하위범주화에 의한 한국어 파서의 설계와 구현 : I

이 호 석
뉴미디어학과 공과대학 호서대학교
hslee@office.hoseo.ac.kr

A Design & Implementation of Korean Parser using Subcategorization: I

Ho Suk Lee
New Media Dept. College of Engineering Hoseo University

요 약

본 논문에서는 의존 문법, 하위범주화, 그리고 조사와 어미의 분석과 처리에 기반 한 한국어 파서를 제시하고 논의한다. 의존 문법과 하위범주화는 BNF(Backus Naur Form)를 확장한 형식을 사용하여 정의하였다. 논문에서 한국어 파서의 개념적 기본 구도를 C 프로그램 형식을 사용하여 나타내었다. 현재 구현된 한국어 파서의 구성을 설명하고 실행결과를 보여준다.

Abstract

We present and discuss a Korean language parser based on dependency grammar, subcategorization, and the analysis of viable postfix such as josa and omi. We employ an extended form of BNF(Backus Naur Form) to define the dependency grammar and the form of subcategorization. We present the conceptual form of Korean language parser in a C program style. We discuss the structure of Korean parser currently implemented and show the execution results.

1. 서 론

한국어에 대한 언어 이론적 연구는 매우 발전되어 있다[1][6][19]~[21]. 근래에는 국립국어원에 의하여 말뭉치를 사용한 실증적 연구도 잘 이루어져 있다[12]~[25]. 한국어의 중요한 구문적 특징은 다음과 같이 제시할 수 있다[1]. (1) 국어는 문장의 주요 근간 성분의 생략이 가능한 언어이다. 즉, 논항이 문맥적 상황에 따라 공범주(empty category)로 실현되는 경우가 많다. (2) 주어가 잇달아 나타나는 문장 구성 방법이 있다. (3) 주어가 없는 문장도 많이 쓰인다. (4) 조사와 어미가 매우 발달하여 다른 유형의 언어에서는 부사나 기타 독립된 어휘로 나타나야 할 의미가 조사나 어미의 형태로 나타난다. 조사의 첨가나 어미의 활용에 의하여 어감의 차이가 구현된다. (5) 문장의 통사적 서법이나 절의 통사적 자격이 어미에 의하여 외형적으로 구현된다. 즉, 어미에 의하여 문장의 의미가 결정된다. 또한 어미의 종류가 매우 많아서 다양한 의미로 두 개의 문장을 접속시킬 수가 있다. (6) 의존 명사가 널리 쓰인다. (7) 한자어에서 온 낱말이 매우 많아서 한국어는 뜻글자인 면이 많다. (8) 문장에서 능동과 수동의 의미가 혼용되어 사용되는 경우가 많다. (9) 형용사가 동사와 거의 비슷한 문법적 역할을 한다. (10) 문장의 기본 구성 성분의 순서는 핵말(head-final) 언어로서 모든 구와 절에서 핵은 뒤에 나타나며, 문장 성분의 순서는 엄격히 정해져서 “주어+목적어+동사”의 순서를 가진다. (11) 문장에 대한 어휘 분석과 구문 분석을 수행할 때, 의미 분석을 함께 고려하는 것이 좋다.

본 논문에서는 이러한 한국어의 특징을 처리할 수 있도록 의존 문법, 하위범주화(subcategorization), 그리고 조사와 어미의 기능에 기반을 둔 새로운 한국어 파서를 구현하고 실행 결과를 제시한다. 한국어 문법에서 논의하는 내용을 그대로 소프트웨어로 구현하는 것이 바람직하다고 판단하였다.

2. 관련 연구

한국어 구문 분석은 오랜 기간 동안 학계의 관심을 받아왔다. 90년대 초중반에는 주로 영어 분석을 위하여 개발된 이론들을 한국어에 적용하는 연구가 많았다고 할 수 있다. 그러나 의존 문법이나 하위범주화 개념이 적용된 연구도 있었다[2][3][4]. 참고 문헌 [5]에서는 이제까지의 주요 한국어 구문 분석 방법들에 대하여 논의하고 평가하였다. 영어를 대상으로 개발된 이론들은 한국어에 적합하지 않다고 설명하였으며 한국어의 문법적 특징을 고려한 구문 분석 방법론을 제시하였다. 특히, 조사와 어미들에 대하여 자세하게 분석하여 제시하였으며 한국어 구문 분석기의 구현에 필요한 중요한 원칙들도 제시하였다. 참고 문헌 [6]에서는 21세기 세종계획의 일환으로 550만 어절의 한국어 말뭉치를 구축하여 어휘 빈도를 조사하였다. 국립국어원은 21세기 세종계획의 일환으로 한국어 말뭉치를 구축하고 컴퓨터를 사용한 한국어 분석 연구를 수행하였으며 형태소 분석기와 구문 분석기를 개발하여 실행 파일은 연구용으로 제공하고 있다[7][8].

또한 비교적 최근에는 LR 파싱 기법을 개선한 GLR(Generalized LR) 파싱 기법을 한국어 파싱에 적용한 주목할 만한 연구가 발표되었다[9]. 본 논문에서 논의하는 연구와의 차이점은 본 연구에서는 의존 문법과 하위범주화를 함께 고려하였다는 것이다. 참고 문헌 [9]에서는 의존 문법은 고려하였으나 하위범주화 개념을 고려하지 않았다. 대신에 명사구에 6개 그리고 동사구에 5개의 표층 구문 타입(surface phrasal type)을 정의하여 파싱 테이블을 구성하여 사용하였다. 그러나 하위범주화 개념이 국어 문법적인 관점에서 더욱 적합하다고 생각한다. 왜냐하면, 하위범주화 개념을 사용하면 국어 동사와 형용사의 문법 현상을 하위범주화 개념만을 사용하여 일관되게 설명할 수 있기 때문이다. 하위범주화 개념은 대부분의 국어 문법책에 언급되어 있으며 참고문헌 [5]에도 자세하게 언급되어 있다. 국어 연구에 있어서는 일반적으로 언어적 상세함(linguistic detail)을 연구한다. 즉, 낱말 형태와 구성의 상세함, 구문의 상세함, 의미와 그에 따른 사용의 상세함을 연구한다. 따라서 국어 연구에 있어서는 표층 구문 타입이라는 복합적인 개념을 정의하여 사용하지 않는다. 참고 문헌 [9]를 보면 PX 단위코드는 명사구와 동사구에 모두 포함되어 있다. 아마 이 개념은 프로그래밍 언어를 위한 LR 파싱 기법을 국어 파싱에 적용하기 위하여 고안해낸 개념으로 생각된다. 본 연구에서는 파싱 테이블은 사용하지 않고 파서의 상태를 나타내는 상태 변수를 사용하였다.

그 밖에 국어의 하위범주화 연구와 관련된 참고 문헌에는 [10][11][12][13] 등이 있다. 참고 문헌 [10][11]에는 국어에서 하위범주화의 중요성이 잘 설명되어 있다.

3. 본 론

3.1 의존 문법, 하위범주화, 조사와 어미

본 논문에서는 의존 문법과 하위범주화 형태는 확장된 BNF 형식을 사용하여 정의하여 보았다. 정의된 BNF는 개념적인 것으로서 참고 문헌 [14] 389쪽에 있는 약 30개의 국어 기본 문형을 하위범주화 개념을 사용하여 BNF 형식으로 수정하여 표현한 것이다. 하위범주화를 사용하면 국어 서술어와 구문의 특징을 간결하게 나타낼 수 있다. BNF 정의에서 SENT는 문장(sentence)을 의미한다. S는 하위범주화를 의미한다. 문장은 제일 상위 문법 범주에서는 하위범주화 단위의 연속으로 생각하며 마지막 하위범주화 단위의 종결형어미에 의하여 종결된다. 하위범주화는 형용사, 자동사, 그리고 타동사에 대하여 정의하였다. 형용사의 경우에는 1가 형용사와 2가 형용사가 있는 것으로 간주하였다. 자동사의 경우에는 1가, 2가, 3가 자동사가 있는 것으로 간주하였으며, 타동사의 경우에는 2가, 3가, 4가 타동사가 있는 것으로 간주하였다[5]. 하위범주화는 마지막에 위치한 활용어의 어미를 전달받는다. NKP(noun phrase kyuk)는 명사구를 의미하며 NKP는 명사구의 마지막 위치에 있는 명사의 조사를 격(kyuk)으로 전달받는다. 그러나 (27)번 NKP의 경우는 예외로서 처음에 위치한 명사의 조사를 전달받는다. (26)번은 수 표현을 나타내고 (24)(25)과 (27)(28)번은 양사 표현을 나타낸다. DEP(dependency)는 수식 관계를 나타낸다. 조사와 어미는 viable postfix 로서 구문 분석 과정에서 상위의 문법 범주로 전달되어 하위범주 정보를 전달한다.

(1) SENT(종결형어미) -> S^{+}/종결형어미
(2) S(어미) -> NKP/조사 + ADJ/어미
(3) S(어미) -> NKP/조사 + NKP/조사 + ADJ/어미
(4) S(어미) -> NKP/조사 + Vi/어미
(5) S(어미) -> NKP/조사 + NKP/조사 + Vi/어미
(6) S(어미) -> NKP/조사 + NKP/조사 + NKP/조사 + Vi/어미
(7) S(어미) -> NKP/조사 + NKP/조사 + Vt/어미
(8) S(어미) -> NKP/조사 + NKP/조사 + NKP/조사 + Vt/어미
(9) S(어미) -> NKP/조사 + NKP/조사 + NKP/조사 + NKP/조사 + Vt/어미
(10) NKP(조사) -> N/조사
(11) NKP(조사) -> ADJ/관형격어미 + N/조사
(12) NKP(조사) -> Vi/관형격어미 + N/조사
(13) NKP(조사) -> Vt/관형격어미 + N/조사
(14) NKP(조사) -> Det + N/조사
(15) NKP(조사) -> Det + ADJ/관형격어미 + N/조사
(16) NKP(조사) -> Det + Vi/관형격어미 + N/조사
(17) NKP(조사) -> Det + Vt/관형격어미 + N/조사
(18) NKP(조사) -> DEP(관형격어미) + N/조사
(19) NKP(조사) -> S(관형격어미) + N/조사
(20) NKP(조사) -> N/와 N/조사
(21) NKP(조사) -> N/과 N/조사
(22) NKP(조사) -> N/의 N/조사
(23) NKP(조사) -> N 그리고 N/조사
(24) NKP(조사) -> N + Number + 의존명사 + Quant/조사
(25) NKP(조사) -> Number + 의존명사 + Quant/조사
(26) NKP(조사) -> Number + 의존명사 + 접사/조사
(27) NKP(조사) -> N/조사 + Number + 의존명사 + Quant
(28) NKP(조사) -> N + 수 + Number + Quant/조사
(29) Number -> 0|1|2|...|9|십|백|천|만|억|조|경
(30) Quant -> 명|개|그루|마리|벌|...|
(31) DEP(어미) -> ADJ/부사격어미 + Vi/어미
(32) DEP(어미) -> ADJ/부사격어미 + Vt/어미
(33) DEP(어미) -> ADV + Vi/어미
(34) DEP(어미) -> ADV + Vt/어미
(35) DEP(어미) -> ADJ/부사격어미 + ADJ/어미
(36) DEP(어미) -> ADV + ADJ/어미
(37) Det -> 그|이

앞에서도 언급하였듯이 한국어 문장은 기본적으로 하위범주화 단위의 연속으로 인식하여 BNF 문법으로 표시하였다.

3.2 자료 구조

자료 구조[15]은 단어에 대한 언어적 정보를 나타낸다. 다음은 C 프로그램[16][17] 형식으로 표현한 단어와 스택에 대한 자료 구조의 일부분이다.

```
struct words {
        char    *POS;
        char    *stem;
        int      affix;
        char    *josa;
        words   *next;
};

...

struct stack {
        int      cnt;
        words   *top;
};
```

3.3 파싱 프로그램 설계

파싱은 지역 처리(local processing)를 고려하며 스택(stack)을 사용하여 구현한다. 파싱 결과는 수식 관계를 나타내는 트리로서 표현된다. 한국어 파싱에서 조사와 어미는 viable postfix 로서 지금까지 처리된 문장 성분의 구성이 완성되었다는 것을 의미한다. 조사는 문장에서 명사구의 완성을 의미하고 어미는 수식 관계 혹은 하위범주화 단위의 완성을 의미한다. 스택에는 관형사 스택, 명사구 스택, 관형형 스택, 부사형 스택 등을 생각할 수 있다. 스택에는 단어의 포인터가 저장된다. 포인터는 단어 혹은 지금까지 파싱되어 구축된 문장 성문을 가리킨다. 즉, 구축된 파싱 트리가 명사구이면 명사구 스택에 주소가 저장된다. 구축된 파싱 트리가 관형형이면 관형형 스택에 저장되고 구축된 파싱 트리가 부사형이면 부사격 스택에 저장된다. 스택들은 의존 문법의 관점에서 수식하는(의존하는) 요소들을 저장하는 곳이다. 수식되는(의존되는) 요소들은 입력에 존재한다.

파싱은 기본적으로 상향식(bottom-up) 입력/축소(shift-reduce) 방식으로 수행된다[18]. 즉, 입력된 단어를 하나씩 읽으면서 품사에 따라 해당되는 스택으로 입력한다. 단어를 스택으로 입력하는 과정에서 BNF 의존 문법으로 인식되는 부분이 있으면 해당 문법으로 축소되는 것으로 생각할 수 있다. 축소되면서 인식된 부분의 마지막 부분에 있는 조사와 어미를 확인하여 축소된 문법 범주(category)의 핵말 요소로 간주한다. 또한 이 과정에서 문장에 존재하는 동사와 형용사를 인식하여 하위범주화 정보를 사전에서 출력한다. 출력된 하위범주화 정보를 스택에 저장된 조사 성분과 비교한다. 그리고 스택에서 출력된 조사와 활용어의 하위범주화 정보를 비교하여 하위범주화 단위를 구성한다. 구성된 하위범주화 단위는 그 어미의 형태에 따라 해당되는 스택으로 입력한다. 하위범주화 단위는 완전한(full) 단위와 부분적인(partial) 단위를 생각할 수 있다. 스택에서 확인이 되지 않는 조사 성분은 생략된 문장 성분으로 간주할 수 있다. 문장의 파싱은 문장을 구성하는 단어를 읽으면서 최종 동사나 형용사의 종결 어미를 만날 때까지 계속되고 종결 어미를 만나면 종료된다. 다음은 상향식 입력/축소 방식의 한국어 파서 프로그램의 개념적 기본 구도이다.

```
// Korean parsing program

parsing() {
  input( );
  syntax_analysis( );
  output( );
}

syntax_analysis( )
{
  int length;
  for(i=0; i<length; i++) {
      switch(words) {
          case (word1->pos == noun)
              proc_noun(word1, word2);
          case (word1->pos == depnoun)
              proc_depnoun(word1, word2);
          case (word1->pos == prefix)
              proc_prefix(word1, word2);
          case (word1->pos == postfix)
```

```
            proc_postfix(word1, word2);
         case (word1->pos == adjective)
            proc_adj(word1, word2);
         case (word1->pos == adverb)
            proc_adv(word1);
         case (word1->pos == iverb)
            proc_verb(word1, word2);
         case (word1->pos == tverb)
            proc_tverb(word1, word2);
         case (word1->pos == conjunctive)
            proc_conj(word1);
         case (word1->pos == number);
            proc_number(word1);
         case (word1->pos == quant)
            proc_quant(word1);
      }
   }
}

proc_noun(word1, word2)
{
   josa_k = josa(word2);
   push(NKP);
   p_state(noun);

// BNF_grammar_trace(p_state, josa_k);

// 하위범주화를 위한 NKP 처리

   switch(josa_k) {
      case (주격) :
      case (주제격) :
      case (보격) :
      case (목적격) :
      case (복수들조사) :
      case (처소격) :
      case (도구격) :
      case (출격) :
      case (탈격) :
      case (향격) :
      case (한달점격) :
      case (공격) :
      case (비교격) :
      case (자격격) :
      case (원인격) :
      case (변위격) :
      case (변성격) :
      case (결과격) :
      case (바탕격) :
      case (호격) :
      case (화격) :
      case (인용격) :
      case (보조사) :
      case (접속조사) :
      default : {
         . . .
         build_NKP(word1);
         push(NKP);
         p_state("NKP구성");
         . . .
         break;
   }
   return;
}

proc_adj(word1, word2)
{
   adj_omi = adj_omi(word2);

   // subcategorization 처리
      . . .

   switch(adj_omi) {
      case(adj_omi == 관형격어미) {
            if (adj_subcattree)
               push(adj);
               . . .

      }
      . . .
      case(adj_omi == 연결형어미) {
            if (adj_subcattree) {
               push(adv);
               . . .
            } else error( );
      }
      case(adj_omi == 종결형어미) {
            pop( );
               . . .
               . . .
            build_parsetree(adj_omi);
         }
      }
   }
}

build_NKP(unit1, NKP)
{
   make a NKP;

build_full_subcategorize( )
{
   make a tree using the predicate as a root
   with all of the subcategorization elements present;
}

build_partial_subcategorize( )
{
   make a tree using the predicate as a root
   with some subcategorization elements absent;
}

build_parsetree( )
{
   make a parse tree using all of the subcategorization
   units comprising the sentence;
}
```

3.4 프로그램 구조 및 실행 결과

자연언어 처리 프로그램에 있어서는 입력되는 자연언어 문장을 제한할 수 없다. 파싱 프로그램은 전체적으로 명사를 처리하는 부분과 서술어를 처리하는 부분으로 구성되어 있다. 명사를 처리하는 부분에 상태를 처리하는 부분이 있다. 서술어를 처리하는 부분에서는 형용사, 타동사, 자동사를 모두 처리한다. 하위범주화 개념을 사용하였기 때문에 형용사와 동사의 구분은 없다. 현재는 문장의 앞에 반드시 주어가 있어야 한다. 주어가 생략되는 경우는 다루지 않고 있다. 구문 분석의 중의성도 아직 다루지 않고 있다. 현재의 파서에서 다음 문장들을 실험하여 보았다. (1)~(8) 문장에서 동사와 형용사는 모두 하위범주화 개념에 의하여 파싱을 한다. 하위범주화 정보는 각 서술어 항목의 사전으로부터 제공된다.

(1) 나는 유명한 사람을 본다.
(2) 나는 저 유명한 사람을 잘 본다.
(3) 나는 공을 잘 차는 영희를 본다.
(4) 그는 재주가 매우 훌륭하다.
(5) 나는 얼굴이 매우 유명하고 아름다운 사람을 본다.
(6) 그는 유명하지 않다.
(7) 그는 책과 연필을 본다.
(8) 그는 훌륭한 책과 연필을 본다.

다음 (그림 1)은 현재 구현되어 있는 파서의 실행 결과이다. 입력 문장은 “나는 공을 잘 차는 영희를 본다.”이다. 출력 결과는 하위범주화 트리이다. 하위범주화는 타동사 “본다”를 트리 루트로 한다. 이 트리 구조는 두 개의 하위범주화 구성 성분 “나는”과 “영희를”를 취하고, “영희”를 수식하는 것으로는 또 다른 관형형 하위범주화 “공을 차는”과 부사 “잘”이 있다. 다음 (그림 2)는 문장 “그는 재주가 매우 훌륭하다.”의 파싱 결과이고, (그림 3)은 문장 “그는 책과 연필을 본다.”의 파싱 결과이다. 그리고 주어가 생략된 문장에서 주어를 삽입하여 처리할 수 있도록 하였다. 중의성은 현 단계에서 고려하지 않고 있으며 마지막 단계에서 고려할 생각이다.

4. 결 론

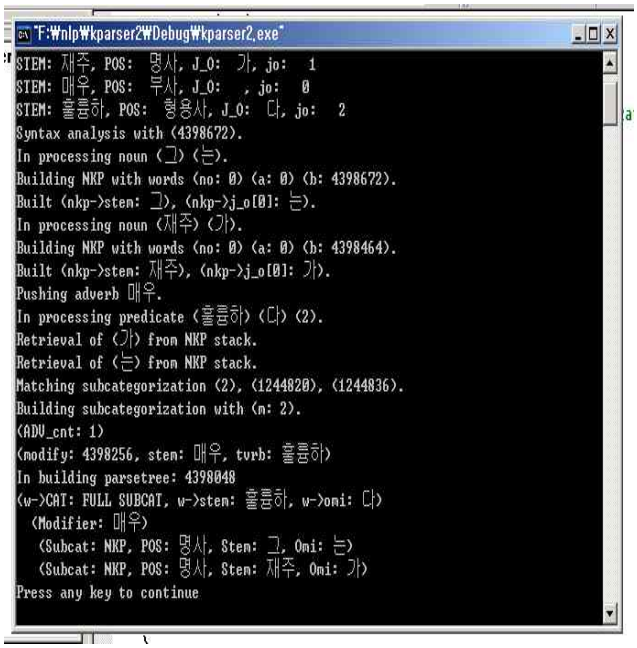
한국어에 대한 언어 이론적 연구는 다른 학문 분야의 연구 이상으로 매우 발전되어 있으며 이미 많은 저작들이 출간되어 있다[1][19][20][21]. 더욱이 근래에는 국립국어원에 의하여 말뭉치를 사용한 실증적 연구도 잘 이루어져 있다[6][22]~[25]. 한국어 구문 분석에 대하여서는 참고 문헌 [5]를 먼저 보는 것도 좋을 것이다.

본 논문에서는 한국어의 특징을 인식하여 의존 문법, 하위범주화, 그리고 조사와 어미의 분석에 기반을 둔 새로운 한국어 파서 프로그램을 구현하여 실행 결과와 함께 제시하였다. 파서의 구현과 실험을 통하여, 본 논문에서 논의한 하위범주화 개념이 한국어 파서의 구현에 가장 적합한 것으로 검증되었다고 생각한다. 왜냐하면

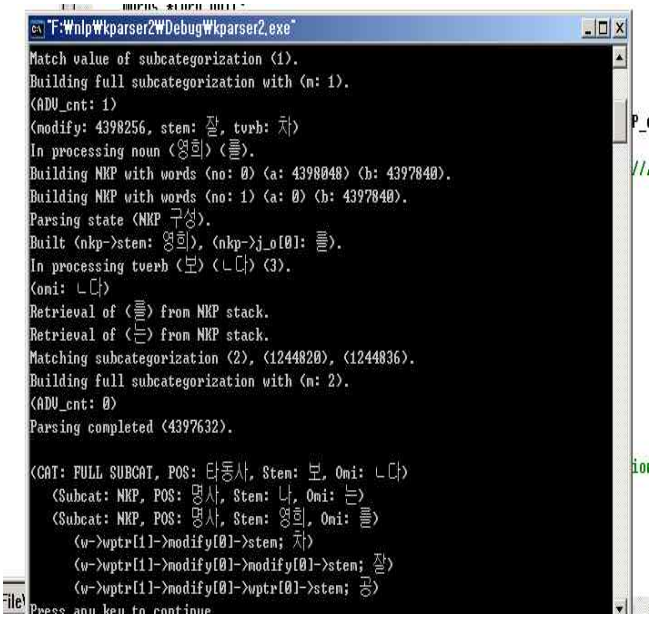
한국어 서술어와 구문의 전반적인 특징을 하나의 개념을 사용하여 나타낼 수 있기 때문이다. 현재 한국어 파서의 기본 내용과 구조를 완성하였다고 생각한다. 앞으로의 연구 계획은 한국어 파서를 계속 개발하여 최종적으로 완성하는 것이다.

참고 문헌

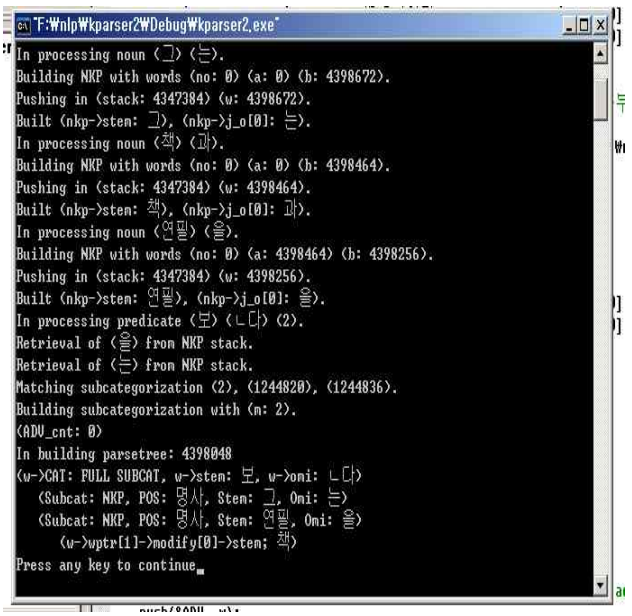
[1] 김상대, 국어 문법의 대안적 접근, 국학자료원, 2001.
[2] 박용호, 외2인, "의존 문법을 이용한 한국어 분석기의 구현," 1990년도 한국정보과학회 봄 학술발표대회논문집, Vol. 17, No. 1, 1990년.
[3] 류법모 외 3인, "술어 중심 제약 전파를 이용한 2-단계 한국어 의존 파서," 1996년도 한국정보과학회 봄 학술발표대회 논문집, Vol. 23, No. 1, 1996년.
[4] 이상국 외 3인, "용언의 하위범주화 정보를 이용한 특수 문형의 처리 방안," 1993년도 한국정보과학회 가을 학술발표대회논문집, Vol. 20, No. 2, 1993년.
[5] 임홍빈, 이홍식 외, 한국어 구문 분석 방법론, 한국문화사, 2003.
[6] 강범모, 김흥규, 한국어 형태소및 어휘 사용 빈도의 분석 2, 고려대학교 민족문화연구원, 2004.
[7] 국립국어원, 한국어형태소분석기, 21세기 세종계획, 2006.
[8] 국립국어원, 한국어구문분석기, 21세기 세종계획, 2006.
[9] 곽용재 외 5인, "표층 구문 타입을 사용한 조건부 연산 모델의 일반화 LR 파서," 한국정보과학회논문지 :소프트웨어및 응용, 제 30권, 제 1호, 2003.2.
[10] 양승현 외 3인, "시소러스 술어 패턴을 이용한 의미역 부착 하국어 하위범주화 사전의 구축," 한국정보과학회논문지: 컴퓨팅의 실제, 제 6권, 제 3호, pp.364-372, 2000.
[11] 양재형, 심광섭, "시소러스 하위범주화 사전을 이용한 격 모호성 해결," 한국정보과학회논문지(B), 제 26권, 제9호, 1999.
[12] 이수선 외 2인, "하위범주화 사전의 구축 및 자동확장," 한국정보과학회 가을 학술발표논문집, Vol. 27, No. 2, 2000.
[13] 이호석, "하위범주화에 의한 한국어 파싱 설계," 한국컴퓨터종합학술대회 논문집, Vol. 35, No. 1(C), 2008.
[14] 이광정, 국어문법연구 i 품사, 도서출판 역락, 2003.
[15] Ellis Horowitz, et al., Fundamentals of Data Structures in C, Computer Science Press, 1993.
[16] Al Kelley, Ira Pohl, A Book on C (4th ed.), Addison-Wesley, 1998.
[17] Brian W. Kernighan, Dennis M. Ritchie, The C Programming Language (2nd ed.), Prentice-Hall, Inc., 1988.
[18] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Compilers Principles, Techniques, and Tools (2nd ed.)(한국어판), 서고당, 1995.
[19] 박창해, 현대 한국어 통어론 연구, 연세대학교출판부, 2007.
[20] 임홍빈, 한국어의 주제와 통사 분석, 서울대학교출판부, 2007.
[21] Chong-Hoon Chun, Towards a Theory of Morphopragmatics of Korean Connectives, 도서출판 박이정, 2007.
[22] 국립국어원, 21세기 세종계획 말뭉치와 활용 도구, 2006.1.
[23] 국립국어원, 21세기 세종계획 균형 말뭉치, 2006.1.
[24] 김한샘, 현대 국어 사용 빈도 조사 2, 국립국어원, 2005.
[25] 김한샘, 한국 현대 소설의 어휘 조사 연구, 국립국어원, 2003.



(그림 1)



(그림 2)



(그림 3)