부분 어절의 기분석에 기반한 고속 한국어 형태소 분석 방법

(A High-Speed Korean Morphological Analysis Method based on Pre-Analyzed Partial Words)

양 승 현 † 김 영 섬 ††
(seung Hyun Yang) (Young-Sum Kim)

요 약 일반적으로 형태소 분석 방법은 실행시에 매 어절마다 코드 변환, 형태소 분리, 원형 복원 규칙 적용을 통한 분석 후보 생성, 사전 탐색을 통한 분석 후보의 여과 등의 절차를 거쳐 형태소 분석을 수행하기 때문에 실행 효율의 관점에서 효율적이지 못하다. 이러한 문제점을 완화시키기 위해 도입된 어절 단위 기분석 사전에 의한 분석은 실행시 처리에 소요되는 계산 부하를 크게 줄일 수는 있지만 어절의 수가 사실상 무한하므로 사전의 크기 문제 때문에 완전한 처리 방법이 될 수 없다는 단점이 있다. 본 논문에서는 이상과 같은 문제점을 해결하기 위해 두가지 대비되는 방법론을 결합하여 부분 어절별로 기분석 결과를 구축하여 형태소를 분석하는 방법에 대해 기술하고 있다. 이 방법에 의하면, 형태소 분리, 원형 복원 등 형태소 분석에 필요한 계산의 대부분을 실행시에 행하지 않고 기분석 결과의 구축 시에 행함으로써 실행시 계산 부하를 크게 줄일 수 있고 불필요한 중간 결과가 생성되지 않아 사전 탐색 횟수가 크게 절감되는 효과가 있으므로, 실행 효율을 크게 개선할 수 있다. 아울러 음소별 연산을 하지 않으므로 코드 변환 등에 소요되는 계산량도 전혀 필요치 않다는 특징도 있다.

Abstract Most morphological analysis methods require repetitive procedures of input character code conversion, segmentation and lemmatization of constituent morphemes, filtering of candidate results through looking up lexicons, which causes run-time inefficiency. To alleviate such problem of run-time inefficiency, many systems have introduced the notion of "pre-analysis" of words. However, this method based on pre-analysis dictionary of surface also has a critical drawback in its practical application because the size of the dictionaries increases indefinite to cover all words. This paper hybridizes both extreme approaches methodologically to overcome the problems of the two, and presents a method of morphological analysis based on pre-analysis of partial words. Under such hybridized scheme, most computational overheads, such as segmentation and lemmatization of morphemes, are shifted to building-up processes of the pre-analysis dictionaries and the run-time dictionary look-ups are greatly reduced, so as to enhance the run-time performance of the system. Moreover, additional computing overheads such as input character code conversion can also be avoided because this method relies upon no graphemic processing.

1. 서 론

한국어 형태소 분석 분야는 비교적 오랜 세월에 걸쳐 많은 연구가 이루어졌기 때문에, 이미 다양한 처리 방법을 이용한 많은 시스템이 개발되어 있으며, 이 중에는 충분히 실용가능한 정도의 성능을 달성하여 상용으로 활용되고 있는 경우도 많다. 그러나 인터넷에서 새로 생성되는 문서의 양만 해도 일일 수 GB에 달하고 있는 등 텍스트 정보의 경우 단순히 양적인 면에서만 볼 때도 이미 폭발적으로 증가하고 있으며, 이러한 상황하에서는 정보 처리에 있어서 형태소 분석에만 수십 시간을

† 종신회원 : (주) 코난테크놀로지 이사
yan@konantech.co.kr
†† 종신회원 : (주) 코난테크놀로지 대표이사
yskim@konantech.co.kr
논문접수 : 1998년 10월 15일
심사완료 : 2000년 1월 11일

소요해야 하므로 정보처리의 자동화 및 실시간화에 큰 장애가 될 수 있다. 이와 관련해 본 논문에서는 대용량의 문서처리에 적합한 고속의 형태소 분석 방법에 대해 논의하고자 한다.

기존의 대표적인 한국어 형태소 분석 방법으로는 잘 알려진 바와 같이 테이블 파싱법(tabular parsing)[1], 최장일치법[2], 음절기반법[3] 등이 있으며 방법론의 분류도 대부분 알고리즘을 중심으로 이루어지지만, 앞으로의 논의를 위해 본 논문에서는 실행시(run-time) 계산량의 정도에 따라 편의상 "전분석(full analysis)"과 "무분석(no analysis)" 처리 방법의 두가지로 분류해서 논의를 진행하고자 한다.

첫째, 전분석 방법이라 함은 실행 시에 형태소 분석의 전 과정이 이루어지는 방법론들을 총칭하며, 기존 형태소 분석 방법의 대부분이 이 부류에 속한다고 할 수 있다. 이 방법은 어절이 입력되면 통상적으로 코드 변환, 형태소 분리, 원형 복원 규칙의 적용을 통한 분석 후보 생성, 사전 탐색을 통한 분석 후보의 여과 등의 절차를 거쳐 형태소 분석을 수행하는데, 처리 범위 면에서는 강건성을 갖지만 관련 절차 및 연산이 많고 또 실행 시에 이들을 수행하기 때문에 실행 효율이 좋지 않아서 사전의 검색 횟수나 중간 분석결과의 조합수 최소화 등 꾸준히 효율 개선이 시도되고 있기도 하다[4-7]. 테이블 파싱법, 최장일치법이나 형태소 분리시에 음절의 통계적 특성을 이용하여 효율을 향상시킨 음절기반 방법 등 대부분의 규칙기반형 형태소 분석 방법이 실행시 분석을 전제로 하고 있으므로 전분석 범주에 속하게 된다.

둘째, 무분석 방법은 기분석 어절 사전을 이용한 분석 방법으로도 알려져 있는데, 이 방법의 핵심은 어절에 대한 형태소 분석 결과를 미리 구축하여 사전에 저장해 놓고, 실행 시에는 전분석 방법과 같은 복잡한 분석 절차를 거치지 않고 사전을 탐색하여 직접 형태소 분석 결과를 출력한다는 것이다. 이 방법은 규칙 중심의 전분석 형태소 방법에 비해 실행 효율을 크게 개선할 수 있고, 또 규칙이나 알고리즘의 수정이 아닌 사전 데이타의 추가 및 수정으로 처리범위를 확장하기 때문에 규칙으로 처리하기 곤란한 예외적인 어절에 대해 효과적인 처리를 할 수 있다는 장점도 있다. 그러나 어절의 수는 사실상 무한하므로 모든 어절에 대한 분석 결과를 미리 등록하는 것이 불가능한 이상, 무분석 방법만으로는 독자적인 형태소 분석 시스템을 구축할 수가 없고 강건한 처리 능력을 갖는 전분석 모듈과 결합되어 쓰여야 한다. 실제로 무분석 방법은 소규모의 기분석 사전을 이용한 예외 어절의 처리나 대규모의 적중률 높은 기분석 어절 사전을 이용한 실행 효율의 향상 등을 위해 전분석 방법과 결합하여 자주 쓰이고 있다[8].

이상과 같이 기존 방법론을 살펴본 바에 의하면 결국 실행 효율을 높이기 위해서는 알고리즘의 부분적 개선보다도 실행시 계산량을 줄이는 것이 효과적이며, 이의 달성과 함께 처리 범위의 강건성을 확보하는 것 역시 중요한 관건이 됨을 알 수 있다.

2. 기존 연구 및 접근 방법

만약 기존의 규칙기반 전분석 방법과 기분석 어절사전 기반 무분석 방법이 서로에 대해 갖는 장점이면서 모순되는 부분이기도 한 강건성과 효율성을 절충하는 것이 가능하다면, 강건하면서도 효율적인 분석 방법을 개발할 수 있다는 것이 앞으로의 논의의 출발점이 된다. 그렇다면 여기에서 우선적으로 고려할 수 있는 사실은, 분석 효율을 높이기 위해서는 무분석 방법론의 기분석 개념을 최대한 수용해야 하고 이 과정에서 발생하는 강건성 저하의 문제는 전분석 방법에서 취하는 분석 후보의 조합 방식을 통해 해결할 수밖에 없다는 것이 된다. 결국 이러한 바탕 위에서 도출될 수 있는 접근 방법은 기분석 개념을 수용하되 전체 어절 단위가 아니라 어절의 일부인 부분 어절에 대해 기분석 결과를 구축하여 이용하는 것이 된다.

물론 그 동안 어절의 일부에 대해 기분석 개념을 도입한 연구 사례도 있었는데, 주로 형태소가 활발히 결합되고 어형 변화도 심하지만 그 수가 유한하여 구축이 용이하고 활용 효과도 큰 문법 형태소를 중심으로 제한적인 범위 내에서 연구가 많이 이루어졌다. [9]에서는 조합 수가 적은 선어말 어미를 대상으로 결합 처리를 시도하고 있고, 복합 조사나 어미 등의 문법 형태소로 구성된 기분석 부분 어절도 적은 수의 유한 집합을 형성하고 구축이 용이하기 때문에 많이 활용되고 있다[10-12]. 문법 형태소 외에도 실질 형태소를 중심으로 기분석 개념을 도입한 연구 사례도 있다. [13]에서는 음운 변동이 심한 형태소인 용언의 활용형과 축약형에 대해 기분석 개념을 도입하여 사전에 용언의 활용형을 등재하여 처리하는 방법과 이때의 사전 환경에 대해 제안하고 있다.

이상에서 실행 효율을 높이기 위해 몇몇 연구에서 도입한 부분 어절별 기분석 방법에 대해 살펴보았다. 이상에서 알 수 있듯이 이 방법은 심한 음운 변동으로 인해 처리가 복잡해지고, 또 비교적 그 수가 적어서 사전 구축이 용이한 경우에 한해 제한적으로 활용되어 왔다. 그러나 이 경우에도 코드 변환, 형태소 분절, 원형 복원,

후보 여과 등 여러 단계의 규칙기반형 처리 절차를 거칠 수밖에 없으므로 실행 효율의 병목을 완전히 해소하기가 어렵다고 할 수 있다. 이때, 만약 모든 형태소와 모든 음운 현상에 대해 나타날 수 있는 모든 변동 결과를 미리 구축하여 부분 문자열별 기분석 결과를 구축할 수 있다면, 전분석 방법의 절차를 모두 생략할 수도 있을 것이다. 그리고 이것이 가능하다면 이러한 부분 어절별 기분석에 기반한 접근 방법은 굳이 예외 현상 처리의 용이 등의 일반적인 장점을 열거하지 않더라도 다음과 같은 면에서 주목할 만하다.

첫째, 형태소 분석에 필요한 계산의 대부분이 실행 시에 발생하지 않고 기분석 데이타의 구축 시에 발생한다는 점이다. 즉, 형태소의 교착 시에 발생하는 형태소의 변이 현상을 실행 시에 처음부터 계산을 하는 것이 아니고 미리 계산하여 저장하고 있다가 그 결과만을 가져다가 바로 분석 결과를 출력할 수 있게 되므로 계산의 반복을 피할 수 있다. 이에 따라 형태소 분리, 음운 변동, 사전 탐색 등의 반복적인 절차로 인해 파생되는 계산량의 대부분을 실행 시에 배제할 수 있고, 문자열 조작에 필요한 정도의 복잡도로 분석 과정을 수행할 수 있으므로, 처리 시간을 크게 감축할 수 있다. 원형 복원 규칙이나 사전 정보 등의 처리 지식이 실행 시에 필요없게 됨은 물론이다. 따라서 실행 시에 필요한 연산은 부분 문자열별 기분석 결과의 탐색과 각 부분 결과를 전체로 연결하는 정도면 되므로, 분석 알고리즘이 매우 단순하게 된다.

둘째, 자소 단위의 연산이 필요없으므로 조합형 코드로의 변환 과정이 필요없게 되어 어떤 코드 체계이든 그대로 처리할 수 있게 된다. 즉, 중간 분석 결과의 생성이 기분석 결과의 탐색만으로 이루어지고, 기분석 결과의 탐색을 위해서는 문자열이 자소 단위든 음절 단위든 아무런 문제가 없으므로 자소별 연산이 필요없다면 어절을 굳이 자소열로 변환할 필요가 없다. 코드 변환이 없으므로 문서 처리 환경에서 실제적인 문제가 되는 다양한 코드 체계, 확장 코드, 비트 오류로 인한 코드 변질 등에도 강건성을 갖는다. 또한 입력 어절이 초·중·종성으로 분리되면 결과적으로 입력 어절의 길이를 3배로 증가시키게 되며, 이에 따라 테이블 파싱 등 중간 결과의 조합 과정에서 공간 복잡도도 9배 증가하는 등의 단점이 있다. 연산의 기본 단위를 자소가 아닌 음절로 할 수 있다면 이러한 복잡도 문제도 크게 줄일 수 있을 것이다.

3. 기분석 부분 어절

3.1 기분석 부분 어절의 구축

위에서 설명한 기분석 부분 어절을 이용한 접근 방법의 핵심은 한국어 어절에서 관측되는 모든 부분 어절에 대해 분석 결과를 미리 저장하고 있다가 분석시에 이용한다는 것이다. 표층 어절에 출현할 수 있는 모든 부분 문자열에 대한 분석 결과를 등록하는 한가지 효과적인 방법은 형태소 사전에 수록된 모든 형태소에 대해 그 형태소가 처할 수 있는 모든 형태·음운적 조건에서 음운 변동을 시켜서 표층형으로 합성하는 방법이다. 즉, 각 형태소가 속할 수 있는 음운 환경에 대한 모든 — 실제로는 그리 많지 않은 — 표층형을 미리 등록할 수만 있다면, 역으로 표층 문자열만 보고도 어떤 형태소들이 그 속에 결합되어 있는지를 알 수 있다. 이와 같이, 부분 어절별 기분석 결과는 기본적으로는 사전에 등재된 형태소 목록에 대해 형태·음운적 제약에 따른 변형을 거쳐 표층형으로 활용하는 과정을 거쳐 만들어진다.

명사, 대명사와 같은 체언이나 수식언 중 부사, 관형사 등은 표층형에서 어형 변화가 거의 없어서 사전에 수록된 원형과 표층형이 동일하게 되므로, 원형 문자열을 그대로 표층형으로 하고 교체, 축약, 음소 병합 등으로 인해 어근이 변하는 경우(예. 누구+가->누가)만 찾아서 추가해 주면 되기 때문에 표층형 등록이 간단하다. 그러나, 용언의 경우에는 정칙은 물론 불규칙 활용(ㄷ, ㅂ,ㅅ,ㅎ,거라,너라,르,러,우,여), 축약(ㅏ/ ㅓ/ ㅐ, ㅒ/ ㅖ, ㅗ/ ㅜ, ㅣ, ㅟ), 탈락(으,ㄹ) 등 각 형태소가 처한 형태·음운적 환경에 따라 어형이 변하는 경우가 많기 때문에, 체언의 경우와는 달리 형태소 원형으로부터 표층형을 합성하는 과정이 필요하다. 이것은 형태소 합성 규칙의 적용을 통해 쉽게 해결할 수 있는 문제이므로 비교적 어형 변화가 많은 용언의 경우에도 표층형 등록에 큰 문제가 없음을 알 수 있다 (5.1절 참조). 또한 어미 등에서 형태소 원형에 음절이 아닌 음소가 있는 경우(예를 들어, 'ㄴ다' 등)는 표층형에서 해당 음소가 인접하는 형태소와 합성되어 표층형 음절에서는 나타나지 않기 때문에, 음소 표현을 표층형에서는 분리하여 내부정보(접속정보)로 저장해야 한다. 이 외에도 예외 현상이나 구어체나 신문기사 등에서 자주 볼 수 있는 조사나 어미의 축약 표현, 붙여쓰기가 가능한 표현 등 규칙의 합성만으로 처리가 곤란한 경우 등록해 줄 수 있다.

이와 같이 얻어진 표층형 데이타는 형태소 원형으로부터 유도된 것이기 때문에 표층형 문자열과 그것을 구성하는 형태소들의 원형 정보를 포함하고 있는 기분석 정보가 된다. 따라서 탐색의 키는 형태소 원형 문자열이 아닌 표층형 문자열이 되며 탐색의 결과는 해당 표층형

문자열에 대한 분석 결과가 된다. 본 논문에서는 이 표층형 문자열을 문자 위치에 따라 순차적으로 탐색하기에 적합한 트라이 구조로 저장하고 있다. 이때 어미, (복합)조사와 같은 기능어에서 유도된 표층 문자열은 입력 어절의 후위에서부터 후진 탐색하기 때문에 표층 문자열의 역순으로 저장하고, 그밖의 문자열은 전진 탐색을 위해 원래대로 저장한다.

3.2 접속 정보(병합 조건, 인접 조건)

한편 이상에서 언급한 방법의 구현에 있어서 발생되는 문제점으로는, 첫째로 전체 어절이 두 개 이상의 부분 어절로 분할될 때 각 부분별 형태소 리스트끼리 결합 가능한지 혹은 결합 불가능한지를 판별해야 하는 문제와, 둘째로 한 형태소의 음절이 다른 형태소의 음절과 음운 변동을 통해 하나로 합쳐져서 표층형에서는 음절 단위로는 형태소간의 경계를 구분지을 수 없다는 문제를 들 수 있다. 이를 위해 기분석 데이타의 구축시에 형태소 리스트 외에도 각 형태소 리스트의 결합 혹은 배제의 판별 근거가 되는 접속 정보도 함께 저장해야 한다. 접속 정보는 형태소가 처할 수 있는 형태·음운적 환경을 나타내는 것이므로, 표층 부분 문자열이 아닌 각 형태소 리스트에 부착된다.

접속 정보를 정의하기 이전에 먼저 음절별 음운 정보에 대해 정의하면 정의 1과 같다. 본 논문의 방법은 문자열을 코드 변환없이 직접 조작하므로 접속 정보에 명시된 음운 조건을 위해 각 음절에 대한 음운 정보를 2,390개의 한국어 음절에 대해 미리 구성해 놓아야 한다.

정 의 1 : 음운 자질 c, l, v, p, n은 각각 유종성 음절, 'ㄹ'로 끝나는 유종성 음절, 무종성 음절, 양성 모음으로 끝나는 유종성 음절, 음성 모음으로 끝나는 유종성 음절을 나타낸다. 음절이 갖는 **음운 정보**는 해당 음절을 구성하는 자소에 따라 결정되며, 자질 집합 $\{c, l, v, p, n\}$의 부분 집합이다.

예를 들어, 음절 '었'은 유종성 음절이고 중성이 음성 모음이므로 음운 정보는 $\{c, n\}$이 된다.

한편 한 형태소의 음절이 다른 형태소의 음절과 음운 변동을 통해 하나로 합쳐져서 표층형에서는 음절 단위로는 형태소 분절점을 찾을 수 없는 경우가 있다. 예를 들어 형태소 '가/VV'와 'ㄴ다/EF'는 표층형에서는 "간다"로 합성되기 때문에 음절 경계로는 두 형태소를 분절할 수가 없다. 만약 각 부분 문자열 "간"과 "다"에 대해 형태소 원형 리스트를 구한다면 각각 [가/VV ㄴ/ER(관형형 어미)], [ㄴ다/EF(일반 어미)]가 되는데, 두 리스트를 단순히 연접하면 [가/VV ㄴ/ER ㄴ다/EF]가 되어 'ㄴ'의 중복을 피할 수 없게 된다. 이런 경우에 올바른 결과를 얻기 위해서는 리스트의 결합 시에 선행 'ㄴ/ER'이 'ㄴ다/EF'에 이미 포함되어 있으므로 삭제되어야 함을 알려주는 정보가 있어야 한다. 이와 같이 형태소의 일부가 선행하는 표층 음절에 병합되어 있다는 조건을 본 논문에서는 **병합 조건**이라 한다 (정의 2 참조).

또한 형태소의 분석을 위해서는 위 병합 조건과 함께 한 형태소 리스트가 다른 리스트와 결합 가능한지를 여부를 나타내는 정보가 필요하다. 예를 들어, 입력 어절 "소설가"의 전위열(prefix), 후위열(suffix)인 "소설"과 "가"에 대해 기분석 사전을 참조하면 각각 [소설/NN(명사)]과 [가/PP(조사)], [가/SN(명사파생접미사)] 등의 형태소 리스트를 얻을 수 있다. 이런 경우에 [소설/NN]과 [가/PP]가 결합되어 [소설/NN 가/PP]와 같은 오분석 결과가 생성되는 것을 배제하고 올바른 결과를 얻기 위해서는 리스트의 결합 시에 [가/PP]는 선행 형태소가 명사류이며 마지막 음절이 무종성 음절, 즉 모음으로 끝날 때에만 쓰인다는 것을 나타내는 조건이 있어야 한다. 이와 같이 형태소의 인접 환경에 대한 제약을 나타내는 조건을 본 논문에서는 **인접 조건**이라 하며 형태 제약, 음운 제약, 품사 제약을 그 내용으로 하고 있다 (정의 2 참조).

이 병합 조건과 인접 조건은 형태소의 분절과 결합 관계에 대한 제약을 나타내는 정보이므로 사전에서 형태소 리스트와 함께 제공되어야 하며 분석 과정에서 중요한 역할을 담당한다. 이를 합쳐 **접속 정보**라 부른다 (정의 2 참조).

정 의 2 : **접속 정보**는 **인접 조건**(adjacency condition)과 **병합 조건**(mergency condition)으로 구성되며 $\langle M, P, T, G\rangle$의 4-투플(quadruple)로 표현된다. 이때 M, P, T가 형태, 음운, 품사 제약을 표현하는 인접 조건이 되며 각각 형태소/품사쌍의 집합 혹은 $-$(NULL 값), 음운 자질의 집합, 품사의 집합을 값으로 갖는다. G는 병합 조건이 되며 형태소/품사쌍 혹은 $-$을 값으로 가진다. 접속 정보가 부착된 형태소 리스트 $l=_{LC}[w_1/t_1, \ldots, w_n/t_n]_{RC}$(단, w_i는 형태소, t_i는 품사, LC, RC는 접속 정보)라고 표기할 때 LC, RC는 각각 l의 좌접속 정보, 우접속 정보라고 한다.

"간다"의 예에서 볼 때, [ㄴ다/EF]의 좌접속 정보 중 병합 조건은 $G=$ㄴ/ER이 된다. 또한 "소설가"의 예에서 [가/PP]의 좌접속 정보 중 인접 조건은 $M=-$, $P=\{v\}$, $T=\{NN, NP, NU, SN\}$이 되며, 병합 조건은 $G=-$이 되므로 전체 좌접속 정보는 $\langle -, \{v\}, \{NN, NP, NU, SN\}, -\rangle$가 된다. 이 접속 정보를 간

단히 'Nv'와 같이 기호로 표기(부록 참조)하기로 한다. 이에 따라 기분석 부분 어절 사전에서는 표층 문자열 "가"에 대해 (Nv)[가/PP]를 포함하게 된다 (물론 조사 '가' 외에도 명사 '가', 접미사 '가' 등 여러 가지 형태소가 존재하므로 실제로는 한 문자열에 대해 여러 개의 리스트가 연관되어 있다.)

3.3 병합 조건의 검사

기분석 트라이를 탐색하면 입력 어절의 부분별 형태소 리스트를 구할 수 있으며, 이로부터 전체 결과를 구하려면 서로 다른 부분 형태소 리스트를 하나로 결합하는 과정이 필요한데, 이 과정에서 접속 정보가 활용된다. 먼저 병합 조건을 검사하게 된다. 병합 조건이 없으면(즉, NULL 값) 형태소 중복이 존재하지 않는 상황이므로 바로 인접 조건 검사를 할 수 있다.

만약 병합 조건이 존재하면 중복 형태소가 존재한다는 의미이므로 먼저 삭제할 수 있는지를 판별하는 과정이 필요하다. 예를 들어, "간다"를 사전에서 탐색하면 "간"과 "다"에 대해 각각 l_1=[가/VV ㄴ/ER], l_2=(<ㄴ/ER>)[ㄴ다/EF]가 탐색된다 (단, 접속 정보 <ㄴ/ER>=<-, {}, {}, ㄴ/ER>) 이때 l_2의 좌접속 정보 중 병합 조건은 NULL이 아니므로 먼저 l_2의 좌측에 위치하게 될 형태소(즉, l_1의 마지막 형태소)가 이 병합 조건에 의해 삭제될 수 있는 형태소인지 아닌지를 판별하는 일이 필요하다. 이 판별은 중복된 형태소가 병합 조건에 명시된 형태소와 같은 형태소인지를 검사하는 것을 주 내용으로 하고 있으며 전·후위열 검사를 통해 이루어진다 (정의 3 참조).

정의 3 : w,x는 형태소, t,y는 품사라고 할 때, 형태소의 전·후위열 검사 함수(각각 σ_P, σ_S로 표기)는 다음과 같이 정의된다.

$$\sigma_{P(S)}(w/t, x/y) = \begin{cases} \text{true} & x\text{가 } w\text{의 전위열(후위열)이면} \\ \text{false} & \text{그렇지않으면} \end{cases}$$

위의 예에서 l_2의 좌접속 정보 중 병합 조건 G=ㄴ/ER이므로 l_1의 마지막 형태소인 'ㄴ/ER'에 대해 σ_S(ㄴ/ER, ㄴ/ER)=true가 되는데 이것은 곧 'ㄴ다/EF'의 일부인 'ㄴ'이 중복되어 있음을 의미하므로, 이 'ㄴ'이 제거되어야 함을 알 수 있다. 이때, l_1의 마지막 형태소 'ㄴ/ER'은 l_2의 첫 번째 형태소 'ㄴ다/EF'에 **병합가능**하다고 한다 (정의 4 참조). 그리고 중복된 형태소가 삭제된 결과로 얻어지게 되는 형태소 리스트는 **병합 리스트**라고 하며 l_1'=[가/VV]가 된다 (정의 4 참조). 만약 병합 조건이 '-'인 경우는 중복 형태소가 없으므로 사전에서 주어진 원래의 형태소 리스트가 그대로 병합 리스트가 된다.

정의 4 : 형태소 리스트 $l_1=[v_1/s_1, \ldots, v_m/s_m]_{RC}$ (v_i는 형태소, s_i는 품사), $l_2={}_{LC}[w_1/t_1, \ldots, w_n/t_n]$ (w_i는 형태소, t_i는 품사)이다. 이때 l_1의 우접속 조건 RC=<M, P, T, G>이고 $\sigma_P(w_1/t_1, G)$=true라면, l_2는 l_1에 **병합가능**하며 l_2에서 w_1/t_1을 제거하면 **병합 리스트** $l_2'={}_{LC}[w_2/t_2, \ldots, w_n/t_n]$가 된다. 만약 G=-(null)이면 l_2는 l_1에 **병합가능**하며 **병합 리스트** $l_2'=l_2$가 된다. 역으로, l_2의 좌접속 조건 LC=<M, P, T, G>이라고 하자. 이때, $\sigma_S(v_m/s_m, G)$=true이면 l_1은 l_2에 **병합가능**하다고 하며 l_1에서 v_m/s_m을 제거하면 **병합 리스트** $l_1'=[v_1/s_1, \ldots, v_{m-1}/s_{m-1}]_{RC}$ 가 된다. 만약 G=-(null)이면 l_1은 l_2에 **병합가능**하며 **병합 리스트** $l_1'=l_1$가 된다.

3.4 인접 조건의 검사

병합 리스트는 중복 형태소의 제거가 이루어져 형태소간의 음절 단위 분절이 명확해진 형태소 리스트이므로, 다음으로 두 병합 리스트가 연접될 수 있는지를 검사하면 된다. 이것은 인접 조건의 검사를 통해 이루어진다. 만약 인접 조건이 존재한다면 두 병합 리스트의 연접 시에 형태·음소적인 제약이 존재한다는 의미이므로 두 병합 리스트가 이를 만족하는지를 판별하는 과정이 필요하다. 예를 들어, 병합 리스트 l_1=[소설/NN]이고 l_2=(Nv)[가/PP]라고 하자 (단, 접속 기호 Nv = <-, {v}, {NN, NP, NU, SN}, ->). 이때 M = -이므로 형태소에 대한 제약은 없음을 알 수 있고, 음운 제약 P = {v}이므로 무종성 음절이 인접해야 함을 알 수 있고, 품사 제약 T = {NN, NP, NU, SN}이므로 명사류가 인접해야 함을 알 수 있다. 각각에 대한 판별은 정의 5와 같이 이루어 진다.

정의 5 : 접속 조건 C=<M, P, T, G>이라고 하고 임의의 문자(음절) a에 대한 음운 정보를 $\pi(a)$라고 하자. 이때 C가 좌(우)접속 정보이면 각 인접 조건 M, P, T에 대한 검사는 $a_m^{L(R)}$, $a_p^{L(R)}$, $a_t^{L(R)}$와 같이 정의된다. (단, w는 형태소, t는 품사라고 하자)

$$a_m^{L(R)}(w/t, M) = \begin{cases} \text{true} & \text{if } M = - \\ \text{true} & \text{if } \exists x \in M, \sigma_{S(P)}(w/t, x) \\ \text{false} & \text{otherwise} \end{cases}$$

$$a_p^{L(R)}(w/t, P) = \begin{cases} \text{true} & \text{if } P = - \\ \text{true} & \text{if } \exists x \in P, x \in \pi(a_k) \text{ where } w = a_1 \ldots a_k \\ \text{false} & \text{otherwise} \end{cases}$$

$$a_t^{L(R)}(w/t, T) = \begin{cases} \text{true} & \text{if } T = \{\ \} \text{ or } t \in T \\ \text{false} & \text{otherwise} \end{cases}$$

위의 "소설가"의 예에서 l_2의 좌접속 조건에 의해 a_m^L(소설/ NN, −)=true, a_f^L(소설/ NN,{NN,NP,NU,SN})= true이지만, 음운 제약 a_p^L(소설/ NN,{v}) =false (π(설)={c,l,n})가 되어 l_2는 l_1에 인접가능하지 않음을 알 수 있다. 이 예는 조사 '가'가 유종성 음절 뒤에는 결합될 수 없다는 것을 의미하고 있다. 이러한 인접 조건의 검사는 정의 5와 6에 정의되어 있다.

정의 6 : 형태소 리스트 $l_1=[v_1/s_1,\ldots,v_m/s_m]_{RC}$ (v_i는 형태소, s_i는 품사), $l_2=_{LC}[w_1/t_1,\ldots,w_n/t_n]$ (w_i는 형태소, t_i는 품사)가 있고, l_2의 좌접속 조건을 $LC=\langle M,P,T,G\rangle$ 이라고 하자. 이때 $a_m^L(v_m/s_m,M)=\text{true}\wedge a_p^L(v_m/s_m,P)=\text{true}\wedge a_f^L(v_m/s_m,T)=\text{true}$이면 l_1은 l_2에 **인접가능**하다고 한다. 역으로 l_1의 우접속 조건 $RC=\langle M,P,T,G\rangle$ 이라고 하자. 이때 $a_m^R(w_1/t_1,M)=\text{true}\wedge a_p^R(w_1/t_1,P)=\text{true}\wedge a_f^R(w_1/t_1,T)=\text{true}$이면 l_2는 l_1에 **인접가능**하다고 한다.

3.5 형태소 배열법 검사

형태소 리스트의 연접에 관한 제약은 위에서 설명한 접속 정보(병합 조건, 인접 조건)에 의한 제약 외에도 어절의 형태소 배열법(morphotactics)에 의한 제약이 있다. 이 제약은 적법한 한국어 어절 내의 품사 전이, 즉 어절 내에서 선행 형태소의 품사 뒤에 어떤 품사가 뒤따를 수 있는지를 나타낸다. 만약 형태소 리스트를 연결할 때 접속 정보에 의한 제약 조건이 충족되었다고 해도, 허용되지 않는 품사 전이가 발생하면 리스트의 연결이 불가하다. 이 품사 전이에 대한 정의는 정의 7과 같다.

정의 7 : **품사 전이**는 어절 내에서 선행 형태소의 품사 뒤에 어떤 품사가 뒤따를 수 있는지를 나타내며 다음과 같이 품사 행과 품사 열의 행렬로 표현된다.

$$\tau(t,u)=\begin{cases}\text{true} & \text{어절 내에서 품사 } t \text{ 다음에 품사 } u\text{가 나타날 수 있으면}\\ \text{false} & \text{그렇지 않으면}\end{cases}$$

형태소 리스트 $l_1=[v_1/s_1,\ldots,v_m/s_m]$ (v_i는 형태소, s_i는 품사), $l_2=[w_1/t_1,\ldots,w_n/t_n]$ (w_i는 형태소, t_i는 품사)라고 할 때, $\tau(s_m,t_1)=true$이면 l_1은 l_2로 **전이가능**하다고 한다.

예를 들어, 품사 전이 행렬은 τ(VV,EF)=true, τ(NN,EF)=false의 값을 갖는다. 이 예는 한국어의 형태소 배열법에서 분명하게 명시되어 있는 경우들이지만 한편으로 보조용언의 붙여쓰기나 띄어쓰기 오류 등의 처리를 위해 τ(EF,VV)≈true 등을 허용할 수도 있다.

3.6 형태소 리스트의 연결

형태소 리스트의 연결은 음절간 경계가 명확해진 병합 리스트를 대상으로 이루어진다. 병합 리스트가 1) 인접가능하고 2) 전이가능하면 두 리스트를 연결할 수 있다. 이상을 바탕으로 형태소 리스트와 형태소 리스트의 연결에 대해 정의해 보겠다.

정의 8 : 임의의 형태소 리스트 l_1, l_2가 있다. 이때 l_1이 l_2에 병합가능하고 l_2가 l_1에 병합가능하다고 하자. 그리고 l_1, l_2의 병합 리스트를 각각 l_1', l_2'로 표기하고 $l_1'=_{LC_1}[v_1/s_1,\ldots,v_m/s_m]_{RC_1}$ (v_i는 형태소, s_i는 품사), $l_2'=_{LC_2}[w_1/t_1,\ldots,w_n/t_n]_{RC_2}$ (w_i는 형태소, t_i는 품사)라고 하자. 이때, l_1'이 l_2'에 인접가능하고 l_2'이 l_1'에 인접가능하며, l_1'이 l_2'으로 전이가능하면, l_1과 l_2는 **결합가능**하다고 하며 다음과 같이 결합($\oplus$로 표기)되어 형태소 리스트 l이 된다.

$$l=l_1\oplus l_2=_{LC_1}[v_1/s_1,\ldots,v_m/s_m\ w_1/t_1,\ldots,w_n/t_n]_{RC_2}$$

4. 형태소 분석

4.1 분석 알고리즘

본 논문에서 임의의 어절 $a_1a_2\ldots a_n$에 대한 분석 후보는 형태소 리스트로 표현된다. 일반적으로 한 어절은 하나 이상의 분석 후보를 갖게 되므로, 한 어절에 분석 결과는 형태소 리스트의 집합으로 표현되는데, 앞으로의 논의를 위해 $a_1a_2\ldots a_n$에 대한 분석 결과를 $\overline{a_1a_2\ldots a_n}$로 표기하기로 한다. 이에 의하면 형태소 분석은 $a_1a_2\ldots a_n$에서 $\overline{a_1a_2\ldots a_n}$을 발견하는 과정이라고 볼 수 있으며, 전분석 방법은 형태소 분리와 원형 복원 규칙에 기반하여 $\overline{a_1a_2\ldots a_n}$을 발견하는 방법이고 무분석 방법은 모든 $a_1a_2\ldots a_n$에 대해 $\overline{a_1a_2\ldots a_n}$을 사상시켜서 직접 분석 결과를 얻는 방법이라고 볼 수 있다.

어절 $a_1a_2\ldots a_n$이 주어졌을 때 부분 어절에 대한 분석은 기분석 트라이를 탐색하는 것으로 간단히 구할 수 있으므로, 실제 처리에서는 규칙에 의해 형태소를 분석하는 과정은 필요없고 부분 결과를 전체로 연결하는 과정이면 충분함을 알 수 있다. 이를 위해 본 논문에서는 조합에 의한 결합에 효과적으로 이용할 수 있는 CYK 알고리즘을 이용하고 있다. CYK 테이블의 (j,i) 셀 ($1\le i\le n$, $1\le j\le n-j+1$)은 $\overline{a_ia_{i+1}\ldots a_{i+j-1}}$(즉, 입력 어절의 i번째 위치부터 길이가 j인 부분 문자열에 대한 분석 후보의 집합)이 된다. $\overline{a_ia_{i+1}\ldots a_{i+j-1}}$에 속하는 모든 분석 후보는 입력 어절의 $i+j-1$번째 문자까지의 분석 결과이므로, 다음 위치인 $i+j$번째 문자부터 시작하는 분석 후보, 즉 $\overline{a_{i+j}a_{i+j+1}\ldots a_k}$ ($k\le n$)에 속하는 분석 후보

와만 결합 가능하다.

정의 9는 부분 어절별 형태소 리스트를 결합하는 연산에 대한 정의이다. 이 정의에 의하면, 본 논문의 형태소 분석 알고리즘은 트라이에서 탐색한 부분 어절별 분석 결과를 기본으로 하여 점점 더 긴 길이의 부분 결과를 결합해 나가면서 최종적으로는 입력 어절 전체를 포함하는 분석 결과를 발견해 나가는 동적 프로그래밍 과정임을 알 수 있다.

정의 9 : 셀 $(k-i+1, i) = \overline{a_i a_{i+1}\ldots a_k} = \{p_1, \ldots, p_m\}$, 셀 $(j-i, k+1) = \overline{a_{k+1}a_{k+2}\ldots a_j} = \{q_1, \ldots, q_n\}$ $(i \le k \le j)$이라고 하자. (p_i, q_i는 형태소 리스트) 이때 셀 $(k-i+1, i)$과 $(j-i, k+1)$의 Cartesion 곱($\odot$로 표기)으로 형태소 리스트를 결합하면 셀 $(j-i+1, i)$의 원소인 형태소 리스트가 만들어 진다.

$$\overline{a_i \ldots a_k a_{k+1} \ldots a_j} \supset \overline{a_i a_{i+1} \ldots a_k} \odot \overline{a_{k+1} a_{k+2} \ldots a_j}$$
$$= \{l \mid l = p_i \oplus q_j \text{ for } i \ge m, j \ge n\}$$

예를 들어, 입력 어절이 a_1a_2="나는"이라고 하면, 트라이 탐색을 통해 $\overline{a_1}$={[나/NN], [나/NP], [나/VV], [나/VV 어/EF], [나/VX], [나/VX 어/EF], [낫/VV]$_{(어/EF)}$, [낫/VV]$_{(었/EP)}$, [낫/AJ]$_{(어/EF)}$, [낫/AJ]$_{(었/EP)}$, [날/VV]$_{(ㄴ\ 。)}$, [낫/VV 으/UC], [낫/AJ 으/UC]}, $\overline{a_2}$={$_{(V)}$[는/EF], $_{(V)}$[는/PP]}를 얻을 수 있다. 이때 정의 9에서 볼 수 있는 바와 같이 $\overline{a_1}$과 $\overline{a_2}$의 원소들을 Cartesian 곱으로 조합하면 $\overline{a_1a_2}$의 원소를 얻을 수 있으며 그 결과는 아래와 같다.

$\overline{a_1a_2} \supset \overline{a_1} \odot \overline{a_2}$={[나/NN 는/PP], [나/NP 는/PP], [나/VV 는/EF], [나/VV 는/EF], [날/VV 는/PP]}

그림 1은 이상에서 설명한 바를 바탕으로 입력 어절 $a_1a_2\ldots a_n$을 분석하는 알고리즘이다. 먼저 모든 $\overline{a_i \ldots a_j}$를 공집합으로 초기화한 다음(1), 트라이를 첫번째 문자에서부터 전진 탐색하고 n번째 문자에서부터 후진 탐색하여 부분 어절별 분석 결과의 초기값을 가져온다(2). 그리고 나서, 부분 어절별 분석 결과를 결합하여 첫 번째 위치에서 시작하여 n번째 위치에서 끝나는 분석 결과, 즉 전체 분석 결과를 만든다(3). 만약 전체 분석 결과가 만들어져 있으면, 즉 $|\overline{a_1a_2\ldots a_n}|>0$이면 알고리즘이 끝나고, 아니면 전체 분석 결과가 아직 만들어지지 않은 것이므로 다음과 같은 연결 과정을 반복한다. 첫 번째 위치에서 시작하는 부분 결과 중 가장 길이가 긴 것을 찾아내서 길이를 i라고 하면(4), $\overline{a_1a_2\ldots a_i}$까지 현재 만들어져 있으므로 $i+1$번째 위치부터 시작하는 모든 부분 결과 $\overline{a_{i+1}a_{i+2}\ldots a_d}(i+1 \le d \le n)$를 트라이에서 탐색한 후(9), 이전에 이미 만들어져 있던 부분 결과 $\overline{a_1a_2\ldots a_j}$ $(1 \le j \le i)$ 와 연결을 시도한다(10). 이때 길의 i의 부분 결과인 $\overline{a_1a_2\ldots a_i}$에 대해서만 시도하지 않고 길이 i 이하의 모든 부분 결과에 대해서도 연결을 시도함으로써 가능한 조합을 모두 시도해 봄을 알 수 있다. 그 결과 길이 k $(i \le k \le n)$인 부분 결과 $\overline{a_1a_2\ldots a_k}$가 만들어지며, 이때 $|\overline{a_1a_2\ldots a_n}|>0$이면, 즉 전체 분석 결과 $\overline{a_1a_2\ldots a_n}$가 만들어졌으면 반복 과정이 끝나 알고리즘이 종료하고(5), 아니면 다시 첫 번째 위치에서 시작하는 부분 결과 중 가장 길이가 긴 것을 찾아서(12) 위 과정을 반복한다.

```
Alogrithm analyze_word( a1a2...an )
;;; assume max{ } = 0.
begin-algorithm
(1)   a_i a_{i+1}...a_j ← φ (1≤i≤j≤n);
(2)   search a1a2...ai and aj a_{j+1}...an (1≤i,j≤n)
(3)
      a1a2...an ← a1a2...an ∪ a1a2...ai ⊙ a_{i+1}a_{i+2}...an (1≤i≤n);
(4)    i ← max{i | |a1a2...ai|>0}
(5)      while ( i≠0 ∧ |a1a2...an|=0) do
(6)      begin-while
(7)       if ( |a1a2...ai|≠0) do
(8)         begin-if
(9)         search a_{i+1}a_{i+2}...ad (i+1≤d≤n)
(10)
      a1a2...ak ← a1a2...ak ∪ a1a2...aj ⊙ a_{j+1}a_{j+2}...ak (1≤j≤i≤k≤n)
(11)            end-if
(12)             i ← max{j | |a1a2...aj|>0 ∨ i-1}
(13)        end-while
end-algorithm
```

4.2 방법론적 장점

위에서 설명한 바와 같이, 중간 분석 후보의 결합 절차는 일반적인 CYK 알고리즘의 진행과 크게 다른 것이 없으나, 분석 후보를 생성하는 부분에서 트라이 탐색으로 모든 것이 끝난다는 점이 다름을 알 수 있다. 이 방법은 2절에서도 개략적으로 언급한 바 있지만 실행 효율면에서 다음과 같이 실제적인 장점을 갖는다.

첫째, 분석 후보를 생성하기 위한 코드 변환, 원형 복원 규칙 적용 등의 절차를 없앨 수 있다. 예를 들어, 입력 어절 "나는"에 대한 예를 살펴보자. 기존의 전분석 방법에 의하면 먼저 입력을 조합형 코드로 변환하여 자소열 "ㄴ ㅏ · ㄴ ㅡ ㄴ"으로 바꾸고, 조사/어미 사전을 탐색하여 문자열의 후위에서 어미로 추정되는 문자열 "ㄴ"과 "는"을 분리해낸 다음 문자열 경계를 중심으로 '불규

칙 활용 복원', '모음 축약 복원', 'ㄹ 탈락 복원' 등 여러 종류의 원형 복원 규칙을 적용해 본다. 규칙의 적용 결과 이 경우에는 [나는/?], [나느/? ㄴ/EF], [나늫/? ㄴ/EF], [나늘/? ㄴ/EF], [나/? 는/PP], [나/? 는/EF], [날/? 는/EF] 등 7개의 분석 후보가 생성된다. 본 방법에 의하면 이와 같은 절차를 모두 생략할 수 있다.

둘째, 무효한 분석 후보 생성을 방지하여 사전 탐색 횟수를 크게 줄일 수 있다. 전분석 방법에서는 생성된 분석 후보 중에 맞는 것을 선택하려면 사전을 탐색하여 분석 후보를 확정해야 하는 과정이 필요하다. 위의 예를 다시 보면, 7개의 분석 후보 중에서 서로 다른 문자열 "나는", "나느", "나늫", "나늘", "나", "날"에 대해 모두 6번의 사전 탐색이 필요하다[5]. "나는", "나느", "나늫", "나늘" 등의 형태를 갖는 어휘가 없으므로 앞의 네 경우는 모두 탈락되고, 최종 분석 결과 [나/NN 는/PP], [나/NP 는/PP], [나/VV 는/EF], [나/VX 는/EF], [날/VV 는/EF]를 얻을 수 있다. 이와 같이, 전분석 방법은 분석 후보를 만들어 내기 위해 여러 종류의 원형 복원 규칙을 적용해야 하므로 일단 규칙 적용에 시간이 많이 소요되고, 규칙이 적용되었다 해도 "나느", "나늫", "나늘"처럼 존재하지도 않은 무효한 중간 분석 결과를 만들어 냄에 따라 사전 탐색 횟수도 증가하여 실행 효율이 떨어지게 된다. 본 논문에서 제시한 기분석 부분 문자열을 이용한 분석 방법에서는 입력 문자열 "나는"에 대해, 정의 9의 예에서 설명하였듯이 트라이 전진 탐색 1회, 후진 탐색 1회로 모든 부분 결과를 가져오고 이의 결합으로 분석이 완료된다. 결과적으로, 기존의 규칙기반 전분석 방법은 이 예의 경우 코드 변환, 원형 복원 규칙의 적용을 필수적으로 거쳐야 하고 조사/어미 사전 탐색까지 포함하여 7회의 사전 탐색을 해야 하기 때문에 분석 효율이 많이 떨어지지만, 본 논문에서 제시한 방법은 이러한 절차를 모두 없애고 단 2회의 트라이 탐색과 결합 연산만으로 분석을 완료하므로 효율이 크게 개선됨을 알 수 있다.

셋째, 복합어, 보조용언 등 붙여쓰기로 인해 여러 어절이 하나로 합쳐진 경우에도 처리가 용이하며, 더 나아가서는 띄붙 오류 처리에도 그대로 적용될 수 있다. 예를 들어, "행복해져갔다"처럼 보조용언의 붙여쓰기로 인해 3개의 실질형태소가 하나의 어절에 포함된 경우의 분석을 보자. 이 경우에 형태소 분절점을 찾아서 원형을 복원하여 기본 형태소 사전을 통해 분석후보를 여과하는 전분석 방법으로는 많은 무효 후보의 생성과 사전 탐색을 피할 수 없으므로 효율이 크게 떨어지게 된다. 그러나 본 방법은 원형 구동 방식인 전분석과는 달리 표층형 구동 방식이므로 붙여쓰기가 일어난 어절간 경계 형태소의 표층형 부분 문자열을 등록시키기만 하면 나머지 과정에 대해서는 모두 동일한 방법으로 처리가 가능하다. 예를 들면 붙여쓰기가 가능한 '어미+보조용언' 쌍 '-어+지'에 대해서는 "어지 [어/EF 지/VX]", "어진 [어/EF 지/VX ㄴ/ER]", "어져 [어/EF 지/VX 어/EF]", "어졌 [어/EF 지/VX 었/EP]", "져 (<어/EF>)[어/EF 지/VX 어/EF]", "졌 (<어/EF>)[어/EF 지/VX 었/EP]" 등이 해당된다. 물론 이러한 어절간 표층 문자열이 등록되지 않았더라도 정의 5에서 설명한 품사 전이를 허용하면 각각의 부분 문자열에 기초하여 분석이 가능하므로 분석의 강건성은 보장되고, 역시 원형 복원이나 무효 후보의 생성 등이 일어나지 않으므로 전분석 방법에 비해서는 훨씬 효율이 좋아짐을 알 수 있다. 대신에 본용언인지, 보조용언인지를 알 수 없으므로 두가지의 분석 결과가 나오게 된다. 위의 예에서는, '-어+지'에 대해서는 표층형이 등록되었다고 해도 "해져"라는 부분 문자열에 대해 형태적 중의성(형용사 '해지다'에 대한 활용형도 가능)이 존재하므로 부분 문자열 "해져"에 대해서도 등록을 해 주면 효과적으로 처리할 수 있다.

표 1은 이 경우에 알고리즘의 동작에 따른 중간 결과의 변화를 예시하고 있다. 처음에 전진과 후진 트라이 탐색을 해서 "행복", "다"에 대한 부분 결과를 가져오고 (결합되지 않는 부분 결과는 생략했음), 이를 결합해도 전체 분석 결과가 나오지 않으므로 반복에 들어간다. 반복 1에서는 다음 부분 문자열 위치인 "해"에서 시작하는 전진 탐색을 하여, 탐색 결과를 기존 중간 결과에 연결한다. 이와 같은 과정을 반복하면 표에 제시된 바와 같이 전체 분석 결과가 만들어졌으므로 반복 3에서 알고리즘이 끝난다. 이 경우, '-어+가'에 대해서는 등록되지 않았다고 해도 기능어 탐색을 포함해도 총 4회(각각 "행복", "해져", "갔", "다")의 사전 탐색만으로 분석이 끝날 수 있음을 알 수 있다.

5. 실험 및 평가

5.1 기분석 부분 어절 구축 실험

전술한 바와 같이 본 논문의 방법론에서는 새로운 부분 어절 엔트리의 추가만으로도 시스템의 확장이 가능하다. 이때 필요한 기분석 부분 어절의 범위에는 특별한 제약이 없으며, 하나 이상의 형태소가 결합되어 표층 어절에서 실현된 형태이면 된다. 여기에는 형태소 자체의 변이형이나 활용형, 또는 어휘형태소+문법형태소, 어휘형태소+어휘형태소, 문법형태소+어휘형태소, 문법형태소

표 1 단계별 진행 및 중간 결과의 예

loop	대응 문자열	사전 탐색 결과	중 간 결 과
0	행복하다	[행복/NN](하/SJ) (<었/EP>)[었/EP 다/EF]	[행복/NN](하/SJ), ... (<었/EP>)[었/EP 다/EF], ...
1	해져	[하/SJ 어/EF 지/VX 어/EF]	[행복/NN 하/SJ 어/EF 지/VX 어/EF], ... (<었/EP>)[었/EP 다/EF], ...
2	갔	[가/VV 었/EP] [가/VX 었/EP]	[행복/NN 하/SJ 어/EF 지/VX 어/EF 가/VV 었/EP] [행복/NN 하/SJ 어/EF 지/VX 어/EF 가/VX 었/EP] (<었/EP>)[었/EP 다/EF]
3	-	-	[행복/NN 하/SJ 어/EF 지/VX 어/EF 가/VV 었/EP 다/EF] [행복/NN 하/SJ 어/EF 지/VX 어/EF 가/VX 었/EP 다/EF]

표 2 사전을 기반으로 한 기분석 부분 어절의 구축

품사	구분	변동 조건	원형 →표층형	원형 갯수	표층형갯수
체언	명사,대명사,		사과/NN → 사과 [사과/NN]	162,725	162,725
용언	정칙		먹 VV → 먹 [먹/VV]	24,861	24,861
	ㄷ불규칙	어,었	듣/VV+어 → 들 [듣/VV](어/EF)	41	82
	ㅂ불규칙	ㄴ,르,ㅁ,으,어,었	아름답/VJ+ㄴ→ 아름다운 [아름답/VJ ㄴ/ER]	523	3138
	ㅅ불규칙	어,었	긋/VV+어 → 그 [긋/VV](어/EF)	51	153
	ㅎ불규칙	ㄴ,르,ㅁ,ㅂ,어,었	파랗/AJ+ㄴ → 파란 [파랗/AJ ㄴ/ER]	107	642
	거라불규칙	거라	가/VV+어라 → 가거라 [가/VV 어라/EF]	59	59
	너라불규칙	너라	오/VV+너라 → 오너라 [오/VV 어라/EF]	44	44
	르불규칙	어,었	가르/VV+어 → 갈라 [가르/VV 어/EF]	135	270
	러불규칙	어,었	이르/VV+어 → 이르러 [이르/VV 어/EF]	8	16
	우불규칙	어,었	푸/VV+어 → 퍼 [푸/VV어/EF]	1	2
	여불규칙	어,었	당하/VV+어 → 당하여 [당하/VV 어/EF]	2837	11348
	'으' 탈락	어,었	쓰/VV+어 → 써 [쓰/VV 어/EF]	242	968
	무종성 용언	ㄴ,르,ㅁ,ㅂ	가/VV+ㄴ → 간 [가/VV ㄴ/ER]	10,416	41,664
	'ㄹ' 탈락	ㄴ,르,ㅁ,ㅂ,ㄴ*,오*,시	살/VV+ㄴ → 산 [산/VV ㄴ/ER]	503	8,048
	'ㅏ/ㅓ/ㅕ'축약	어,었	가/VV+어 → 가 [가/VV 어/EF]	523	1,046
	'ㅐ/ㅔ' 축약	어,었	매/VV+어 → 배 [매/VV 어/EF]	2,012	4,024
	'ㅗ/ㅜ' 축약	어,었	오/VV+어 → 와 [오/VV 어/EF]	473	946
	'ㅣ' 축약	어,었	이기/VV+어 → 이겨 [이기/VV 어/EF]	3,986	7,972
	'ㅚ' 축약	어,었	되/VV+어 → 돼 [되/VV 어/EF]	111	222
	예외 현상	ㄷ*,ㅈ*,라	말/VV+라 → 마 [말/VV](라/EF)	2	6
수식언	부사		조금/AD → 조금 [조금/AD]	6,282	6,282
	관형사		그/DT → 그 [그/DT]	144	144
독립언	감탄사		후유/IJ → 후유 [후유/IJ]	399	399
기능어	어미		었/EP+는가/EF+는/PP → 었는가는 (n)[었/EP 는가/EF 는/PP]	1,906	51,605
	(복합)조사		으로/PP+보다/PP+는/PP → 으로보다는 (c)[으로/PP 보다/PP 는/PP]	2,415	2,415
기타	중의적 표제어		가르/VV+마/EF → 가르마 [가르/VV 마/EF]	0	15,208
	조사 축약		나/NP+는/PP → 난 [나/NP 는/PP]		
	어미 축약		하/SV+지/EF → 치 [하/SV 지/EF]		
	어미+보조용언		어/EF+지/VX+어/EF → 어져 [어/EF 지/VX 어/EF]		
총합				220,806	344,289

+문법형태소의 결합형 등 여러 가지가 가능하며 사실상 하나 이상의 형태소의 표층형이면 전체 어절이라도 상관이 없음을 알 수 있다. 가장 기본이 되는 부분 어절 엔트리는 3.1절에서 설명한 바와 같이 사전에 수록된 형태소를 형태소 합성 규칙을 통해 음운 변동시켜서 얻은 것이다. 또한 자동으로 구축하기 어려운 것들에 대해서는 수작업으로 등록을 할 필요가 있다. 이러한 것들은 대개 예외 현상처럼 알고리즘의 동작을 보완하는 성격의 데이타인 경우가 많은데, 예를 들어 구어체나 신문기사 등에서 자주 볼 수 있는 조사나 어미의 축약 표현, 붙여쓰기가 가능한 보조용언 표현 등을 들 수 있다. 전자의 경우는 기본 분석 알고리즘만으로는 과생성의 위험 때문에 처리가 곤란하므로 등록을 하는 것이 좋고, 후자의 경우는 기본 알고리즘으로도 처리가 가능하지만 본용언과 보조용언의 구분 등 좀더 개선된 결과를 얻기 위해서 등록하는 것이 좋은 경우이다.

본 논문에서는 이상과 같은 내용으로 실제 기분석 부분 어절 사전을 구축했는데, 표 2에서 보듯이 220,806개의 형태소가 수록된 사전을 이용해서 344,289 개의 표층 부분 문자열을 얻었으므로 한 형태소 당 평균 1.56개의 표층형을 갖는다는 것을 알 수 있다. 이러한 사실은 표층형 등록을 형태소 단위의 부분 어절로만 한정할 경우 기본 사전의 2배 미만 정도로 사전 크기가 커지면서도 표층 어절에서 발견되는 모든 부분 어절을 등록할 수 있다는 의미이므로, 어절 단위로 기분석 결과를 등록할 때 문제가 되었던 사전 크기가 부분 어절별로 등록할 때는 문제가 되지 않음을 알 수 있다.

5.2 실험 및 평가

구축된 기분석 부분 문자열 사전을 이용한 형태소 분석기를 구현하여 평가를 하였다. 평가용 문장으로는 일일 증가량이 크므로 최근 들어 처리의 자동화 요구가 매우 큰 신문 기사를 택했고, 평가 결과의 신뢰성을 위해 특수문자, 로마자 등은 모두 제외시켰다. 평가용 집합의 크기는 82,949 어절이며 평균 어절 길이는 3.42 음절이다. 표 3에 제시된 실험 및 평가 결과를 보면 본 논문의 실험을 위해 구현된 형태소 분석기의 재현율은 98.57%로서 신조어, 외래어, 고유명사 등의 미지어 부류나 띄어쓰기 오류 등이 많은 신문 기사의 특성을 고려한다면 분석 능력에는 문제가 없음을 알 수 있다. 오히려 실험 경험에 의하면 본 방법은 띄붙 오류가 발생할 경우에도 부분 문자열별로 분석 결과를 가져와서 결합하므로 분석 과정이 간단, 강건해져서 재현에 실패하는 경우를 상당수 방지할 수 있었다.

다음은 분석 속도에 대한 평가 결과이다. 통상 분석 속도의 측정치로는 "word/sec"나 "byte/sec"를 많이 이용하지만 이 측정값은 CPU 처리 속도, 디스크 입·출력 성능 등 특정 하드웨어마다 다름은 물론, 품사 분류 체계, 사전의 크기 등에도 크게 영향을 받는 값이기 때문에, 본 논문에서는 "word/sec", "byte/sec" 단순 속도 측정치보다는 실행 성능에 영향을 미치는 항목별 통계를 주요 성능 평가값으로 제시하고자 한다[1]. 첫째, 코드 변환, 원형복원 규칙 적용 횟수는 모두 어절당 0회로서 이 부분의 처리에 소요되는 시간이 전혀 없다. 둘째, 분석 효율에 가장 큰 영향을 미치는 사전 탐색 복잡도에 대해 알아보겠다. 사전 탐색 요구 횟수의 최악 복잡도는 다음과 같이 계산할 수 있다. 어절의 길이(=음절수)를 n이라 할 때, 어절 위치 i에서 보면 다음 어절 위치에서 사전 탐색이 필요한 중간 분석 후보는 길이별로 모두 $(n-i)$ 개가 된다. 이때 최악의 경우는 $(n-i-1)$ 개에 대해 사전을 탐색해서 모두 실패하고 마지막에 가서야 다음 위치와 연결 가능하게 되는 탐색 결과를 가져오는 경우가 된다. 이런 식으로 계산하면 최악 사전 탐색 요구 횟수는 $\sum_{i=1}^{n}(n-i)+2=\frac{n(n-1)}{2}+2$(2는 반복에 들어가기 전의 초기 사전 탐색 횟수)가 되어 최악의 경우를 상정하더라도 사전 탐색 요구 횟수의 복잡도는 $O(n^2)$이 됨을 알 수 있다[2].

표 3 실험 및 평가 결과

평가 항목		평가치	비고
재현율		98.57 %	
재현 실패 원인	미지어(신조어)	19.15 %	재현에 실패한 1.43%에 대한 원인별 분석
	미지어(외래어,고유명사)	38.29 %	
	띄어쓰기 오류	17.02 %	
	접두사	12.77 %	
	기타	12.77 %	
	소계	100.00 %	

1) 참고로 신문기사 텍스트에 대해 Pentium-II 333MHz PC에서 대략 172분/GB의 성능을 보였다.

2) 본 방법에서 사전 탐색은 입력 어절의 위치별로 이루어지기 때문에 실제로 사전 탐색의 경우의 수는 각 위치별로 모두 n개의 서로 다른 경우만이 가능하다. 따라서 실행시에 같은 위치에서 반복적으로 사전 탐색 요구가 발생하여 최악 사전 탐색 요구 횟수가 $O(n^2)$이 되었다 하더라도, 같은 위치에서 이전에 탐색한 결과가 있다면 중복해서 탐색이 일어날 필요가 없으므로 탐색 요구 횟수와는 달리 실제 탐색이 일어나는 횟수는 최대 n 번이 되어 실제 탐색 횟수의 최악 복잡도는 $O(n)$이 된다.

표 4 어절 길이별 분포와 사전 탐색 횟수

길이	길이별 구성비(누적분포)	사전탐색횟수
1	25.05% (25.05%)	1.02
2	19.57% (44.62%)	2.03
3	24.89% (69.50%)	2.20
4	15.63% (85.14%)	2.49
5	8.36% (93.49%)	2.77
6	4.00% (97.49%)	2.87
7	1.51% (99.00%)	3.44
8	0.56% (99.57%)	3.93
9	0.23% (99.80%)	4.52
10	0.11% (99.91%)	4.88
11	0.05% (99.96%)	5.62
12	0.02% (99.98%)	6.42
13	0.01% (99.99%)	7.48
14	0.01%(100.00%)	7.20
15	0.00%(100.00%)	8.00
평균		2.31

다음으로 평균 탐색 복잡도에 대해 알아보겠다. 표 4에서 어절 길이(음절수)별로 보면 대략 $\frac{n}{2}$번의 사전 탐색이 발생함을 알 수 있으므로 평균 탐색 복잡도는 $O(n)$이 됨을 알 수 있다. 또한 어절 길이별 구성비와 누적 분포를 보면 97.49%의 어절들, 즉 거의 모든 어절들이 3회 미만의 사전 탐색으로 분석이 완료됨을 알 수 있다. 전체적으로 보면, 평균 사전 탐색 횟수는 어절당 평균 2.31회가 된다. [7]에서는 전분석 방법인 최장일치법에서 사전 탐색 횟수를 절감하는 방법을 통해 결과를 제시하였는데, 이에 의하면 실질어 2.04회와 기능어 2.08회를 합쳐 어절당 평균 4.12회의 사전 탐색이 이루어진다. 이에 비해 볼 때 본 논문에서 제시한 결과는 기존의 방법들에 비해 효율이 크게 개선된 결과임을 알 수 있다. 이러한 결과는 본 논문에서 제시한 방법의 특성상 무효 중간 분석 후보가 생성되지 않기 때문이며, 사전 탐색횟수가 줄어든 만큼 성능의 개선이 이루어졌다고 할 수 있다.

이상 실험 및 평가 결과에서 살펴 보았듯이, 본 논문에서 제시한 방법은 기존의 규칙기반 전분석 방법에서 많은 처리시간을 소모하던 부분을 크게 개선시키고 있음을 알 수 있다. 또 한편으로는, 부분 어절별 기분석 데이타를 어절 전체 단위로 확장하여 이용한다면, 즉 기분석 어절도 다수 등록시켜 이용한다면 성능을 더욱 개선할 수 있다. 이러한 기분석 어절 사전을 이용하기 위해 별도의 모듈 구성이 필요치 않으며 동일 방법으로 단지 데이타를 추가하는 수준에서 이를 수용할 수 있다.

6. 결 론

이상에서 부분 어절별 기분석을 이용한 고속의 형태소 분석 방법에 대해 설명하였다. 본 방법은 기분석 데이타를 이용하기 때문에 형태소 분석 시스템의 실행시에 행해지던 규칙의 적용을 데이타 구축시에 행함으로써 실행시 소요 시간을 없앨 수 있으며 사전 탐색 횟수의 절감 등의 효과도 가져올 수 있다. 정리해 보면, 본 논문에서 제시한 형태소 분석 방법은 기존의 방법론에 비해 다음과 같은 특징을 갖는다.

- 분석 과정 중에 반복적으로 발생하는 계산을 미리 계산 및 저장하여 이용하는 방법이므로, 처리 성능이 크게 개선되어 대용량 문서의 고속 분석에 적합하다.
- 분석 후보를 만들어 내는 데 소요되는 원형 복원 규칙을 적용하지 않기 때문에 규칙의 적용에 소요되는 절차 및 처리 시간을 없앨 수 있다.
- 원형 복원 규칙을 적용하지 않기 때문에 무의미한 중간 분석 후보를 생성하는 것을 방지할 수 있다.
- 분석 후보 중에서 문법적으로 올바른 것을 선택하기 위해 사전을 탐색해야 하는 절차를 없애서 사전 탐색에 소요되는 처리 시간을 없앨 수 있다.
- 일반적인 처리 방법으로 분석되지 않는 예외적인 현상을 처리하기 위해 별도의 모듈, 규칙, 사전 구성을 하지 않고 부분 어절별 기분석 데이타의 등록만으로 쉽게 예외 어절을 분석해 낼 수 있다.
- 신조어와 같은 미분석 어절을 분석하기 위해서는 규칙, 사전 등을 수정할 필요없이 부분 어절별 기분석 데이타를 등록하기만 하면 되므로 시스템의 확장 및 유지보수가 용이하게 된다.
- 연산의 기본 단위가 음절에 해당하는 2바이트 코드이므로 완성형이든 조합형이든 코드 변환이 필요없고 따라서 코드 변환에 소요되는 처리 시간을 없앨 수 있다.
- 코드 변환 절차가 필요없으므로 특정 한글 코드와 무관하게 형태소 분석 장치를 구성할 수 있다.

이상에서 살펴본 바와 같이 방법론적으로 볼 때 본 논문에서 제시한 방법의 특징은, 매우 단순한 알고리즘으로 형태소 분리나 불규칙 활용, 탈락, 축약 등 복잡한 형태론적인 현상을 처리할 수 있다는 것이다. 이것은 물론 언어 지식에 관련된 부분이 실행시가 아닌 사전 구축시로 옮겨가기 때문이며, 또한 실행시에는 언어독립적 방법으로 분석이 행해지므로 타언어에 대한 형태소 분석에도 그대로 적용할 수 있다는 장점도 있다.

참 고 문 헌

[1] 김성용, 최기선, 김길창, "Tabular Parsing 방법과 접속정보를 이용한 한국어 형태소 분석기", 한국정보과학회 춘계 인공지능발표논문집, pp. 133--147, 1987.

[2] 최재혁, 이상조, "양방향 최장일치법을 이용한 한국어 형태소 분석기", 한국정보과학회 봄 학술발표논문집, Vol. 20, No. 1, pp. 769--772, 1993.

[3] 강승식, "다층 형태론과 한국어 형태소 분석 모델", 제6회 한국어 정보처리 학술발표 논문집, pp. 140--145, 1994.

[4] 강승식, 김영택, "사전 정보에 기반한 효율적인 한국어 형태소 분석기의 설계 및 구현", 한국정보과학회 봄 학술발표논문집, Vol. 18, No. 1, 1991.

[5] 김영택, 자연언어처리, 교학사, 1994.

[6] 이은철, 이종혁, "계층적 기호 접속정보를 이용한 한국어 형태소 분석기의 구현", 제 4회 한글 및 한국어 정보처리 학술발표 논문집, pp. 95--104, 1992.

[7] 최재혁, 이상조, "양방향 최장일치법을 이용한 한국어 형태소 분석기에서의 사전 검색 횟수 감소 방안", 한국정보과학회 논문지, Vol. 20, No. 10, 1993.

[8] 김재한, 옥철영, "통합형태소를 이용한 한국어 형태소 분석기", 한국정보과학회 가을 학술발표논문집, Vol 21, No. 2, pp. 653--656, 1994.

[9] 강승식, 김영택, "한국어 형태소 분석기에서 선어말어미의 분석 모형", 한국정보과학회 논문지, Vol. 18, No. 5, pp. 505--513, 1991.

[10] 권혁철, 채영숙, 김재원, 김민정, "한국어 철자 검색을 위한 형태소 분석 기법", 우리말 정보화 큰잔치, pp. 179--186, 1991.

[11] 김재한, 안미정, 옥철영, "활용 형태소에 기반한 한국어 형태소 분석기", 한국정보과학회 가을 학술발표논문집, Vol. 20, No. 2, 1993.

[12] 김재한, 옥철영, "어절 사전을 이용한 한국어 형태소 분석", 한국정보과학회 봄 학술발표논문집, Vol. 21, No. 1, pp. 813--816, 1994.

[13] 조영환, 차희준, 김길창, "확장 사전 환경에서의 한국어 형태소 해석과 생성", 제5회 한글 및 한국어 정보처리 학술발표 논문집, pp. 355--368, 1993.

부록. 접속정보(인접 조건, 병합 조건)의 예

기호	〈M, P, T, G〉	기호	〈M, P, T, G〉
J	<-,{ },{AJ, AX, SJ, CP},->	c	<-,{c},{ },->
Jc	<-,{c},{AJ, AX, SJ, CP},->	n	<-,{n},{ },->
Jn	<-,{n},{AJ, AX, SJ, CP},->	p	<-,{p},{ },->
Jp	<-,{p},{AJ, AX, SJ, CP},->	v	<-,{v},{ },->
Jv	<-,{v},{AJ, AX, SJ, CP},->	v+ㄹ	<-,{v,1},{ },->
N	<-,{ },{NN, NP, NU, SN},->	Vv+ㄹ	<-,{v,1},{VV, VX, SV},->
Nv	<-,{v},{NN, NP, NU, SN},->	Jv+ㄹ	<-,{v,1},{AJ, AX, SJ, CP},->
V	<-,{ },{VV, VX, SV},->	<ㄴ/ER>	<-,{ },{ },ㄴ/ER>
Vc	<-,{c},{VV, VX, SV},->	<ㄹ/ER>	<-,{ },{ },ㄹ/ER>
Vn	<-,{n},{VV, VX, SV},->	<ㅁ/EN>	<-,{ },{ },ㅁ/EN>
아/EF	<{아/EF},{ },{ },->	<아/EF>	<-,{ },{ },아/EF>
어/EF	<{어/EF},{ },{ },->	<어/EF>	<-,{ },{ },어/EF>
았/EP	<{았/EP},{ },{ },->	<았/EP>	<-,{ },{ },았/EP>
었/EP	<{었/EP},{ },{ },->	<었/EP>	<-,{ },{ },었/EP>
ㄴ-*	<{나/EF, 냐/EF, 네/EF, 니/EF,시/EP,...},{ },{ },->	...	

양 승 현

1990년 서울대 공과대학 컴퓨터공학과 학사. 1992년 서울대 대학원 컴퓨터공학과 석사. 1997년 서울대 대학원 컴퓨터공학과 박사. 1997년 ~ 1999년 한국전자통신연구원 선임연구원. 1999년 ~ 현재 (주)코난테크놀로지 이사 겸 부설 멀티미디어정보기술연구소장. 관심분야는 교차언어 정보검색, 멀티미디어 정보검색, 문서 분류, 문서 요약, 기계번역, 자연언어처리, 진화 알고리즘

김 영 섬

1983년 한양대학교 전자통신공학과 공학사. 1985년 한양대학교 공과대학원 공학석사. 1989년 한양대학교 공과대학원 공학박사. 1989년 ~ 1999년 한국전자통신연구원 선임연구원. 1992년 ~ 1993년 미국 Bellcore Basic Research Center 객원연구원. 1999년 ~ 현재 (주)코난테크놀로지 대표이사. 관심분야는 디지털 아카이브, 멀티미디어 정보검색, 교차언어 정보검색, 기계번역, 자연언어처리