확률적 차트 파싱에 기반 한 한국어 의존 구조 분석기*

은지현 정민우 이근배
포항공과대학교 컴퓨터공학과
{tigger,stardust,gblee}@postech.ac.kr

Korean Dependency Structure Analyzer based on Probabilistic Chart Parsing

Jihyun Eun, Minwoo Jeong Gary Geunbae Lee
Dept. of Computer Science and Engineering, POSTECH

요 약

정형적인 프로그래밍 언어에서는 언어를 기계적으로 해석하기 위해 입력의 구조적인 형태를 구축하는 파싱이 필수적인 과정으로 여겨진다. 기계에 기반 해서 개발된 프로그래밍 언어와 달리, 인간의 자유로운 의사소통을 위해 형성된 자연어는 특유의 다양성으로 인해 어휘, 구문, 의미 분석이 매우 어렵다. 반대로 자연어 구조 분석이 성공적으로 이루어지면 응용 시스템의 성능 향상에 상당한 기여를 할 것이라고 여겨지고, 이로 인해 끊임없이 자연어 처리, 특히 구문 분석에 많은 연구가 이루어지고 있다. 본 논문에서는 파싱에 사용되는 문법 전체를 말뭉치로부터 자동 구축하여 영역별 이식성 및 문법의 효율성을 도모했다. 또한 확률적 차트 파싱 기법과 immediate-head 파싱 모델을 적용하여 기존 파싱 시스템의 성능 향상을 시도했다. 세종 말뭉치를 이용한 파서의 성능은 각각 LP/LR 78.98%/79.55%로 나타났다.

1. 서론

정형적인 프로그래밍 언어 (programming language)가 컴파일러 (compiler)에 의해 처리되기 위해서 일련의 분석 과정-어휘 분석 (lexical analysis), 구문 분석 (syntax analysis), 의미 분석 (semantic analysis)을 거치게 된다. 이 분석 과정들은 정상적인 언어 처리를 위한 컴파일러의 필수적인 부분이다[1]. 기계에 기반 해서 개발된 프로그래밍 언어와 달리, 인간의 자유로운 의사소통을 위해 형성된 자연어 (natural language)를 기계를 통해 처리하고자 하는 시도가 이어졌고, 이를 위해 자연어를 대상으로 어휘, 의미, 구문 분석에 대한 연구가 본격적으로 시작됐다. 그러나 정형적인 프로그래밍 언어와 달리, 자연어에는 N개로 제한할 수 없는 단어와 현상이 존재하고 그에 따른 다양한 규칙이 존재한다. 또한 하나의 단어가 여러 의미를 가지고 있고 이로 인한 모호성 (ambiguity) 때문에, 실세계의 데이터를 이용하여 확률적으로 분석 과정을 다루는 방법이 대두되었다.

자연어 분석/처리 과정은 그 자체로도 의미를 가지지만 여러 응용 시스템에 융합되어 보다 다양한 측면에서 활용되고 있다. 자연어 이해의 응용 시스템은 크게 텍스트 기반 시스템과 대화 기반 시스템으로 나눌 수 있다. 문서 분류 (document classification), 정보 추출 (information retrieval), 기계 번역 (machine translation), 문서 요약 (text summarization) 등의 텍스트 기반 시스템과, 음성 인식 (speech recognition), 음성 언어 이해 (spoken language understanding), 음성 합성 (text-to-speech) 등의 대화 기반 시스템이 있다[2]. 응용 시스템에 적용되는 한 예로, 음성 합성

*본 연구는 산업자원부 뇌신경정보학 특정연구과제의 지원을 받아 수행되었음.

시스템에서 구문 끊어 읽기 (phrase break)에 문장의 구조적인 구문 정보를 활용할 수 있다면 보다 자연스러운 발음을 생성해 낼 수 있다는 점에서 구문 분석의 필요성을 찾을 수 있다[그림 1].

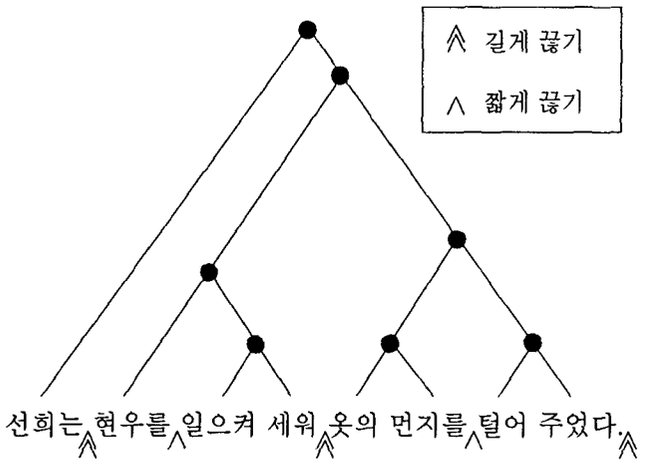

그림 1. 구문 구조에 따른 구문 끊어 읽기.

그러나 실질적인 자연어 응용 시스템에서는 파싱 (parsing) 단계의 정보를 배제한, 품사 태깅 (POS tagging)과 구 청킹 (phrase chunking) 단계의 정보만을 사용하고 있다. 이는 파싱 정보를 활용하여 응용 시스템의 성능을 블랙 박스 (black box) 형식으로 평가했을 때, 눈에 띌만한 성능 향상을 보이지 않거나 오히려 성능 저하를 가지고 오기 때문이다. 따라서 기본적인 자연어 처리 뿐 아니라 자연어 응용 시스템에도 긍정적이 효과를 기대할 수 있는 고정밀의 구문 분석기 혹은 파서가 필요하다.

본 논문에서는 이러한 문제 제기를 바탕으로 확률적인 한국어 의존 구조 분석에 대한 연구 내용을 담고 있다. 2절에서는 자연어 구조 분석에 관련된 기존 연구를 소개하고, 3절에서는 전체적인 시스템에 대해 언급한다. 4절에서는 본 연구에서 자연어 구조 분석을 위해 사용한 방법에 대해서 설명한다. 5절에서는 실험 및 분석을 싣는다. 마지막으로 6절은 결론 및 향후 연구 계획에 대해 논한다.

2. 관련연구

자연어 구문 구조에 대한 연구는 기계 기반의 자연어 처리 학문이 발생된 시점부터 끊임없이 이어지고 있다. 현재까지 성능이 뛰어나다고 알려진 대부분의 통계적 파서 (statistical parser)는 분석될 문장의 어휘 정보를 기반으로 조건부 확률을 적용한 어휘화된 파서 (lexicalized parser)이다. E. Charniak의 immediate-head parsing model[3]은 파스 트리 (parse tree)의 한 구성요소 c에서 하위 노드로 확장될 때, 구성요소 c의 head의 어휘정보를 조건부로 하는 확률에 따른다. 이것의 성능은 40단어 미만의 문장을 대상으로 했을 때 평균 90.1% LP(labeled precision)/LR(labeled recall)을 보인다. M. Collins의 dependency-based parsing model[4]도 성능 좋은 어휘화된 파서 중의 하나로서, 하나의 head와 이것과 관련된 여러 modifiers 사이의 확률적인 의존 관계를 고려한 모델이다. 40단어 미만의 문장으로 성능을 측정했을 때 평균 88.6% LP/LR을 보인다. 또한 이들은 Collins의 기본 parsing model을 바탕으로 서로 다른 reranking 방법을 적용하였는데, Collins의 reranking 모델은 평균 90.3% LP/LR, Charniak의 reranking 모델은 평균 91.0% LP/LR의 성능을 나타냈다[5,6]. 이 밖에도 의존 관계 기반의 MINIPAR[7]가 잘 알려져 있다.

앞서 언급한 어휘화된 파서와 달리, 파싱 과정에서 문장의 어휘 정보를 사용하지 않는 어휘화되지 않은 파서 (unlexicalized parser)에 대한 연구도 진행되고 있다. 이 파서는 어휘화된 파서에 비해 상대적으로 성능은 떨어지지만, 어휘를 고려하지 않기 때문에 처리 속도가 효율적이라는 특징이 있다. Stanford parser는 대표적인 어휘화되지 않은 파서로서 평균 86.36% LP/LR을 나타낸다[8]. 확률적인 문법 유도 없이 파스 트리의 일부를 하나의 단위로 취급하고 데이터로부터 바로 선택하는 데이터 지향 (data-oriented) 파싱[9]도 어휘화되지 않은 파서 중의 하나이다.

또한 파싱 하는 기법에 대한 연구로 chart parsing을 기반으로 하되 agenda에서 구성요소를 단순한 FILO (first-in-last-out)를 통해 선택하는 것이 아닌, 각 요소의 확률값 비교를 통해 우선순위에 따라 선택하는 기법도 있다[10]. 이는 모든 경우에 대해 chart를 형성하는 소모적인 기본 방식과 달리, 가장 확률이 높은 파스 결과를 우선적으로 선택함으로써 우선순위가 떨어지는 일부를 무시할 수 있다는 특징이 있다.

3. 전체 시스템 개요

이 장에서는 본 연구에 적용된 방법론에 대해 설명하기에 앞서, 한국어 파서의 전체 시스템 개요에 대해서 설명한다.

파싱은 주어진 문장으로부터 구조적인 형태를 구축하는 과정으로써, 실질적인 파싱이 이루어지기 전에 품사 태깅과 구 청킹 과정이 수행된다. 본 시스템의 전체 파싱 과정을 도식화하면 다음과 같다[그림 2].

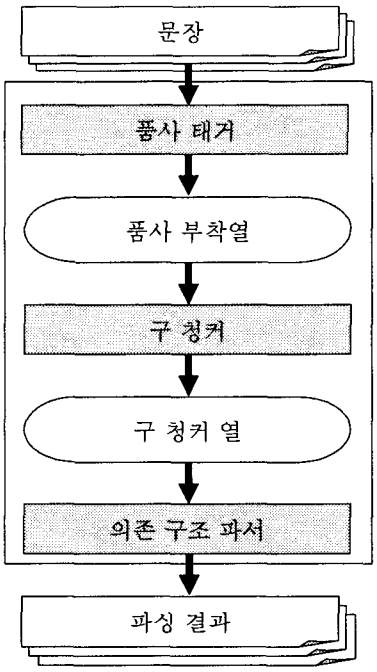

그림 2. 한국어 파싱 과정.

3.1. 품사 태거

한국어 품사 태거는, 세종 계획 품사 부착 말뭉치(2001년~2003년)를 이용하여 Hidden Markov Model (HMM) 기반으로 구현됐다. 훈련용 데이터로 12만 문장, 테스트용 데이터로 3만 문장을 사용했고, 이 때 태거의 성능은 정확률 93.86%, 재현율 93.21%, F1 값 93.53을 나타낸다. 품사 부착 열은 다음과 같은 형태로써 구 청커의 입력으로 사용된다[표 1].

선희/NNP 는/JX 현우/NNP 를/JKO 일으키/VV 어/EC 세우/VV 어/EC 옷/NNG 의/JKG 먼지/NNG 를/JKO 털/VV 어/EC 주/VX 었/EP 다/EF

표 1. 품사 부착 열 (단어/태그).

3.2. 구 청커

한국어 구 (phrase) 구조 분석 청커는 세종 계획 의미 구조 분석 말뭉치(2002년~2004년)를 이용하여 Conditional Random Fields (CRF) 기반으로 구현됐다. 세종 말뭉치 중 임의로 3만 문장을 선택하여 훈련용 데이터로, 나머지 약 3천 문장은 테스트용 데이터로 사용했다. 이 때 구 청커 성능은 정확률 91.10%, 재현률 91.99%, F1 값 91.54이다. 태거와 마찬가지로 다음과 같은 형태로 파서의 입력으로 사용된다[표 2].

Label	Constituent
NP_SBJ	선희/NNP 는/JX
NP_OBJ	현우/NNP 를/JKO
VP	일으키/VV 어/EC
VP	세우/VV 어/EC
NP_MOD	옷/NNG 의/JKG
NP_OBJ	먼지/NNG 를/JKO
VP	털/VV 어/EC
VP	주/VX 었/EP 다/EF

표 2. 구 청커 열.

3.3. 의존 구조 파서

의존 관계 분석용 파서 역시 세종 계획 의미 구조 분석 말뭉치를 이용했다. 최종적인 파스 결과는 다음과 같은 형태를 가진다[그림 3]. 파서에 대한 구체적인 내용은 4, 5절에서 설명하도록 한다.

4. 한국어 의존 구조 분석

본 논문에서 한국어 구조 분석 (즉, 파싱)을 위해 고려한 점은 크게 세 가지로 나눌 수 있다. 하나는 말뭉치로부터 구문 분석용 문법을 전부 자동 구축한다는 점이다. 다른 하나는 효율적인 처리를 위해 변형된 차트 파싱 기법을 적용했다는 것과 마지막으로 immediate-head 파싱 모델을 활용했다는 점이다. 파싱이 이루어지는 과정은 크게 3단계로 나눌 수 있다[그림 4]. 1단계에서는 말뭉치에 사용된 문법을 자동 추출하여 문법 집합을 확정 짓는다. 2단계에서는 추출

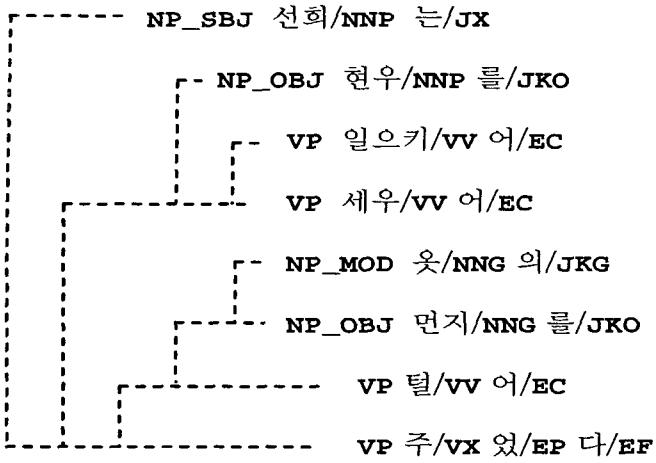

그림 3. 최종 파싱 결과.

된 문법을 바탕으로 bottom-up 방식의 차트 파싱을 수행하고, 마지막으로 차트 파싱의 결과를 의존관계에 따라 점수화하여 최종 결과를 확정짓는다.

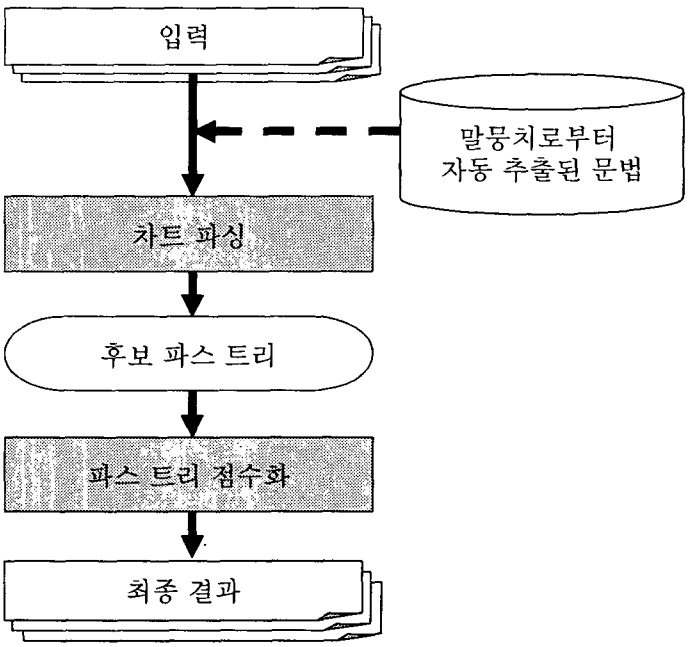

그림 4. 파싱 과정.

4.1. 구문 분석용 문법 자동 구축

본 파서에서는 구문 분석을 위한 문법을 손수 제작하지 않고, 파서 구축에 사용되는 말뭉치로 부터 자동으로 문법을 추출하여 사용한다. 문법을 자동 구축함으로써 모든 문법을 손수 제작하는 기존의 소모적인 방법을 탈피하고, 실제 사용되는 문법을 사용한다는 측면에서 문법의 효율성, 실재성이 두드러지게 된다. 또한 말뭉치 자체가 특정 영역에 한정되지 않기 때문에 이 말뭉치로부터 추출된 문법 역시 모든 영역에서 사용할 수 있다는 점에서 영역 이식성이 뛰어나다.

4.2. 확률적 차트 파싱 기법

차트 파싱은 동적 프로그래밍 (dynamic programming) 기법의 한 가지로써, 동일한 구성요소에 대해서는 중복의 여지없이 단 한번만 고려한다는 점에서 효율적이라고 알려져 있다. 그러나 문법 (grammar 혹은 rule)이 상당히 많은 경우와 입력 문장의 길이가 길어질 경우에는 기본적인 차트 파싱조차 효율성의 관점에서는 부정적일 수밖에 없다. 실제로, 구문 분석된 세종 말뭉치 3만여 문장에는 약 1,200개의 문법이 등장한다. 또한 영어권의 Penn Treebank II WSJ 말뭉치에는 $1.6x10^4$개의 문법이 존재한다.

따라서 본 연구에서는 보다 효율적이고 효과적인 파싱을 위해서 다음과 같이 변형된 차트 파싱 기법을 적용했다. 기존의 차트 파싱 알고리듬[2]과 변형된 알고리듬[표 4]의 차이는 다음과 같다.

기본적인 차트 파싱에서는 LIFO 또는 FILO 방식으로 agenda에서 한 구성요소를 선택해서, 이것이 문법의 left-hand의 첫 번째 요소와 같으면 active arc list에 추가하고 arc extension을 한다. 이 때, 어울리는 문법의 개수에 따라 수많은 active arc가 추가로 발생하게 된다.

이와 달리 변형된 차트 파싱에서는, agenda에서 모든 구성요소를 한 번씩 선택하지 않고 해당 구성요소가 파스 트리 내에서 등장할 확률을 계산해서 의도하는 임계값 (threshold) 미만이면 버리고, 임계값 이상일 경우에만 선택하는 방법을 취한다. 또한 파스 트리 내에서 깊이에 따라 적당히 임계값을 조정할 필요가 있다. 즉, 확장 가능한 모든 문법을 고려하지 않고 확률적으로 효용이 있음직한 active arc만 추가하기 때문에 기존의 방법 보다 효율적이다. 또한 일률적으로 임계값을 적용하기보다 파스 트리 깊이에 따른 임계값을 적용한다는 점에서 합리적이기도 하다.

입력이 있는 동안:
1. agenda가 비었을 경우, 다음 입력을 agenda에 추가.
2. **파스 트리 내의 깊이(depth)에 따라 특정 임계값 보다 큰 확률 값을 가지는 agenda의 구성요소 c를 선택함.**
3. $X \rightarrow C\ X_1 \ldots X_n$, 형태의 각 문법 규칙마다, 구간 $[p_1, p_2]$를 차지하는 active arc 추가.
4. Arc 확장 알고리듬에 따라 C를 차트에 추가.

표 3. 변형된 차트 파싱 알고리듬.

4.3. Immediate-head 파싱 모델

앞서 관련 연구에서 언급했듯이, 파스 트리 (parse tree)의 한 구성요소 c에서 하위 노드로 문법이 확장될 때, 구성요소 c의 head의 어휘정보를 조건부로 하는 확률에 따르는 파서를 immediate-head 파서라고 한다 [3]. [그림 5]에서 보듯이, **VP1**이 **NP_OBJ2 VP2**로 확장될 때의 확률은 **VP1**의 어휘 head인 '주'에 따라 영향을 받게 된다. 또한 **NP_OBJ2**가 **NP_MOD1 NP_OBJ3**으로 확장될 때에는 **NP_OBJ2**의 어휘 head인 '먼지', **VP2**가 **VP3 VP4**로 확장될 때에는 **VP2**의 어휘 head인 '주'에 따라 영향을 받게 된다.

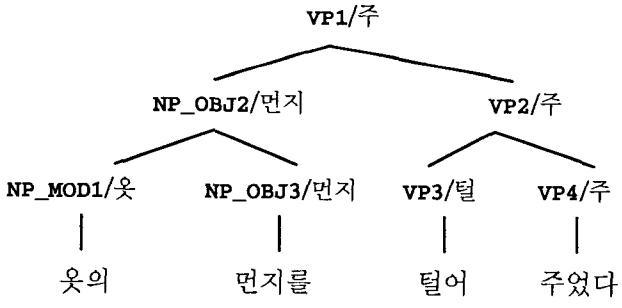

그림 5. Head 정보를 나타내는 파스 트리 예제.

이 파싱 모델은 파스 π의 확률을 할당하기 위해서 π 내의 각 구성요소 c에 대해 태그 $t(c)$를 결정하고, c의 어휘 head $h(c)$를 추측한 다음, c의 확장 결과인 $e(c)$를 추측하는 과정을 따른다. 따라서 한 파스의 확률은 다음과 같은 [식 1] 으로 표현될 수 있다[3].

$$p(\pi) = \prod_{c \in \pi} p(t(c)|l(c), H(c)) \cdot p(h(c)|t(c), l(c), H(c)) \cdot p(e(c)|l(c), t(c), h(c), H(c)) \quad (1)$$

여기서 $l(c)$은 **VP**와 같은 c의 label이고, $H(c)$는 c가 확장되어진 히스토리를 나타낸다. 이것은 c의 부모 노드의 label, 어휘 head, 어휘 head의 태그 정보로 나타내어 진다.

본 논문에서는 차트 파싱 이후 생성된 후보 파스 트리를 대상으로 각 트리 π의 우선순위를 정할 때, immediate-head 파싱 모델에 의한 확률 $p(\pi)$ 을 사용한다. 즉 bottom-up 방식으로 차트 파싱을 한 후, top-down 방식으로 후보의 순위 조정을 하는 것이다.

5. 실험 및 분석

5.1. 실험환경

본 실험에서 세종 의미 구조 분석 말뭉치 (2002년~2004년)를 사용했다. 세종 말뭉치의 통계적인 정보는 다음과 같다[표 5]. 이 중 40 형태소 미만의 문장 3,000개를 임의로 선택하여 테스트용으로 사용하고, 나머지는 훈련용으로 사용했다. 아래 표에서 보듯이 세종 말뭉치에서 등장하는 문법은 총 1,189개이지만, 효율성을 고려해서 N번 이상 등장하는 규칙을 최종 규칙으로 사용했다. 빈도수를 고려해서 선택된 규칙은 총 793개 (N=20) 이다.

정보	값
문장 개수	33,941
총 단어 개수	836,338
어휘 개수	33,689
문법 개수	1,189

표 4. 세종 말뭉치 정보.

성능 평가는 PARSEVAL[11] 척도를 사용했다. PARSEVAL 척도는 트리 전체가 완전히 일치해야만 맞은 것으로 인정하는 exact match 방법과 달리, 부분적으로 맞은 파스의 구성요소도 인정한다는 특징이 있다 [12].

$$Labeled\ precision = \frac{결과\ 파스\ 내에서\ 올바른\ 구성\ 요소\ 개수}{결과\ 파스\ 내의\ 구성\ 요소\ 개수}$$

$$Labeled\ recall = \frac{결과\ 파스\ 내에서\ 올바른\ 구성\ 요소\ 개수}{정답\ 파스\ 내의\ 구성\ 요소\ 개수}$$

5.2. 실험결과

본 논문에서 제시한 자동 구축된 문법을 이용한 확률적 차트 파싱 기법 그리고 immediate-head 파싱 모델을 적용한 파서의 성능은 다음과 같다[표 6]. 앞서 제안한 차트 파싱 기법의 성능은 LP/LR 76.27%/77.82% 이고, 차트 파싱의 결과를 대상으로 점수화를 통한 최종 결과의 성능은 LP/LR 78.98%/79.55%이다. 차트 파싱의 결과를 immediate-head 파싱 모델로 점수화를 함으로써 10.16%의 성능 향상을 보였다.

Model	LP	LR
Basic model	76.27 %	77.82 %
+ Immediate-head scoring	78.98 %	79.55 %

표 5. 파싱 성능 측정 결과.

기본 차트 파싱과는 다르게 확률적 차트 파싱 기법을 사용하면, 테스트용 3,000문장을 파싱하는데 192분 30초가 소요되고, 문장 당 파싱 속도는 3.85초이다. 이 때 생성되는 평균 후보 파스 트리의 개수는 문장 당 57.9개이다. 실험 환경은 Intel Xeon CPU 2.66GHz, Redhat/Linux이다.

6. 결론

지금까지 한국어 의존 구조 분석을 위한 파서에 대해서 살펴보았다. 본 논문에서는 기본적인 차트 파싱 기법이 아닌 효율성을 고려한 새로운 차트 파싱 기법을 사용하였다. 또한 문장 구성요소의 head를 문법 확장 하는데 활용함으로써 10.16% 의 성능 향상을 도모하였다.

서론에서 언급했다시피, 아직까지 파싱 단계의 정보가 실질적인 자연어 처리 응용 시스템에 눈에 띌 정도의 영향을 미치지 못하고 있는 것이 현실이다. 본 논문에서는 파서 자체의 성능에 대해 다루고 있지만 추후에는 다양한 자연어 처리 응용 시스템에 적용했을 때, 전체 시스템에서 성능 향상이 얼마나 이루어지는지 확인해 볼 필요도 있다.

또한 파서의 N-best 결과를 대상으로 다양한 자질 정보를 이용해서 다시 우선순위를 정하는 reranking 방법을 적용하는 것도 파서의 성능 향상을 위해 필요하다. 마지막으로 자연어의 궁극적인 처리 목표인 음성 언어를 다룰 수 있는 음성 언어를 위한 파서에 대한 연구도 계획하고 있다.

참고문헌

[1] Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman. 1986. *Compilers: principles, techniques, and tools.* Addison-Wesley.

[2] James Allen. 1995. *Natural Language Understanding.* The Benjamin/Cummings.

[3] Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *The Proceedings of North American Chapter of the Association for Computational Linguistics*, pages 132-139.

[4] Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, San Francisco, Morgan Kaufmann.

[5] Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML 2000)*, pages 175-182, Stanford, California.

[6] Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *The Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics.*

[7] Dekang Lin. 1994. PRINCIPAR - An efficient, broad-coverage, principle-based parser. *In The Proceedings of COLING-94,* pp.482-488

[8] Dan Klein and Christopher D. Manning. 2003.

Accurate unlexicalized parsing. In *The Proceedings of the 41st Meeting of the Association of Computational Linguistics.*

[9] Khalil Sima'an. 1995. An optimized algorithm for Data Oriented Parsing. In *The Proceedings International Conference on Recent Advances in Natural Language Processing,* Tzigov Chark, Bulgaria.

[10] Eugene Charniak, Sharon Goldwater and Mark Johnson. 1998. Edge-based best-first chart parsing. In *The Proceedings of the Fourteenth National Conference on Artificial Intelligence.* pages 127-133.

[11] E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *The Proceedings of the 1991 DARPA Speech and Natural Language Workshop,* pages 306-311.

[12] Christopher Manning and Hinrich Schutze. 1999. *Foundations of Statistical Natural language Processing,* MIT Press, Cambridge.