구간 분할 기반 한국어 구문분석

김광백[0], 박의규[◊], 나동렬[◊], 윤준태[+]
연세대 컴퓨터과학과[0], 연세대 정보기술학부[◊], 다음소프트[+]
kbkim@cs.yonsei.ac.kr[0], {ekpark, dyra}@dragon.yonsei.ac.kr[◊], jtyoon@daumsoft.com[+]

A Method of Koean Parsing Based on Sentence Segmentation

Kwang-baek Kim[0], Eui-Kyu Park[◊], Dong-Yul Ra[◊], Joon-Tae Yoon[+]
C.S. Dept.[0], Div. of I.T.[◊], Yonsei University
Daum Soft[+]

요 약

오늘날 자연어 구문 분석 기술은 만족할 만한 수준에 도달하지 못하고 있고 한국어 구문분석 기술 역시 만족할만한 수준과는 거리가 멀다. 특히 문장의 길이가 긴 문장의 경우 구문분석기가 너무 많은 계산 량으로 인해 제대로 동작하지 못하는 경우가 빈번히 발생하고, 비록 구문구조 결과를 내더라도 정확도가 낮은 경우가 많다. 그 이유는 문장의 길이가 길어질수록 중의성이 매우 증가하여 많은 수의 구문분석 결과가 가능하기 때문이다. 이 중에서 정확한 구문구조를 선택하는 문제는 매우 어려워서 기존의 긴 전체 문장에 대한 구문구조를 한번에 계산하려는 시도는 앞으로도 계속 좋은 결과를 기대하기 어렵다. 따라서 우리는 문장의 길이에 상관없이 항상 안정적으로 결과를 내며, 구문분석에 소요되는 시간이 비교적 짧고, 정확도 역시 높은 구문분석기를 개발하고 자 한다. 이를 위하여 전체 문장을 여러 개의 구간으로 분할하여 각 구간을 독립적으로 구문 분석한다. 그 다음 각 구간의 결과를 통합하여 전체 문장에 대한 결과를 생성하는 기법을 택하였다.

1. 서론

파싱이란 문장을 분석하여 문장을 구성하는 각각의 요소와 용언과의 연결관계를 밝히는 작업이라고 할 수 있다. 한국어에 대해서도 많은 파싱 연구와 이론들이 많이 만들어졌고 실험 되어 왔다[1,3~5,9]. 하지만 한국어의 잦은 문장 요소 생략과 자유로운 어순은 완전한 한국어 파서의 개발에 걸림돌이 되어 왔다. 또한 문장이 길어지면 길어질수록, 파서의 결과로써 제시되는 구문 구조의 수가 너무 많아서 좋지 않은 결과를 내는 원인이 되어 왔다.

긴 문장을 여러 개의 단문으로 나누어서 구문 분석을 하고 자 하는 시도가 있었는데[8], 이 기법은 용언마다 그 용언이 취하는 문형을 미리 알고 있어야 한다는 제약을 갖는다. 따라서 본 논문은 논문을 구간이라고 부르는 절단위로 우선 분리한 뒤에, 각 구간에 대하여 완전한 분석(full parsing)을 통하여 가장 좋은 구문구조(파스트리)를 알아낸다. 그 다음 구간들에 대한 구문구조를 합하여 전체 문장의 파스 트리를 알아내는 방식을 취한다.

이러한 방법을 사용함으로써 구문분석의 주요 대상이 되는 구간의 단어의 수를 줄여서 생성 가능한 파스 트리의 수를 줄일 수 있고, 따라서 전체 문장의 구문분석의 복잡도를 감소시켜 보다 안정적인 구문분석기를 구현할 수 있게 된다.

2. 구문 분석 과정

본 논문에서 제시하는 한국어 구문분석기는 4 단계를 가진다.

2.1. 전처리(chunking)

문장을 구성하는 단어들 중에는 서로 연관된 단어와 합쳐서 한 덩어리로 만들어도 무방한 것들이 있다[2,6,11]. 이러한 작업을 보통 구묶음(chunking)이라 한다. 이러한 작업을 하는 이유는 구묶음 과정을 통하여 구문분석의 대상이 되는 단어의 수를 감소시켜 구문분석의 복잡도를 줄일 수 있기 때문이다. 본

논문에서는 다음과 같은 구묶음 전처리 과정을 수행한다.

- 복합명사 처리
복합명사를 구성하는 단어들을 합하여 하나의 단위로 만든다 (단위의 마지막 명사가 헤드가 되어 복합명사를 대표하게 된다.)

- 본용언/보조용언 처리
보조용언들은 자체적으로 독립적인 의미를 갖지 않고 본용언의 의미에 추가적인 성질을 부여하는 역할만 하므로 본용언에 붙인다.

- 의존 명사 처리
"먹을 수 있는 사과" 에서 구묶음 과정을 통하여 "먹을+수+있는 사과" 가 된다. "패배할 것 같은 예감이" 은 구묶음을 통하여 "패배할+것+같은 예감이" 로 된다. 여기에서 "먹을+수+있는" 은 "구문분석기가 볼 때에는 "먹+은" 처럼 보이도록 한다.

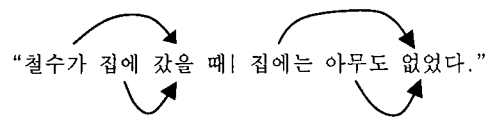
2.2 구간 분할

한국어 문장에서 구문분석이 쉽지 않은 이유는 긴 문장과 한국어의 자유로운 어순에 있다. 따라서 본 논문에서는 문장을 여러 개의 구간으로 나눈 후에 각 구간 단위의 구문분석을 수행하고 구간에 대한 구문분석 결과를 결합하여 전체에 대한 구문분석 결과를 도출하는 기법을 사용한다.

구간은 절과 같은 의미로 사용되며 관형형 어미를 갖지 않은 용언 바로 다음에서 구간이 나누어 지게 된다. 또한 "~때", "~인 이유로" 등 이유, 시간 등을 나타내는 구 바로 다음에서 구간분할을 수행한다.

관형형 어미를 갖는 용언에서 구간을 나누지 않는 이유는 관형형 어미를 갖는 용언 앞의 명사구가 이 관형형 어미를 갖는 용언의 구간을 벗어나 뒤 구간의 용언에 연결되는 경우가 많기 때문이다.

아래 예문에서 문장은 | 에 의하여 두 구간으로 분할되어 있다. 각 구간마다 그 안의 명사구는 자신의 구간 안의 용언에만 연결이 되는 것을 볼 수 있다. 따라서 전체를 한번에 구문 분석하는 대신에 각 구간을 구문 분석한 뒤 그 결과를 결합하여 전체 문장에 대한 구문분석 결과를 구할 수 있다.



이런 방법을 수행 하면 구문분석기의 계산 부담이 많이 감소하게 되는데, 그 이유는 문장의 길이가 길수록 복잡도 및 계산 량의 부담이 큰 폭으로 증가하기 때문이다.

그러나 위 방법으로 구간분할을 한 경우에 한 구간의 명사구가 다른 구간의 용언에 연결되어야만 하는 경우들이 발생한다. 다음 예문을 보자.

"철수의 아버지가 철수를 꾸짖은 선생님께서 지나가시자 | 인사를 드렸다."

여기서 "아버지가"는 첫 구간 에 속해 있다. 그러나 이 명사구는 두번째 구간의 용언과 연결되어야 한다. 첫 구간의 용언들과는 결합되지 않는다. 결국 구간 구문분석에서 이 명사구는 홀로 동떨어지고 구간내의 구문구조의 일부가 되지 못한다.

결국 예문의 "아버지가" 와 같이 구간 분석에서 구간내의 용언과 결합되지 못한 명사구들은 뒷부분에서 설명하는 구간통합 단계를 거쳐 다른 구간의 용언과 결합하는 과정을 거치게 된다.

2.3 CYK 기반 구간 구문분석

이 단계에서는 문장에서 나누어진 각 구간에 대하여 CYK 파싱 기법을 이용하여 모든 가능한 구문구조를 구하고 그 중에서 가장 적합한 구문구조를 선택한다. 이것은 기존의 전체 문장에 대하여 구문분석을 행하던 것을 그보다 길이가 작은 각 구간에 대하여 구문분석을 시행하게 되므로 구간마다에 대한 구문분석 작업의 복잡도가 많이 감소한다.

구간 구문분석 작업은 통계적인 정보를 이용하여 최적의 구문구조를 구하도록 한다. 이용되는 통계정보로는 [명사,조사,용언]의 공기정보를 상호정보 형태로 구하여 이용한다.

2.4 구간통합

각 구간마다에 대하여 구문구조가 구해진 후 전체 문장에 대한 구문구조를 구해야 한다. 보통은 각 구간에 대한 구문구조를 통합하면 된다.

그러나 위 2.2에서도 설명하였듯이 구간 분석 과정에서 같은 구간의 용언과 결합하지 못한 명사구를 다른 구간의 용언과 결합시켜 주는 작업을 수행하여야 한다.

그리고 구간에 대한 구문분석 후 필수격이 빠진 용언에 대하여 다른 구간의 명사구를 이용하여 이 필수격을 채워 주려는 시도를 수행하여야 한다.

"철수가 노는 동생을 꾸짖어 | 학교에 보냈다"

이 문장에서 구간 분석 후에 두번째 구간의 "보냈다"의 필수격인 주어 및 목적어가 빠져 있다. 실제로는 주

어와 목적어가 다른 구간 (첫구간)에 존재하고 있다. 결국 구간 파싱 후에 첫 구간의 "철수가"를 둘째 구간의 "보냈다"의 주격으로, "동생을"을 목적격으로 채워주는 작업을 수행해야 하는데 이는 뒤의 4절의 구간통합과정에서 자세히 알아보기로 한다.

3. 구간에 대한 CYK 구문 분석

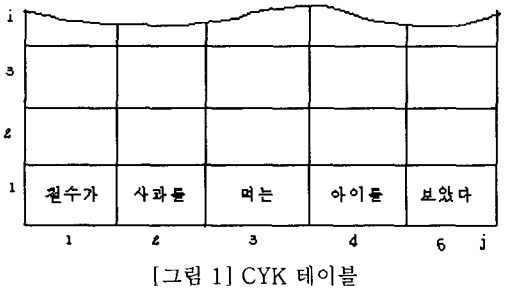
분할된 각 구간에 대하여 CYK 파싱 기법을 이용하여 완전 구문분석을 수행한다. 그 결과로 생성되는 트리 중에서 가장 적합한 것을 선택하여 구간의 구문구조로 한다.

다음의 간단한 예문에 대하여 구간 구문분석이 행해지는 것을 예로 들어 설명하자. (이 문장은 구간이 1개 밖에 없는 문장이다.)

"철수가 사과를 먹는 아이를 보았다."

- CYK 테이블

그림1은 초기의 CYK 테이블을 나타내고 있다. 처음에는 이것은 길이가 1인 구문구조들 즉 각 어절들을 가지고 있는 모습이다. t[i,j] (i^{th} 행, j^{th} 열)는 j번째 단어에서 시작하여 i 개의 단어로 이루어진 구문구조를 나타낸다.



[그림 1] CYK 테이블

각 구문구조는 헤드 단어를 가지고 있다. 헤드는 그 구문구조를 이루는 단어열 내에서 가장 중심이 되는 단어로 정해진다.

- 두 구조의 결합에 의한 새 구조의 생성

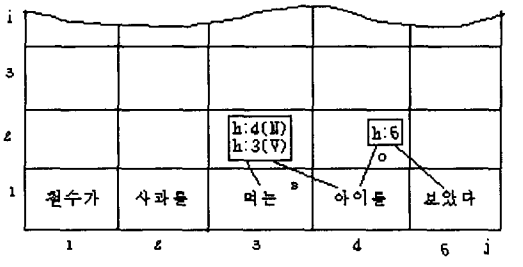
CYK 의 가장 기본이 되는 작업은 두 개의 *인접한* 구문 구조를 합쳐서 하나의 더 큰 (두 구조의 단어열을 합한) 구문구조를 만드는 것이다. 두 인접한 두 구문구조를 각각 C, C' 이라 했을 때 두 구조가 합쳐질 조건은 다음과 같다.

(i) C 의 헤드인 명사가 C' 의 헤드인 용언의 격을 채울 수 있을 경우:
합쳐진 새 구문구조의 헤드는 C'의 헤드이다.

(ii) C 가 관형형어미의 용언이고 C'가 피수식명사구로서 C'의 헤드 명사가 C 의 헤드인 용언의 한 격을 채우는 경우: 헤드는 C 의 용언이 될 수도 있고 C'의 명사가 될 수 도 있다.

(iii) C 가 C' 를 수식하는 경우: 이 경우는 관형사에 의한 명사의 수식, 부사에 의한 용언의 수식, 명사에 의한 명사의 수식 등 다양한 수식현상을 망라한다. 새로운 구조의 헤드는 C'의 헤드이다.

길이가 2인 구문구조는 길이가 1인 (서로 인접한) 2개를 합쳐서 만들며 그 결과는 그림2과 같다. "아이를"이 "보았다"의 목적격(o)을 채워 만들어진 구조가 t[4,2]에 있다. h:5 는 헤드가 단어 5 즉 "보았다"임을 나타내며 o 표시는 이 화살표가 가리키는 것이 목적어 o 가 됨을 나타낸다.



[그림 2] 길이 2인 구문구조

- 관형형 용언의 처리

관형형 어미를 가진 용언은 분석의 복잡성을 일으킨다. 이러한 문제로 인해 용언마다 상태번호를 붙인다. 비관형형 용언은 항상 상태를 1로 둔다. 상태가 1인 용언을 포함한 구조는 항상 상태가 1이며 앞의 명사구와 결합하게 된다.

관형형 어미의 용언은 처음에는 상태가 0으로 주어진다. 상태가 0인 관형형어미의 용언은 반드시 오른쪽에 인접한 구문구조의 헤드가 명사인 구문구조와 결합하여야 한다. 이때 격은 주격,목적격,부사격 모두 가능한 것으로 보아야 한다. 그림2에서는 관형형 어미 '먹는' 이 인접한 구문구조의 헤드인 '아이를' 을 주격으로 채운 경우만을 보여 주고 있으나 실제는 목적격, 부사격에 대한 구조도 t[3,2] 에 존재한다. 이때 피수식 명사구와 결합한 관형형어미의 용언은 상태가 2로 바뀌게 된다.

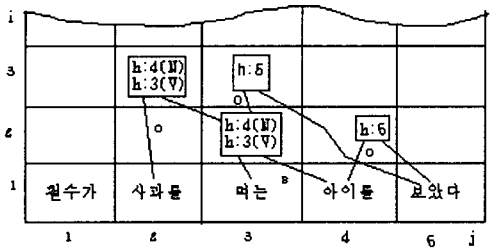
상태가 2인 용언이 포함되어 만들어진 구문구조는 용언으로 동작할 수도 있고 명사구로도 동작할 수 있다. 따라서 각 경우에 대하여 헤드가 다르게 존재한다. 그림2의 t[3,2]에 보여진 구조에서 4(N)이란 헤드가 단어 4 "아이를" 이며 이 구조를 명사로 볼 수 있음을 나타내며 3(V)란 헤드가 단어 3 "먹은" 이며 이 구조를 용언으로 볼 수 있음을 나타낸다.

이 구조가 용언으로 간주되어 앞의 '사과를'과 결합하여 만들어진 구조가 그림3의 t[2,3]에 나타나 있

으며, 명사로 간주되어 뒤의 '보았다'와 결합하여 생긴 구조가 t[3,3]에 나타나 있다. t[2,3] 의 구조는 여전히 명사 또는 용언으로 작용할 수 있음을 나타낸다. 그러나 t[3,3]의 구조는 용언으로만 동작하게 됨을 나타낸다.

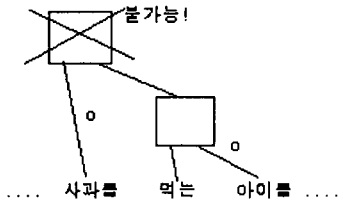
• 격 정보의 활용

명사구에 붙어 있는 조사는 그 명사구가 동작할 수 있는 격이 무엇인지를 나타낸다. 예를 들면 '를'은 목적격으로만 동작할 수 있음을 나타낸다. 의존문법에서는 "용언의 필수격은 오직 하나의 명사구에 의해서 만 채워질 수 있다"라는 "유일 필수격" 원리를 이용하면 많은 중의성을 해소할 수 있다.



[그림 3] 명사 또는 용언으로 동작할 수 있는 구조

앞에서 "먹는 아이를" 에 대한 구문구조로 "아이를"이 주격 뿐만 아니라 목적격, 부사격도 가능하므로 이를 t[3,2]에 만들게 된다고 하였다. 이 경우 목격격으로 된 경우에 대하여 생각하여 보자. 이 구조와 앞의 "사과를"이 목적격으로 합쳐지는 것은 불가능하다. (그림4 참조.) 그 이유는 "아이를"이 이미 "먹은"의 목적격을 채우고 있는데 다시 "사과를"이 용언의 목적격을 채울 수는 없기 때문이다.



[그림 4] 조사의 격정보의 이용

• 격관계에 대한 통계적인 상호정보의 활용

앞에서 "먹는 아이를" 에 대한 가능한 구문구조로 "아이"가 "먹는"의 목적격이나 부사격으로 채운 경우에 대한 구문구조도 만든다고 하였다. 그러나 사실상 이런 경우는 의미적으로 거의 설득력이 없다. 현 시스템이 약간의 의미정보라도 이용할 수 있는 시스템이라면 이런 의미적으로 가능성이 없는 구조를 만들 지 않는 것이 좋다.

그러나 문제는 현재로서 임의의 즉 넓은 분야에 대한 일반 문장에서도 동작할 수 있는 안정되고 신뢰성 있는 의미정보 제공 수단이 없다는 점이다. 하지만 의미정보가 없으면 중의성을 해소할 방법이 없다.

이러한 문제에 대한 대안으로 우리는 명사구와 용언간의 상호 정보(mutual information)를 이용하고 자 한다[7,11]. 명사 N 이 조사 P 로 용언 V 에 이 조사가 나타내는 격으로 연결될 (즉 격을 채울) 가능성의 정도를 나타내는 상호정보는 다음처럼 정의된다.

$$I(N,P,V) = \log \frac{P(N,P,V)}{P(N,P)\times P(V)} = \log \frac{f(N,P,V)\times N}{f(N,P)\times f(V)}$$

f(N, P,V) : N이 P로 V에 연결된 횟수
f(N,P) : 명사구 N+P 가 나타난 횟수
f(V) : 용언 V 가 나타난 횟수
N : 모든 명사_조사_용언 결합의 총 개수

"먹는 아이를"에 대한 3 가지 구문 구조의 상호정보는 다음과 같다:

I("아이", s, "먹다") = 2.7
I("아이", o, "먹다") = -∞
I("아이", a, "먹다") = -∞

(주: s 는 주격에 대한 모든 조사, o는 목적격에 대한 모든 조사, a 는 부사격에 대한 모든 조사를 나타냄.)

여기에서 I("아이", o, "먹다"), I("아이", a, "먹다")는 상호정보 값이 -∞ 인데 그 이유는 주어진 격으로 주어진 명사와 동사가 연결된 경우가 훈련 말뭉치에서 한번도 없었기 때문이다.

따라서 -∞ 인 연결은 만들지 않으면 좋을 것이다. 그러나 문제가 그렇게 간단하지 않다. 통계정보 기반 자연어 처리에서 가장 골치거리인 "데이터 부족" 문제 때문에 통계정보를 액면 그대로 믿을 수 없다는데 문제가 있다. 즉 통계정보가 부족하여 실제로는 일어날 수 있는 연결인데 상호정보에서는 발견되지 않은 경우일 수도 있다는 것이다.

결국 우리는 *비록 통계정보에 따르면 발생하기 힘든 구조라 하더라도 구문구조를 생성한다.* 단, 문법적으로 불가능한 것은 만들지 않는다. 그러나 문법적으로 가능한 것은 비록 의미정보 (여기서는 통계적인 상호정보)가 나쁘더라도 만들어 나간다. (통계적으로 나쁜 의존관계를 이용하여 만들어진 트리는 나중에 통계 값을 비교하여 제거하는 전략을 사용한다.)

따라서 시스템은 각 의존관계마다 구조의 타당성을 나타내는 상호정보 값을 가지고 있게 된다. 상호정보 값이 클수록 그 관계는 가능성이 높은 즉 의미적으로 좋은 관계임을 나타낸다. 본 시스템이 사용하는 통계

정보로는 4000만 어절 말뭉치로부터 추출된 300만 개의 <명사, 격 관계, 서술어> 공기정보 리스트를 이용하여 구해진다. 얼마만큼 정확한 공기정보를 얼마나 많이 확보하는가에 따라서 본 시스템의 성능에 큰 차이가 있게 된다. 앞으로 이 부분에 대한 장기적이고 지속적인 연구 개발이 필요하다.

• 최종 구문구조(트리)의 결정

파싱 후에 각 구간에 대하여 여러 개의 구문트리가 생성된다. 이 중에서 가장 적합한 것을 선택하는 것이 바로 중의성 해소의 핵심이다. 본 시스템에서는 각 트리마다 점수(score)를 부여한다. 트리의 점수는 그 트리를 구성하는 의존관계 (아크)의 상호정보의 합으로 나타낸다.

$$score(T) = \sum_{arc_i \in T} I(arc_i)$$

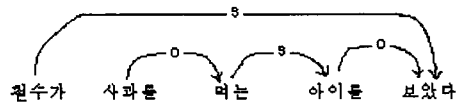
(arc_i : 각 연결(의존관계)을 나타냄.)

구간에 대하여 선택되는 최종트리는 최대의 점수를 갖는 트리이다.

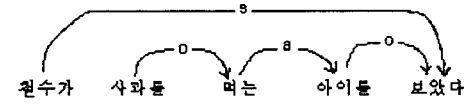
$$T_{chosen} = \arg\max_{T_i} score(T_i)$$

위의 예문에 대하여 최종적으로 가능한 구문 분석 결과는 다음과 같이 3 가지이다 (그림의 복잡성을 피하기 위하여 분석 결과를 위의 CYK 테이블에서 처럼 나타내지 않고 다음 그림과 같이 각 연결을 아크로 나타내었다.)

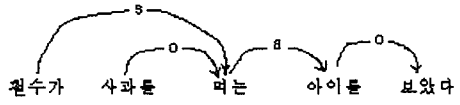
(T1)



(T2)



(T3)



T1 과 T2 의 차이는 "먹는"과 "아이" 사이의 연결의 격이 전자는 s, 후자는 a 라는 차이이다. I(아이,a,먹다) = -∞ 이다. 따라서 I(아이, s, 먹다) > I(아이,a,먹다) 이므로 score(T1) > score(T2). T1과T3 간의 비교에서도 아무리 I(철수, s, 먹다) 가 I(철수, s, 보다) 보다 크더라도 I(아이,a,먹다) = -∞ 인 점 때문에 I(아이,s, 먹다)보다 점수가 낮으므로 score(T1) > score(T3). 결국 T1 의 score 가 가장 크므로 이 문장에 대한 구문구조는 T1 으로 결정된다.

4. 구간 통합

• 구간에서 용언과 결합하지 못한 명사구의 처리:

구간 파싱 후 어떤 구간의 명사구가 용언과 연결되지 못한 경우가 발생할 수 있다. 다음 문장을 보자.

그 지주가 농부들이 일을 마쳤을 때 | 수당을 나누어 주었다.

구간 1의 파싱 결과 "그 지주가"는 용언에 연결되지 못한 상태이다. 이 경우 이 명사구를 뒤의 구간의 필수격이 비어 있는 "나누어 주었다"의 주어로 채워 주어야 한다.

• 용언의 필수격이 비어 있는 경우의 처리:

모든 구간들의 구간 파싱이 종료된 후 어느 구간의 용언이 필수격이 비어 있는 경우가 종종 발생할 수 있다. 이 경우에는 다른 구간의 명사구가 이 비어있는 필수격을 채울 수 있는 경우가 많다. 예를 들어 다음 문장을 보자.

사람들은 산봉우리에서 횃불을 피워 | 연락을 시도했다.

구간 2의 "시도했다"의 필수격인 주어가 비어 있다. 이 경우 구간 1에서 주어로 사용되는 "사람들은"을 가져다가 "시도했다"의 주격을 채우도록 한다.

목적격이 필요한 타동사의 경우 목적격이 비어 있다면 앞 구간에서 목적격으로 사용된 명사구를 가져다가 채우도록 한다. 여기서 문제가 되는 것은 어떤 용언이 타동사인지 판단하여야 하는 점이다. 모든 용언은 주격을 필수격으로 한다. 그러나 목적격을 필수격을 원하는 용언은 타동사뿐 이다. 어떤 용언의 목적격이 비어 있는 경우 이것이 타동사라면 다른 구간에서 가져다가 이 목적격을 채우도록 시도해야 한다.

임의의 용언 (V) 이 타동사인지 판단하기 위해서 우리는 이 용언이 목적어를 취한 경우가 한번이라도 있는지 상호정보테이블에서 찾아 본다. 이런 경우가 있으면 타동사로 간주한다.

위와 같은 처리가 필요한 경우 이러한 작업을 수행한 후 결국 전체 문장에 대한 구문 구조는 각 구간의 구문 구조를 결합함으로써 구해진다.

5. 실험

본 논문에서 제안한 시스템의 초기 프로토타입 버젼이 개발되어 실험 중이다. 본 논문에서 제안한 기법의 성능을 확인하기 위해서는 보다 완벽한 시스템에 대하여 많은 데이터에 의한 실험이 필요하다. 현재의 초기 시스템에 대하여 초등학교 교과서, 신문, 소설 등에서 추출한 예문 127 문장을 가지고 정확률을 측정하여 본 간단한 실험 결과는 표 1 과 같다. 여기서의 정확률은 다음과 같은 식으로 구해진다.

$$\text{정확률} = \frac{\text{파서가 제안한 마크 중 올바른 것의 수}}{\text{파서가 제안한 마크의 수}}$$

[표1] 구간 분할 기반 한국어 구문분석 정확률

	10≥길이	10<길이	장르별 평균
교과서	93%	87%	90.0%
신문	87%	82%	84.5%
소설	91%	85%	88.0%
평균	90.0%	84.6%	87.3%

6. 결 론

길이가 긴 한국어 문장의 구문분석이 매우 어렵다. 계산량이 매우 증가할 뿐만 아니라 중의성이 폭발적으로 증가하여 올바른 구문구조의 결정이 매우 어렵게 된다.

이점을 극복하기 위하여 문장을 여러 개의 구간으로 나눈 다음 각 구간에 대한 구문분석을 수행한다. 구간에 대한 구문구조를 구한 후 이를 통합하여 전체 문장에 대한 구문 구조를 구하는 접근 방법을 취하였다. 그 결과 문장의 길이에 민감하지 않은 구문분석기를 개발할 수 있게 되었다.

실험 결과 본 논문의 기법이 실제 문장에서 잘 적용될 수 있음을 알게 되었다.

7. 참고문헌

[1] 권혁철, 최준영, "단일화 기반 의존문법을 이용한 한국어 분석기", 한국정보과학회논문지, 19권5호, pp 467-476, 1992.9.

[2] 김미영, 강신재, 이종혁, "규칙과 어휘정보를 이용한 한국어 구문 모호성 해결", 제12회 한글 및 한국어 정보처리 학술대회, 성공회대, pp 103-109, 2000.10.

[3] 류범모, 이태승, 이종혁, 이근배, "술어 중심 제약 전파를 이용한 2-단계 한국어 의존 파서", 한국정보과학회논문지 1996 봄 학술발표 논문집, pp 923-926, 1996.4.

[4] 양재형, 김영택, "통계정보를 활용한 한국어 미지격 명사구의 문법기능 결정", 한국정보과학회논문지, 21권5호, pp 808-815, 1994.5.

[5] 양재형, 김영택, "다중 지식원을 이용한 한국어의 분석", 한국정보과학회논문지, 21권7호, pp 1324-1332, 1994.7.

[6] 윤덕호, "숙어 정보를 활용한 한국어 파싱", 서울대학교 박사학위 논문, 1993

[7] 윤준태, "공기 관계 기반 어휘 연관도를 이용한 한국어 구문 분석", 연세대학교 박사학위 논문, 1997

[8] 이현영, 황이규, 이용석, "문형과 단문분할을 이용한 한국어 구문 모호성 해결", 제12회 한글 및 한국어 정보처리 학술대회, pp 116-123, 2000.10.

[9] 홍영국, 이종혁, 이근배, "의존문법에 기반을 둔 한국어 구문 분석기", 한국정보과학회논문지 1993 봄 학술발표논문집, pp 781-784, 1993.4.

[10] Juntae Yoon, Seonho Kim, and Mansuk Song, New Parsing Method Using Global Associaiton Table, *Advances in probabilistic and other parsing technologies*, Text, Speech and Language Technology , vol 16, pp 125-139, 2000

[11] Juntae Yoon, Efficient Dependency Analysis for Korean Sentence Based on Lexical Association and Multi-layered Chunking, Literary and Linguistic Computing, vol 16(3), pp 265-285, 2001