論文
6-5-5

형태소 사전 기반 구문 형태소 생성
(syntactic morpheme generation using morpheme dictionary)

박인철(Park In Cheol)[1)]

요 약

구문 형태소는 형태소 분석 과정에서 생성된 노드들을 최소한으로 줄이기 위해 제안되었다. 구문 형태소는 불필요한 노드를 제거해 줌으로 구문 분석기의 부담을 매우 크게 줄이는 효과가 있다. 그러나 기존의 시스템에서 구문 형태소 생성은 형태소 분석 단계와 분리되어 별도의 분석 시간을 요구하며, 띄어쓰기 오류에 대한 고려를 하지 않았다. 본 논문에서는 이러한 문제점을 해결하기 위해 형태소 사전을 기반으로 한 구문 형태소 생성 방법을 제안한다. 실험 결과 기존의 방법에 비해 제안된 방법은 100배 이상의 생성 속도 향상을 보였다.

ABSTRACT

Syntactic morpheme is proposed for reducing morpheme units generated by korean morpheme analyzer. It is proved that syntactic morpheme remarkably diminished the overhead of syntactic analyzer. However, the syntactic morpheme generation is so separated from the morpheme analyze phase in the existing system that it needs an extra execution time. Moreover, the method do not consider spacing-free statements. In this paper, we propose a syntactic morpheme generation using morpheme dictionary in order to resolve the problems. Experiments show that our proposed method reduce generation time more than one hundred times as compared with the existing one.

논문접수 : 2005. 11. 10.
심사완료 : 2005. 11. 30.

1) 정회원 : 호원대학교 컴퓨터학부 교수

본 논문은 「2004년도 호원대학교 교내학술연구조성비」에 의해 연구되었음.

1. 서론

한국어 문장의 분석 과정에서 발생하는 주요 문제점 중의 하나는 과중한 형태론적 및 구문적 모호성의 발생이다. 이러한 모호성은 전체 분석 시간에 큰 영향을 미치고 의미 분석과 같은 후처리에도 영향을 주기 때문에, 모호성을 줄이기 위한 연구가 다각도로 진행되었다. 이러한 연구는 크게 문법에 의한 구문적 모호성을 줄이는 접근[1, 2, 3]과 구문 분석의 입력인 형태론적 모호성을 줄이는 접근으로 구분될 수 있다[4, 5, 6].

[5] 등에 의해 제안된 구문 형태소는 여러 개의 형태소가 묶여져 하나의 의미를 갖는 확장된 형태소를 의미한다[3]. 예를 들어, 어절 "너를 위하여"는 전통적인 형태소 분류에 따르면, '너/대명사)', '를/조사/', '위하(다)/동사', '여/어미' 등으로 분류된다. 구문 분석기는 대명사 '너'와 동사 '위하다'와의 구문 관계를 분석해야 하므로 별도의 분석 시간이 필요하고, 이 과정에서 다양한 구문적 모호성이 발생할 수 있다. 그러나 구문 형태소를 사용하면 위 어절은 '너/대명사'와 '를_위하여/확장_조사'라는 두 개의 후보만을 생성하므로 구문 분석의 부담을 크게 줄일 수 있음을 알 수 있다.

기존의 구문 형태소 생성 방법은 형태소 분석 단계에서 생성된 형태소들을 오토마타를 이용하여 규칙 혹은 구문 형태소 사전을 이용하여 생성하는 후처리 방식을 채택했다. 따라서 기존의 생성 방법은 전체 형태소 분석 노드들을 탐색해야 하는 부과적인 시간을 필요로 하며, 불필요한 사전 탐색 과정과 형태소 결과 생성 과정을 거친다. 예를 들어, '를_위하여'라는 구문 형태소 사전을 탐색하기 전에, 형태소 분석 단계에서 조사 '를', 동사 '위하(다)', 어미 '여'를 형태소 분석 사전에서 탐색한 후, 이에 대한 결과를 생성해야 할 것이다.

기존 방법에서 고려하지 않은 주요 요소 중의 하나는 구문 형태소는 그 특성상 띄어쓰기 오류가 빈번할 수 있다는 것을 고려하지 않았다는 것이다. 이는 사람이 구문 형태소를 하나의 의미로 인지하기 때문에 발생하기 쉬운 오류이다. 예를 들어, 구글 검색엔진[7]을 이용하여 "에대하여"라는 용어가 포함된 문서를 검색하면 적지 않은 문서가 검색됨을 알 수 있다.

기존의 구문 형태소가 갖는 별도의 오버헤드를 줄이기 위해 본 논문에서는 형태소 분석과 구문 형태소 생성을 통합한다. 이를 위해, 구문 형태소는 하나의 단어로 취급되어 형태소 사전에 등록하며 이에 따라 발생할 수 있는 문제점을 살펴보고 해결 방안을 제시하고자 한다. 아울러, 구문 형태소의 띄어쓰기 오류를 허용하기 위해 발생 가능한 모든 구문 형태소 조합을 사전에 등록하는 대신에 사전 검색 알고리즘을 수정하는 방안을 제시할 것이다.

본 논문의 구성은 다음과 같다. 2장에서 구문 형태소에 대해 심도 있게 살펴보고, 3장에서는 형태소 사전을 이용하여 띄어쓰기 오류를 허용하는 구문 형태소 생성 방법을 기술한다. 4장에서는 제안된 방법과 기존의 방법을 실험을 통해 비교하며, 5장에서 결론을 맺는다.

2. 구문 형태소

2.1 구문 형태소 체계

[5] 등은 구문 형태소를 여러 기능 형태소들이 결합하여 하나의 구문/의미적 단위를 형성하는 형태소 열로 정의하였다[3]. 이러한 구문 형태소의 종류로는 1) 보조 용언을 매개로 한 구문 형태소, 2) 의존 명사를 매개로 한 구문 형태소, 3) 주로 불완전 용언에 의한 의사 조사 형태의 구문 형태소로 나누었다. <표 1>은 이런 분류 체계에 대한 각각의 예를 보여준다. 여기서 문자 '_'는 공백 문자를 의미한다.

종 류	예
보조 용언 매개	어_보, 고_싶, ...
의존 명사 매개	ㄹ_수_있, ㄹ_것,,...
의사 조사	을_위해, 로_향해서

<표 1> 구문 형태소의 종류

이러한 분류 체계는 구문 형태소 인식을 위한 오토마타 설계를 위해 채택되었다. 본 논문

에서는 구문 형태소를 하나의 형태소 단위로 사전에 등록할 것이므로 이러한 분류 체계보다는 형태소 사전에서 가질 수 있는 품사 체계로 분류하는 것이 바람직하다. 구문 형태소는 그 구문적 역할에 따라 조사, 어미, 보조 용언 등의 역할을 갖는 것으로 분류할 수 있다. <표 2>는 품사 체계에 따른 구문 형태소의 분류 체계를 나타내고 있다.

(확장) 품사		예
조사		에_대해, 를_위해, ...
어미	어말 어미	기_위해, 는_데서, ...
	전성 어미	ㄹ_것, 기_때문, ...
보조 용언		어_보, ㄹ_수_있, ...

<표 2> 품사에 의한 구문 형태소 분류

조사 혹은 어미의 특성을 갖는 구문 형태소는 단순 형태소와 형태론 및 구문적 특징이 같으므로 기존의 형태소 및 구문 분석 방법과 똑같이 취급하도 무방하다. 그러나 보조 용언 특성을 갖는 구문 형태소는 그 결합 구조상 어미의 특징과 용언의 특징을 모두 가지므로 그 처리에 주의해야 한다. 즉, 어미처럼 반드시 용언의 뒤에 붙어야 하며, 용언처럼 뒤에 반드시 어미가 뒤따라와야 한다. 이는 보조 용언의 특성을 갖는 구문 형태소는 '보(다)'와 같은 일반 보조 용언과 다른 품사를 가져야 함을 의미한다. 그러나 코퍼스 분석 결과 모든 보조 용언을 보조 어미와 결합한 하나의 구문 형태소로 보아도 무방함을 알 수 있었다. 즉, '보(다)'를 독립된 보조 용언으로 분석할 필요가 없다는 의미이다. 따라서 본 논문에서 보조 용언은 보조 용언의 특성을 갖는 구문 형태소로 간주한다.

2.2 구문 형태소의 특징

구문 형태소는 여러 형태소가 묶여 하나의 구문적 혹은 의미적 단위를 형성한다. 여러 형태소는 한국어 구문 규칙에 따라 어느 정도 자유롭게 묶일 수 있으므로 여러 변형을 가질 수 있다. 예를 들어, 영어의 for에 해당하는 구문 형태소는 아래와 같이 다양한 형태로 나타날 수 있다.

(2-1) 을_위해/를_위해
(2-2) 을_위해서/를_위해서
(2-3) 을_위하여/를_위하여
(2-4) 을_위하여서/를_위하여서
(2-5) 을_위한/를_위한

여기에서 (2-1)부터 (2-4)는 동사와 묶일 때 나타나는 형태이며 (2-5)는 명사를 수식할 때 나타나는 형태이다. 이러한 다양한 형태의 구문 형태소를 처리하기 위한 간단한 방법은 모두 형태소 사전에 등록하는 것이다. 구문 형태소의 특성 상 앞의 문자열이 중복되는 경우가 많으므로 트라이 사전은 구문 형태소 사전을 구축하는데 매우 효율적이다[8].

또한 고려해야 할 구문 형태소의 또 다른 특징은 띄어쓰기 오류가 매우 빈번하게 나타난다는 것이다. 이에 대한 정확한 통계가 없으나, 일반적인 웹문서를 읽다보면 구문 형태소의 띄어쓰기 오류가 빈번하게 나타난다. 앞서 언급했듯이 이는 사람이 구문 형태소를 하나의 의미 단위로 인식하기 때문에 쉽게 발생하는 오류일 가능성이 있다. 이를 처리하는 간단한 방법은 형태소 사전에 가능한 모든 구문 형태소 나열을 저장하는 것이다.

그러나 이는 사전의 크기를 크게 증가시킬 뿐만 아니라, 동일한 정보를 여러 구문 형태소 변이에 저장해야 하므로 사전 관리를 어렵게 한다. 예를 들어, "ㄹ_수_없"는 다음과 같이 모두 4개의 변이를 가질 수 있다.

(2-6) ㄹ_수_없: 보조용언, +cannot
(2-7) ㄹ_수없: 보조용언, +cannot
(2-8) ㄹ수_없: 보조용언, +cannot
(2-9) ㄹ수없: 보조용언, +cannot

또한, "ㄹ_듯_말_듯_하"는 모두 16개의 변이를 가질 수 있다. 따라서 본 논문에서는 띄어쓰기에 따라 조합 가능한 모든 형태를 형태소 사전에 저장하는 대신에 사전 검색 알고리즘을 수

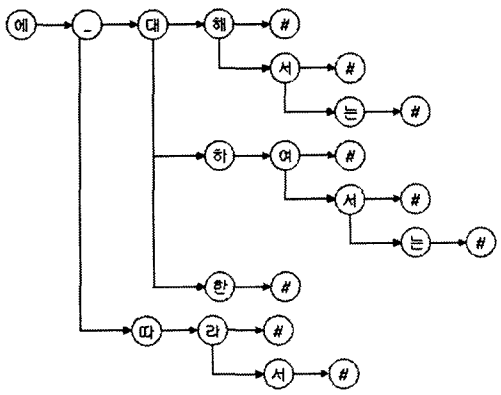
정하는 방식을 적용한다.

3. 구문 형태소 생성

3.1 사전 검색 알고리즘 수정

트라이 사전은 단어의 기본이 되는 음절 혹은 자소 단위가 하나의 노드가 되어 구축된 사전이다. 예를 들어, <그림 1>은 단어 "에_따라", "에_따라서", "에_대해", "에_대해서", "에_대해서는", "에_대한", "에_대하여", "에_대하여서", "에_대하여서는"에 대한 음절 단위로 구축된 한국어 트라이 사전의 형태를 보여준다. <그림 1>에서 '#'은 종단 노드를 의미한다.

트라이의 특성상 트라이 사전 검색과 형태소 분석을 동시에 진행할 수 있다[9]. 이러한 접근을 사용하면 사전 검색 횟수를 크게 줄일 수 있으므로, 띄어 쓰지 않은 어절을 분석할 때 매우 유용하다. 예를 들어, "에대하여서"라는 구문 패턴을 찾을 때 사전 검색과 형태소 분석이 분리된 방식을 사용하면, "에대하여서"의 검색을 실패하면 어절을 "에_대하여서"로 재구성한 후 검색해야 한다. 그러나 이 방법은 해당 어절에 대해 다시 검색을 시도해야 한다. 또한, 어떤 어절 사이에 공백을 넣어야 할 지 알 수 없으므로 어절 "알지도모르"에 대한 사전 검색은 매우 복잡해진다.



<그림 63> 트라이 사전의 구성 예

위 문제는 트라이 사전 검색과 형태소 분석을 동시에 진행하면 어렵지 않게 해결할 수 있다. 위의 예를 가지고 설명하면, 음절 "에" 다음에서 더 이상 탐색을 진행할 수 없으므로 현재 노드에서 공백으로 진행하는 노드가 있는지 검사하고 발견되면 공백 노드를 지나 계속 검색을 수행하면 된다. 이를 알고리즘으로 기술하면 다음과 같다.

(단계 1) 현재 노드에서 다음 노드 전이가 가능한지 판별한다.
(단계 2) 전이가 가능하면 다음 노드로 이동한 후 (단계 1)을 수행한다.
(단계 3) 전이가 불가능하면 종단 노드를 찾는다.
(단계 4) 종단 노드가 발견되면 검색이 성공했음을 알린다.
(단계 5) 종단 노드가 없으면 공백 노드를 찾는다.
(단계 6) 공백 노드가 없으면 검색이 실패했음을 알린다.
(단계 7) 공백 노드가 발견되면 공백 노드로 전이하고 현재 문자열에 있는 공백 문자들을 건너뛴 후 (단계 1)을 수행한다.

위 검색 알고리즘은 음절 '에'에서 '대'로 가는 노드가 존재하면 문제를 일으킨다("에대"라는 단어가 등록되었다고 가정하자). 음절 '대'까지 이동한 후 더 이상 탐색 노드가 없으므로 검색은 실패하게 된다. 이를 해결하려면, 역추적을 시도해야 하는 문제가 생긴다. 그러나 구문 형태소의 구성 요건 상 구문 형태소들 간에는 이러한 문제가 발생하지 않으므로, 구문 형태소를 포함한 어미 및 조사 사전과 기타 사전을 분리하면 이런 문제가 발생하지 않는다.

3.2 구문 형태소의 결정

앞 절에서 구문 형태소를 형태소 사전에서 검색하는 방법에 대해 살펴보았으며 트라이 사전 알고리즘을 사용함으로써 효율적으로 구문

형태소를 검색할 수 있었다. 그러나 단순한 구문 형태소 검색을 사용하면 구문 형태소가 아닌 경우에도 구문 형태소로 판별할 수 있다. 이러한 검색을 구문 형태소 우선 검색이라 부르기로 한다. 이러한 문제를 살펴보기 위해 아래의 문장에 대한 형태소 분석을 고려해 보자.

(3-1) 그 도독을 잡아 **보고**를 하였다.
(3-2) 이 문제를 해결하여 **가뭄**에서 벗어날 수 있었다.
(3-3) 시도**해볼장치**가 없었다.

위 예제에서, 밑줄 친 부분은 구문 형태소가 가능함을 나타낸 것이며, 굵은 글씨 부분은 명사가 가능함을 나타낸 것이다. 구문 형태소 우선 검색 방법을 사용하면, 문장 (3-1)은 "아_보"를 보조 용언으로, "고"를 어미로, "고"를 명사로 잘못 분석한다. 또한 문장 (3-2)에서는 "어_가"를 보조 용언으로, "어"를 생략된 어미로, "뭄"을 미지어로 잘못 분석하게 된다. (3-3)은 구문 형태소 우선 검색 방법 하에서도 올바른 결과를 생성하는 예제를 보인 것이다.

형태소 분석과 구문 형태소 생성 과정을 분리하면 (3-1)과 (3-2)와 같은 오류는 발생하지 않는다. 형태소 분석기가 이미 각각 "보고"와 "가뭄"을 명사로 분석하였기 때문에 앞에 있는 보조 어미와 묶일 염려가 없기 때문이다. 그러나 (3-3)은 기존 형태소 분석기를 사용하면 "볼장"이라는 명사가 먼저 묶이고 '치'가 접미사로 잘못 분석된다. (다른 형태소 분석기는 띄어쓰기를 가정하거나 명사의 나열과 같은 단순한 형태만을 다루므로 비교할 수 없다. 예를 들어, 연구용으로 공개된 형태소 분석기인 coran은 "시도해볼장치" 어절 전체를 하나의 명사로 가정한다)[10].

문장 (3-1)과 (3-2)의 잘못된 분석을 막기 위해서는 구문 형태소의 탐색 도중 뒤에 체언이나 용언과 같은 독립어가 나타나는지를 검사할 필요가 있다. 즉, 위의 예에서는 각각 노드 '보'와 '가'에서 매치되는 독립어가 발견되는지 조사해 보면 된다(이 경우에는 각각 '보고'와 '가뭄' 명사가 매치된다). 이런 현상은 띄어쓰기가 가능한 부분에서 일어나므로 독립어 매치 검사는 공백 노드 '_'가 발견된 후 시작하면 된다. 따라서 앞에서 기술한 알고리즘에서 (단계 7)은 다음과 같이 세분화 될 수 있다.

(단계 7) 공백 노드가 발견되면 공백 노드를 전이하고 현재 문자열에 있는 공백 문자를 건너뛴다.
(단계 8) 현재 노드에서 독립어가 가능한지 추정한다.
(단계 8-1) 독립어가 발견되면 공백 이전 노드에서 종단 노드를 검사하고 발견되면 검색이 성공했음을 알린다.
(단계 8-2) 종단 노드가 없으면 (단계 1)을 수행한다.
(단계 9) 독립어가 가능하지 않으면 (단계 1)을 수행한다.

독립어의 추정은 단순한 문제가 아니다. 예문 (3-3)에서 볼 수 있는 바와 같이, 단순히 바로 뒤에 나타나는 독립어만 찾으면 안 되고 그 뒤에 나오는 단어가 더 독립어에 적절한지를 판단해야 하기 때문이다. 독립어 추정에 대한 알고리즘은 다음과 같다.

(단계 1) 문자열의 현재 위치에서 독립어를 사전에서 검색한다.
(단계 2) 검색이 실패하면 독립어가 뒤따르지 않음을 알린다.
(단계 3) 검색이 성공하면 매치된 문자열(A)의 길이를 변수 n1에 저장하고 하나의 음절을 건너뛴 후 독립어를 사전에서 검색한다.
(단계 4) 다음 음절에서 독립어가 발견되지 않으면 독립어 추정 성공을 알린다.
(단계 5) 다음 음절에서 독립어가 발견되면 매치된 문자열(B)의 길이를 변수 n2에 저장한다.

(단계 6) 매치된 문자열 A와 B에서 각각 조사 혹은 어미를 검사하고 각각의 길이를 변수 t1과 t2에 저장한다.
(단계 7) n1 + t1의 합이 t1 + t2 합보다 크면 독립어 추정 성공을, 작으면 실패를 알린다.
(단계 8) 만일 두 개의 합이 같으면 n2의 길이가 2(즉, 한 글자짜리 명사)이면 독립어 추정 성공을, 2보다 크면 실패를 알린다.

위의 알고리즘은 미지어 문제를 고려하지 않았다. 즉, 사전에서 검색이 실패하더라도 그 문자열이 미지어이면 위의 알고리즘은 문제를 일으킨다. 따라서 독립어를 추정할 때 단순히 사전 검색만을 할 뿐만 아니라, 그 문자열이 미지어가 될 수 있는지도 추정해야 한다. 미지어 추정 문제는 별도의 어려운 문제로 본 논문에서는 이에 대해 언급하지 않는다. 단지, 실제 구현에 있어서는 기존의 미지어 추정 방식[9, 11]을 참고하여 독립어 추정 시 미지어 추정도 함께 고려하였다.

3.3 형태소 분석과의 통합

앞에서 살펴본 바와 같이, 구문 형태소를 형태소 사전에 등록함으로써 구문 형태소의 생성은 사전에서 구문 형태소를 검색하고 형태론적 규칙에 적합한지 검사하는 것으로 충분하다. 2.1절에서 구문 형태소는 조사, 어미 또는 보조 용언의 역할을 할 수 있음을 살펴보았다. 또한, 보조 용언은 기존의 체계와는 달리 어미와 용언의 성격을 모두 가짐을 알 수 있었다. 따라서 구문 형태소 생성을 위해 형태소 분석 단계에서 고려 사항은 보조 용언의 처리 문제이다.

보조 용언형 구문 형태소는 어미의 역할을 함께 수행하므로 형태소 분석기는 용언이 발견되면 어미와 마찬가지로 보조 용언도 함께 검색한다. 2.1 절에서 언급하였듯이 원래의 보조 용언은 사전에 등재되지 않으므로 검색된 보조 용언은 모두 구문 형태소이다. 만일 구문 형태소가 검색되면 형태소 분석기는 다시 어미 혹은 보조 용언을 검색하고 실패하면 보조 어미를 일반 어미로 바꾼다. 이를 알고리즘으로 기술하면 다음과 같다.

(단계 1) 어미 혹은 보조 용언을 사전에서 검색한다.
(단계 2) 검색이 실패하면, 이전 분석 결과가 보조 용언인지를 판별한다.
(단계 2-1) 이전 결과가 보조 용언이면 보조 용언의 보조 어미 부분을 일반 어미로 변경하고 생성 성공을 알린다.
(단계 2-2) 보조 용언이 없으면 다시 이전 결과에서 보조 용언을 찾아 발견되면 (단계 2)를 수행하고 발견되지 않으면 생성 실패를 알린다.
(단계 3) 성공한 단어가 보조 용언이면 이를 스택에 저장하고 (단계 1)을 수행한다.
(단계 4) 성공한 단어가 일반 어미이면 일반적인 어미 처리를 수행하고 생성 성공을 알린다.

<표 3>은 어절 "가고 싶지 않기 때문이다"를 위의 알고리즘에 따라 분석한 형태소 결과를 보여준다. 여기서, 대문자 V는 용언류, E는 어미류, J는 조사류를 의미하며, 소문자 v는 동사, x는 보조 용언, k는 명사형 전성 어미, s는 술어 조사, e는 일반 종결 어미를 의미한다.

[가고 싶지 않기 때문이다]
가/Vv
고 싶/Ex +hope
지 않/Ex +neg
기 때문/Ek +cause
이/Js
다/Ee

<표 3> 구문 형태소 분석의 예

4. 실험 및 평가

4.1 실험 환경 및 결과

실험을 위한 코퍼스는 수집한 한겨레신문의 인터넷 기사를 이용하였다. 수집된 문장은 50,784 개이었으며, 평균 문장 길이는 134.67바이트였다. 띄어쓰기를 기준으로 하였을 때 전체 어절은 916,823개로(기호는 제외하고 영어, 한자, 숫자 등은 포함), 한 문장 평균 어절 수는 18.08개였다. 심벌을 제외한 전체 형태소 개수는 2,816,287개였으며, 이 중 구문 형태소는 45,543개로 전체 형태소 중 차지하는 비율은 1.62%였다.

본 형태소 분석기는 복합 형태소를 생성한다. 복합 형태소는 괄호나 인용 부호로 묶인 어절을 하나의 형태소 단위로 보는 것으로 이 또한 구문 분석의 부담을 줄이는 데 기여한다. 예를 들어, 문장 "소렌스탐(스웨덴)이 우승하였다"에 대한 형태소 분석 결과는 <표 4>와 같다.(z는 미지어를, r은 고유 명사를, l은 문장 부호, q는 용언화 접미사, f는 선어말 어미를 각각 의미한다. 아울러, Y는 심볼류를 의미한다.) 분석 결과 이러한 복합 형태소는 9,984개가 나타났으며, 전체 형태소 중 차지하는 비율은 0.35%였다.

```
[소렌스탐(스웨덴)이]  ⇒ 복합형태소
   소렌스탐/Nz
   (/Yl 스웨덴/Nr )/Yl
   이/Jj
 [우승하였다]
   우승/Nn
   하/Jq
 었/Ef
   다/Ee
```

<표 4> 복합 형태소의 예

제안된 통합형 형태소 분석기는 펜티엄 PC 3.2GHz 상에서 Windows XP에서 동작하는 cygwin 환경 하에서 평가되었다. 컴파일러는 gcc 버전 3.4.4를 사용하였으며 최적화 옵션은 -O2를 사용하였다. 전체 문장을 분석하는 데 걸린 시간은 3.019초였으며, 1M당 평균 0.477초 걸렸다.

4.2 구문 형태소 생성 평가

구문 형태소 생성의 정확도 평가는 오분석율과 누락율로 결정할 수 있다. 생성된 전체 구문 형태소 개수를 A이라 하고 그 중 잘못 분석된 결과를 B라 할 때 오분석율은 'B/A'으로 결정할 수 있다. 누락율은 생성하지 못한 구문 형태소의 비율을 의미하며, 생성하지 못한 구문 형태소의 개수를 C라 할 때 누락율은 'C/(A-B)'로 결정할 수 있다.

오분석율의 계산은 전체 생성된 구문 형태소 중에서 잘못된 결과를 찾아내면 되므로 그리 어려운 작업이 아니다. 하지만, 누락율은 전체 형태소 결과를 살펴봐야 하기 때문에 만만치 않은 작업이다. 이러한 과중한 검사 작업을 피하기 위해 별도로 구문 형태소로 검색되었으나 뒤에 독립어가 나타나 구문 형태소 생성이 취소된 경우를 저장하였다. 이렇게 해서 얻어진 취소된 구문 형태소의 개수는 987개였다.

내용	각 항목 개수		비율
오분석	전체 구문형태소(A)	45,543	0.16%
	잘못된 구문형태소(B)	72	
누락	취소된 구문형태소	987	0.02%
	누락된 구문형태소(C)	8	

<표 5> 구문 형태소의 오분석율과 누락율

이러한 구문 형태소 생성 방법은 기본적으로 구문 형태소를 먼저 찾고 취소 여부를 판별하므로 이러한 검증 방법을 사용해도 무방하다. 그러나 체언 혹은 용언 자체가 제대로 분석되지 못하면 구문 형태소를 아예 찾지 않으

므로 누락될 가능성이 있다. 본 평가에서 이런 경우는 논외로 한다. <표 5>는 구문 형태소 생성에 대한 오분석율과 누락율을 보여준다. 구문 형태소 생성 시 발생하는 오분석과 누락의 대부분 문제는 미지어의 잘못된 추정 때문이었다. 미지어 추정이 완전해지면 거의 100%의 성공율을 보일 것으로 기대한다.

기존 방법과 구문 형태소의 생성 속도를 비교를 위해 분석된 형태소를 입력으로 구문 형태소를 생성할 때 걸린 시간은 위 코퍼스 문장에 대해 동일한 구현 및 실행 환경 하에서 17.64초가 소요되었다. 생성의 주된 시간은 형태소 분석 노드를 탐색하고 각 노드의 태그를 비교하는 데 소요되었다. 제안된 방법은 형태소 분석과 통합하여 구문 형태소를 생성하므로 생성 시간만을 따로 측정하는 것이 어렵다. 하지만, 본 논문의 목적이 생성 시간의 정확한 측정이 아니라 기존 방법과 비교이므로 생성 시간을 어림잡아도 문제는 없을 것이다.

우선 구문 형태소는 평균 2.2개의 기본 형태소로 구성되어 있고, 사전 검색은 단 한번만 수행하므로 기본 형태소를 분석하는 시간에 3배정도 더 소요된다고 가정해도 무방하다. 또한, 4.1절에서 살펴본 바와 같이 구문 형태소와 복합 형태소의 비율은 각각 1.62%와 0.35%로 전체에서 차지하는 비중이 그리 크지 않기 때문에 하나의 기본 형태소를 생성하는 시간은 전체 걸린 시간을 전체 형태소 개수로 나누어 구해도 무방할 것이다. 전체 구문 형태소 생성 시간은 구문 형태소의 개수를 하나의 형태소를 생성하는 데 걸린 평균 시간을 곱한 것의 3배로 추정할 수 있다. 이렇게 해서 얻어진 결과는 0.15초로 기존의 방법에 비해 100배 이상 빠른 것으로 나타났다.

5. 결론

본 논문에서는 형태소 사전을 이용한 구문 형태소 생성 방법을 제안하였다. 이러한 접근은 형태소 분석과 구문 형태소 생성 과정을 통합할 수 있게 해 줄 뿐만 아니라 띄어쓰기 오류가 있는 구문 형태소도 분석 가능하게 함을 알 수 있었다. 아울러, 미지어가 포함되고 띄어쓰기 정보가 없는 문장에서 구문 형태소가 잘못 생성될 수 있음을 보였다. 이는 미지어에 대한 정확한 추정 없이는 본질적으로 불가능한 작업이다. 따라서 띄어쓰기 정보가 없는 문장에 대한 미지어 추정에 대한 연구가 요구된다.

참고 문헌

[1] 김창제, 정천영, 김영훈, 서영훈. "부분적인 어절 결합을 이용한 효율적인 한국어 분석기", 제22회 정보과학회 가을 학술발표 논문집 pp. 597-600, 1995.

[2] 박상규, 정창민, 조준모, 이상조. "최장 묶음을 이용한 효과적인 한국어 구문분석기", 제22회 정보과학회 춘계 학술발표 논문집, pp. 961-964, 1995

[3] 안미정, 옥철영. "한국어 구문 구조 분석을 위한 복수 동사 처리", 제21회 정보과학회 추계 학술발표 논문집, pp. 625-628, 1994

[4] 김재훈. "어휘정보를 이용한 형태적 모호성 축소", 한국정보처리학회 논문지, 5권 2호 pp. 351-360, 1998.

[5] 황이규, 이현영, 이용석. "형태소 및 구문 모호성 축소를 위한 구문단위 형태소의 이용", 한국정보과학회 논문지 B, 27권 7호 pp. 784-793, 2000.

[6] 김재훈. "오류-보정 기법을 이용한 어휘 모호성 해소", 한국과학기술원 전산학과 대학원 박사학위 논문, 1996.

[7] http://www.google.co.kr

[8] 배우정, 김철수, 이용석, J. I. Aoe. "이중배열 트라이 구조를 이용한 한국어 전자 사전 구축", 한국정보과학회 논문지, 23권 1호 pp. 85-94, 1996.

[9] 이근용, 박기선, 이용석. "Two-level 한국어 형태소 해석에서의 복합명사 처리", 한국정보과학회 2002년 춘계학술대회 29권 1호 pp. 505-507, 2002,

[10] http://nlp.kookmin.ac.kr/HAM/kor/

[11] 박성배, 장병탁. “최대 엔트로피 부스팅 모델을 이용한 품사 모호성 해소”, 한국정보과학회 2003년 춘계학술대회 30권 1호 pp. 522-524, 2003.

박인철

1984 전북대학교 전산통계학과 이학사

1986 전북대학교 전산통계학과 이학석사

1998 전북대학교 전산통계학과 이학박사

1992 - 현재 호원대학교 컴퓨터학부 교수

관심분야 : 한국어정보처리, 정보검색, 임베디드 소프트웨어