한국어 형태소 처리와 사전

——접속정보를 이용한 한글 철자 및 띄어쓰기 검사기——

김덕봉 · 최기선 · 강재우

1. 서 론

오늘날 사람들은 문서작성, 출판 등을 하는 데 컴퓨터를 이용하는 일이 많아졌다. 이것은 컴퓨터 기술의 급격한 발달로 인한 컴퓨터 가격의 인하와 고급 문서 편집기의 개발에 기인한다. 특히 컴퓨터를 이용한 출판 기술의 발달은 개인 뿐만 아니라 출판사, 신문사 등에서도 문서 작성과 출판을 종래보다 빠르고 저렴하게 할 수 있게 하였다. 다시 말하면 출판사나 신문사 등이 납활자를 일일이 사람 손으로 찾아서 판을 짜던 전통적인 방식에서 벗어나 이제는 컴퓨터를 이용하여 온라인으로 편리하게 사식을 함으로써 이들이 좀 더 생산적인 일을 할 수 있게 하였다는 것이다. 그러나 컴퓨터를 이용한다고 해서 원고의 입력에서부터 출판까지 완전히 자동으로 되는 것은 아니다. 현재의 컴퓨터 기술이 이것을 처리하기에는 너무 부족한 점이 많기 때문이다. 따라서 아직까지는 원고의 입력은 사람이 해야하고, 입력된 원고의 교정도 사람이 해야만이 완벽한 결과를 얻을 수 있다. 앞으로 이와 같은 일은 영상 이해 시스템(image understanding system), 언어 이해 시스템(language understanding system) 등의 연구 결과에 따라 컴퓨터가 대신하여 줄 수 있을 것이다. 우리의 관심 사항은 입력된 원고를 어떻게 하면 컴퓨터가 자동으로 교정하여 줄 수 있을까 하는 데 있다.

일반적으로 원고를 교정하는 것에는 두 가지 형태가 있다〔미승우 1988〕. 우선 그 하나를 살펴보면 원고에서 틀린 철자나 띄어쓰기 오류 등을 찾아 맞춤법에 맞게 고치는 작업이 있다. 이를 보통 교정이라고 한다. 둘째는 원고에서 철자는 틀리지 않았으나 의미상 맞지 않는 단어나 구문상 위치가 잘못된 단어, 구두점이나 의미상 나와야 할 단어가 나오지 않은 단어, 문맥상 어울리지 않은 문장을 찾아 이를 바르게 고치는 교열이 있다. 예를 들어 살펴보자.

(1) 꽃이 피어 넜다. -> 꽃이 피어 있다.(철자오류)

(2) 꽃이 있다 피어. -> 꽃이 피어 있다.(구문오류)
(3) 꽃이 피어 없다. -> 꽃이 피지 않는다./꽃이 없다.(의미오류)

(1)의 경우는 '있다'를 '닜다'로 잘못 쓴 철자 오류이다. 이와같은 오류를 찾아 바르게 고치는 것을 교정이라 한다. (2)와 (3)은 철자 오류는 없으나 구문 오류와 의미 오류가 있음을 알 수 있다. 이런 오류를 고치는 것을 교열이라 한다.

본 논문은 한글 텍스트를 분석하여 철자오류와 띄어쓰기 오류를 찾는 시스템인 HSPELL에 대하여 한국어 형태소 처리와 사전 구성 관점에서 기술한다. HSPELL은 한 어절 내에서 형태소들의 접속 가능성을 나타내는 좌우접속정보표, 사전, 그리고 사전 편집기로 구성되어 있다. HSPELL은 최단일치법을 이용하여 한 어절 내의 형태소를 분리하고, 좌우접속정보를 이용하여 이 형태소들이 접속 가능한가를 검사한다. 만약 접속될 수 없으면, 철자 또는 띄어쓰기 오류로 간주하고 그 어절을 출력한다. 이와 같은 HSPELL은 UNIX 4.3 BSD 상에서 C 언어〔KERN 78〕로 구현되어 동작되고 있다.

HSPELL은 다음과 같은 특성을 갖고 있다. 첫째, HSPELL은 한 어절에서 추출할 수 있는 가능한 모든 형태소들 중에서 가장 짧은 형태소를 먼저 선택하는 방법 즉, 최단일치법(Shortest Match Strategy)을 이용하여 철자 및 띄어쓰기 오류를 검사하기 때문에 처리 속도가 비교적 빠르다. 둘째, HSPELL은 추출된 형태소들이 접속 가능한가를 검사하기 위하여 좌우접속정보가 들어가 있는 접속행렬(connection matrix)을 이용한다. 셋째, HSPELL은 사전의 기본단위를 형태소로 하고, 사전에는 형태소와 좌우접속정보만을 갖도록 함으로써 비교적 사전의 크기가 작으면서도 제기능을 충분히 발휘하며, 또 사전에 새로운 정보의 추가가 용이하다. 넷째, HSPELL은 한글에 나타나는 언어 현상을 처리하는 데 있어 기존의 관련 시스템 〔김성용 1987, 김효준 1988, 한국과 1988〕에서 한 프로시듀얼 처리(procedural processing)를 피하고, 이를 테이블화하여 처리함으로써 유지, 보수가 용이하다.

앞으로 HSPELL은 철자 및 띄어쓰기 오류 검사 뿐만 아니라 철자 교정도 가능하게 할 예정이다. 선진 외국에서는 1960년대부터 철자 검사 및 교정기(speller)에 관한 연구가 계속되어 현재 실용적인 시스템이 많이 나와 문서 작성이나 데이타베이스 제작에 상당히 많은 도움을 주고 있다. 우리도 이에 대한 연구를 서둘러 한글 문서 작성에 도움을 주고, 한글 데이타베이스 제작에 일조하며, 더 나아가서는 정보화 사회를 앞당기는 역할을 해야 할 것이다.

2. 관련 연구

한글 철자 및 띄어쓰기 오류를 찾는 데에는 앞의 예문에서도 보았지만 하나의 어절에 대해서 형태론적인 분석만을 하면 된다. 따라서 이와 밀접한 관련이 있는 형태소 분석에 관한 기존의 연구와 영어 및 한글 철자 교정에 관한 기존의 연구를 살펴보는 것은 매우 중요하다. 다음은 기존의 형태소 분석에 관한 연구와 철자 교정에 관한 연구에 대하여 기술한다.

2.1. 형태소 분석 시스템

형태소는 일정한 음성에 일정한 뜻이 결합되어 있는 말의 가장 작은 단위이다. 즉, 최소의 유의적 단위(minimum meaningful unit)로 정의된다〔남기심 1988, 이기동 1987〕. 형태소분석기(morphological analyzer)는 기계 번역 또는 한국어 처리를 하기 위하여 하나의 어절이 어떠한 형태소들로 구성되어 있고, 각 형태소와 관련된 정보(품사, 원형, 의미 등)가 무엇인가를 분석하는 시스템이다. 이러한 의미에서 한글 철자 및 띄어쓰기 검사기는 하나의 어절 내에서 형태소들이 접속 가능한가만을 파악하므로 형태소 분석기의 하부 시스템(subsystem)이라 할 수 있다.

기존의 한국어 형태소 분석기에서 주로 사용한 방법들을 살펴보면 다음과 같다.

2.1.1. 최장일치법(Longest Match Strategy)

한 어절 내에서 분절이 가능한 여러 형태소 중에서 가장 긴 형태소를 선택하는 방법이다〔김효준 외(1988), 한국과학기술원(1988)〕. 예를 들면 '감기는'에 대해 앞 부분에서 최장일치법을 이용한 경우 '감기(cold)+는'을, 뒷 부분에서 적용한 경우 '감(wind)+기는'을 추출한다. 이와 같이 이 방법은 모호성(ambiguity)에 대한 처리를 전혀 하지 않기 때문에 다음 단계에서의 백트래킹(backtracking)을 유발한다.

2.1.2. Head-Tail 구분법

어절을 변형되지 않는 부분(Head)과 변형되는 부분(Tail)으로 구분하여 어절의 뒷 부분에서부터 가능한 모든 Tail을 찾아가면서 Tail 테이블을 구성한다. 그 다음 어절의 앞 부분에서는 가능한 Head를 찾고, 그 Head와 각 Tail들과의 접속 가능성에 대하여 Tail 테이블에 들어 있는 정보를 이용하여 검사하는 방법이다〔최형석 1984〕.

이 방법은 음운 현상이나 불규칙 현상을 처리하기 위하여 용언을 스물 여

섯 가지, 어미를 열 한 가지로 세분하고 이들 사이의 결합관계를 나타낸 표를 이용하고 있다. 그러나 이 방법 역시 어절의 형태를 제한하고 결합관계를 나타내는 표가 미흡하기 때문에 모든 형태의 어절에 대해 완전한 분석을 하기가 어렵다.

2.1.3. Tabular parsing 방법

CYK(Cocke-Younger-Kasami) 알고리즘의 변형으로 각 열(row)에 대해서 그 열의 첫 자소로 시작되는 문자열을 한번의 탐색을 거치는 동안 모두 찾는다.

길이가 n인 문자열에 대해 크기가 $(n \times n)/2$인 삼각 테이블을 T라 하면, $T(i, j)$는 i번째 자소로부터 시작하여 j개의 자소로 구성된 형태소이다. 이때 $T(i, k)$와 $T(i+k, j-k)$가 접속가능하면 $T(i, j)$를 만든다. 길이가 n인 어절일 경우 $T(1, n)$을 찾는 것이 목적으로서 동적 프로그래밍 기법을 이용한다[김성용 1987].

여기서는 불규칙 현상 및 음운 현상을 검사하기 위해 오토마타(automata)를 사용한다. 또한 형태소 분석 중 발견된 현상들을 기록하기 위해 참조표(reference table)를 사용한다. 참조표에 기록된 현상은 규칙틀(rule frame)을 이용하여 원형으로 복구되고, 이 원형을 가지고 사전을 찾는다. Head-Tail 방법과 유사한 시도로서 형태소들의 접속 가능성을 검사하기 위하여 접속정보행렬을 사용한다. 이 접속정보행렬은 형태소를 좌접속 범주와 우접속 범주로 세분하여 고유번호를 부여하고, 이를 표로 나타낸 것이다.

2.2. 철자 검사 및 교정 시스템

2.2.1. 영어 철자 교정 시스템

영어 철자 교정 시스템은 1962년 항공기 승객 이름의 오류를 수정하는 방법[DARI62]이 최초로 제시된 후, 많은 연구가 진행되었다[BLAI60, DAME 64, ALBE67, PETE80]. 특히 UNIX에서는 문서 준비 과정에서 필요한 여러 가지 일을 지원하는 WWB(Writer's Workbench) 소프트웨어를 제공하고 있다[CHER83, FRAS83].

영어는 단어 단위로 띄어쓰기 때문에 띄어쓰기 오류가 거의 일어나지 않는다. 또한 한 단어의 굴절이 심하지 않으므로 비교적 간단한 방법으로 어휘 분석(lexical analysis)을 할 수 있다. 현재 영문을 위한 철자 교정은 상당한 연구가 진행되어 실용적인 단계에 있다.

2.2.2. 한글 철자법 교정 시스템

한글 철자 검사 및 교정에 관한 연구는 최근에 시작되었고, 기존의 시스

템으로는 한글 철자법 교정 시스템〔김영웅 1984〕이 있다.

이 시스템은 문서 편집 중에 철자를 교정하는 대화식 시스템으로서 COBOL 언어로 NEC/S100 상에서 구현되었다. 이 시스템은 입력 문서가 한글 맞춤법에 맞게 되어 있어야 하는 등 많은 가정을 하고 있다. 접사를 찾는 방법은 최장일치법과 조사/어미를 역순으로 구성한 테이블을 이용하여 어절의 뒤에서부터 찾는 방법을 택하였다. 어절의 뒤에서부터 조사/어미를 분리하고 남은 나머지를 사전에서 찾아, 그것이 없으면 오류로 간주하여 사용자에게 교정 및 사전등록 여부를 묻는다. 교정된 어절은 교정단어사전에 등록하여 다시 오류가 발생할 때 자동으로 교정한다. 따라서 일단 한번 에러를 범한 단어에 대해서는 다시 같은 오류를 범하지 않는다.

3. 한국어에 대한 고찰

자연 언어를 컴퓨터가 처리하기 위해서는 대상 언어에 대한 체계적인 연구가 필요하다. 다음은 한글 철자 및 띄어쓰기 검사기를 구현하는데 필요한 한국어의 특성 및 어절의 형태에 대해 살펴보고, 그 오류의 종류를 제시한다. 그리고 좌우접속정보표를 만드는 데 필요한 품사의 분류와 한국어 처리를 하는데 필요한 불규칙 현상 및 음운 현상의 처리에 대하여 기술한다.

3.1. 한국어의 특성

한글 철자 및 띄어쓰기 검사를 위해 고려해야 할 한국어의 특성〔조규빈 1988, 성균관 1988, 미승우 1988, 김성용 1987〕은 다음과 같다.

○ 음절의 특징으로서 한글은 모아쓰기 특징을 가진다. 즉, 자음과 모음이 조합되어 단음절을 형성한다.
○ 띄어쓰기가 일정하지 않고 복잡하다. 따라서 띄어쓰기 오류를 범하기 쉽다.
○ 한국어는 첨가어로서 각 낱말의 의미 변화에 의해 문장의 성분을 결정하며, 첨용과 활용이 자유롭다.
○ 한국어는 불규칙 현상 및 음운 현상이 발달하였다.
○ 한글과 한문 및 영문을 혼용하여 사용하는 경우가 많다.

3.2. 품사의 분류

본 논문에서는 좌우접속정보표의 작성과 사전 구성을 위한 품사 분류를 그림 3-1과 같이 한다.

이 분류는 학교 문법 체제를 준용하면서 컴퓨터에서의 처리를 고려한 분류이다. 여기서 조용사는 명사나 의성어 또는 의태어에 붙어서 이것들을 용

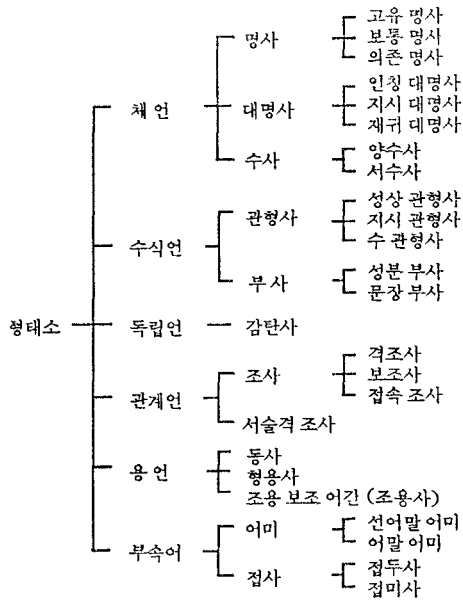

그림 3-1. 접속정보표 구성을 위한 한국어 품사 분류

언화 시키는 역할을 해 주는 어근을 말한다[김성용 1987]. 조용사로는 '하다', '이다', '되다' 등이 있는데, 조용사는 따로 분류하지 않으면 등록해야 할 표제어의 수가 상당히 많이 늘어난다. 따라서 우리는 조용사를 따로 분류하는 것이 사전의 크기를 줄일 수 있다고 생각한다.

3.3. 어절의 형태

한국어의 문장 구성은 형태소, 어절, 구, 문의 순으로 이루어진다. 가장 작은 의미 단위인 형태소(morpheme)가 모여 어절을 형성하고, 어절이 모여 구가 되며, 구가 연결되어 문장을 이룬다. 그런데 한국어는 어절 단위의 띄어쓰기를 하므로, 한 어절 내의 형태소를 파악하여 오류를 찾는 것이 철자 및 띄어쓰기 검사의 과제이다[한국과 1988].

한국어 어절의 개략적인 구조는 그림 3-2와 같이 나타난다[성균관 1988, 조규빈 1988].

화살표(->)는 화살표의 좌측의 범주와 우측의 범주가 접속 가능함을 의미한다. 예를 들면, 어절 '과학자는'은 '과학+자+는'과 같이 명사 '과학'에 접미사 '자'가 붙고, 여기에 다시 조사(보조사) '는'이 붙은 형태이다. 또한 어절 '먹었다'는 '먹+었+다'로 동사 '먹다'의 어간 '먹'과 과거시제

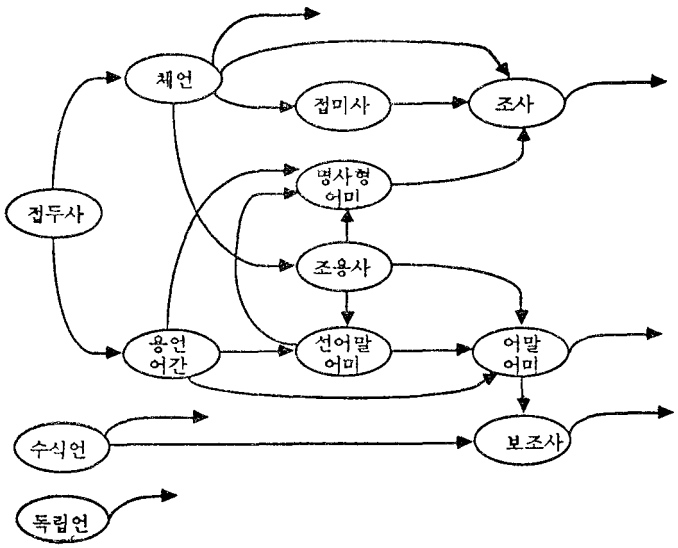

그림 3-2. 한국어 어절의 형태에 대한 네트워크

보조어간(선어말어미) '었'이 붙고, 여기에 평서형 종결어미(어말어미) '다'가 붙은 형태이다.

3.4. 각 품사별 처리

다음은 좌우접속정보표(부록 A, B 참조)에서 형태소를 어떻게 처리하는지를 3.2절에서 분류한 각 품사에 대하여 각각 설명한다.

3.4.1. 체 언

체언에는 명사, 대명사, 수사가 있다. 명사는 고유명사, 보통명사, 의존명사, 외래어로 나누며, 보통명사는 용언어간 겸용과 용언어간 비겸용으로 나누고, 이들을 각각 한자형과 순한글형으로 다시 나눈다. 또한, 우접속 범주에서는 각 체언의 종성 유무에 따라서 무종성과 유종성(ㄹ종성 제외), 그리고 ㄹ종성으로 나눈다. 고유명사에는 좌측에 접속되는 것이 없고, 보통명사에는 접두사나 보통명사가 접속된다. 그리고 단위성 의존명사인 경우는 좌측에 숫자나 양수사가 접속되고, 체언의 우접속은 접미사와 조사가 종성 유무나 체언의 특성에 따라서 달리 접속된다. 복합명사는 일반적으로 보통명사가 접속되는데, 이를 처리하는 것은 쉽지 않다. 그렇다고 모든 복합명사를 사전에 등록하면 잘못된 복합명사를 쉽게 검사할 수는 있으나 사전의 크기

가 문제이다. 본 논문은 기본적으로 복합명사는 보통명사끼리 접속해서 만들 수 있다고 가정한다.

3.4.2. 수식언

수식언에는 관형사와 부사가 있다. 관형사인 경우에는 좌우에 아무 것도 접속될 수 없고, 부사인 경우에는 좌측에는 아무 것도 접속될 수 없지만 우측에는 보조사가 접속될 수 있다.

3.4.3. 독립언

독립언으로는 감탄사가 있는데, 좌우에 아무 것도 접속될 수 없다.

3.4.4. 관계언

관계언에는 조사와 서술격조사가 있다. 조사는 격조사, 보조사, 접속조사가 있는데, 종성 유무에 따라서 좌측에 체언, 접미사, 용언의 명사형, 부사, 종결어미, 보조적 연결어미가 접속될 수 있다. 우측에는 어떠한 것도 접속될 수 없다. 한국어에서 조사는 몇 개 안되지만 조사들이 결합되어 복합조사를 형성할 수 있어 복잡하다. 즉, 복합조사의 접속 형태는 규칙적이지 못하고 복잡하여 조사끼리의 접속 양상을 규칙으로 나타내더라도 잘못된 조사를 허용할 수 있다. 그런데 한국어에서 가능한 모든 복합조사를 조사하면 대략 500여 개 정도 밖에 안되기 때문에, 이를 모두 등록하여 처리하면 문제가 간단하다.

3.4.5. 용 언

한국어에서 동사와 형용사가 차지하는 비율은 30% 정도된다〔유재원 1985〕. 특히, 용언들 대부분이 불규칙 활용 (표 3-1) 및 음운현상 (표 3-2)을 일으키고 있어 이에 대한 처리가 필수적이다. 활용이란 용언이 일정한 문법적 관계를 표시하기 위하여 그 끝을 여러 가지로 바꾸는 현상을 말한다〔조규빈 1988〕. 용언이 활용할 때에 어간과 어미의 모습이 달라지는 일이 있는데, 이들 중 국어의 일반적 음운 규칙으로 설명할 수 없는 것을 불규칙 활용이라 한다. 본 논문에서는 불규칙 활용 및 음운 현상의 처리를 다음과 같이 한다.

(1) 규칙 어간 :
규칙 어간은 그대로 사전에 등록한다.

(2) 어간이 바뀌는 불규칙 어간 :
불규칙 어간 중 변하지 않는 부분만 사전에 등록하고 좌우접속정보

표로 그들의 접속 형태를 제한한다.

(3) 어미가 바뀌는 불규칙 어간 :
규칙 어간을 사전에 등록하고 좌우접속정보표로 어미가 접속되는 형태를 제한한다.

(4) 어간과 어미가 바뀌는 불규칙 어간 :
불규칙 어간 중 변하지 않는 부분을 사전에 등록하고 좌우접속정보표로 어미의 접속되는 형태를 제한한다.

표 3-1. 한국어 불규칙 현상의 분류

구 분	불규칙 현상	활용의 형태	예
어간의 바뀜	ㅅ불규칙*	ㅅ+모음어미->모음어미	짓+어->지어
	ㄷ불규칙*	ㄷ+모음어미->ㄹ+모음어미	깨달+아->깨달아
	ㅂ불규칙	ㅂ+모음어미->오/우+어미 첫음절의 마지막 음소	돕+아->도와
	르불규칙	르+모음어미->ㄹ+ㄹ+모음어미	흐르+어->흘러
	우불규칙	ㅜ+모음어미->어미 첫음절의 마지막 음소	푸+어->퍼
어미의 바뀜	여 불규칙	어간+아/어->어간+여	하+아->하여
	러 불규칙	어간+아/어->어간+러	이르+어->이르러
	거라 불규칙*	어간+어라/아라->어간+거라	가+아라->가거라
	너라 불규칙*	어간+어라/아라->어간+너라	오+어라->오너라
어간과 어미의 바뀜	ㅎ불규칙 (형용사에만 해당)	ㅎ+모음어미->모음어미 ㅎ+ㄴ->ㄴ	파랗+어->파래 파랗+ㄴ->파란

*는 동사에만 해당

표 3-2. 한국어 음운 현상의 분류

구 분	음운 현상	활용의 형태	예
어간의 바뀜	ㅡ 탈락	ㅡ+모음어미->모음어미	쓰+어->써
	ㄹ 탈락	ㄹ+어미(ㄴ, ㅂ, ㅅ오)->모음어미	살+니->사니
어미의 바뀜	어미 아/어 (모음 조화)	양성어간+아/어->양성어간+아 음성어간+아/어->음성어간+어	잡+아/어->잡아 먹+아/어->먹어
	ㅡ삽입	어간+어미(ㄴ, ㄹ, ㅂ, ㅇ, ㅅ, ㅁ) -> 어간+ㅡ+어미	잡+며->잡으며
어간과 어미의 바뀜	간음화	아/어 탈락, 이+어->여 오+아->와, 외+어->왜 우+어->워, 오+이->외 우+이->위, ㅡ+이->의	차+아->차 서+어->서 즐기+어->즐겨 보+아->봐

3.4.6. 부속어

부속어로는 어미와 접사가 있는데, 접사는 접속형태가 복합명사의 문제와 같이 접사가 붙은 형태를 모두 사전에 넣든가, 그렇지 않으면 모든 보통명사에 접속 가능하도록 할 수 있는데, 본 논문에서는 사전의 크기를 줄이기 위해 후자를 택했다. 어미는 다양하고 복잡하지만, 수가 제한되어 있고, 접속 형태가 잘 알려져 있어 쉽게 처리할 수 있다.

3.5. 오류의 종류

우리는 철자 및 띄어쓰기 오류를 철자 및 띄어쓰기를 검사하는 시스템이 범하는 오류인 시스템 오류와 원고작성자(원고 입력자 포함)가 범하는 오류인 입력문 오류로 나누어 생각한다.

3.5.1. 시스템 오류

시스템이 범하는 오류로는 다음과 같은 것들이 있다.

(1) 제1형 오류 :
시스템이 철자 및 띄어쓰기가 맞는 어절을 틀렸다고 취급하는 오류
(2) 제2형 오류 :
시스템이 철자 및 띄어쓰기가 틀린 어절을 맞았다고 취급하는 오류

제1형 오류의 감소는 제2형 오류의 증가를, 제2형 오류의 감소는 제1형 오류의 증가를 가져 올 수 있다. 제1형 오류는 어절 중 미등록어가 있는 경우에 발생될 수 있고, 사전에 정보가 잘못된 경우에도 있을 수 있다. 제2형 오류는 좌우접속정보가 잘못된 경우에 발생할 수 있다. 시스템은 이와같은 오류를 절대 범하지 말아야 되겠지만, 이것이 여건상 힘들다면 최소한 제2형 오류만이라도 범하지 말아야 된다. 틀린 어절을 맞았다고 하는 것은 시스템 신뢰도에 크게 영향을 주기 때문이다.

3.5.2. 입력문 오류

입력문의 오류로는 다음과 같은 것들이 있다.

(1) 철자 오류 :
맞춤법에 맞지 않게 철자를 쓴 오류
(2) 띄어쓰기 오류 :
띄어쓰기를 잘못한 오류

철자 오류는 맞춤법을 제대로 알지 못하고 잘못 쓴 경우와 타자가 잘못된 경우가 있다. 맞춤법을 제대로 알지 못하고 쓴 경우는 일관되게 잘못 쓰는

일이 많다. 이에 비해 타자가 잘못된 경우는 일관되게 틀리지는 않지만, 철자 오류의 상당 부분을 차지할 것으로 생각된다.

한국어 문서에서 가장 흔히 볼 수 있는 오류 중 하나가 띄어쓰기 오류이다[미승우 1988]. 한국어는 영어나 일본어와는 달리 복잡한 띄어쓰기 규칙을 가지고 있다. 또한 이러한 띄어쓰기 규칙을 많은 사람들이 제대로 알지 못하고 있어서 문제는 더욱 심각하다. 실례로서, 중고교 교과서에서의 오류를 조사한 자료[미승우 1988]를 보면 이들 오류 중 띄어쓰기 오류가 약 29.4%로 높은 비율을 차지하고 있음을 알 수 있다. 우리는 띄어쓰기 오류를 다음과 같은 두 종류로 정의한다.

(1) 띄붙 오류 : 띄어 써야 할 것을 붙여 쓰는 오류
(2) 붙띄 오류 : 붙여 써야 할 것을 띄어 쓰는 오류

4. 시스템의 설계 및 구현

4.1. 시스템 개요

본 시스템은 좌우접속정보표와 사전, 사전 편집기, 그리고 한글 철자 및 띄어쓰기 검사기(HSPELL)로 구성되어 있다(그림 4-1).

시스템의 각 부분에 대하여 개략적으로 기술하면 다음과 같다.

(1) 좌우접속정보표 :
한 어절 내에서 형태소들의 접속 형태에 따라 그들을 범주화하여 좌우 접속 관계를 나타낸 표이다.

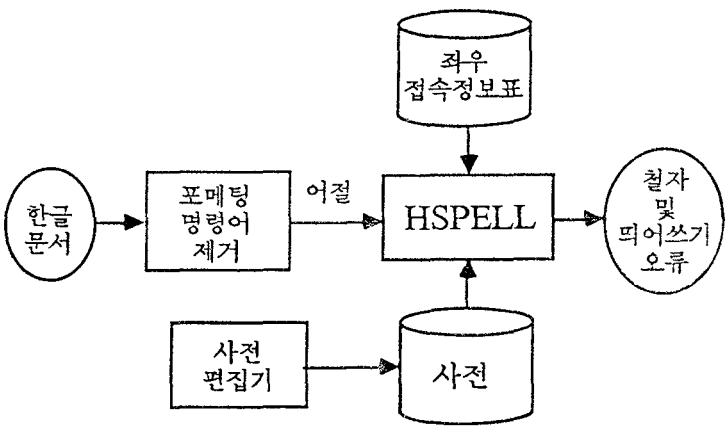

그림 4-1. 시스템의 개괄적인 구성도

(2) 사전 :

형태소와 이에 대한 좌우접속정보의 쌍으로 구성되어 있다. 하나의 형태소가 여러 개의 접속 범주에 속할 수 있으므로 여러 개의 좌우접속정보의 쌍이 있을 수 있다.

(3) 사전 편집기 :

사전 항목의 삽입, 삭제, 변경 등을 용이하게 하기 위하여 사전 전용 편집기를 개발하여 사전 관리에 이용할 수 있도록 하였다.

(4) HSPELL :

사전과 좌우접속정보를 적재(load)한 후, 주어진 입력문서를 어절 단위로 형태소 분석을 하여 철자 및 띄어쓰기를 검사하는 부분이다.

4.2. 좌우접속정보표

한국어 어절은 3.3절에서 본 바와 같이 다양한 형태로 형태소들이 결합된 것이다. 어절은 기본적으로 홀로 사용되는 자립어와 홀로 사용되지 못하는 부속어들로 구성된다. 부속어들도 그 붙는 자립어의 종성의 종류에 따라 몇 가지로 나누어 진다. 이와 같이 형태소들의 접속 유형을 파악하여 표로 구성한 것이 좌우접속정보표이다[강재우 1990].

형태소들의 접속 형태를 다시쓰기 규칙(rewriting rule)으로 기술하는 대신 테이블(table) 형식으로 사용하는 것의 장점으로는 다음과 같은 것을 들 수 있다.

첫째, 두 형태소가 접속될 수 있는지를 빨리 결정할 수 있다. 규칙으로 기술하는 경우 부합(matching)되는 규칙을 찾는 데 많은 시간이 걸린다.

둘째, 규칙으로 기술하는 것은 계층 구조(hierarchical structure)를 파악하는 데 좋으나, 형태소들의 접속 구조는 선형구조(linear structure)이기 때문에 두 형태소 간의 접속 여부만 파악하면 된다. 따라서 테이블 형식이 더 적합하다.

셋째, 한국어 어절의 접속 양상을 규칙으로 표현하는 것은 그 규칙의 수가 상당히 많아 복잡할 뿐 아니라 예외적인 경우가 많아 이를 다 규칙으로 나타내는 것은 힘들다. 따라서 새로운 규칙의 추가라든가 삭제 및 갱신을 하기가 어렵다. 그러나 테이블 형식은 이러한 복잡한 어절의 접속양상을 쉽고 간단하게 표현함으로서 이의 유지, 보수가 쉽다.

좌우접속정보는 그 정확도에 따라 시스템 성능에 상당한 영향을 준다. 즉, 좌우접속정보표가 잘못 구성되었거나, 이 좌우접속정보표로부터 만들어지는 좌우접속행렬이 잘못되었거나, 사전에 좌우접속정보가 잘못 기입되었거나 하면 시스템 오류를 유발한다.

4.3. 사 전

4.3.1. 사전 항목

사전은 실용적인 철자 검사기를 만드는 데 중요한 역할을 한다. 접속 정보가 아무리 잘 되어 있더라도 사전 항목이 적거나 정보가 잘못 들어가면 시스템 오류가 많이 발생한다. 따라서 사전 내의 항목의 수가 많으면 많을수록 제 1 형 오류는 그만큼 적어진다. 왜냐하면, 대부분의 제 1 형 오류는 사전에 등록되어 있지 않은 미등록어로 인해서 발생하기 때문이다. 그러나 사전의 크기는 시스템 성능에 영향을 주기 때문에 되도록이면 그 크기를 줄이는 것이 좋다. 이러한 점을 감안한다면 형태소를 표제어로 하여 사전을 구성하는 것이 좋다고 본다.

4.3.2. 사전의 종류

우리는 사전을 기본적으로 마스터사전과 사용자 정의 사전으로 구분하고 있다(그림 4-2). 마스터 사전에는 단어 등록시 그 단어에 대한 좌우접속정보를 조사하여 이를 반드시 기입해야 하는 것들(예, 조사, 연결어미, 종결어미, 전성어미, 보조어간, 접사, 관형사, 대명사, 수사, 부사, 용언어간 등)을 등록하고, 사용자 정의 사전에는 좌우접속정보를 부여하지 않아도 되는 명사만을 둔다. 이는 한국어의 50% 이상이 명사라는 점(유재원 1985)을 감안한다면 단어 등록시 가장 부담이 되는 좌우접속정보의 부여와 같은 일을 명사에 대해서만이라도 하지 않음으로써 상당한 이득을 볼 수 있다고 본다.

현재 사전은 마스터사전에 약 2,500 단어와 사용자 정의 사전에 약 4,500 단어가 등록되어 있어 도합 약 7,000 단어가 구축되어 있다,

4.3.3. 사전의 구조

사전은 형태소와 그 좌우접속정보의 쌍으로 구성된다. 사전의 구조는 일반적인 편집기(editor)로 편집이 가능하도록 하기 위하여 텍스트화일(text file) 형태를 취했으며 한 줄에 하나의 항목과 그에 대한 정보를 갖도록 하였다. 하나의 항목은 처음에 형태소가 나오고, 다음에 좌우접속정보의 쌍들로 구성된다. 마스터 사전을 형식적(formal)으로 기술하면 다음과 같다.

$M_1,\ CL_{11},\ CR_{11},\ \cdots,\ CL_{1n_1},\ CR_{1n_1}$

$M_2,\ CL_{21},\ CR_{21},\ \cdots,\ CL_{2n_2},\ CR_{2n_2}$

……

$M_i,\ CL_{i1},\ CR_{i1},\ \cdots,\ CL_{in_i},\ CR_{in_i}$

……

$M_N,\ CL_{N1},\ CR_{N1},\ \cdots,\ CL_{Nn_N},\ CR_{Nn_N}$

여기서 M_i는 i번째 형태소이고, CL_{ij}는 M_i의 j번째 좌접속 정보이고, CR_{ij}는 M_i의 j번째 우접속 정보이다. 그리고 N은 사전 항목의 수이고, n_i는 i번째 사전 항목의 접속 정보의 수이다. 또한 모든 $1 \leq i \leq N$에 대하여 $1 \leq n_i \leq 10$이다. 예를 들면, 형태소 '가장'에 대한 사전 항목은 다음과 같다.

가장 (1 2) (8 18) (37 81)

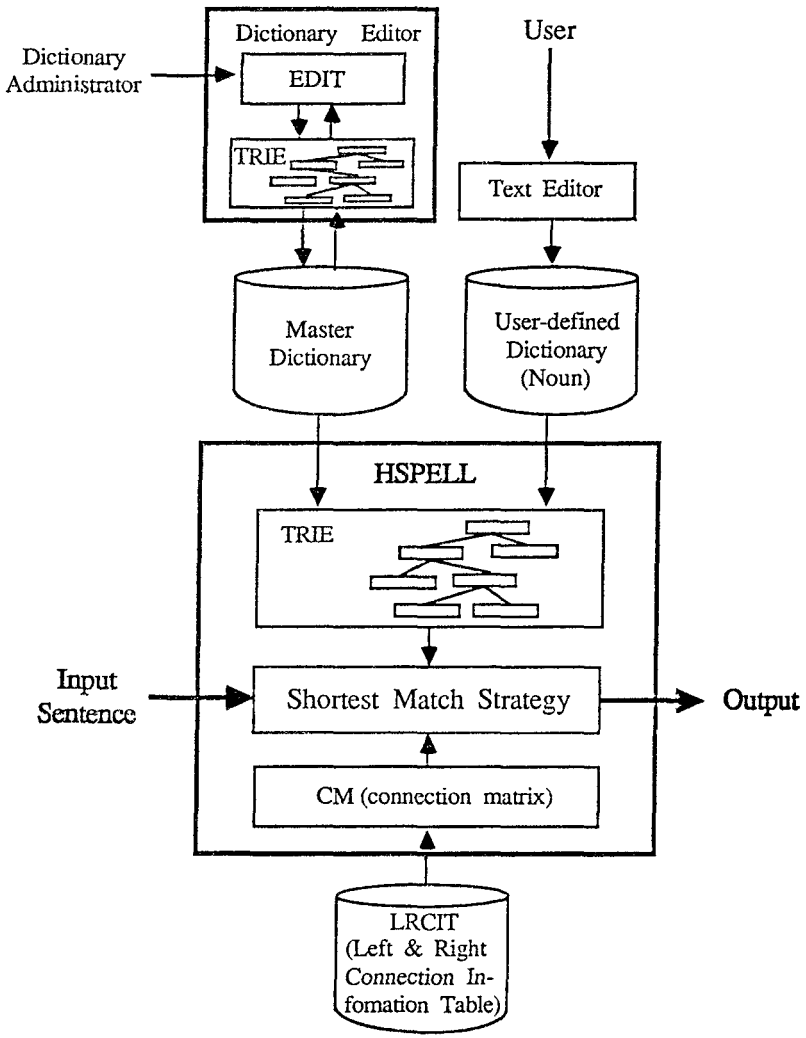

그림 4-2. 시스템의 자세한 구성도

명사 사전 및 사용자 정의 사전은 구축의 편의를 위해 사전에는 동사어간 겸용 명사와 동사어간 비겸용 명사로만 구분을 한다. 그러면 시스템이 사전을 적재할 때 이 명사의 종성 유무 등을 판별하여 접속정보를 자동으로 구성한다. 명사 사전의 예를 들면 동사어간 겸용 명사 '연구'는 사전에 '연구-'를 등록하고, 동사 어간 비겸용 명사 '도경'는 이것을 그대로 등록한다.

4.4. 사전 편집기

사전을 구축하는 데 있어서 사전이 커지는 경우 관리하기가 어려워진다. 즉, 사전에 새로운 단어를 추가할 경우, 이 단어가 이미 사전에 있는지를 파악하기 위해서는 사전이 정열(sort)되어 있어야 하는데, 많은 양의 단어를 추가할 경우는 이것도 문제가 발생한다. 따라서 사전편집기는 사전 항목의 추가, 삭제, 변경이 용이해야 한다.

4.5. 최단일치법(Shortest Match Strategy)

최단 일치법은 가장 먼저 사전항목과 일치되는 형태소를 선택하는 방법을 말한다. 철자 검사기는 하나의 어절에 대해 단지 형태론적인 분석만을 하기 때문에 구문론적 또는 의미론적인 오류는 고려 대상이 아니다. 따라서 철자 검사기는 한 어절이 의미적으로 애매한 경우라도 형태소 분석기에서와 같이 모든 가능한 형태소 결합을 찾아 이를 해결하는 것이 아니라, 단지 하나의 가능한 형태소 결합이 발견되면 올바른 철자로 간주하고 만다. 예를 들면, 어절 '감기는'에 대해 형태소 분석기는 '감기(cold)+는'과 '감(wind)+기는', '감(wind)+기+는'을 모두 찾는 것이 목적이지만, 철자 검사기는 그 중의 하나만을 찾아 철자를 검사해도 된다. (이 경우 최단 일치법은 '감+기+는'를 찾는다.)

4.5.1. 자료구조

본 시스템에서 사용하는 자료 구조는 다음과 같다.

(1) 인덱스 트리(index tree)

사전 검색을 용이하게 하기 위하여 사전을 적재하여 *trie*[HORO84] 구조를 만든다. *trie*는 형태소와 같이 찾고자 하는 키(key)의 길이가 일정하지 않을 때 유용한 자료 구조이다. *trie*는 한 레벨(level)에서 다음 레벨로의 분기(branch)의 결정이 전체 키에 의하지 않고 키의 일부로 결정되는 정도(degree)가 2 이상인 트리(tree)이다. *trie*의 구체적인 예는 그림 4-3과 같다.

(2) 스택(stack)

스택은 형태소 분석을 하는 과정에서 발견된 형태소를 저장하기 위하여 사용된다. 스택에 들어 있는 인접한 형태소들은 서로 접속 가능함을 의미한다. 예를 들면, E_i가 하나의 자소를 나타내고, 어절 E가 $E_1, \cdots, E_n$으로 구성되어 있을 때, $M_1, M_2, M_3, M_3', M_4, M_5$가 사전에 있고, M_1과 M_2, M_2와 M_3', M_2와 M_5가 접속 가능하다면, 형태소 분석 도중의 스택의 상태는 그림 4-4와 같이 나타난다. 이 어절이 가능한 어절인지 분석하는 과정은 다음과 같다. 처음 E_1에서 시작하여 M_1을 사전에서 발견한다. 이 M_1을 스택에 넣고 다음 E_{i+1}에서 시작하는 형태소를 찾는다. M_2가 발견되고 스택에 있는 M_1과 접속이 가능하므로 스택에 넣는다. 다음 E_{j+1}에서 시작하여 M_3를 발견하나 스택에 있는 M_2와 접속이 불가능하므로 더 긴 형태소를 찾는다. M_3'가 발견되고 M_2와 접속 가능하므로 스택에 넣는다. 다음에 E_l에서 시작하는 형태소 M_4를 찾는다. 이 M_4는 M_3'와 접속이 불가능하므로 더 긴 형태소를 찾는다. E_l에서 시작하는 더 긴 형태소가 발견이 안 되므로 스택에 있는 M_3'를 빼고(pop), 여기부터 시작하는 더 긴 형태소를 찾는다. M_5가 발견되고 M_2와 접

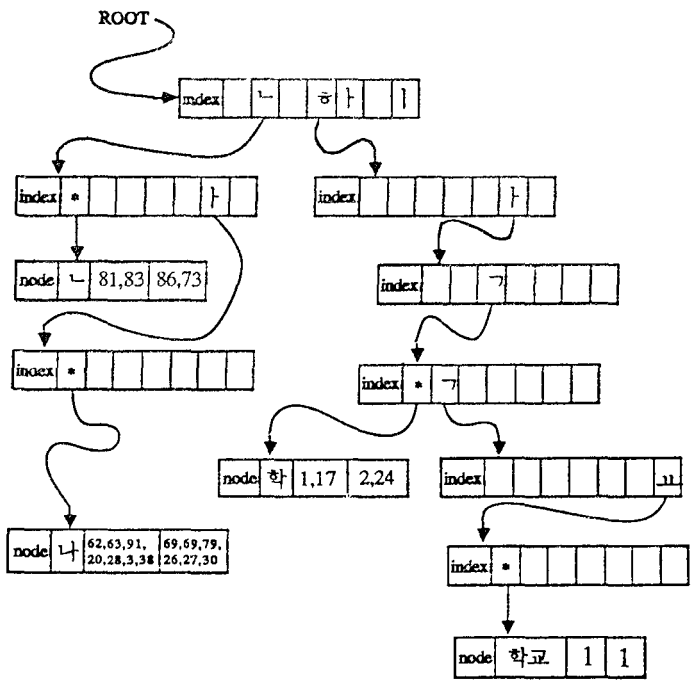

그림 4-3. TRIE의 구체적인 예

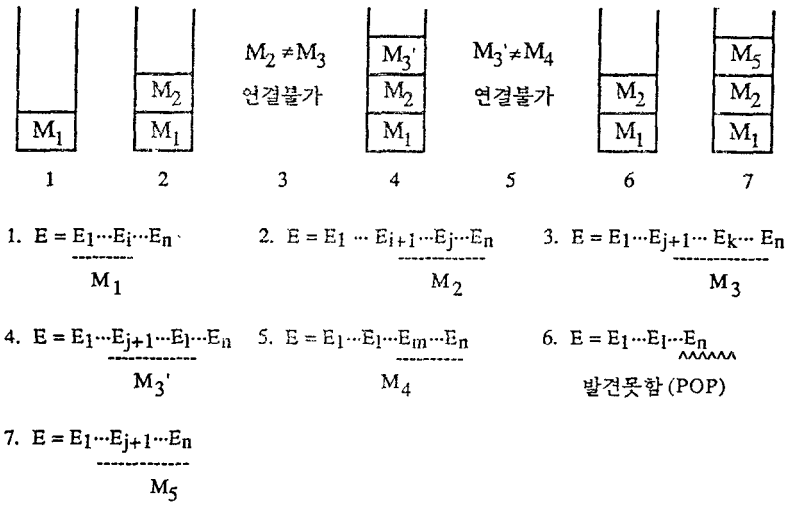

그림 4-4. 어절 *E*를 분석하는 도중의 스택

속이 가능하므로 스택에 M_5를 넣는다. 어절의 끝까지 처리하였으므로 이 어절은 올바른 어절이다.

하나의 어절에 대한 분석이 끝난 후, 스택에 쌓여 있는 형태소들은 그림 4-5와 같이 적어도 하나의 경로(path)를 형성한다.

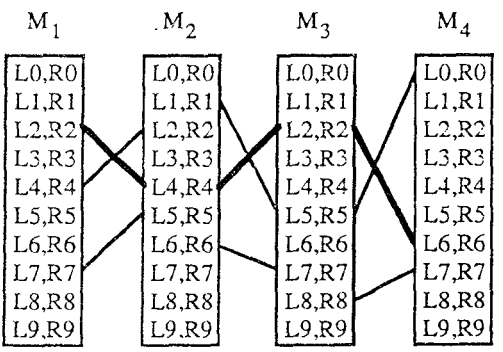

그림 4-5. 스택에 쌓인 형태소들의 접속 경로

(3) 접속 행렬(CM: connection matrix)

행(row)은 좌접속 번호를, 열(column)은 우접속 번호를 나타내는 테이블

이다. CM[i][j]가 참(true)이면 좌접속 번호 i와 우접속 번호 j가 접속 가능함을 의미하고, 거짓(false)이면 불가능함을 의미한다. 이 접속 행렬은 좌우접속정보표(LRCIT: Left and Right Connectivity Information Table)를 적재하여 초기화된다.

4.5.2. 알고리즘

이제 어절내의 철자 및 띄어쓰기 오류를 찾는 방법에 대하여 자세히 알아보자. 그림 4-6은 이러한 방법을 기술한 알고리즘이다.

```
Algorithm Hangul_Spelling_and_Word-spacing_Checker
var
   CM : array[][] of Boolean;
   LRCIT : file of integer;
   TRIE : index tree of dictionary;
   n : integer;
   E1, ..., En : character;
   E : array[] of character;
begin
   load LRCIT and initialize CM;
   load dictionary and make TRIE;
   while there is an eojeol do
      get an eojeol, E = E1 · · · En;
      if E is not stand-alonable then
         Shortest_Match_Strategy(E, CM, TRIE);
      endif;
   endwhile;   /* while there is an eojeol */
end;
```

그림 4-6. HSPELL의 알고리즘

위에서 접속행렬 CM의 내용은 화일 LRCIT로부터 적재되며, TRIE는 사전을 적재하여 만들어진 인덱스 트리이다. E_i는 한글의 한 자소를 의미하며, E는 n개의 자소로 이루어진 문자열(string)이다. 어절 E가 한 개의 형태소로 되어 있지 않다면, 어절 E의 분석을 위해 Shortest_Match_Strategy라는 프로시듀어(procedure)를 부른다. 이때, 어절 E, 접속행렬 CM과 사전의 TRIE 구조를 그 프로시듀어에 넘긴다. 만약 어절 E가 홀로 쓰일 수 있으면, 이 어절은 오류가 없으므로 다음 어절을 읽어 처리한다.

앞에서 간략히 언급한 최단 일치법에 대한 알고리즘인 Shortest_Match_Strategy 프로시듀어는 그림 4-7에 자세히 나타나 있다.

```
Algorithm Shortest_Match_Strategy(E, CM, TRIE)
var
   i, j, p, n : integer;
   LC, RC, R, C : array[] of set of integer;
   M : array[] of character;
   S : stack;
begin
   set stack S to empty;
   set p to 1;   /* index of E1 */
   while p <= n do
      try to find a morpheme M based on TRIE,
            M  = Ep  · · ·  Ej,
            LC = {l1, ..., l10},
            RC = {r1, ..., r10};
      if M is found then
         if S is empty then
            PUSH(p,RC,j);
            set p to j+1;
         else   /* if S is not empty */
            get S's top value, (i,R,p-1);
            set C to connectable set (R,LC); /* by CM */
            if C is not empty then
               PUSH(p,C,j);
               set p to j+1;
            else   /* if C is empty */
               retry to find another longer morpheme;
      else   /* if M is not found */
         if S is empty then
            print E as error;
            break;
         else   /* if S is not empty */
            POP(i,R,j);   /* S's top morpheme */
            set p to i;
            retry to find another longer morpheme;
   endwhile;  /* while p <= n */
end;
```

그림 4-7. 최단일치법에 대한 알고리즘

스택 S에는 최단일치법에 의하여 발견된 형태소 정보가 저장된다. 저장되는 정보의 상태는 (형태소 시작 자소번호, 가능한 접속번호의 쌍들, 형태소 종료 자소번호)로 나타난다. 즉, 위에서 (p, C, j)는 어절 E의 형태소 E_p $E_{p+1}\cdots$ E_j를 뜻하며, C는 이 형태소의 우접속번호와 스택의 꼭대기(top)의 좌접속정보의 접속가능 집합이다. 형태소들이 접속가능하다는 것은 스택 S의 꼭대기에 있는 정보의 상태가 $(i, R, p-1)$이고, 현재 발견된 형태소가 $(M=E_p\cdots E_j,\ LC,\ RC)$라면 RL와 C가 접속이 가능한가를 CM을 참조하여 살피고, 만일 불가능하다면 최단의 형태소에 의한 접속은 불가능한 것이

므로, 그 보다 긴 형태소를 찾아 다시 접속가능한 형태소를 찾는다.

5. 비교 및 성능 평가

5.1. 최장일치법과의 비교

본 절에서는 형태소분석트리를 정의하고, 이 정의에 따른 형태소분석트리의 탐색에 있어서 최단일치법과 최장일치법을 정의한다. 그리고 탐색방법에 따른 최단일치법과 최장일치법의 차이점을 비교한다.

어떤 어절이 주어졌을 때, 이 어절에 대한 형태소분석트리는 사전에 들어있는 형태소에 의해서 결정된다. 따라서, 형태소분석트리를 정의하기 전에 다음과 같은 변수가 주어졌다고 가정하자.

○ 모든 가능한 형태소로 구성된 사전(dictionary)을 D라 하자.

○ 어떤 어절 E는 $E=E_1 \cdots E_n$이라 하자. 여기서 $E_i(i \leq n)$는 하나의 자소이다.

○ $M_{i,j}(i \leq j)$는 $E_i \cdots E_j$로 구성된 사전 D에 있는 형태소라 하자.

○ $M'_{i,n}(i \leq n)$는 $E_i \cdots E_n$로 구성된 사전에 없는 E의 suffix라 하자.

〔정의 1〕 형태소분석트리(MAT: Morphological Analysis Tree)

형태소분석트리는 다음과 같은 조건을 만족하는 하나 이상의 노드로 구성된 순서화된 트리(ordered tree)이다.

i) 루트(root)라 불리우는 특별히 지정된 노드(node) $M_{0,0}$가 있다.

ii) 한 노드 $M_{i,j}$의 자식 노드(child node)는 $M_{j+1,k}(k \leq n)$로 사전 D에 있는 모든 가능한 노드이다.

iii) 한 노드의 자식 노드 중 가장 오른쪽 노드 $M_{i,j}$가 $j \neq n$이면 새로운 $M'_{j+1,n}$을 이 노드의 가장 오른쪽 노드로 둔다.

iv) 같은 부모(parent)를 가진 노드들은 좌에서 우로 문자열 길이(string length)가 커지는 순(ascending order)이다.

〔정의 2〕 최단일치법(Shortest Match Stratege)

두 노드 간의 접속성을 검사하면서 MAT를 Depth-first Search 방법으로 루트 노드로부터 단말(terminal)노드까지의 모든 노드들이 접속 가능한 경로(path) 하나를 찾는 방법이다.

〔정의 3〕 최장일치법(Longest Match Strategy)

두 노드 간의 접속성을 검사하면서 MAT를 Breadth-first Search 방법으로 루트 노드로부터 단말 노드까지의 모든 노드들이 접속 가능한 경로 하나를 찾는 방법이다.

이 정의에 의해서 어절 '감기는'에 대한 MAT를 그리면 그림 5-1과 같다. 여기에서 '감기는'에서 가능한(사전 D에 있는)어절은 '가', '감', '감기',

'ㅁ', '기', 'ㄴ', '는'이라고 가정한다. 그림에서 원은 $M_{i,j}$인 형태소 노드를 표시하고, 사각형은 $M'_{i,j}$인 비형태소 노드를 표시한다. 빗금친 원은 이 어절의 가능한 해석 중의 하나를 나타낸다. 이 노드를 가능한 해석 노드라 부른다.

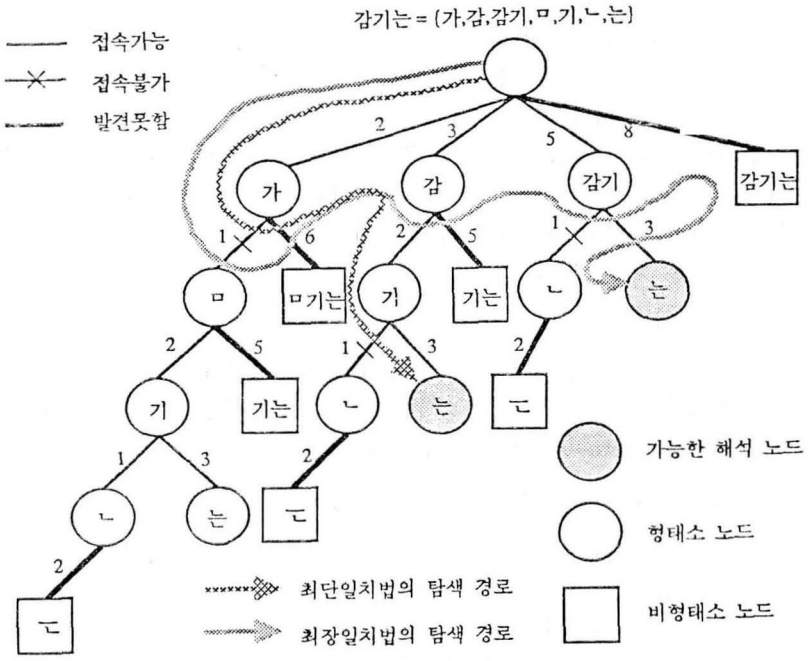

그림 5-1. '감기는'에 대한 형태소분석트리

최단일치법과 최장일치법를 비교할 수 있는 척도(criteria)로는 접속가능성 검사 횟수와 패턴 비교 횟수를 들 수 있다.

○ 접속가능성 검사 회수는 MAT를 여행한 경로의 합이다.

○ 패턴 비교 횟수는 여행(traverse)한 경로의 길이의 합이다. 두 개의 노드 $M_{i,j}$와 $M_{j,k}$(또는 $M'_{j,k}$) 사이의 경로의 길이는 $k-j+1$이다.

그림 5-1에서 최단일치법의 경우, 접속 가능한 검사의 횟수는 7회이고, 패턴 비교 횟수는 2+1+6+3+2+1+3=18회이다. 최장일치법의 경우, 접속가능성 검사 횟수는 6회이고, 패턴 비교횟수는 2+3+5+7+1+3=21회이다.

최단일치법은 Depth-first Search를 하고 최장일치법은 Breadth-first Search를 하므로, 최단일치법은 가능한 해석 노드가 왼쪽 가지에 치우친 경우에,

최장일치법은 오른쪽에 치우친 경우에 유리하다.

MAT는 오른쪽으로 치우칠수록 깊이가 낮아지고 경로의 길이(path length)가 커지는 특성을 가지고 있다. 최장일치법은 자식 노드를 전부 검사해야 하므로 패턴 비교 횟수가 더 크다. 반면에 최단일치법은 접속 가능성 검사의 횟수가 커질 수 있다.

형태소 노드로 끝나는 단말 노드는 어절이 경로 상에 있는 형태소들로 분절될 수 있음을 의미한다. 이들 중 접속 가능성 검사를 하여 루트 노드로부터 단말 노드까지 접속 가능한 경로의 수(즉, 가능한 해석 노드의 수)는 이 어절의 애매성의 수이다. 애매성이 발생했을 경우, 최단일치법은 최초 적합(first-fit)을 찾으려는 방법이고, 최장일치법은 이러한 경우 더 긴 형태소가 맞을 가능성이 높다는 한국어의 특성을 이용하여 최적 적합(best-fit)을 찾으려는 방법으로 볼 수 있다.

5.2. 성능 평가

본 시스템의 성능 평가를 두가지 측면에서 시도하였다. 즉, 시스템 오류에 의한 시스템의 성능 평가와 최단일치법에 대한 시간 복잡도(time complexity) 측면에서의 알고리즘의 평가가 그것이다.

5.2.1. 시스템 오류에 의한 평가

한글 철자 및 띄어쓰기 검사기의 성능은 입력 문서에 대해서 어느 정도의 정확도로 검사를 하는가에 달려 있다. 이러한 정확도는 시스템 오류에 의해 결정된다. 따라서 시스템 오류를 계산함으로써 성능을 평가할 수 있다. 다음과 같은 변수가 주어졌을 때 시스템 오류율을 정의하고 이에 따라 성능을 평가한다.

T_e : 입력 문서 내의 어절의 갯수
E_1 : 입력 문서 중 제1형 오류의 갯수
E_2 : 입력 문서 중 제2형 오류의 갯수
U : 입력 문서 내의 미등록어의 갯수

입력 문서에 따라서 시스템 오류의 수가 달라질 수 있으므로 입력문 의존적 성능과 입력문 독립적 성능을 정의한다. 입력문 의존적 성능(P_D)은

$$P_D = \{1-(E_1+E_2)/T_e\} \times 100$$

이다. 입력문 독립적 성능(P_I)은

$$P_I = \{1-(E_1+E_2-U)/T_e\} \times 100$$

이다. 여기에서, 미등록어를 뺀 이유는 제 1 종 오류의 대부분이 미등록어로 인한 오류이고, 이 미등록어는 입력 문서 의존적으로 발생하기 때문이다.

위 방법에 의해 예문을 대상으로 성능을 평가한 결과를 보면 표 5-2와 같다.

표 5-2. 본 시스템의 성능 평가 결과

Sample Text	T_e	E_1	E_2	U	$P_D(\%)$	$P_I(\%)$
정보과학회 초록	1589	17	19	17	97.734	98.804
고등학교 교과서	5761	210	58	102	95.348	97.118
매뉴얼 번역문	8514	286	216	66	94.104	94.879
[강재우 1990]논문 제 1 장	508	29	1	19	94.094	97.834
TOTAL	16372	542	294	204	94.893	96.139

5.2.2. 알고리즘의 시간 복잡도

최단일치법의 최악의 시간 복잡도(worst case time complexity)를 계산하기 위한 척도로서 접속 가능성 검사 횟수와 패턴 비교 횟수를 사용한다.

최악의 시간 복잡도를 계산하기 위하여 MAT의 최대 가능한 노드의 수를 계산하여 보자. 최대 가능한 노드는 어절의 길이가 n일 때 그림 5-2과 같은 MAT $T(n)$이 만들어진다.

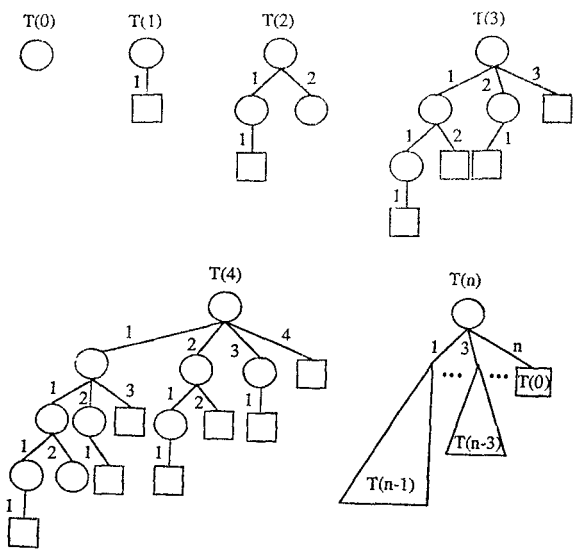

그림 5-2. 자소의 길이가 n일 때 최대노드 MAT

$Z(n)$을 길이가 n인 어절의 최대 노드 수라고 하면, $Z(n)$은 다음과 같다.

$$
\begin{aligned}
Z(0) &= 1 \\
Z(n) &= Z(n-1)+\cdots+Z(0)+1, \quad n \geq 1 \\
&= 2\{Z(n-2)+\cdots+Z(0)+1\} \\
&\quad \cdots \\
&= 2^{n-1}\{Z(0)+1\} \\
&= 2^n, \quad n \geq 1
\end{aligned}
$$

이 때, 이 $T(n)$의 경로 길이의 합을 $V(n)$이라 하면,

$$
\begin{aligned}
V(0) &= 0 \\
V(n) &= V(n-1)+\cdots+V(0)+n+(n-1)+\cdots+1, \quad n \geq 1 \\
&= 2\{V(n-2)+\cdots+V(0)+(n-1)+\cdots+1\}+n \\
&= 2^2\{V(n-3)+\cdots+V(0)+(n-2)+\cdots+1\}+n+2(n-1) \\
&\quad \cdots \\
&= 2^{n-1}\{V(0)+1\}+n+2(n-1)+\cdots+2^{n-2}\cdot 2 \\
&= \sum_{i=1}^{n} 2^{n-i}\cdot i \\
&= 2^{n+1}-n-2
\end{aligned}
$$

이다. 최단일치법의 최악의 시간 복잡도는 내부 노드들 간에는 접속이 가능하고 모든 단말 노드가 비형태소 노드인 MAT에서 발생한다. 이 경우는 MAT를 모두 탐색하고 마지막에 불가능한 어절임이 판명된다. 접속 가능성 검사시간과 패턴 비교 시간이 각각 상수 χ와 ρ라면 최악의 시간 복잡도 $T_w(n)$은 다음과 같다.

$$
\begin{aligned}
T_w(n) &= \chi Z(n)+\rho V(n) \\
&\leq \chi 2^n+\rho 2^{n+1} \\
&= O(2^{n+1})
\end{aligned}
$$

한국어에서는 근본적으로 길이가 1인 형태소로 시작하는 어절이 없으므로 이러한 경우를 제외하고, 또 형태소의 길이가 1인것끼리 접속되는 것도 없으므로 이러한 경우도 제외하면 최악의 시간 복잡도는 $O(2^n)$으로 줄어든다. 또한, 한 어절의 자소 길이가 평균 8정도 되고, 형태소분석트리의 정도(degree)가 보통 4 이하이고, 접속 정보를 이용하여 트리 탐색시 하부트리(subtree)를 자르기 때문에 평균 시간 복잡도는 크지 않다.

6. 결 론

본 논문은 접속 정보를 이용한 한글 철자 및 띄어쓰기 검사기 HSPELL에 대하여 한국어 형태소 처리와 사전 구성 관점에서 기술하였다. 우리는 한글 철자 및 띄어쓰기 검사를 위한 형태소분석 방법으로서 최단일치법이라는 새로운 방법을 제시하였다. 철자 및 띄어쓰기 오류 검사에서는 주어진 어절이 가능한 어절인지 아닌지가 중요하지 실제 문장 중에서 의미하는대로 형태소가 분리되었는가는 중요하지 않으므로 최단일치법은 적절한 방법이라고 생각한다.

HSPELL은 좌우접속정보표가 얼마나 정확하게 기술되어 있는가에 따라 그 성능이 영향을 받는다. 그리고 이와같은 좌우접속정보는 사전에도 들어가기 때문에 상대적으로 사전을 구성하기가 쉽지 않다. 본 논문에서는 이런 문제점을 극소화시키기 위하여 사전을 마스터 사전과 사용자 정의 사전으로 구분하여 놓았다. 즉, 마스터 사전에는 그 수가 제한적이라 할 수 있는 조사, 연결어미, 종결어미, 전성어미, 보조어간, 접사, 관형사, 대명사, 수사, 부사, 용언어간 등을 두도록 하였다. 이의 단어를 등록하는 데에는 좌우접속정보가 절대적으로 필요하기 때문에 마스터 사전을 구축하는 것은 다소 어려움이 있을 수 있다. 그리고 사용자 정의 사전에는 좌우접속정보를 넣지 않아도 되는 명사만을 두도록 하였다. 실제로 사용자 정의 사전에는 시스템에서 자동으로 생성한 좌우접속정보가 들어간다. 이렇게 함으로써 얻을 수 있는 장점은 마스터 사전만 어느 정도 구축이 끝나면 사용자 정의 사전은 쉽게 구축할 수 있기 때문에 사전구축 문제가 간단해진다는 것이다.

HSPELL은 UNIX 4.3 BSD 상에서 C 언어로 구현되어 동작되고 있다. HSPELL의 성능을 평가하기 위하여 우리는 정보과학회 초록문, 고등학교 교과서, IBM 매뉴얼 번역문, 본 논문 1장을 대상으로 약 2만 어절에 대해 테스트하여 보았다. 그 결과 HSPELL은 94% 이상의 정확도를 냈다. 그러나 현재 HSPELL은 입력 문서로 한글과 영문이 혼용된 것을 가정하고 있지만, 한글에 대해서만 처리가 가능하고, 한글 코드도 N 바이트 조합형만이 가능한 상태이다. 앞으로 HSPELL은 이와같은 제한된 점을 개선하고, 지금까지의 철자 및 띄어쓰기 오류를 검사하는 기능 뿐만 아니라 이런 오류를 자동으로 교정해 주는 기능까지도 가능하게 할 예정이다.

참 고 문 헌

강재우(1990) 접속정보를 이용한 한글 철자 및 띄어쓰기 검사기의 설계 및 구현, 한국과학기술원 석사학위논문.

김성용(1987) Tabular Parsing 방법과 접속 정보를 이용한 한국어 형태소 분석기, 한국과학기술원 석사학위 논문.

김영웅(1984) 한글 철자법 교정 시스템, 한국과학기술원 석사학위논문.

김효준, 박재득, 김길창(1988) '이식성이 있는 한국어 형태소 해석기개발 연구', '87특정연구과제 결과 발표 논문집.

남기심, 이정민, 이홍배(1988) 언어학 개론, 탑 출판사.

문교부 고시(1989) 한글 맞춤법/표준어 규정/외래어 표기법, 삼성출판사.

미승우(1988) 새 맞춤법과 교정의 실제, 어문각.

박상규(1984) 기계번역을 위한 한국어 품사의 자동 분류 방법, 한국과학기술원석사학위논문.

성균관 대학교 대동 문화 연구원(1988) 고등학교 문법, 대한교과서주식회사.

손우형(1986) 한국어 기계번역을 위한 형태소 분석에 관한 연구, 한국과학기술원석사학위논문.

송춘환, 강재우, 김연배, 최기선, 권용래, 김길창(1989) '한글 철자 및 띄어쓰기 검사기—두가지 접근 방법—', '89 가을 한국정보과학회 발표 논문집.

신기철, 신용철 편저(1989) 새우리말 큰사전, 삼성출판사.

안동언(1987) 기계번역을 위한 한국어 해석에서 형태소로부터 구문요소의 형성에 관한 연구, 한국과학기술원 석사학위논문.

유재원(1985) 우리말 역순 사전, 정음사.

이기동, 신현숙(1987) 언어학 개론, 한신문화사.

이종혁, 김성용(1987) 좌우 접속 정보표, 한국과학기술원, CS Lab. Memo.

장석진(1989) 한·영 동사의 하위범주화와 대응에 관한 연구, 한국전자통신연구소.

조규빈(1988) 하이라이트 고교문법, 지학사.

최형석, 이주근(1984) '자연어 어절 처리 알고리즘', 한국 정보과학회 가을 학술발표논문집, 11권, 2호.

편집부(1989) 한글바로쓰기, 글쓰기 서당.

한국과학기술정보센터(1980) 우리말 표준화 보고서—2(단어 빈도조사 일람표).

한국과학기술원 전산학과 컴퓨터시스템 연구실(1985) 한글 조사 목록, CS Lab. Memo.

한국과학기술원 전산학과 컴퓨터시스템 연구실(1988) 한국어 처리시스템 개발 환경의 연구, 과기처 보고서.

Cyril N. Alberga(1967) 'String Similarity and Misspelling,' *CACM, Vol.*

10, No. 5, May, pp. 302-313.

Charles R. Blair(1960) 'A program for correcting spelling errors,' *Information and control 3*, pp. 66-67.

L.L. Cherry and W. Vesterman(1983) 'Writing Tools—The STYLE and DICTION Programs,' *Bell system tech. jour. Vol. 62, No. 6, July-August.*

Fred J. Damerau(1964) 'A Technique for computer Detection and Correction of Spelling Errors,' *CACM, Vol. 7, No. 3, March*, pp. 171-176.

Leon Daridson(1962) 'Retrieval of Misspelled Names in an Airlines passenger System,' CACM, Vol. 5, No. 3, March, pp. 169-171.

L.T. Frase(1983) 'The *UNIX*™ Writer's Workbench Software: Philosophy,' *Bell systemtech. jour. Vol. 62, No. 6, July-August.*

E. Horowitz and S. Sahni(1984) *Fundamentals of Data Structure in Pascal*, Computer Science Press Inc.

Brian W. Kernighan & Dennis M. Ritche(1978) *The C Programming Language*, Prentice-Hall.

James L. Peterson(1980) 'Computer Programs for Detecting and Correcting Spelling Errors,' *CACM, Vol. 23, No. 12, December*, pp. 676-687.

Computer Science Division(1984) *UNIX User's Manual Reference Guide*, Univ. of California, March.

ABSTRACT

Morphological Analysis of Korean and its Dictionary
—Hangul Spelling and Word-spacing Checker Using the Connectivity Information—

Deok-Bong Kim, Key-Sun Choi, Jae-Woo Kang

This paper describes the Hangul spelling and word-spacing checker, HSPELL. HSPELL was implemented in C and runs under UNIX™ 4.3 BSD. The system finds spelling or word-spacing errors in the text of written Hangul and points out it by batch mode. For this processing, HSPELL uses the connectivity information between morphemes and the dictionary with 7,000 morphemes.

In this paper, a morphological analysis method called ***shortest match strategy*** is proposed for the Hangul spelling and wordspacing checking. This method allows efficient search for separable morphemes in a word phrase.

130-012
서울시 동대문구 청량리동 207-43
한국과학기술원 전산학과

UNIX™ is a trade mark of Bell Laboratories.

<부록 A>

문법 범주				예	번호	좌접속
명사	고유명사			김유신, 한국, 광주	1	
	보통명사	용언 어간 겸용	한자형	분석, 설계, 연구	2	한자형접두사, 보통명사
			순한글형	자랑, 노래, 사랑	3	순한글형접두사, 보통명사
		용언 어간 비겸용	한자형	학생, 권리, 자유	4	한자형접두사, 보통명사
			순한글형	하늘, 비늘, 수박	5	순한글형접두사, 보통명사
	의존명사	단위성		채, 개, 분, 마리, 섬	6	숫자, 양수사
		기타		것, 등, 뿐, 데, 바	7	
	외래어			컴퓨터, 데이타	8	
...				...	...	...
접미사	인간 고유명사형			씨, 군, 양	360	인명 고유명사
	지명 고유명사형			도, 시, 군, 읍, 구	361	지명 고유명사
	한자 보통명사형			국, 실, 군, 학, 적	362	한자 고유명사
	순한글 보통명사형			매, 장이	363	순한글 고유명사
	복수 접미사			들	364	가산 보통명사, 부사, ...
숫자				1, 2, 3, 4, 5	370	숫자
기호 (특수문자)				@, #, $, %, &, *, !	380	용언 어간, .. 제외
파생어 (위 표에 예외적인 것)					390	

<부록 B>

문법 범주					예	번호	우접속
명사	…				…	…	…
	보통명사	용언어간 겸용	한자형	무종성	설계, 연구, 처리	19	한자 보통명사형 접미사, 조사, 서술격 조사, 보통명사 조용보조어간
				유종성	분석, 가난, 번역	20	
				ㄹ종성	청결, 거절, 자살	21	
			순한글형	무종성	노래, 빨래, 이야기	22	순한글 보통명사형 접미사, 조사, 서술격 조사, 보통명사 조용보조어간
				유종성	밥, 사랑, 자랑	23	
				ㄹ종성	일, 절, 말	24	
		용언어간 비겸용	한자형	무종성	권리, 자유, 우유	25	접미사, 조사, 서술격 조사, 보통명사
				유종성	학생, 책상, 병원	26	
				ㄹ종성	글, 사실, 지하철	27	
			순한글형	무종성	다리, 나무, 거리	28	접미사, 조사, 서술격 조사, 보통명사
				유종성	손, 도장, 땅, 바닥	29	
				ㄹ종성	달, 건달, 발, 티울	30	
…					…	…	…
숫자					1, 2, 3, 4, 5	370	숫자,숫자형 접미사, 단위성 의존명사
기호(특수문자)					@, #, $, %, &, *, !	380	체언, 조사, 기호, 숫자
파생어(위 표에 예외적인 것)						390	