형태소 분석 주도의 한국어 복합동사 처리

이기오, 김기철, 이용석
전북대학교 전자계산학과

(Morphological Analysis Driven Processing of Compound Verbal in Korean)

Gi-O Lee, Ki-Cheol Kim, Yong-Seok Lee
Dept. of Computer Science, Chonbuk National University

요약

복합동사의 처리는 기계번역이나 자연어 이해 시스템의 질에 상당한 영향을 주기 때문에 복합동사의 정확한 분석과 처리는 중요하다. 기존의 형태소 분석에서는 복합동사에 대한 처리를 복합동사를 구성하는 각 용언들에 대한 분석결과를 생성하여 주고 구문분석단계에서 문법규칙을 이용하여 처리함으로써 문법이 커지고 파싱테이블과 심볼테이블이 커져 메모리 효율이 저하되고 형태소 분석에서의 품사 모호성이 구문분석단계에 영향을 주어 구문구조 모호성을 야기하며 복합동사의 정확한 의미를 나타내기 어려운 문제를 가진다. 본 논문은 한국어의 복합동사를 형태소분석단계에서 처리해 주고 여기에서 처리하기 힘든 복합동사는 사전과 구문분석단계 등의 모듈에서 처리하는 총괄적인 복합동사 처리방법을 제안한다.

1. 서 론

최근 자연어 처리에 대한 연구가 관심을 모으면서 기계번역이나 자연어 이해 시스템 등의 응용분야에 대한 연구가 활발히 진행중이다. 기계번역이나 자연어 이해 시스템 등에 사용하기 위해서는 먼저 대상 언어에 대한 정확한 분석이 필수적으로 요구된다. 한국어 처리에서 분석의 중심이 되는 성분은 용언(inflected word)이다. 용언은 문장의 주체를 서술하는 기능을 띄고있는 말로 여기에는 사물의 동작을 설명하는 동사(verb)와 사물의 성질이나 상태를 설명하는 형용사(adjective)가 있다. 한국어의 특징 중의 하나는 한 문장에 이런 용언들이 연속적으로 나타날 수 있다는 것인데 이렇게 연속적으로 나타나는 용언들을 복합동사(compound verbal)라 부른다. 복합동사가 한국어에서 차지하는 비중은 매우 크기 때문에 복합동사를 정확하고 효율적으로 분석하는 것은 매우 중요하다 할 수 있다. 복합동사에 관한 연구는 주로 국문학에서 수행되어 왔는데 이들 연구에서는 복합동사를 이은말과 합성동사로 구별하여 이들중 합성동사의 생성 원리와 형성 규칙을 밝히려는 시도였다[4,5]. 또한 언어학에서 일부 형태의 복합동사를 LFG로 표현하려는 시도도 있었지만[1] 이들 연구의 결과는 전산언어학에서 응용하기에는 적합하지 못하고 전산언어학적으로 복합동사를 정확히 처리하는 것은 결코 쉽지않은 작업이기 때문에 복합동사 처리에 관한 연구는 미진하다. 본 논문에서는 복합동사를 용언들이 연속적으로 나와 선행 용언의 어말 어미를 매개로 하나의 서술어 기능을 하는 용언들외에 용언들 사이에서 '-ㄹ 수'를 매개로 연결되어 하나의 서술어 기능을 하는 용언들을 포함한다.

최근 자연어 처리에서 활발하게 연구되는 기계번역 시스템과 자연어 이해 시스템 등의 한국어 처리에서 복합동사는 많은 문법적 기능을 담당하고 화자의 양태적 관계를 나타내므로 이들에 대한 정확한 분석과 처리는 번역의 질과 자연어 이해에 상당한 영향을 미치므로 복합동사를 정확하고 효율적으로 처리하지 못한다면 좋은 결과를 획득하기 어렵다[11].

복합동사에 관한 기존의 시스템에서의 처리를 살펴보면 일반적으로 형태소 분석 단계에서 분석된 형태소와 그에 따른 자질구조를 생성하여 주고 파싱 단계에서 파서가 주어진 결과를 가지고 문법을 이용하여 처리하는 파서 주도의 방법이다. 그러나 이 방법은 여러가지 유형의 복합동사를 처리하기 위하여 부가적으로 많은 문법이 요구되므로 기억공간과 시간의 낭비가 많고 또한 복합동사가 나타내고자 하는 의미를 정확하게 표현하는 것이 쉽지 않은 단점을 가진다. 본 논문은 복합동사를 가능한 한 형태소 분석 단계에서 처리하고 여기에서 처리하지 못하는 복합동사는 파서와 사전을 이용하여 처리하는 형태소 분석 주도의 총괄적인 복합동사 처리 방법을 제안한다.이를 위하여 기존의 형태소 분석 방법으로는 형태소 분석 주도로 처리할 수 없으므로 기존의 형태소 분석 방법을 확장시킨 형태소 분석 방법을 제안하고 본 논문의 방법이 복합동사를 효율적이고 정확하게 처리할 수 있음을 보인다.

2. 한국어 복합동사의 분류

복합동사는 독자적인 의미를 갖는 두개 이상의 단어가 결합하여 하나의 동사 역할을 하는 품사이다. 한국어에서 복합동사는 매우 빈번하게 나타나기 때문에 복합동사를 정확하고 효율적으로 분석하고 처리하지 못하면 기계번역 시스템에서 고품질의 번역 결과를 획득하는데 어려움이 있다. 복합동사를 효율적으로 처리하기 위해서는 먼저 복합동사를 형성하는 유형을 세밀하게 분류하는 일이 선행되어야하고 그런 다음, 각 유형별로 적절하게 처리하여야 한다. 현재까지 복합동사에 관해 명확히 분류된 것은 아니지만 국문학적 관점에서 한국어에는 크게 다음과 같은 세가지 형태의 복합동사가 존재한다[4,5,6,7,8].

1) 명사(N) + 동사(V) (샘솟다, 바람나다, 공부하다, ...)
2) 부사(ADV) + 동사(V) (가로막다, 다하다, 잘나다, ...)
3) 동사(V1) + 동사(V2) (돌아가다, 파고들다, 쓰디쓰다, 내다보다, 껴안다, 여닫다, ...)

이중 1), 2)형태의 복합동사는 특별한 규칙이 존재하지 않고 생산성도 높지 않으므로 전산언어학적 관점에서 이 형태의 복합동사들은 사전에 등록하여 처리하는 것이 타당하다. 단, 1)의 경우 중 동사가 '하다'인 경우는 매우 많은 복합동사가 생성되고 '하다'가 명사의 원래 의미에 '진행' 또는 '상태'를 나타내는 역할을 수행하므로 이 복합동사들은 사전에 기술하지 않고 해당하는 명사에 '하다'와 결합할 수 있음을 나타내어 주어 형태소 분석 단계에서 하나의 용언으로 취급하면 효율적으로 처리할 수 있다. 특히 '하다'는 의존명사 + '-체', '-듯', + '-양' 따위에 붙어 보조 형용사나 보조 동사를 만들어주기 때문에 이들 경우는 형태소 분석 단계에서 처리하는 것이 타당하다[9]. 그러나 3)의 형태는 생산성이 높고 여러가지 유형이 존재하기 때문에 처리하기가 쉽지 않다[5,7]. 국문학에서는 3)의 형태에 대하여 아래와 같은 세가지 분류 기준을 가지고 통사적인 이은말과 합성동사를 구별하여 이들 유형에 대한 판별이 끝나면 생성 규칙을 발견하려고 시도하였다[5].

첫째, 'V1 + -아/어'와 'V2'사이에 '서'를 넣어서 의미가 통하면 접속법이고 그렇지 않으면 자격법으로 판별한다.
둘째, 대동사화와 대등접속 등의 통사론적 검증법을 거친다.
세째, 의미론적 기준을 사용한다.

그러나 국문학에서의 분류 기준은 사람이 지니고 있는 풍부한 지식을 이용하여 판단하기 때문에 유형 구별이 가능하지만 전산언어학적 측면에서 앞의 분류는 적당하지 않고 분류기준 또한 사용할 수 없다. 따라서 복합동사는 전산언어학에서 기계론적 분석에 적합하게 사용될 수 있도록 재분류되어야 한다. 본 논문은 전산언어학적 관점에서 복합동사를 모듈별로 효율적으로 분석할 수 있도록 복합동사 형태를 다음과 같이 재분류한다.

1) '본용언 + 본용언'으로만 분석되는 복합동사
2) '본용언 + 보조 용언'으로만 분석되는 복합동사
3) '본용언 + 본용언'과 '본용언 + 보조 용언' 두가지 다 분석되는 복합동사
4) '단일용언화'된 복합동사

위에서 분류한 복합동사의 유형중 첫번째 유형은 각 동사나 형용사가 자립성을 가지고 실질적인 의미를 나타내는 반면, 두번째 유형은 보조 용언이 원래 가지고 있는 뜻으로 자립할 수 없기 때문에 이들을 결합하여 하나의 서술어로 취급해야 한다. 세번째 유형은 문장에 따라 '본용언 + 본용언' 또는 '본용언 + 보조 용언'으로 분석되므로 두 가지 결과를 모두 생성해야하며 마지막 유형은 분리되면 본 뜻을 나타낼 수 없으므로 하나의 동사로 처리해야 한다. 영어에서도 본동사와 조동사가 결합하여 하나의 동사로서 작용하지만 이 경우에는 조동사를 식별할 수 있기 때문에 '조동사 + 본동사'라는 사실을 쉽게 알 수 있다. 그러나 한국어에서는 '용언 + 용언'이 첫번째 형태인지 두번째의 형태인지를 파악하기가 매우 난해하다. 주어진 복합동사가 어떤 형태에 해당되는지를 식별하기 위하여 본 논문은 다음과 같은 분류 기준을 이용한다.

i) 단일용언화된 복합동사를 조사하여 사전에 등록한 후, 입력된 동사가 사전에 등록되어 있으면 4)형태의 복합동사로 판별한다.
ii) 보조 용언으로 사용되는 결합 관계를 보조 용언 테이블에 구성하고 이 테이블에 들어있으면 2)형태의 복합동사로 판별한다.
iii) 복합동사 분류시 '본용언 + 본용언'과 '본용언 + 보조 용언' 두가지 다 분석되는 복합동사에 대해서는 보조 용언 테이블에 '*'를 표시하고, 보조 용언 테이블 참조시 '*'를 만나면 3)형태의 복합동사로 판별한다.
iv) 사전과 보조 용언 테이블에 등록되어 있지 않으면 1) 형태의 복합동사로 판별한다.

위에서 제시한 4가지 분류 기준을 거치면 전산언어학적 관점의 복합동사를 식별할 수 있다. 본 논문에서 복합동사를 위에서 언급한 형태로 재분류한 이유는 이들이 각

모듈별로 처리가 가능하다는데 있다. 그러나 기존의 형태소 분석 방법을 사용하여 복합동사를 분석하고자 하면 위의 분류 기준에 따라 정확히 식별할 수 없기 때문에 형태소 분석 ㅋ 방법을 확장해야하는 필요성이 요구된다.

3. 확장된 형태소 분석

한국어의 형태소 분석 방법에 관해서는 현재까지 많은 연구가 진행되어 왔다. 현재까지 이들 분석기의 분석 방법을 분류하여 보면 단어를 분석하는 방향에 따라 분류한 좌-우 분석법, 우-좌 분석법, 양방향 분석법 등이 있고, 단어를 이루는 형태소의 길이에 따라 단어를 가능한 모든 형태소로 분리한 다음, 최고로 긴 형태소를 포함하는 결과를 선택하는 최장일치법과 가장 짧은 형태소를 포함하는 결과를 선택하는 최단일치법, 접속 정보를 이용한 tabular 파싱법 등이 있다. 또한 형태소 분리 방법을 기준으로 분류하여 보면 중심어와 피중심어의 접속 관계를 이용하는 Head-Tail법 등이 있다. 한국어 불규칙 어절의 원형을 복원하는 기법으로는 two-level 형태론과 음절기반 복원법 등이 있다. 본 논문에서는 복합동사 처리를 형태소 분석 단계에서 주도적으로 처리하기 위하여 기존의 최장일치법을 확장시킨 확장된 최장일치법(extended longest match strategy)을 사용하는데, 기존의 최장일치법은 입력 문자열을 가능한 모든 형태소로 분리한 다음, 가장 긴 형태소를 포함하는 분석 결과를 선택하는 방법이므로 형태소 분석 단계에서 복합동사를 처리하기가 어려운데 비하여 확장된 최장일치법은 가능하며 형태소 분석 단계에서 복합동사를 처리하고자 다음과 같은 작업을 수행하도록 확장시킨 형태소 분석 방법이다.

1. 입력 문자열을 가능한 모든 형태소로 분석한 다음, 최장의 형태소를 포함하는 분석 후보를 찾는다.
2. 최장의 형태소로 분석된 후보중에서 체언으로 분석된 후보와 용언으로 분석된 후보가 결합하여 하나의 범주로 취급될 수 있으면, 이들 후보 각각에 대한 결과를 생성하여 주고, 이들을 묶어서 분석한 결과를 생성하여 준다.
3. 최장의 형태소로 분석된 후보중에서 용언으로 분석된 후보들끼리 결합하여 하나의 범주로 취급될 수 있으면서 각각의 용언이 자립성을 가지는 경우, 각각의 용언에 대한 분석 결과를 생성하여 주고, 이들을 묶어서 분석한 결과를 생성하여 준다.
4. 최장의 형태소로 분석된 후보중에서 용언으로 분석된 후보들끼리 결합하여 하나의 범주로 취급될 수 있으면서 뒤의 용언이 자립성을 상실하는 경우, 이들을 묶어서 하나의 용언으로 분석한 결과만을 생성하여 준다.

본 논문에서 제시하는 확장된 최장일치법을 사용하는 형태소 분석의 개략적인 알고리즘은 다음과 같다.

```
begin
 fetch token
 do while not (end of input string)
   search dictionary for noun
   if (success)
     then search and analyze 접미사
          search and analyze 조사
          produce 형태소 분석 결과 1, 분석된 위치 p_end1
   search dictionary for verb
   if (fail)
     then recover the root of irregular forms
   search and analyze 선어말어미
   search and analyze 어말어미
   produce 형태소 분석 결과 2, 분석된 위치 p_end2
   if (보조 용언을 가질 수 있는 어미인가)
     then fetch next token
          search 보조용언 테이블
          if (exist in 보조용언 테이블)
             then search and analyze 선어말어미
                  search and analyze 어말어미
                  produce 형태소 분석 결과 3, 분석된 위치 p_end3
                  append 보조용언 테이블의 자질 to 본 용언의 자질
/* 보조용언의 자질을 본 용언의 자질과 결합하여 용언의 자질 생성 */
                  produce 형태소 분석 결과 4, 분석된 위치 p_end4
             else analyze that token
                  produce 형태소 분석 결과 5, 분석된 위치 p_end5
   if (exist 형태소 분석 결과 4)
     then if ((p_end1 + p_end3) == p_end4)
             then return 형태소 분석 결과 4,
                         형태소 분석 결과 1 + 형태소 분석 결과 3
             else return 형태소 분석 결과 4
     else if (p_end1 == p_end2) /* 최장일치를 위하여 길이 비교 */
             then return 형태소 분석 결과 1 + 형태소 분석 결과 5,
                         형태소 분석 결과 2 + 형태소 분석 결과 5
             else if (p_end1 > p_end2)
               then return 형태소 분석 결과 1 + 형태소 분석 결과 5
               else return 형태소 분석 결과 2 + 형태소 분석 결과 5
 endwhile
end
```

[그림 1] 확장된 최장일치법의 개략적인 알고리즘

위에서 언급한 확장된 최장일치법을 형태소 분석 방법으로 사용할 경우, '숙희가 사과를 먹어 보았다'라는 입력 문자열에서 '먹어(eat)'와 '보았다(see)'의 두개의 용언을 생성하여 주고 파서에서 문법에서 처리하는 것이 아니라 '먹어 보았다'를 하나의 용언으로 생성하여 준다. 이는 '먹어 보았다'에서 '보았다'가 영어의 'see'라는 의미로 쓰인 것이 아니라 '먹다(eat)' 동작의 시도(try)를 나타내는 보조 용언으로 사용된 것이므로 이를 하나의 복합동사로 묶어서 처리하는 것이 타당하기 때문이다.

4. 형태소 분석 주도의 복합동사의 처리

앞에서 기술한 형태소 분석 방법을 사용하여 한국어의 복합동사를 가능한 한 형태소 분석 단계에서 처리하면 파서의 부담을 최소화시킬수 있다. 그러나 형태소 분석 단계에서 처리하지 못하는 복합동사들이 존재하는데 이들은 사전의 도움을 받아서 처리하거나 구문분석 단계에서 문법 규칙과 의미 정보를 이용하여 처리도록 한다. 기존의 형태소 분석 시스템들은 형태소 분석 자체에 중점을 두어 처리했기 때문에 복합동사를 기존의 형태소 분석 시스템의 형태소 정보를 이용하여 구문 분석단계에서 문법 규칙을 이용하여 처리하게 되면, 보조 용언과 선행 어휘 형태소의 품사 모호성때문에 구조적 모호성을 야기시킨다. 또한 복합동사를 처리하기 위한 문법이 복잡해지고 파싱 테이블과 심볼 테이블이 커짐으로써 발생하는 메모리의 낭비가 크고 전체적인 시스템의 효율성이 저하된다. 그러나 복합동사를 구문 분석 이전의 형태소 분석 단계에서 처리하게 되면 용언의 품사 모호성과 이로 인해 발생하는 구조적 모호성을 사전에 방지할 수 있고 사전의 크기를 줄일수 있으며 전체적인 시스템의 효율성을 증가시킬수 있다.

전산학적 관점에서 재분류한 복합동사 구성 형태중에서 '본용언 + 보조 용언'인 경우는 보조 용언들이 자신의 속성을 상실하고 본용언에 필요한 속성을 제공한다. 이를 구문분석 단계에서 문법을 이용하여 처리한다면 각각의 경우를 고려해야 하므로 문법이 기하급수적으로 증가되고 보조 용언이 나타내고자 하는 의미를 표현하기가 쉽지 않으므로 정확한 자질을 모으기 힘들다. 따라서 기존의 방법과 같이 처리한다면 정확한 의미를 나타내기 어렵고 처리시간이 길어지며 기계번역을 할 때 고품질의 결과를 얻을 수 없다. 예를 들어, "철수가 숙희에게 과자를 먹게 하였다"에서 '먹게 하였다'는 '먹다(eat)'와 '하다(do)'라는 용언의 결합으로 이 경우는 '먹다'의 의미와 '하다'의 의미가 각각 자립하는 것이 아니라 '하다'가 지신이 가지는 의미를 상실하고 '먹다'라는 용언이 가지는 의미에 '사역'의 의미를 나타낸다. 한국어의 조동사에 대한 연구는 [11]에서 시도하였는데 여기에서는 기존의 방법과 같이 파서가 주도적으로 처리한 다음, 조동사 테이블을 참조하여 번역해주는 후처리 형태의 복합동사 처리 방법이다. 본 논문에서는 어미에 따라 뒤에 오는 용언이 본용언인지 보조 용언인지를 결정할 수 있는 특성을 이용하여 부록에 있는 <표 1>과 같은 보조 용언 테이블을 구성하고 이 테이블을 이용하여 복합동사가 '본용언 + 보조 용언'임을 식별한다. 따라서 어떤 복합동사가 이 테이블에 있는 구조에 속하면 뒤에 있는 용언은 보조 용언으로 사용된 것이고 또한 보조 용언으로 판별된 용언은 본래의 의미를 상실하고 본용언의 의미에 보조적으로 테이블에서 분류한 역할을 나타내므로 전처리 형태로 형태소 분석 단계에서 앞에 있는 본용언의 속성에 테이블에서 보여주는 속성 자질값을 첨가시켜 분석 결과로 생성하여 줌으로써 복합동사를 처리한다.

여러개의 용언들이 연속하여 나오는 복합동사는 가장 마지막에 위치한 용언은 시제정보와 문장의 형을 결정하는 종결어미를 가지며, 그 앞에 위치하는 나머지 용언들은 이 용언을 뒤따르는 용언의 유형에 따라 각기 다른 유형의 어미를 취한다[1]. 이러한 경우는 어미를 매개로 뒤에 나오는 용언들이 보조 용언으로 사용되는 경우에 한해서이다. 따라서 이러한 유형의 복합동사는 앞에서 제시한 보조 용언 테이블을 참조로 하여 적절한 자질값을 처음에 나오는 본용언에 첨가시켜 주어 형태소 분석 단계에서 처리할 수 있다. 형태소 분석 단계에서 처리할 수 있는 이러한 유형의 구조는 다음과 같다.

1) V1 + IM + V2 + IM + V3 (먹어보게하였다, 먹어보지않았다, ...)
2) V1 + IM + V2 + IM + V3 + IM + V4
   (먹어보게하여보았다, 헤엄쳐보지않게하였다, ...)
3) VIM + VIM + .. (여기서 VIM = V + IM)
   (먹어보게하여보지않았다, ...)
4) V1 + KM + V2 + IM + V3
   (먹을수있게하였다, 볼수있게만듣었다, ...)
5) V1 + KM + V2 + IM + V3 + IM + V4
   (먹을수있게하여보았다, 볼수있게만들지않았다, ...)
6) VKM + VKM + ... (여기서 VKM = V + KM)
   (먹을수있게하여보지않았다, ...)

여기에서 IM은 '아/어', '게', '지', '고'의 보조적 연결어미를 나타내고[1] KM은 '-ㄹ수'를 나타낸다.

형태소 분석기의 자질구조 생성에 관해서는 [12]에서 제시하였는데 본 논문은 이를 확장하여 <표 2>의 자질과 부록에 있는 <표 1>의 보조 용언 테이블에서 보여준 자질값을 이용하여 자질구조를 생성한다.

속성	값	내용	속성	값	내용
root		용언의 원형		CNT	연결
cat	VI	자동사		REL	관형
	VT	타동사		DEC	평서
	ADJ	형용사	mood	EXEC	감탄
tense	PAST	과거시제		QUES	의문
	PRESENT	현재시제		INST	명령
	FUTURE	미래시제		LET	청유

<표 2> 복합동사를 처리하기 위해 요구되는 자질

위에서 제시한 자질구조를 가지고 다수의 연속된 용언들이 보조 용언으로 사용되어 하나의 복합동사를 구성하는 형태중에서 IM과 KM을 매개로 구성하는 예를 분석해 본

다. 주요 약어(abbreviation)는 다음과 같다.

Abbreviations :
NOM(nominative case marker)
ACC(accusative case marker)
IM(infinitive marker)
DEC(declarative)
ADV(adverbial case marker)

① VC1 + IM + VC2 + IM + VC3의 구조

㉠ 철수가 숙희를 헤엄치 게 하 여 보 았 다
NOM ACC swim IM amake IM atry past DEC

㉡ 철수가 숙희를 헤엄치 지 못 하 게 하 여 보 았 다
NOM ACC swim IM neg do IM amake IM atry past

위 예문 ㉠과 ㉡에서 보는 것과 같이 이들 복합동사들은 '용언1 + 용언2 + 용언3'의 형태에서 용언들이 보조적 연결어미 IM을 매개로 하여 확장된 형태들이다. 따라서 이러한 구조의 복합동사들은 형태소 분석 단계에서 <표 1>의 자질값과 <표 2>의 자질을 참조하여 'amake', 'atry', 'past', 'DEC' 자질값을 본용언(swim)이 가지고 있는 속성에 삽입하여 줌으로써 처리할 수 있다. 용언들이 IM을 매개로 하여 연속적으로 네개 이상 연결되어 하나의 복합동사를 형성하는 경우도 똑같은 방법을 적용하여 처리한다. ㉡의 형태소 분석 결과는 다음과 같다.

(@NP (form 기오가) (cat N) (pform 가) (cform 주격))	(@NP (form 숙희를) (cat N) (pform 를) (cform 목적격))	(@V (form 헤엄치지못하게 하여보았다) (cat VT) (root 헤엄치) (neg +) (amake +) (atry +) (tense PAST) (mood DEC)))

[그림 2] 예문 ㉡의 형태소 분석 결과

② VC1 + KM + VC2 및 VC1 + KM + VC2 + IM + VC3의 구조

㉠ 철수가 숙희를 볼 수 있 었 다
NOM ACC see KM aposs past DEC

㉡ 철수가 숙희를 집에 갈 수 없 게 만들 어 버 렸 다
NOM ACC ADV go KM aimpo IM amake IM aend past

위의 예문은 KM과 IM을 매개로 복합동사를 형성한 형태데 이러한 구조를 가지는 복합동사는 형태소 분석 단계에서 '가능'을 나타내는 자질값을 VC1에 삽입하여 처리할 수 있다. 따라서 ㉠은 본용언(see)이 가지고 있는 속성값에 'aposs', 'past', 'DEC' 자질값을, ㉡은 본용언(go)이 가지고 있는 속성에 'aimpo', 'amake', 'aend', 'past', 'DEC' 자질값을 삽입하여 처리한다. 용언들이 많은 수의 KM과 IM을 매개로 여러개 연결되어 하나의 복합동사를 성하는 경우도 같은 방법을 적용하여 처리한다. ㉡의 형태소 분석 결과는 다음과 같다.

(@NP (form 기오가) (cat N) (pform 가) (cform 주격))	(@NP (form 숙희를) (cat N) (pform 를) (cform 목적격))	(@NP (form 집에) (cat N) (pform 에))	(@V (form 갈수없게 만들어버렸다) (cat VT) (root 가) (aimpo +) (amake +) (aend +) (tense PAST) (mood DEC))) (@V (form 갈수없게 만들어버렸다) (cat VT) (root 갈) (aimpo +) (amake +) (aend +) (tense PAST) (mood DEC)))

[그림 3] 예문 ㉡의 형태소 분석 결과

본 논문은 가능한 한 형태소 분석 단계에서 주도적으로 복합동사를 처리하고자 하지만 형태소 분석 단계에서 처리하기 어려운 복합동사들이 존재한다. 이러한 복합동사들은 다른 모듈에서 처리하는 것이 훨씬 효율적이다. 이러한 구조들은 '본용언 + 본용언'의 형태를 취하는 것과 '단일 용언화'의 형태로 우리는 이 구조에 속하는 복합동사는 파서와 사전을 이용하여 처리한다.

4.1 파서에서 처리

"철수가 가방을 들고 간다"에서 '들고 간다'는 '들다'와 '가다'라는 용언들이 결합하여 복합동사를 형성한 것으로 '들다'의 의미와 '가다'의 의미가 자립하여야 한다. 따라서 '들고 간다'는 '본용언 + 본용언'의 구조로 이루어진 복합동사이다. '본용언 + 본용언'의 경우는 형태소 분석기에서 각각의 본용언들에 대한 결과를 생성해주고 구문분석 단계에서 파서가 문법을 이용하여 처리한다. 우리는 [그림 4]와 같은 조건 단일화[10]기반의 문법을 사용하여 복합동사를 처리한다.

```
(<s> <==> (<s> <s>)
   (((x1 mood) =c CNT)
    ((x0 mood) = (x2 mood))
     (*OR*  (((x2 cat) = vt)
         (*OR*  (((x2 subj) = *EXIST*)
             (*OR*  (((x2 obj) = *EXIST*)
                    ((x0 sadjunct) > x2)
                    ((x0 sadjunct) > x1))
                   (((x2 obj) = *NONEXIST*)
                    ((x2 obj) = (x1 obj))
```

```
                ((x0 sadjunct) > x2)
                ((x0 sadjunct) > x1)))
               (((x2 subj) = *NONEXIST*)
                 (*OR*  (((x2 obj) = *EXIST*)
                         ((x2 subj) = (x1 subj))
                         ((x0 sadjunct) > x2)
                         ((x0 sadjunct) > x1))
                        (((x2 obj) = *NONEXIST*)
                         ((x2 subj) = (x1 subj))
                         ((x2 obj) = (x1 obj))
                         ((x0 sadjunct) > x2)
                         ((x0 sadjunct) > x1))))))
        (((x2 cat) = vi)
          (*OR*  (((x2 subj) = *EXIST*)
                  ((x0 sadjunct) > x2)
                  ((x0 sadjunct) > x1))
                 (((x2 subj) = *NONEXIST*)
                  ((x2 subj) = (x1 subj))
                  ((x0 sadjunct) > x2)
                  ((x0 sadjunct) > x1)))))))
(<s> <==> (<vc>)
       ((x0 = x1)))
(<vc> <==> (<vc> <im>)
       ((x0 = x1)
       (x0 = x2)))
```

[그림 4] 복합동사 처리를 위한 조건 단일화 문법

[그림 4]에서는 단일화의 수행 조건을 검사할 수 있는 몇 가지 연산자가 나타나 있는데 이들은 다음과 같다.

1) 명세의 이접기술 기능이다. 우리의 분석기에서는 이 접기술의 기호를 *OR*로 쓴다. *OR*로 시작되는 명세에는 대등한 위치를 차지하는 동일문의 집합을 나열할 수 있다. 이 기능을 통하여 규칙 밖에 별도의 제어기구를 두지 않아도 'IF (조건식) THEN S1 ELSE S2'의 기능을 수행하도록 명세를 기술할 수 있다.
2) 동일성 검사기능 '=C'이다. '=C'는 LFG의 검사장치로 단일화 연산인 '='과는 달리 좌우변 자질구조간의 완전한 단일화는 일어나지 않으며 단지 좌우항의 동일성만 검사해서 같으면 참, 다르면 거짓이 된다.
3) 존재여부 검사기능이다. 이 기능은 특정 속성의 값이 이미 존재하는 지의 여부를 판단하기위해 사용된다. 이 기능은 특정 자질구조 안의 임의의 속성의 존재 여부를 동적으로 검사하기 위하여 도입하였다. 이 기능을 위한 기호로는 '*EXIST*', 이것의 부정인 '*NONEXIST*'를 사용한다.
4) 부가기능이다. 영어의 'and' 문장이나 한국어의 등위문의 경우, 두개의 대등한 문장을 표현할 때 단일화가 아닌 부가 연산을 통해 해결한다. 즉, S_1과 S_2가 대등 접속일 경우에 이 연산을 사용하면 [*CONJ* (S_1) (S_2)]와 같은 자질구조를 얻을 수 있다. 우리의 시스템에서는 이 기능을 위한 연산자를 '>'로 쓴다.

앞의 예문은 형태소 분석 단계를 거쳐 '들' + '고' + '가' + 'ㄴ' + '다'가 구문 분석기에 입력으로 들어오고 '들고'가 앞 용언, '간다'가 뒤 따르는 용언이 된다. 이 복합동사는 '고'라는 어미에 연결의 의미(mood CNT)가 사전을 통해 올라오게 된다. 이러한 연결 정보를 가지고 이 문장의 정확한 분석이 이루어지게 된다. '철수가 가방을 들고 철수가 간다'가 이 문장의 구문 분석 결과가 되는데, 이러한 구문 분석을 위해 위와 같은 문법기술이 필요하게 된다. 즉 뒷 용언에 해당하는 주어나 목적어가 생략된 경우 앞 문장의 주어나 목적어가 이러한 생략을 처리하게 되는 것이다. 결국 '본용언' + '본용언'은 뒷 용언의 주어나 목적어와 같은 문장성분의 생략에 따라 발생하게 됨을 알 수 있다. 그림 [4]의 절차적 표현 방법에 대한 자세한 내용은 [10]에서 다루고 있다.

4.2 형태소 분석기와 파서를 연계한 복합동사 처리

"운전사가 차를 길옆으로 몰아 대었다"라는 예문을 보자. 위 예문의 경우 '몰아 대다'가 <표 1>에서 보여준 보조 용언 테이블에 보조 용언으로 사용된 것으로 분류되어 '몰다'의 의미에 강조를 나타내는 'aemph' 자질값을 첨가시켜 결과를 생성하여 준다. 그러나 이 문장에서 '몰아 대다'가 '차를 길 옆 한쪽으로 바치다'라는 의미로 사용되는 경우도 있다. 이 경우는 '강조'를 나타내는 것으로만 보는 것은 잘못된 분석이 된다. 이 구조에 속하는 것은 어미 테이블을 참조할 때 '*'를 만나면 '본용언 + 보조 용언'으로 분석된 후보와 '본용언 + 본용언'으로 분석된 후보를 아래와 같이 생성해주고 이들 분석 후보에 대한 처리는 구문 분석 단계에서 파서가 [그림 4]의 조건 단일화 문법과 의미지식을 이용하여 처리해야 한다. 위 예문에 대한 형태소 분석 결과는 부록에 나타나 있는 [그림 5]와 같다.

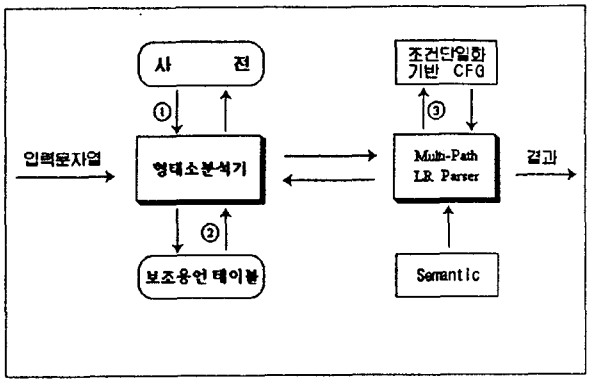
4.3 사전에서 처리

복합동사 중에는 특별한 생성 규칙이 존재하지 않아서 '본용언 + 본용언'으로도 분류되지도 않고 '본용언 + 보조 용언'으로도 분류되지 않는 복합동사가 존재한다. 예를 들어 '껴안다', '여닫다' 등의 복합동사는 이런 부류의 복합동사에 속하는 말들이다. 따라서 이 경우에 속하는 복합동사는 앞에서 언급한 어떤 부류에도 포함시킬 수 없기 때문에 사전에 등록하여 처리한다.

5. 시스템 구성 및 처리 알고리즘

앞 장에서는 전산언어학적인 관점에서 분류한 복합동사를 각 유형별로 처리 방법을 제시하였는데 이들을 처리

하기 위해서는 사전과 형태소 분석기 그리고 파서가 유기적으로 연계되어 각각의 재정의된 역할을 수행하여야 한다. 본 논문이 제안하는 시스템의 구성은 다음과 같다.



[그림 6] 복합동사 처리를 위한 시스템의 구성

입력 문장이 들어오면 최장일치 전략을 사용하는 형태소 분석기는 [그림 6]에서 보는 것같이 ①번 단계에서 어휘 형태소 사전에 등록되어 있는지 검색한다. 사전에 등록되어 있으면 그 결과를 생성하여 준다. 만약 사전에 등록되어 있지 않으면 ②번 단계로 보조 용언 테이블을 검색하는데 이 단계에서 '본용언 + 보조 용언'의 복합동사와 '본용언 + 본용언'과 '본용언 + 보조 용언'을 동시에 가지는 복합동사를 처리하여 형태소 분석 결과로 내어준다. 이들 결과들은 ③번 구문분석 단계에서 다중-경로 LR 파서가 LFG 계열의 조건 단일화 문법[10]과 semantic을 이용하여 처리한다. 복합동사 처리시, '나와 있다', '가게는 해주다' 등과 같은 문자열은 구문 범주가 'v' 로 분석될 수 있고 'np + v'로 분석될 수 있다. 이와 같이 다른 갯수의 구문 범주로 분석되는 문자열을 Tomita의 파서(GLR)에서는 처리할 수 없기 때문에 이들을 처리하기 위하여 GLR의 개념을 수정한 다중-경로 LR 파서를 사용한[13]. 그러나 이미 테이블에 포함되지 않는 복합동사는 '본용언 + 본용언'의 복합동사이므로 형태소 분석기는 각각의 '본용언'에 대한 결과를 생성하여 주고 ③번 단계에서 처리한다.

본 논문이 제안하는 복합동사 처리 방법은 사전에 등록하는 어휘를 줄여주고 형태소 분석기의 역할을 강화시켜 파서의 부담을 줄임으로써 전체적인 효율성을 획득하는 방법이다. 이 방법은 형태소 분석기의 부담이 다른 형태소 분석기들보다 크지만 구문분석 단계에서 파서가 문법을 이용한 단일화에 의해 복합동사를 처리하는 시간과 비교해볼 때 매우 효율적임을 볼 수 있었다. 특히 우리의 방법은 'V1 + IM + V2'의 구조와 'V1 + KM + V2'의 에서 용언들이 여러개 모여 하나의 복합동사를 형성할 경우에 파서에서 문법에 의해 처리하는 것보다 더욱 더 좋은 성능을 보였다. 그러나 현재까지는 복합동사 중 보조 용언으로 사용되는 유형에 관하여 명확하게 규명되지 않았기 때문에 보조 용언으로 사용된 용언을 형태소 분석 단계에서 처리하지 못하는 경우도 있다.

6. 결론 및 향후 연구과제

본 논문에서는 한국어의 복합동사를 형태소 분석 단계에서 처리함으로써 구문분석기의 부담을 줄임으로써 전체적인 시스템의 효율성을 향상시키는 형태소 분석 주도의 복합동사 처리에 대하여 기술하였다. 이 전략은 형태소 분석기의 역할을 확장시켜 가능한 한 복합동사를 형태소 분석 단계에서 처리해주고, 이 단계에서 처리하기 어려운 복합동사는 사전을 참조하여 처리하거나 파서에서 문법 규칙과 다양한 의미지식을 이용하여 처리하도록 하는 방법이다. 그러나 본 논문에서 제시하는 형태소 분석 방법을 사용하여 처리할 경우, 극소수의 예외적인 복합동사들이 존재하는데 이에 대한 처리 방법은 현재 연구중이다. 본 논문의 방법을 사용하여 실험해 본 결과 형태소 분석 시간이 더 걸리지만 전체적인 효율성은 좋게 나타났다. 현재까지 한국어의 복합동사에 관해서는 국문학에서도 명확히 분류하지 못하였고 전산언어학에서도 이를 처리하고자 하는 연구는 미비하였다. 그러나 복합동사는 한국어를 처리하기 위해서는 필수적으로 처리되어야 할 부분으로서, 본 논문의 연구는 다음과 같은 의의를 가진다.

첫째, 형태소 분석 단계에서 복합동사를 처리함으로써 구문분석 단계에서 문법 규칙을 이용하여 처리하는 것보다 메모리 효율성이 좋아진다.

둘째, 전산언어학적인 분류에 따라 각 유형별로 사전과 형태소 분석기 그리고 구문 분석기를 유기적으로 연결하여 전체적인 시스템의 효율성을 획득할 수 있고, 사전의 크기와 용언의 품사 모호성을 줄이고, 구문분석 단계에서 발생하는 구조적 모호성을 사전에 방지할 수 있다.

세째, 한국어에서 빈번하게 사용되는 복합동사 처리 방법을 제시함으로써 기계번역, 자연어 이해 시스템 그리고 음성인식 등의 자연어 응용 분야에서 효율적으로 사용할수 있다.

본 논문에서는 보조 용언으로 사용되는 유형에 대해 완전하게 분류되지 않았기 때문에 보조 용언 테이블에 등록되어 있지 않은 보조 용언에 대해서는 명확한 처리방법을 제시하지 못하였다. 따라서 현재 향후 연구로 한국어의 복합동사 중 보조 용언으로 사용되는 유형 분류에 대

한 세밀한 분류 작업과 형태소 분석 단계에서 처리하는 복합동사에 대하여 기계번역 시스템이나 자연어 이해 시스템 등에서 효율적으로 응용될 수 있는 자질에 대한 연구가 수행중이다.

[참고 문헌]

[1] 김수길, "The Korean Compound Verbal Constructions in Lexical Functional Grammar", 전북대학교 어학연구소 제 17집, pp. 55-71, 1990.

[2] Deok Ho Yoon, Yung Taek Kim, "Analysis Techniques for Korean sentences Based on Lexical Functional Grammar", *International Parsing Workshop*, pp. 369-378, 1989.

[3] 서영훈, 의미정보를 이용하는 중심어 주도의 한국어 파싱, 서울대 공학박사 학위 논문, 1991.

[4] 김창섭, "현대국어의 복합동사 연구", 국어 연구 47.

[5] 김창섭, 국어의 단어 형성과 단어 구조, 서울대 문학박사 학위 논문, 1994.

[6] 강운국, 조선어 문형 연구, 서광 학술 자료사, 1993.

[7] 서정수, "보조동사에 관한 토론", 국어문법의 연구 II, 한국문화사, 1990.

[8] 서정수, "{고}와 {어서}", 국어문법의 연구 II, 한국문화사, 1990.

[9] 남기심, 국어 표준 문법론, 탑출판사, 1985.

[10] Seung-Won Yang, Gi-O Lee, Yong-Seok Lee, "An Analysis Technique for Korean Sentences Using the Conditional Unification", *International Conference on Computer Processing of Oriental Language*, pp. 257-262, May, 1994.

[11] 안동언, 조정미, 김길창, "영한 기계번역의 한국어 생성 시스템에서의 조동사의 형성", 제 5회 한글 및 한국어 정보처리 발표 논문집, pp.533-544, 1993.

[12] 송연정, 배우정, 이기오, 이용석, "한국어 형태소 분석기의 자질구조 생성에 관한 연구", 한국 정보과학회 발표지, Vol. 21, No. 1, pp. 817-820, 1994.

[13] Gi-O Lee, Yi-Gyu Hwang, Yong-Seok Lee, "Multi-path LR Parsing for Nonsegmental Words Using Interactive Strategy", *The 3rd Pacific Rim International Conference on AI*, Vol. 2, pp. 668-672, August, 1994.

부록

의미	속성	어미 + V2의 테이블	의미	속성	어미 + V2의 테이블
완료	acomp	(-아/어) 있다 (-아/어) 두다	가정	asupp	(-다) 치다 (-(으)ㄴ/는) 셈치다
지속	acont	(-아/어) 계시다	금지	aban	(-지) 말다
진행	aprog	(-고) 있다 (-아/어) 가다 (-아/어) 오다 (-고) 계시다 (-게) 굴다	강조	aemph	(-고) 말고 (-다) 말다 (-아/-어) 대다 (-아/-어) 빠지다 (-아/어) 쌓다 (-아/-어) 터지다
습관	ahabt	(-고는) 하다 (-곤) 하다	가식	apret	(-(으)ㄴ/-는) 양하다 (-(으)ㄴ/-는) 척하다 (-(으)ㄴ/-는) 체하다
시작	abegn	(-기) 시작하다 (-어) 들다	피동	apass	(-아/-어) 지다
보유	ahold	(-아/어) 놓다 (-아/어) 가지다 (-아/-어) 두다	부정	neg	(-지) 않다 (-지) 아니하다 (-지) 못하다
의도	aintd	(-려) 들다 (-려고) 들다 (-려) 하다 (-려고) 하다 (-고자) 하다	봉사	aserv	(-아/-어) 주다 (-아/-어) 드리다
소원	awish	(-(으)면) 하다 (-고) 싶프다 (-고) 싶다 (-고) 지다	사동	amake	(-게) 하다 (-도록) 하다 (-게) 만들다 (-도록) 만들다
추측	agues	(-는가/-ㄴ가/-나) 보다 (-ㄹ) 것같다 (-(으)ㄴ/-는/-(으)ㄹ) 듯싶다 (-(으)ㄴ/-는/-(으)ㄹ) 듯하다 (-(으)ㄹ) 법하다 (-(으)ㄴ/-는/-(으)ㄹ) 성싶다 (-(으)ㄴ가/는가/-(으)ㄹ까/를까) 시프다 (-(으)ㄴ가/는가/-나/-(으)ㄹ까/를까) 싶다 (-(으)ㅂ) 직하다	종결	aend	(-고) 나다 (-고) 말다 (-고야) 말다 (-아/-어) 내다 (-아/-어) 버리다 (-아/-어) 치우다
가능	aposs	(-ㄹ) 수있다 (-어도/-아도) 되다	짐작	ainte	(-아/-어) 보이다 (-게) 생기다
가치	awort	(-(으)ㄹ) 만하다	시도	atry	(-아/-어) 보다
불가능	aimpo	(-ㄹ) 수없다	재귀	aretn	(-게) 되다
당위	amust	(-아야/-어야) 한다 (-아야/-어야) 되다	시인	appro	(-기는) 하다 (-기도) 하다 (-아/-어) 마지아니하다 (-아/-어) 마지않다
요행	afluk	(-(으)ㄹ) 뻔하다	불가피	ainev	(ㄹ수밖에) 없다

〈표 1〉 보조 용언을 식별하기 위한 보조 용언 테이블

((@NP (form 운전사가) (cat N) (pform 가) (cform 주격))	(@NP (form 차를) (cat N) (pform 를) (cform 목적격))	(@N (form 길))	(@NP (form 옆으로) (cat N) (pform 으로))	(@V (form 몰아대었다) (cat VT) (root 몰) (aemph +) (tense PAST) (mood DEC)))	
				(@V (form 몰아) (cat VT) (root 몰) (mood CNT))	(@V (form 대었다) (cat VT) (root 대) (tense PAST) (mood DEC)))

[그림 5] 예문의 형태소 분석 결과