중한번역에서 구묶음을 이용한 파싱 효율 개선
(Improving Parsing Efficiency Using Chunking in Chinese-Korean Machine Translation)

양 재 형 † 심 광 섭 ††
(Jaehyung Yang) (Kwangseob Shim)

요 약 본 논문은 기계번역 시스템에서 파싱의 전처리 단계로 도입되는 구묶음 시스템을 제안한다. 구묶음 모듈은 구묶음의 결과로 얻어지는 의존관계 제약을 통하여 분석 시스템의 성능향상에 기여할 수 있다. 중국어를 위한 구묶음 시스템을 변형 기반 학습 기법에 근거하여 구현하며, 의존관계를 효과적으로 파서에 넘겨줄 수 있는 인터페이스를 고안한다. 구현된 모듈을 중한 기계번역 시스템에 통합하고, 중국 관련 웹사이트로부터 수집한 말뭉치를 이용한 실험을 통해 구묶음의 도입이 기계번역에서 분석시스템의 성능 향상에 기여할 수 있음을 보인다.

키워드 : 청크, 구묶음, 변형기반학습, 자연언어분석, 기계번역

Abstract This paper presents a chunking system employed as a preprocessing module to the parser in a Chinese to Korean machine translation system. The parser can benefit from the dependency information provided by the chunking module. The chunking system was implemented using transformation-based learning technique and an effective interface that conveys the dependency information to the parser was also devised. The module was integrated into the machine translation system and experiments were performed with corpuses collected from Chinese websites. The experimental results show the introduction of chunking module provides noticeable improvements in the parser's performance.

Key words : chunk, chunking, transformation-based learning, natural language parsing, machine translation

1. 서 론

기계번역 시스템에서 파서의 전처리 단계로 도입되는 구묶음(chunking) 시스템을 제안한다. 구묶음은 부분 파싱(partial parsing) 기법으로 주로 제안되어 왔으나, 기존의 기계번역 시스템에는 쉽게 통합되기 어려운 단점을 가진다. 현존하는 실용적인 기계번역 시스템은 주로 변환(transfer) 모델에 근거하고 있는데, 이러한 변환 모듈은 흔히 전체 파스 구조를 입력으로 요구하는 경우가 많다. 그러나 구묶음 시스템과 같은 부분 파서들은 일반적으로 청크의 내부 구조를 완전히 기술하지 않으므로 변환 모델에 근거한 기계번역 시스템에서 청크 정보를 직접적으로 이용하기 어렵다.

† 종신회원 : 강남대학교 컴퓨터미디어공학부 교수
jhyang@kangnam.ac.kr
†† 종신회원 : 성신여자대학교 컴퓨터정보학부 교수
shim@sungshin.ac.kr
논문접수 : 2003년 10월 29일
심사완료 : 2004년 6월 16일

본 연구에서는 기계번역 시스템의 성능 향상을 위해 구묶음 시스템을 전체 파싱의 전처리 단계로 도입하였다. 구묶음 모듈은 입력 문장으로부터 유용한 청크들을 인식하여 이로부터 얻어진 정보를 단어 간의 의존관계에 대한 제약으로 파싱 단계에 넘겨주게 된다. 이러한 방법으로 구묶음 모듈이 기존의 파싱 시스템에 쉽게 통합될 수 있으며 이를 통하여 파서의 전반적인 성능 향상에 기여하게 된다.

본 연구에서 구묶음 시스템은 중한 번역에서의 응용을 전제로 하여 인식 대상이 되는 중국어 청크의 타입과 성격을 결정하였고, 자연어처리를 위한 비통계적인 말뭉치 기반 학습 기법의 하나인 변형 기반 학습(transformation-based learning)에 기반하여 구현되었다. 중국 관련 웹사이트로부터 수집된 말뭉치를 이용한 실험을 통하여 구묶음 자체의 성능과 파싱 시스템에의 기여도 등을 검증하였다.

2. 관련 연구

청크(chunk)의 개념은 [1]에서 소개되었는데, 단어들의 의미있는 연속체로서 하나의 내용어(content word)와 몇 개의 인접한 기능어(function word)들로 이루어진 것으로 여겨졌다. 청크는 통사적 단위와 연관되지만 성분(constituent)에 정확히 대응하는 것은 아니다. 최근의 여러 연구들이 다양한 구묶음 및 부분 파싱 기법을 제안하였다.

[2,3]은 직렬로 연결된 여러 단계의 유한자동(finite-state cascade)을 제안하였는데, [2]는 이에 근거하여 효율적이고 신뢰할 만한 부분파서를 구현할 수 있음을 주장하였고, [3]은 이를 대규모 말뭉치로부터 정보추출(information extraction)에 사용하였다. [4]는 변형 기반 학습 기법을 영어의 구묶음 문제에 적용한 결과를 보여주는데, 기반 명사구의 학습뿐 아니라 문장을 명사 및 동사 타입 청크로 분할하는 문제에도 적용하였다. [5,6]에서는 말뭉치로부터 기반 명사구를 이루는 패턴을 찾아내는 일종의 규칙추출 알고리즘과, 비교적 단순한 형태의 오류기반 가지치기(pruning) 기법에 의해 기반 명사구를 인식하는 방안이 논의되었다. [7]은 메모리 기반 학습(memory-based learning)에 의해 입력문장으로부터 패턴을 찾아내는 얕은 파싱(shallow parsing)을 제안하는데, 여기서는 학습용 말뭉치 자체를 그대로 트라이(trie) 형태로 저장하여 두었다가 필요할 때 이 '메모리'로부터 긍정적인 증거와 부정적인 증거를 조사하여 패턴을 인식하게 된다. [8]은 최대 엔트로피(maximum entropy) 기법을 이용하는 부분 파서를 제안하고 독일어의 명사구 및 전치사구 인식에 적용하고 있다.

CoNLL-2000 텍스트 청킹 태스크[11]에서는 텍스트 청킹에 대한 최근의 이러한 여러 연구 결과들을 동일한 말뭉치 데이타에 대해 적용하여 그 결과를 비교·분석해 보는 기회를 제공했는데, 이에 따르면 Penn Treebank의 Wall Street Journal 말뭉치 일부를 학습말뭉치(211,727 토큰)와 검증말뭉치(47,377 토큰)로 사용한 영어의 구묶음의 결과는 대략 $F_{\beta=1}$: 85%~93% 정도의 성능을 보였다.

국내에서도 기계 학습 기법에 기반한 기본구의 구묶음 자체에 대한 [12,13]과 같은 연구나, 한국어 분석의 전처리 단계로서 구묶음을 도입한 [14,15] 등의 연구가 있으며, 대체로 복합명사 및 관형사+명사 구조의 결합과 본용언+보조용언 등의 결합을 중심으로 구묶음이 제안되어 왔다.

구묶음 혹은 부분 파싱에 대한 이러한 기존 연구들은 기법적으로는 기계학습을 이용한 것들이 다수였고, 구묶음의 대상에 있어서는 많은 연구들이 기반 명사구(base noun phrase)와 같은 단순한 청크를 인식하는 데 주력하였으나, 명사구, 동사구 청크를 포함하여 구절에 대한 청크를 모두 인식하여 실질적으로 부분 파싱을 목표로 하는 연구도 있었다.[1] 이와 같은 부분 파싱 혹은 얕은 파싱은 전통적인 전체 파싱(full parsing)에 대한 대안으로 제안된 것으로서, 문장 전체에 대해 하나의 전체적인 문법 구조를 만들어 내려는 전통적인 개념의 자연언어 파서가 실세계의 응용에 합당한 정확도나 효율성, 강건성을 달성하는 데 어려움을 겪어 왔기 때문에 이를 극복하려는 노력의 일환으로 연구되어 왔다. 부분 파싱은 입력 문장 전체에 대한 문법구조를 생성하려 하지 않고 입력 문장을 일련의 서로 겹치지 않는 청크들로 분할한다. 이와 같이 하여 전체 파서의 복잡도를 피하면서 어느 수준의 언어 정보를 효과적으로 획득할 수 있는데, 이런 정도의 결과들로도 정보추출이나 어휘정보 습득 등과 같은 대규모 자연언어 처리 응용에서 유용하게 사용될 수 있다는 점이 주장되어 왔다[3,9,10].

이와 같이 구묶음 자체의 유용성에 대한 많은 긍정적인 주장들이 제안되어 왔으나, 구묶음을 기계번역과 같은 전체 파스 구조를 필요로 하는 실용적인 대규모 응용에 통합하는 문제에 대해서는 별다른 연구가 없었다고 생각된다. 본 연구에서는 구묶음 및 적절한 인터페이스 모듈의 도입으로 기계번역에서 파서의 성능 향상을 꾀할 수 있음을 보이고자 한다.

3. 변형기반 학습에 의한 구묶음

여기서는 본 연구에서 제안하는 구묶음 모듈에 대해 설명한다. 이 모듈은 일반 도메인의 웹문서에 대한 실시간 온라인 번역의 제공을 목표로 현재 개발이 진행 중인 중한 기계번역 시스템에 통합되어 동작되어야 한다.

3.1 청크 타입

청크는 연속된 단어들의 모임으로, 비재귀적이며 겹치지 않는다는 특성을 갖는다. 아래의 예는 괄호로 청크의 경계와 청크 타입을 표시한 중국어 예문이다 .

(1) **[NP** 公安/NN 消防/NN "/PU 119/NU "/PU 接警台/NN] **[NP** 凌晨/NT] **[QP** 3/NU 时/MW 43/NU 分/MW] **[VP** 接到/VV] **[NP** 报警/NN] , /PU **[NP** 9/NU 部/MW 消防车/NN] **[VP** 及时/AD 赶到/VV] **[NP** 火灾/NN 现场/NN] /PU **[QP** 3/NU 时/MW 30/NU 分/MW] **[NP** 大火/NN] **[VP** 被/PO 扑灭/VV] 。/PU

(공안 소방 119 신고처가 이른 새벽 2시 43분에 신고를 접수하여, 9대의 소방차가 즉시 화재 현장으로

1) 물론 구절 청크(phrase chunk)라고 하여도 거의 모든 구묶음 관련 연구에서 구묶음의 대상이 되는 청크는 비재귀적(non-recursive)으로 정의된다. 이는 청크가 재귀적으로 정의되면 실제로 전체 파싱과 거의 동일한 복잡도를 갖게 되기 때문에 전통적인 전체 파싱에 대한 대안이 될 수 없기 때문이다.

달려 갔고, 3시 30분에 대형 화재가 진압되었다.)

청크 타입은 파스 트리에 사용되는 구절 태그와 같은 명칭을 사용하고 있는데, 본 연구에서 사용한 구절 태그들은 중국어 Penn Treebank[16]에서 사용된 것들과 유사하며, 몇가지 중국어에 고유한 태그를 제외하고는 보편적으로 사용되는 NP, VP 등과 같은 것들이다. 예를 들어 청크 타입 NP는 구절 명칭 NP에 대응하고, 청크 타입 VP는 구절 명칭 VP에 대응하는 등이다. 그러나 청크가 비재귀적이기 때문에 하나의 청크 안에 다른 청크가 내포되는 경우가 없다. 따라서 트리 상의 하나의 구절은 일반적으로 여러 개의 청크들로 분할되는 경우가 많게 된다. 또한 S와 같은 상위 구절들은 대응하는 청크 타입이 정의되지 않기도 한다. 위의 예에서와 같이 문장 내의 어떤 단어들은 어떠한 청크에도 포함되지 않을 수도 있다. 표 1은 본 연구에서 설정한 청크 타입들의 목록을 보이고 있다.

표 1 청크 타입

청크 타입	설명
ADJP	형용사구 청크
ADVP	부사구 청크
NP	명사구 청크
QP	수량사구 청크
VP	동사구 청크

각 청크는 항상 하나의 머리(head)를 갖는 것으로 정의한다. 이때의 머리는 통사론적인 관점에서의 머리를 말한다. 예를 들어 ADJP 청크의 머리는 AJ(형용사)이고, NP 청크의 머리는 NN(명사)이다. 또한 관례에 따라 청크는 일반적으로 머리의 왼쪽편에 놓인 선수식어(premodifier)들은 포함하지만 머리의 오른편에 놓인 후수식어(postmodifier)나 보어(argument)들은 포함하지 않는 것으로 정의한다([11]의 청크 정의 참고). 따라서 대부분의 경우에 청크의 마지막 요소가 머리가 된다. 그러나 예외적인 경우가 있는데, 첫째로 청크 전체가 괄호나 인용부호 혹은 기타 인용 심볼에 의해 앞뒤로 둘러싸여 있을 때에 이들 심볼들을 청크의 일부로 간주하였다. 따라서 이러한 구문부호들이 머리의 뒤에 나타날 수 있다. 둘째, VP 청크의 일부가 예외에 해당할 수 있는데, 연속된 두 개의 동사들이 결과보어구(verb-resultative compound)나 방향보어구(verb-directional compound)를 이룰 때에 앞의 동사가 머리가 되고 뒤따르는 동사는 선행 동사의 결과나 방향을 나타내면서 보충해주는 역할을 한다. 이때는 후행 동사가 선행 동사의 후수식어 내지는 보어가 되는 것이지만, 이러한 패턴이 중국어에 고유한 매우 밀접히 연관되는 복합어구이므로 VP 청크에 포함되는 것으로 간주하였다. 따라서 이러한 예외 경우에는 청크의 머리가 청크 내의 최후 요소가 아닐 수도 있다. 등위접속의 경우는 접속되는 마지막 요소를 통사적 머리로 간주했으므로 예외가 되지 않았다.

표 1의 목록에서 QP는 수량사구(quantifier phrase)에 해당하는 청크로서, 일반적으로 관형사나 수사 다음에 양사(measure word)가 나와서 이루어지는 중국어에 고유한 구절이다. 수량사구가 명사구 앞에서 선수식어로 사용되었을 때에는 NP 청크에 포함될 수 있는데, (2)에서 수사+양사의 구조로 이루어진 QP "9 部"가 그와 같은 경우이다. 다른 경우에는 QP가 단독으로 QP 청크를 이루게 된다.

(2) **[NP** 9/NU 部/MW 消防车/NN]

(9 대의 소방차)

한편 전치사구에 해당하는 PP와 같은 청크가 목록에 없다는 사실에 유의할 필요가 있다. 트리의 구성 성분으로서 전치사구는 전치사와 뒤따르는 목적어로 이루어지는데, 이 목적어 자신이 하나 혹은 그 이상의 독립된 청크를 이룰 것이기 때문에 결과적으로 PP 청크라는 것을 두더라도 거의 항상 전치사 한 단어로 이루어지는 청크가 될 것이다. 아래의 예 (3)에 포함된 전치사구인 "向交警中队"를 살펴보면, 전치사 "向"의 목적어인 "交警中队"가 명사구이고 이 명사구가 NP 청크로 인식되어야 한다. 청크는 비재귀적으로 정의되므로 이 경우 PP 청크를 둔다 해도 목적어를 제외한 전치사 한 단어만 남게 되는 것이다.

(3) 车祸/NN 发生/VV 后/LC 《/PU 英雄/NN 》/PU 剧组/NN 向/PO **[NP** 交警/NN 中队/NN] 交纳/VV 了/AS 5/NU 万/NU 元/MW 医疗/NN 保证金/NN 。/PU

(교통사고가 발생한 후 《영웅》 영화팀은 교통경찰 중대에 5만 위안의 의료 보증금을 납부했다)

본 연구에서는 이와 같이 거의 항상 하나의 단어로만 이루어지는 청크 타입은 구묶음의 대상으로 설정하지 않았다. 이는 여타의 구묶음 관련 연구와 대비되는 점으로, 예를 들어 [11]은 거의 항상 전치사 하나로만 이루어지는 PP 청크를 설정하고 있다. 이러한 처리는, 본 연구의 목적이 파서를 보조하는 역할로서의 구묶음에 대한 탐구에 있는데, 이러한 한 단어짜리 청크는 유용한 의존 제약을 제공하지 못하기 때문이다. 같은 이유로 LCP(장소사 LC가 머리인 구절)나 CP(보문소 DE가 이끄는 관계절)에 대해서도 대응하는 청크 타입이 설정되지 않았다.[2)]

2) 여기서 정의된 청크 타입이 지나치게 단순하여 구문 분석시 별다른 도움이 되지 못한다는 지적이 있을 수 있으나, 실제로는 구문 분석 상의 모

본 연구에서 사용한 청크 표기된 말뭉치(chunk-annotated corpus)는 모두 파스 트리로부터 자동적으로 청크를 추출하여 생성한 것이다. 예를 들어 NP 청크는 명사구의 머리와 모든 선수식어를 포함함으로써 얻어질 수 있다. 그러나 만약 선수식어들 가운데 하나가 스스로 NP 혹은 VP와 같은 청크를 갖는다면 청크의 비재귀성에 따라 이 구절은 몇 개의 더 작은 청크들로 나누어져야 한다. 이와 같은 과정의 결과로 어떤 단어는 아무 청크에도 소속하지 않게 될 수도 있다. 이러한 자동적인 변환 과정이 어느 정도의 오류를 수반할 가능성이 있지만, [11]에서도 이미 지적된 바와 같이 현재로서는 충분한 규모의 청크 표기된 말뭉치가 존재하지 않고, 학습 기반의 방법론에서 일정 수준 이하의 오류는 심각한 문제가 되지 않을 것으로 판단한다.

3.2 변형기반 학습

본 연구에서 중국어를 위한 구묶음 시스템은 Brill[17]에 의해 처음 제안된 변형기반 학습(transformation-based learning) 기법을 이용하여 개발되었다. 변형기반 학습은 기본적으로 비통계적인 말뭉치 기반의 학습 기법인데, 학습말뭉치로부터 일련의 변형규칙들의 순서 집합(ordered set)을 학습하는 과정으로 요약된다. 먼저 문제에 대한 간단한 임시적인 해결책(baseline)으로부터 시작하여 점진적으로(incrementally) 변형을 적용해 가면서 매 단계마다 최대의 개선을 가져오는 변형을 선택해 나간다. 이와 같이 하여 얻어진 일련의 변형 규칙들이 학습의 결과가 된다. 변형기반 학습의 장점은 학습 결과가 비통계적인 규칙의 형태를 갖게 되므로 가독성이 뛰어나며 학습 결과의 분석과 이해 및 적용이 용이하다는 점에 있다.

본 연구의 청크 태그는 CoNLL-2000[11]에서 사용된 {I, O, B} 스타일의 코드를 청크 타입에 결합하여 나타내었다.[3] 예를 들어 청크 태그 B-NP는 NP 청크의 시작 단어임을 나타내고 I-VP는 시작단어가 아니면서 VP 청크에 포함되는 단어임을 나타낸다. 또한 O 태그는 어떤 청크에도 속하지 않는 외부의 단어임을 뜻한다. 학습 말뭉치의 단어들은 이러한 청크 태그와 더불어 품사(part of speech) 태그가 표기되어 있다. 학습 과정을 간단히 설명하면 다음과 같다. 먼저 간단한 알고리즘(baseline)에 의해 초기 청크 태그들이 학습말뭉치의 각 단어들에 할당된다. 본 연구에서 사용한 초기할당 알고리즘은 품사 태그 X를 갖는 단어들을 학습말뭉치 내에서 조사하여 청크 태그 Y가 가장 많이 할당되었다면 품사 X를 갖는 단어들은 전부 청크 태그 Y로 할당하는 방법이다. 초기 할당 후에는 학습말뭉치의 현재 할당된 청크 태그 들을 개선할 가능성이 있는 모든 가능한 후보 변형규칙들을 조사한다. 학습말뭉치 내의 한 곳에서 틀린 태그를 옳게 변경하는 규칙이 말뭉치의 다른 곳에서는 옳은 태그를 틀리게 바꿀 수도 있기 때문에, 각 후보 규칙들의 순기여도(net contribution)가 계산되어야 한다. 하나의 후보 규칙이 학습말뭉치 내에서 틀린 태그를 옳게 바꾸는 회수를 f_{pos}라 하고 옳은 태그를 틀리게 바꾸는 회수를 f_{neg}라 하면 이 후보 규칙의 순기여도는 단순히 $(f_{pos}-f_{neg})$로 계산된다. 생성된 후보 규칙들 가운데 최대의 기여도를 갖는 하나의 규칙이 선택되고 학습말뭉치에 적용되어 청크 태그 할당을 수정한 다음, 학습된 변형규칙의 집합에 추가된다. 전체 과정이 태그 할당이 변경된 학습말뭉치에 반복되는데, 더 이상 변형 규칙이 없거나 선택된 규칙의 순기여도가 미리 정한 임계값보다 적으면 학습을 종료한다.

학습의 매 단계에서 후보 변형규칙들을 생성해 내기 위해 규칙 템플릿(rule template)의 집합이 사용되는데, 이 규칙 템플릿이 시스템이 학습하고자 하는 규칙의 성격을 규정한다. 즉, 변형규칙을 만들어 낼 때 어떤 자질(feature)들을 사용할 것인지를 결정한다. 이 규칙 템플릿의 집합은 미리 수동으로 만들어져 학습 알고리즘에 주어지며, 알고리즘이 변형 규칙을 학습하고자 할 때의 탐색 공간을 정의하게 된다. 그러므로 학습을 위해 규칙 템플릿의 집합을 설계할 때에는 학습하고자 하는 대상 문제가 어떤 지식을 필요로 하는지, 그리고 탐색 공간의 크기를 어느 정도 수준으로 제한할 것인지에 대한 판단이 필요하다.

(4)는 규칙 템플릿의 한 예인데, 이 템플릿은 현재의 청크 태그와 선행하는 두 단어의 품사 태그에 근거하여 특정 단어의 청크 태그를 변경하는 규칙들을 정의하고 있다.

(4) chunk_0 pos_-2 pos_-1 => chunk

다시 말하여 이 템플릿을 학습말뭉치의 특정 위치에 적용하여 얻게 되는 변형 규칙은, 특정 단어의 현재 청크 태그와 선행하는 두 단어의 품사 태그를 '보고' 해당 단어의 청크 태그를 변경하는 방식으로 구성된다. (5)는 이 템플릿에 의해 얻어진 실제의 변형 규칙의 예를 보이고 있다.

(5) chunk_0=B-NP pos_-2=NN pos_-1=CJ => chunk= I-NP

이 규칙은 어떤 단어의 현재 청크 태그가 B-NP, 즉

호성을 상당부분 미리 제거하는 효과를 가지며 이러한 사실이 실험에서의 효율 개선으로 확인되고 있다. 예를 들어 수량사구 QP는 통사적으로 선행 혹은 후행 명사구와 결합할 수도 있고 후행 동사구를 수식할 수도 있으나 미리 결합된 QP 혹은 NP 청크들이 이러한 가능성을 상당히 줄일 수 있다. 그리고 인접한 명사복합체와 동사복합체, 인접한 명사접속과 동사접속 등이 청크로 미리 인식되면 이와 관련하여 상당 부분의 복잡도가 감소되는 것이 사실이다.

3) 이는 원래 [4]에서 제안된 표현이다.

NP 청크의 시작이고, 명사(NN)와 접속사(CJ)에 바로 이어지는 단어라면, 그 단어의 청크 태그를 I-NP, 즉 NP 청크의 시작이 아닌 내부 단어인 것으로 고치라는 의미이다. 즉, "명사1+접속사+명사2"로 구성된 예에서 명사2가 NP 청크의 시작(I-NP)으로 되어 있을 때 규칙 (5)가 적용되면, 선행 명사인 명사1도 동일 청크에 속하는 것으로 보고 명사2의 청크 태그를 I-NP로 고치는 것이다. 이러한 변형 규칙은 학습말뭉치의 특정 위치에 템플릿을 적용함으로써 템플릿에 기술된 chunk, pos 등의 자질(feature)에 특정한 실제 값을 할당하여 얻게 된다. 본 연구에서 사용된 자질들의 종류는 (6)에 나열된 바와 같다.

(6) a. 인접한 단어들의 어휘 형태(word)
b. 인접한 단어들의 품사 태그(pos)
c. 인접한 단어들의 청크 태그(chunk)

(6)에 나열된 자질들이 어떤 영역 내에서의 존재 여부가 검사되는 방식으로 사용되기도 한다. 예를 들어 pos[-3,-1]=NU라고 하면 현재 단어에서 좌측으로 세 번째에서 첫 번째 사이에서, 즉 현재 단어가 w_n이라면 w_{n-3}, w_{n-2}, w_{n-1} 가운데서 품사 태그 NU가 발견되는지 여부를 검사하는 것이다. 이러한 기본적인 자질들이 체계적으로 결합되어 규칙 템플릿을 구성하게 된다.

실험에서는 템플릿에서 고려할 주변 문맥을 [-3,+3]의 영역으로 제한하였다. 이는 변형 규칙들이 좌우로 최대 3단어까지만 볼 수 있다는 뜻이다. 이 문맥 영역이 학습 과정상의 탐색 공간의 크기를 결정하게 되므로 전체 학습 과정의 효율에 큰 영향을 미치게 된다. 즉, 문맥을 넓게 잡으면 성능이 개선될 가능성이 있지만 학습 시간이 많이 걸린다. 문맥의 크기에 따른 성능의 개선 정도는 적용하고자 하는 문제가 어느 정도의 주위 문맥에 의존하느냐 하는 점에 달려 있다. 문제 자체가 국부적인(local) 특성을 갖고 있다면 문맥을 넓혀도 별다른 성능 개선이 없을 것이다.[4]

이와 같은 방식으로 (6)에 나열된 자질과 주변 문맥을 나타내는 인덱스를 결합하면 규칙 템플릿을 구성하기 위한 기본 소자를 얻게 되는데, 이들 가운데 임의로 몇 개를 결합하면 (4)의 예와 같은 템플릿이 되고, 이 템플릿이 특정 문맥에 적용되어 실제 값을 할당받게 되면 학습 대상이 되는 (5)와 같은 규칙 후보가 되는 것이다. 규칙 템플릿은 범위 내의 모든 가능한 조합을 다 구성하여 사용하는 방법도 있지만 그럴 경우에 학습 부담이 상당히 커지게 되므로, 설계자의 직관에 의거해 제한적으로 구성하는 것이 상례이다[4]. 본 연구의 실험에서는 (6)의 자질들과 [-3,+3]의 문맥 영역 범위 내에서 150개의 템플릿을 구성하여 사용하였다. 구체적으로는 청크 태그와 어휘 형태에 대해서는 [-2,+2] 범위를 포함했고, 품사 태그에 대해서만 [-3,+3] 범위를 포함하였다. 템플릿의 크기에 따른 학습 결과의 차이가 있을 수 있는데, 이것도 위의 문맥 범위처럼 성능 개선 가능성과 학습 효율이 상호의존적으로 관련되므로 실험적으로 선택되어야 하며 아래 실험에서 사용한 템플릿의 집합이 최종적인 것은 아니다.

3.3 실험 결과

중국어를 위한 구묶음 시스템을 Fast Transformation-Based Learning Toolkit[18]을 이용하여 구현하였다.[5] 앞절에서 설명한 150개의 규칙 템플릿이 사용되었다. 충분한 규모의 청크 표기된 중국어 말뭉치를 구할 수 없으므로 본 연구에서는 파서가 만든 파스 트리로부터 청크 정보를 뽑아내어 학습말뭉치로 이용하였는데, 파스 트리가 어느 정도의 오류를 포함하지만 학습기법에 의해 유용한 결과를 얻을 수 있을 것으로 기대하였다. 이러한 가정을 확인하기 위해 소규모의 예비실험을 통한 비교를 수행하였다. 두 종류의 독립된 소규모 말뭉치를 준비하였는데, Mtrain(45,684토큰)과 Mtest(10,626토큰)는 중국어를 모국어로 하는 화자들에 의해 수동으로 구축된 파스 트리로부터 만들어진 청크 표기된 말뭉치이고, Atrain(45,848토큰)과 Atest(10,757토큰)는 파서가 생성한 파스 트리로부터 자동 변환하여 얻은 말뭉치이다. 실험은 이 두 종류의 말뭉치로부터 구묶음을 학습 및 검증하여 결과를 비교한 것인데, 이 실험을 위한 학습 임계값을 1로 하여 순기여도가 1인 규칙에 도달하면 학습 알고리즘이 종료되도록 하였다.

표 2는 실험의 결과를 일반적으로 통용되는 정확율(precision), 재현율(recall) 및 F-rate(Fβ=1)에 의해 표기한 것이다. 첫 번째 행은 수동 교정하여 비교적 정확할 것으로 기대되는 학습 및 검증 말뭉치로부터 얻어진 결과이고 두 번째 행은 사람이 수정하지 않은 파스 트리로부터 자동 구성한 말뭉치를 이용한 결과이다. 두 실험에서 학습된 변형 규칙은 각각 353개와 399개였다. 세 번째 행은 Atrain으로부터 얻은 변형규칙을 Mtest에 적용하여, 같은 검증말뭉치를 사용했을 때 Mtrain과 Atrain으로부터 얻은 규칙들이 어떠한 성능차를 보이는지 알아보기 위해 수행된 실험 결과이다. 표에 나타난

4) 본 연구에서는 예비실험에서 문맥을 다소 넓혔을 때에도 별다른 성능의 개선을 얻을 수 없었고, 지나치게 넓은 문맥으로는 큰 규모의 학습말뭉치에 대해 학습시간이 너무 오래 걸리는 문제가 있었으므로 문맥을 [-3,+3]으로 제한하였다. 그러나 이러한 문맥의 크기는 최종적인 것이 아니며 말뭉치의 크기나 실험 조건에 따라 달라질 것이다.

5) Fast Transformation-Based Learning Toolkit(혹은 fnTBL Toolkit)은 변형 기반 학습기법을 이용하여 분류(classification) 문제를 해결할 수 있도록 지원하는 공개 소프트웨어이다(http://nlp.cs.jhu.edu/~rflorian/fntbl/).

실험 결과는, 비교적 부정확할 것으로 생각되는 Atrain 말뭉치를 학습에 사용하였을 때에도 별다른 성능의 저하를 보이고 있지 않다. 이러한 점은 구묶음이 비교적 국지적인 현상을 학습 대상으로 하고 있어서 파스 트리의 구조적 오류에 그다지 영향받지 않기 때문인 것으로 추정된다. 실제로 실험의 결과는 기대한 바와 달리 Atrain으로부터 얻어진 규칙이 Mtrain으로부터 얻은 규칙보다 Mtest에 약간 더 나은 결과를 보였으나, 그 차이가 유의미한 것으로 보이지 않는다. 실험의 결과는 구묶음의 관점에서 볼 때 Mtest가 Atest보다 좀더 '쉽다'는 점을 시사하고 있다.

표 2 구묶음 결과(예비실험)

학습말뭉치	검증말뭉치	정확율	재현율	Fβ=1
Mtrain	Mtest	94.88	93.91	94.39
Atrain	Atest	93.79	92.08	92.93
Atrain	Mtest	95.07	93.91	94.49

위 실험의 결과에 따라, 자동 생성된 파스 트리로부터 변환하여 얻은 학습말뭉치(157,835 토큰)와 검증말뭉치(45,879 토큰)를 이용하여 구묶음 실험을 행하였다. 두 말뭉치는 모두 여러 종류의 중국 웹사이트로부터 수집된 실제 문장들로 이루어졌으며 파스 트리로부터 자동 변환하여 사용하였다. 규칙 템플릿은 동일한 150개가 사용되었고 학습 임계치는 3으로 하였는데,[6] 이 실험에서 학습된 변형규칙의 수는 303개였다. 표 3에 초기 할당 후의 결과 및 최종 결과를 보였다.

표 3 구묶음 결과

	정확율	재현율	Fβ=1
Baseline	40.64	64.53	49.56
Final	95.75	94.13	94.93

결과는 소규모 말뭉치를 이용하였던 Atest 결과보다 F-값($\beta=1$) 기준으로 약 2% 정도의 성능개선을 보였다. 기존의 출판된 결과와는 대상언어나 실험조건 등이 모두 다르므로 직접적인 수치 비교를 할 수 없지만 비교적 정확한 결과라고 볼 수 있다. 그러나 예를 들어 CoNLL-2000[11]과 같은 실험과 비교한다면 본 연구가 대상으로 한 문제 자체가 약간 '쉽다'는 사실을 지적할 수 있다. 이는 본 연구의 목적이 파서 전처리기로서의 구묶음을 다루기 때문에 일반적인 구묶음보다는 제한적이고, 그 결과로 인식 대상인 청크 타입의 종류가 더 적은 점에서도 발견할 수 있다.

3.4 오류 분석

검증말뭉치(45,897 토큰)에 대한 실험의 오류를 분석한 결과, 아래와 같은 몇 개의 유형으로 나누어 볼 수 있었다.

1) [{ } .. { }] : 54.3%

중괄호({..})는 올바른 청크의 범위를 나타내고 대괄호([..])는 예측된 청크의 범위를 나타낸다. 따라서 이 유형은 예측된 청크의 범위가 너무 커서 둘 이상의 실제 청크를 포괄하고 있으나, 청크의 시작과 끝은 일치하는 경우이다. 가장 많이 발생한 오류 유형으로 전체 오류의 반 이상을 차지하였다. 이 유형에 대해서는 예측된 청크의 태그 패턴을 분석하여 후속 단계에서 필요하다면 추가 분할하는 방안을 생각해 보아야 할 것이다.

2) { [ ] .. [ ] } : 19.9%

위의 1)과는 반대로 예측된 청크가 실제 청크를 더 잘게 분할한 경우이며, 역시 시작과 끝은 일치하고 있다. 이 유형은 큰 범위의 청크를 파악하는 데 실패하였지만, 이로 인해 구문분석 단계에서의 오류가 초래되지는 않는다.

3) [..{ }..] : 9.0%

예측된 청크가 실제 청크보다 크면서 시작이나 끝이 일치하지 않는다는 것은, 불필요한 요소가 청크에 포함되었다는 뜻이다. 전형적으로 구두점 등의 요소가 청크에 잘못 포함된 (7)과 같은 예를 말하는데, 이러한 유형의 오류도 종류에 따라서는 후속 단계에서 수정할 여지가 있는 것으로 보인다.

(7) ... "/PU [{vp 取胜/VV }] 的/DE [{np 最/AD 主要/JJ 原因/NN } "/PU] 。/PU

4) {..[]..} : 3.2%

위의 3)과 상반된 경우로서 예측된 청크가 올바른 청크의 일부이다. 이 경우도 2)와 마찬가지로 대부분의 경우에 심각한 문제가 되지 않는다.

5) 청크 태그 불일치 : 6.3%

이 유형은 청크의 범위가 정확히 인식되었으나 잘못된 청크 타입이 할당된 경우이다. 즉 "[np{qp 一場/NU }]"와 같이 수량사구(QP) 청크로 인식되어야 할 것이 명사구(NP) 청크로 인식된 경우이다. 이는 본 연구가 전제로 하는 중국어 분석 문법에서 위와 같은 일부의 수사(NU)들이 수량사구로 분석이 되기도 하기 때문에 발생하는데, 대부분 단일 단어이고 청크의 범위가 정확히 인식되었기 때문에 구문 분석에 미치는 악영향은 없다.

6) 기타: 7.2%

나머지 경우는 예측된 청크와 올바른 청크의 범위가 서로 엇갈려서 별다른 대책을 기대할 수 없는 오류들을

6) 말뭉치가 커지면 학습되는 변형규칙의 수가 늘어나지만, 순기여도가 매우 적은 낮은 순위의 규칙들은 실제로 성능에 미치는 영향이 미미하거나 부적절하다. 이를 걸러내기 위해 적절한 임계치를 두는데, 임계치를 낮추면 학습되는 변형규칙의 수가 늘어난다. 본 실험에서는 임계치를 3 아래로 내려도 별다른 성능개선이 없었으므로 3을 사용하였다.

말한다.

4. 파싱 시스템에서의 구묶음

본 연구의 실행 환경이 되는 중한 기계번역기의 분석 시스템의 구조는 그림 1과 같다. 실세계 문장들의 과도한 길이와 그로 인한 복잡도에 대응하기 위해 시스템은 구두점과 인접 문맥 정보에 근거하여 먼저 문장을 일련의 절(clause)들로 분할한다. 다음에 각 절에 대해 구묶음이 시도되고 관련 정보가 차트(chart) 기반의 파서 모듈로 전달된다. 분석의 결과로 둘 이상의 트리가 발생할 수 있으므로, 트리선택을 담당하는 후처리 모듈을 두어 여러 가지 지식원을 이용하여 하나의 트리를 선택하는 역할을 수행하게 한다.

그림 1 분석시스템 구조

4.1 청크와 의존관계 정보

구묶음에 의해 얻어진 정보가 단어들 간의 결합을 제한할 수 있으므로 파서에 도움을 줄 수 있다. 그러나 기계학습 방법에 기반한 대부분의 구묶음 시스템은 청크의 시작과 끝 부분만을 인식해 낼 뿐 청크 내부의 구조는 제공하지 않는다. 따라서 파서가 구묶음 시스템이 제공하는 정보를 직접 이용하는 데에는 어려움이 있다. 본 연구에서는 구묶음에 의해 얻어진 의존관계에 대한 제약 정보를 효과적으로 파서에 넘겨줄 수 있는 인터페이스를 고안하였다.

단어의 수가 n인 문장에서 의존관계(dependency relation)를 n×n 행렬 Dep로 정의한다. 여기서 Dep[i,k]는 i-번째 단어가 k-번째 단어에 대해 의존관계를 갖는지 여부를 나타낸다. 만약 i-번째 단어가 k-번째 단어의 의존소(dependent, 다시 말해 보어나 수식어)라면 Dep[i,k] = 1이 되고, 이 사실은 본 연구의 문법에서 어떤 단어도 둘 이상의 머리(head)를 갖지 못하기 때문에 모든 j(j ≠ k)에 대해 Dep[i,j] = 0임을 함축한다. 또 만약 구묶음에서 파악된 어떤 청크가 파싱 트리에서의 성분(constituent)과 일치한다면 청크 내부의 요소들은 그 머리를 제외하고는 청크 외부의 어떤 요소와도 의존관계를 가질 수 없게 된다. 즉, 만약 문장이 n 개의 단어 $w_1, w_2, ..., w_n$으로 이루어져 있고 w_c에서 w_k에 이르며 머리가 w_h인 청크 C[c:k]가 발견되었다면($1 \le c \le h \le k \le n$), $c \le i \le k$, $i \ne h$인 모든 i와, $1 \le j < c$, $k < j \le n$인 모든 j에 대해 Dep[i,j] = 0이 된다. 그림 2는 청크가 인식되었을 때 얻을 수 있는 이와 같은 관계를 의존관계 행렬의 상태로 보여주고 있다. 처음에는 모든 셀들이 ?로 표시되어 의존관계에 대한 제약이 없다가, 두 단어 간에 의존관계가 있을 수 없음이 밝혀지면 해당 셀이 0으로 표시된다. 구묶음 단계에서 얻어진 정보는 이와 같이 그 문장의 의존관계 행렬의 상태를 결정하고, 이 행렬이 파싱 단계에서 참조되어 0으로 표시된 구조들 사이의 결합이 금지되는 방식으로 동작하게 된다.

	1	...	*c*	...	*h*	...	*k*	...	*n*
1	-	?	0	0	?	0	0	?	?
...	?	-	0	0	?	0	0	?	?
c	0	0	-	?	?	?	?	0	0
...	0	0	?	-	?	?	?	0	0
h	?	?	0	0	-	0	0	?	?
...	0	0	?	?	?	-	?	0	0
k	0	0	?	?	?	?	-	0	0
...	?	?	0	0	?	0	0	-	?
n	?	?	0	0	?	0	0	?	-

그림 2 의존관계 행렬

청크의 통사적 머리는 대부분의 경우에 청크에 속한 단어들의 품사 태그를 살펴 봄으로써 알아낼 수 있다. 예를 들어 NP 청크의 머리는 그 청크의 마지막 명사 요소가 되어야 하고, QP 청크의 머리는 그 청크의 마지막 수량사(measure word) 요소여야 한다. 만약 청크가 인식되었고 그 청크가 정확히 성분에 대응하며 그 머리가 알려졌다면 그림 2와 같은 의존제약은 자동적으로 구성될 수 있다.

그러나 둘 이상의 동사 요소를 갖는 VP 청크의 경우가 문제가 된다. 이러한 VP 청크에는 다음과 같은 두 가지 이유로 인해 위에서 설명한 절차가 적용되지 못한다. 첫째, VP 청크는 트리상의 동사구 성분에 정확히 대응하지 않을 수도 있다. 예를 들어 타동사는 명사구를 목적어로 취할 수 있는데, 이 명사구는 일반적으로 스스로 NP 청크가 될 것이다. 그럴 경우 본 연구에서의 청크는 재귀적으로 정의되지 않으므로 이 동사구는 (VP 청크와 NP 청크라는) 최소한 두 개의 연속된 청크로 쪼개지게 된다. 이와 같이 VP 청크는 동사구로부터 다른 청크들을 뽑아내고 난 후에 남는 요소들로 이루어지게 되는 경우가 많아, 트리 상의 성분과 어긋나게 된다. 만약 청크가 성분에 정확히 대응하지 않는다면 머리를 제외한 청크의 다른 단어들도 청크 외부의 단어들과 의존관계를 가질 수 있어 위에 말한 바와 같은 제약을 사용할 수 없다.

둘째로 VP 청크가 둘 이상의 동사 요소를 가지면 단순히 품사 태그를 살펴보는 것만으로는 청크의 머리를 결정하지 못한다. 이는 앞서 청크를 정의할 때에 VP 청

크에 대해서만은 후수식어를 허용했기 때문이다. 예를 들어, VP 청크의 품사태그가 "AD VV VV"로 이루어져 있다면 첫째 동사가 둘째 동사의 부사적 수식어(상황어) 구실을 할 수도 있지만 반대로 둘째 동사가 첫째 동사의 결과보어나 방향보어일 수도 있다. 전자의 경우는 두 번째 동사가 머리가 되겠지만 후자의 경우는 첫 번째 동사가 머리가 되어야 한다. 또는 두 동사가 접속(연동) 관계에 놓여 있을 수도 있는데, 이런 때에는 청크에 대한 본 연구의 가정에 따라 두 번째 동사가 머리로 가정된다. 이와 같은 이유로 둘 이상의 동사 성분을 갖는 VP 청크에 대해서는 위에서 설명한 절차에 따라 의존관계 제약을 구축하는 것이 불가능하다.

그러나 그런 경우에도 제한된 범위의 의존관계 제약은 여전히 얻을 수 있는데, 앞서 든 예의 "AD VV VV"로 이루어진 VP 청크의 경우 첫 번째 단어 AD는 청크 내에서 자신의 머리를 가져야 한다. 그러므로 이 부사와 청크 외부의 모든 단어들 간의 의존관계는 0으로 지정할 수 있다. 일반적으로 둘 이상의 동사 요소를 갖는 VP 청크에서는, 청크에 속하는 모든 비머리(non-head) 요소 w_i와 청크 외부의 모든 단어 w_j에 대해 Dep[i,j] = 0이 된다.

4.2 실험 결과

앞서 구현한 구묶음 모듈을 중한 기계번역 시스템의 파서 직전 위치에 삽입하고, 구묶음 학습과정에 사용하지 않은 독립된 검증말뭉치(10,160 토큰)에 대해 실험을 행하였다. 표 4는 구묶음 모듈을 사용했을 때와 사용하지 않았을 때의 파싱 결과를 비교하고 있다.

표 4 파싱실험 결과

	구묶음 미사용시	구묶음 사용시	차이
정확도	90.35%	90.87%	+0.52%
평균 트리수	6.1	3.8	-37.8%
초당 평균 처리단어	866.3	1012.8	+16.9%
평균 메모리 사용량	237.2KB	184.0KB	-22.4%

구묶음 모듈을 사용했을 때 평균 트리 수가 상당히 줄어들었으며, 속도와 메모리 사용량 면에서 현저한 개선을 보였다. 속도는 초당 파싱된 단어의 수로 측정하였는데 약 17% 증가되었고, 파싱 과정에서 할당된 총 메모리의 평균 크기는 약 22% 정도 감소되었다.[7] 정확도는 전체 의존관계에 대한 올바른 의존관계의 비율을 의미하는 것인데, 파싱 결과 얻어진 트리를 중국어를 모국어로 하는 화자들에 의해 수동 교정된 정답 트리와 비교하여 구하였다. 예를 들어 n 개의 단어를 포함하는 문장은 n-1 개의 의존관계를 갖지만, 문장 전체의 머리(root)가 더 이상 자신의 머리를 갖지 않아야 한다는 점을 하나의 의존관계로 간주하면 n 개의 의존관계로 이루어지는 것으로 볼 수 있다. 각 단어들이 대응하는 정답 트리에서와 같은 머리를 할당받으면 해당 의존관계가 올바른 것으로 간주한다. 구묶음 모듈을 사용한 결과는 약 0.5% 정도의 정확도 향상을 보였는데, 그다지 극적인 개선이라고 볼 수는 없다. 이것은 현재 분석 시스템에 포함되어 있는 트리선택 모듈이 비교적 믿을 만하다는 점을 시사한다. 그러나 효율성에 있어서의 개선점만으로도 구묶음 모듈을 기존의 기계번역 시스템에 추가할 만한 충분한 가치가 있다고 판단되며, 파싱 전처리 단계로 구묶음을 도입하는 운용의 특성상 효율과 정확도 양면에서 현저한 개선을 기대하기는 어렵다고 생각된다.

7) 속도와 메모리 사용량에 대한 측정은 펜티엄-III 930Mhz PC(Linux 운영체제)에 192 MByte RAM이 설치된 상태에서 이루어진 것으로, 5회 반복하여 평균을 구한 것이다.

5. 결 론

본 논문은 기계번역 시스템에 포함된 파서의 성능을 개선하기 위해 구묶음 모듈을 도입하는 방안을 제안하였다. 이를 위해서는 구묶음 모듈과 기존의 파서를 연결하여 청크 정보를 효과적으로 파서에 넘겨줄 수 있는 적절한 인터페이스가 필요하다. 본 연구에서는 변형 기반 학습 알고리즘을 이용하여 중국어 구묶음 시스템을 구현하였으며, 이를 중한 기계번역 시스템의 일부로 통합하였다. 실험 결과는 파싱 효율과 정확도의 면에서 개선을 가져왔는데, 특히 효율의 개선이 현저하였다. 대규모 자연어처리 시스템이 감당해야 할 실세계 문장들이 갖는 과도한 복잡도를 고려할 때, 본 연구가 제안한 바 구묶음 시스템이 가져오는 파싱 효율상의 상당한 개선은 실제적인 응용을 개발하는 데 큰 도움이 되리라고 판단된다.

참 고 문 헌

[1] S. Abney, "Parsing by Chunks," in Berwick, Abney, Tenny eds., Principle-Based Parsing, Kluwer Academic Publishers (1991) 257-278.

[2] Abney, S., "Partial Parsing via Finite-State Cascades," in Proc of Robust Parsing Workshop ESSLLI'96 (1996) 8-15.

[3] Hobbs, J., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M., Tyson, M., "FASTUS: A Cascaded Finite-State Transducer for Extracting Information From Natural Language Text," in Roche, Schabes eds., Finite-State Language Processing (1997) 383-406.

[4] Ramshaw, L. A., Marcus, M. P., "Text Chunking

Using Transformation-based Learning," in Proc of 3rd ACL Workshop on Very Large Corpora (1995) 82-94.
[5] Cardie, C., Pierce, D., "Error-driven Pruning of Treebank Grammars for Base Noun Phrase Identification," in Proc of ACL/Coling (1998) 218-224.
[6] Cardie, C., Pierce, D., "The Role of Lexicalization and Pruning for Base Noun Phrase Grammars," in Proc of AAAI-99 (1999).
[7] Argamon-Engelson, S., Dagan, I., Krymolowski, Y., "A Memory-based Approach to Learning Shallow Natural Language Patterns," in Proc of ACL/Coling, (1998) 67-73.
[8] Skut, W., Brants, T., "A Maximum Entropy Partial Parser for Unrestricted Text," in Proc of 6th Workshop on Very Large Corpora (1998).
[9] Briscoe, E.J., Carroll, J., "Automatic Extraction of Subcategorization from Corpora," in Proc of ACL Conference on Applied Natural Language Processing (1997).
[10] Carrol, J., Minnen, G., Briscoe, T., "Corpus Annotation for Parser Evaluation," in Proc of EACL'99 Workshop on Linguistically Interpreted Corpora (1999).
[11] Kim Sang, E. F. T., Buchholz, S., "Introduction to the CoNLL-2000 Shared Task: Chunking," in Proc of CoNLL-2000 (2000) 127-132.
[12] 양재형, "규칙기반 학습에 의한 한국어의 기반 명사구 인식", 정보과학회논문지:소프트웨어및응용, 27권10호, pp.1062-71, 2000.
[13] 황영숙, 정후중, 박소영, 곽용재, 임해창, "자질집합선택 기반의 기계학습을 통한 한국어 기본구 인식의 성능향상", 정보과학회논문지:소프트웨어및응용, 29권9호, pp.654-68, 2002.
[14] 김미영, 강신재, 이종혁, "단위 분석과 의존문법에 기반한 한국어 구문분석", 27회 정보과학회 춘계학술발표 논문집, pp.327-29, 2000.
[15] 김광백, 박의규, 나동렬, 윤준태, "구간 분할 기반 한국어 구문분석", 14회 한글및한국어정보처리 학술대회 논문집, pp.163-8, 2002.
[16] Xue, N., Xia, F., The Bracketing Guidelines for the Penn Chinese Treebank, IRCS Repost 00-08 available at http://www.cis.upenn.edu/~chinese/ (2000).
[17] Brill, E., "Transformation-based Error-driven Learning and Natural Language Processing," Computational Linguistics, 21(4) (1995) 543-565.
[18] Ngai, G., Florian, R., "Transformation-Based Learning in the Fast Lane," in Proc of North American ACL 2001 (2001) 40-47.

양 재 형
정보과학회논문지 : 소프트웨어 및 응용
제 31 권 제 1 호 참조

심 광 섭
정보과학회논문지 : 소프트웨어 및 응용
제 31 권 제 1 호 참조