산대학교 컴퓨터정보통신공학부 석사. 2000년 ~ 현재 (주) 퓨처시스템 정보통신연구소 연구원. 관심분야는 한국어정보처리, 정보검색, 정보보안

옥 철 영
1982년 서울대학교 컴퓨터공학과 학사. 1984년 서울대학교 컴퓨터공학과 석사. 1993년 서울대학교 컴퓨터공학과 박사. 1984년 ~ 현재 울산대학교 컴퓨터정보통신공학부 교수. 1994년 러시아 TOMSK 공과대학 교환교수. 1996년 영국 GLASGOW 대학교 객원교수. 관심분야는 한국어정보처리(의미중의성, 시소러스), 기계학습, 문서분류