철자오류의 통계자료에 근거한 철자오류 교정시스템

임 한 규† 김 응 모††

요 약

본 논문에서는 우리가 실제 문서 편집기에서 범하는 철자 오류를 수집하고 분석하여, 이러한 자료를 근거로 철자 오류를 교정할 수 있게 후보를 제시하는 시스템을 구축하였다. 오류가 발견된 어절에서, 각 음절의 후보를 생성함에 있어서 자소별/음절별 빈도수를 고려하여 개수를 최소화했으며, 따라서 후보 어절의 개수도 최소화할 수 있었다. 후보 어절의 개수는 평균 3.1개에서 8개였으며, 제시된 후보 어절 중에는 맞는 어절이 62.1%에서 84.1 % 포함되어져 있었다.

A Spelling Correction System Based on Statistical Data of Spelling Errors

Hankyu Lim† Ungmo Kim††

ABSTRACT

In this paper, the spelling errors which are made by human beings in the real word processors are collected and analyzed. Based on these data, we make a prototype which can perform spell aid function providing candidate words. The number of candidate characters are minimized by the frequency of Jaso and character, so the number of candidate words could be minimized. The average number of candidate words presented are 3.2 to 8, and 62.1 % to 84.1 % of the correct words are presented in the candidate words.

1. 서 론

한글 및 한국어 정보처리에 관한 연구 및 개발의 활성화로 인하여 그 동안의 노력이 상당한 결실로 나타나고 있으며, 이러한 결과를 상품에 적용하여 많은 제품들이 현재 시중에 나와 있다. 정보검색, 기계번역, 문서 편집기, 문자 인식 및 음성 합성 등이 그 예이다. 한국어 처리를 위해서는 형태소 분석, 통사 분석, 의미 분석 및 화용 분석 등이 필요한 데, 이 중 가장 기초가 되는 형태소 분석기를 이용하여 제품에 응용한 것이, 철자 검사 기능을 수행하는 문서 편집기인 '가나다'로서, 1992년 한국아이비엠에서 발표되었다.

이후 여러 회사에서도 문서편집기에 철자 오류나 맞춤법 오류를 검사하고 교정하는 기능을 추가하였으며, 요즈음에는 이러한 기능은 문서 편집기에 당연히 포함되어야할 기능으로 인식되고 있다. 그럼에도 불구하고 철자 오류 또는 맞춤법 오류의 검사 및 교정 기능은 사용자가 느끼기에 부족한 점이 많다. 가장 최고의 시스템이라고 하면, 오류가 있는 모든 어절만 찾아내어 그 오류에 대한 최소 개수의 후보를 제시하고 그 후보 중에는 맞는 어절이 반드시 들어있는 시스템이라 할 수 있다. 그러나 실제 지금 시중에서 유통되고 있는 문서 편집기는 실제로 오류를 찾아내지 못하거나, 맞는 어절을 틀리는 어절로 해석하거나, 후보 어절을 만들어 내지 못하거나, 후보 어절을 과다하게 생성하는 등의 여러 문제점이 드

†종신회원 : 한서대학교 전산정보학과 전임강사
††정 회 원 : 성균관대학교 정보학과 교수
논문접수 : 1995년 8월 7일, 심사완료 : 1995년 11월 9일

러나고 있다. 물론 사전에 수록되어 있지 않은 복합어나 고유명사 등으로 인해 처리의 한계가 있는 것은 사실이다.

본 논문에서는 먼저 우리 주위에 흔히 있는 각종 전자 파일을 대상으로 맞춤법에 어긋나는 각종 오류를 수집하여 이를 체계적으로 분류하였다. 분류된 오류의 유형에 의거하여 오류 어절에 대한 후보 어절을 제시하는데 있어서, 한글 자소의 자판에서의 위치에 따른 오류 유형을 분석하여, 이 유형에 의거하여 후보 자소를 만들고, 이 자소를 조합하여 음절별 빈도순을 고려하였다. 따라서 실생활에서 거의 사용되지 않는 음절은 제외할 수 있었다. 이렇게 함으로써 틀린 어절에 대한 최소 개수의 후보 어절을 제시하고, 이중 맞는 어절이 포함될 확률을 높일 수 있게 되었다.

한국어의 처리에 있어서 가장 기본이 되는 형태소 분석은 한국어 어절을 의미를 가지는 최소의 단위인 형태소로 구분해 내는 것을 말한다[1]. 따라서 형태소의 예로는 어근, 조사, 어미, 접두사 및 접미사를 들 수 있다. 형태소 분석은 여러 응용분야, 즉, 핵심어 추출 및 색인, 철자 오류 및 맞춤법 오류의 검사/교정 등에 널리 쓰이고 있으며, 통사 분석이나 의미, 화용 분석에의 입력으로 쓰인다. 현재 학교 연구실이나 회사 등에서 만들어진 10여개 전후의 형태소 분석 프로그램 및 사전이 개발되었는데, 그중 몇몇은 상당한 정도의 분석 정확도와 빠른 처리 속도를 보여주고 있으며 제품에 응용되기도 했다[2, 3, 4].

2. 맞춤법 및 철자 오류

문서 편집기를 사용하여 문서를 작성할 때, 우리는 여러 가지 오류를 범하곤 한다. 문서 편집기가 아닌 인쇄물인 경우, 띄어 쓰기, 맞춤법 오류 및 어미와 조사의 오용이 전체오류의 약 60%를 차지한다[5].

그러나 문서편집기에서는 〈표 1〉과는 다소 다른 유형을 나타낸다. 문서편집기를 사용하여 문서를 작성할 때 해당 문서의 내용을 자판을 눌러서 직접 입력하게 되므로 자판을 잘못 누르는 경우가 자주 발생한다. 문서편집기를 사용하여 작성한 문서인 경우, 작성자가 문서의 교정을 보았는지 여부에 따라 결과의 정확도가 크게 차이가 난다. 아니면 맞춤법 검사 및 교정 기능이 있는 문서 편집기에서 이 기능을 실행하였는지 여부에 따라서도 크게 달라진다. 오류를 넓게 분류하면, 맞춤법오류, 구문오류 및 의미오류 등으로 나눌 수 있다. 맞춤법오류는 문서 입력시 자판을 잘못 누르는 경우, 맞춤법을 몰라서 발생한 경우, 띄어쓰기를 틀린 경우 등이다. 구문 오류는 문의 구조가 언어의 구문에 맞지 않는 경우 등이며, 기타 의미 상의 오류 등이 있다. 이 중에서 구문오류 및 의미 오류를 찾아내기 위해서는 완전한 구문분석이나 의미 분석을 행할 수 있는 시스템이 필요한데, 아직은 요원한 실정이다. 철자 및 맞춤법 오류의 경우, 형태소 분석을 수행하여 오류를 찾을 수 있으나, 형태소 분석만으로 맞춤법 오류를 전부 발견할 수는 없다[2, 4].

〈표 1〉 중고 교과서 오류
〈Table 1〉 Errors in the text

오류유형	오류개수
띄어쓰기 오류	808
어미와 조사 오용	689
적절하지 못한 낱말	379
문맥이 호응하지 않는 문장	147
맞춤법 오류	132
기타	589
계	2744

선생님께서 열성적으로 국어를 '가리키고' 있는데, 철수는 창밖의 풍경을 손가락으로 '가르치며' 영희와 얘기를 하고 있다.

위 문장의 경우, 형태소 분석만으로 맞춤법 오류를 찾을 수가 없다. 이런 오류는 어절 단위로서는 해결할 수가 없다. 이를 위해서는 해당 어절 전후의 어절 또는 전후의 문장에서 더 많은 정보를 필요로 하게 된다. 맞춤법오류는 어절에 발생한 오류로서 철자오류, 문장부호오류, 띄어쓰기오류 등이 있는데, 문장부호오류 등은 드물게 나타나며[2], 띄어쓰기오류는 시중의 문서 편집기에서 쉽게 처리되고 있다. 따라서 철자 오류가 맞춤법오류에서 가장 큰 비중을 차지하며, 이를 해결하게 되면 맞춤법오류의 대부분을 해결한다고 볼 수 있다.

〈표 2〉 문서편집기에서의 철자오류
〈Table 2〉 Spelling Errors in Word Processors

오 류 유 형	오류개수
자소가 대체된 경우	398
자소가 탈락된 경우	78
자소가 추가된 경우	42
기타	27
계	545

위의 〈표 2〉는 PC통신에 올려져 있는 여러 신문 기사와 공지사항, 문서편집기를 이용하여 작성한 대학생 들의 보고서 등에서 수집한 오류를 모아서 분류한 것이다. 총 105,000어절에서 찾아내어 분류한 철자오류는 545어절이었다. 입력한 사람이 입력을 끝내고 오류를 찾기 위한 시도를 하였는지, 자판을 보면서 입력했는지, 자판을 보지 않고 원고만 보고 입력을 했는지, 입력한 사람의 타이핑 실력이 어떤지, 아니면 맞춤법 검사 기능을 수행하였는지는 알 수 없었다. 오류의 양과 유형은 입력시 또는 입력한 후의 행동 여하에 따라서도 많은 차이가 날 수 있으므로 전체 어절 중에서 발견된 어절의 양이 얼마인지는 큰 의미가 없다. 단지 발견된 오류를 어떻게 하면 올바른 단어로 대체할 수 있는지가 주 관심사이다.

위 표에서 보듯이 철자 오류에서도 여러 발생 원인이 있다. 가장 빈번한 오류는 자판을 누를 때 해당 자판을 잘못 누르는 경우이다. 즉, 자판의 유사성에 의한 오류이다. 가령 'ㄱ'을 누르려고 했는데 'ㄱ'옆에 있는 'ㅅ'을 누른 경우이다. 또 다른 오류는 자소를 누락한 경우이다. '김'을 입력해야 하는데 '기'를 입력한 경우이다. 필요없는 자소가 추가되었거나 한 음절이 누락, 또는 추가된 경우도 발견되었다. 복자음이나 복모음을 누를 때, Shift키를 누르지 않아 발생한 오류도 있었다. 그러나 자리 바꿈 오류의 빈도는 상당히 낮았으며, 발음의 유사성에 의한 오류, 즉, 'ㅖ'와 'ㅐ'의 혼용은 거의 발견되지 않았다.

〈표 3〉에서는 자소가 대체된 경우에서의 오류를 더 상세하게 분류한 표이다. 이 표에서 보면 1째 음절과 2째 음절에서의 오류 빈도는 거의 동일하나, 3째 음절부터는 다소 낮아진다. 이는 국어사전에 수록된 단어의 평균길이가 2.77음절인 것을 보면 당연한 현상이다[2]. 그러나 하나

〈표 3〉 자소가 대체된 경우에서의 음절별 자소별 유형
〈Table 3〉 Error Type by Jaso and Character in Jaso Substitution

	1째 음절	2째 음절	3째 음절	기타음절	계
초성	41	45	9	6	101
중성	110	82	32		224
종성	38	25	10		73
계	189	152	51	6	398

재미있는 현상은 중성에서의 오류가 제일 많다는 것이다. 종성인 경우는 한글의 음절이 종성이 있는 경우와 없는 경우로 나누어지므로, 이 또한 당연한 결과로 보인다.

영어에서의 철자오류는 대부분 한 단어의 첫 문자는 옳다는 가정 하에 철자 검색 시스템을 구축한 경우가 많다. 이는 영어의 경우, 첫 문자가 철자오류를 범할 확률이 상당히 낮다는 통계 자료에 의한 것인데, 이렇게 함으로써 사전검색에 있어서 회수를 줄이거나 유사어가 많이 나올 때 계산의 복잡성 등을 줄일 수 있게 된다.

이러한 철자 오류어를 검사하기 위해서 형태소 분석을 행하면 어절의 생성 규칙에서 벗어난 관계로 분석오류가 되는 게 바람직하지만 간혹 틀린 어절 자체도 형태소분석에서 성공하는 경우도 나타난다. 예를 들어 '사슴을'을 잘못 입력하여 '가슴을'로 입력하였더라도, '가슴을'이 옳은 어절이 되는 경우이다. 이러한 오류를 유사어 오류라 한다[4]. 따라서 형태소 분석만으로 철자오류를 행하는데 있어서 한계가 있는 것은 사실이다.

3. 철자 오류의 검사 및 수정

어절 단위로 처리할 수 있는 철자 검사인 경우는 '정의가'가 '저의가'로 입력되었거나 '압력을'을 '입력을'로 입력한 유사어 오류인 경우는 형태소 분석만으로 해결할 수 없다. 이러한 유사어 오류는 구문검사나 의미검사를 수행하여 처리할 수가 있는데, 대량의 말 뭉치에 근거한 통계적 연어정보를 이용하거나 의미네트워크와 같은 지식베이스를 이용하여 처리하는 방법 등이 연구되고 있다[13]. 형태소 분석은 철자오류를 가진 어절이 틀린 형태소가 되거나 어절 형성규칙에 어긋나는 등의 비어를 찾아내는 데 이용된다[2].

물론 영어의 경우, 사전 검색과 간단한 규칙 만으로 기본적인 철자 검사의 수행이 가능하지만 한국어의 경우는 그렇지 못하다. 형태소 분석 시스템은 기본적으로 옳은 어절은 옳게 분석을 해야하며, 틀린 어절에 대해서는 분석에 실패해야 지만 완전하다고 할 수 있다.

본고에서 사용하는 형태소 분석기의 분석 예를 보면 아래와 같다.

"나는"
(명사 "나")+(조사 "는")
(동사 "나")+(어미 "는")
(동사 "날")+(어미 "는")

"감을"
(명사 "감")+(조사 "을")
(동사 "가")+(어미 "음")+(조사 "을")
(동사 "감")+(어미 "을")

처리결과 중 미등록어와 복합명사 등으로 인해 맞는 어절을 틀리게 분석한 경우도 가끔 나타나기도 하는데, 이럴 경우는 미등록어나 복합명사를 사전에 추가하는 방식으로 해결할 수도 있다.

문서 편집기에서 원문의 입력이 끝나면 원문에 대해 철자 검사를 행하기 위해서 형태소 분석을 하면 철자오류는 발견된다. 철자 오류가 발견되면 교정하는 과정이 필요하다.

〈표 2〉의 결과에서 수집된 오류 어절과 맞는 어절을 비교분석한 결과 72%는 해당 자모 중 한 자가 잘못 입력된 것으로 나타났다. 즉, '변경'을 '변겅'으로 입력했다거나, '대책'을 '대칙'으로 입력한 경우 등이다. 여기서 발견된 중요한 사실 하나는 잘못 입력된 자소의 약 86%가 원래 맞는 자모의 자판에 바로 인접하여 위치한다는 것이다. '대책'을 '대칙'으로 입력한 경우에서 보듯이 중성 '이'는 '애'의 바로 인접한 위치에 존재한다. 또, '사슴을'을 '가슴을'로 입력한 경우에서 보듯이 초성 'ㄱ'은 'ㅅ'과 이웃하여 있다는 사실이다. 이러한 관찰 결과를 바탕으로 작성한 것이 〈표 4〉이다.

'변경'이라는 어절에서 우선 이 어절이 무엇이 틀리는지 알 수가 없다. 따라서 지금까지 관찰된

〈표 4〉 한글 자모와 바로 인접한 자모
〈Table 4〉 Adjacent Jamo List

해당 자음	바로 인접한 자음	해당 모음	바로 인접한 모음
ㄱ	ㄲ, ㄸ, ㅇ, ㄹ, ㅎ, ㅅ	아	어, 으, 이 애, 얘, 야, 여
ㄴ	ㅁ, ㅋ, ㅌ, ㅇ, ㄷ, ㅈ, ㅂ	야	여, 어, 아, 애, 얘
ㄷ	ㄸ, ㅈ, ㄴ, ㅇ, ㄹ, ㄱ	어	오, 우, 으, 아, 야, 여, 요
ㄹ	ㅇ, ㅊ, ㅍ, ㅎ, ㅅ, ㄱ, ㄷ	여	요, 오, 어, 아, 야
ㅁ	ㅋ, ㄴ, ㅈ, ㅂ	오	유, 우, 어, 여, 요
ㅂ	ㅃ, ㅁ, ㄴ, ㅈ	요	오, 어, 여
ㅅ	ㅆ, ㄱ, ㄹ, ㅎ	우	유, 으, 어, 오
ㅇ	ㄴ, ㅌ, ㅊ, ㄹ, ㄱ, ㄷ, ㅈ	유	우, 오
ㅈ	ㅉ, ㅂ, ㅁ, ㄴ, ㅇ, ㄷ	으	우, 아, 어
ㅊ	ㅍ, ㄹ, ㅇ, ㅌ	이	아, 에, 예, 애, 얘
ㅋ	ㅌ, ㄴ, ㅁ	애	애, 야, 아, 이, 에, 예
ㅌ	ㅋ, ㅊ, ㅇ, ㄴ	에	애, 야, 아, 이, 에, 예
ㅎ	ㅎ, ㄹ, ㅊ	얘	애, 야, 아, 이, 에, 예
ㄲ	ㄱ, ㄷ, ㄸ, ㅇ, ㄹ, ㅎ, ㅅ, ㅆ		
ㄸ	ㄷ, ㅈ, ㅉ, ㄴ, ㅇ, ㄹ, ㄱ, ㄲ		
ㅃ	ㅂ, ㅁ, ㄴ, ㅈ, ㅉ		
ㅆ	ㅅ, ㄱ, ㄲ, ㄹ, ㅎ		
ㅉ	ㅈ, ㅂ, ㅃ, ㅁ, ㄴ, ㅇ, ㄷ, ㄸ		

여러 오류의 유형에 의하고 또, 지금까지 연구되어 발표된 한글의 여러 특징들에 기반하여 여기서 제안하는 알고리즘은 다음과 같다.

```
while(문자열의 끝이 아니면){
if(문자가 한글인가?){
  한글코드생성
  글자, 코드,자모코드출력
  원래의 한글코드저장
  /*초성을 변형시킨 후보생성*/
  i=0
  code1=초성코드값
  while(자판 배열정보가 0이 아니고 7보다 작으면)
{
    if(2350자 안에 포함안되면)continue;
    후보배열에 저장
  }
  저장했던 원래의 한글코드를 변수에 복사
```

```
/*중성을 변형시킨 후보생성*/
i=0
code2=중성코드값
while(자판 배열정보가 0이 아니고 7보다 작으면)
{
  if(2350자 안에 포함안되면)continue;
  후보배열에 저장
}
저장했던 원래의 한글코드를 변수에 복사
/*종성을 변형시킨 후보생성*/
i=0
code3=종성코드값
while(자판 배열정보가 0이 아니고 7보다 작으면)
{
  if(2350자 안에 포함안되면)continue;
  후보배열에 저장
}
}
c+=2 /*다음 문자를 읽기 위해 포인터 증가*/
}
```

위의 알고리즘에서 2350자에 속하는지의 검사는 다음의 자료에 근거한다. 한글은 이론적으로 초성 19자, 중성 21자, 종성 27자로 이루어지므로 종성이 없는 경우(28자가 됨)를 고려하여 초성, 중성 및 종성의 개수를 곱하면 이론적으로 11,172자가 생성가능하다. 그러나 이중에서 약 2350자가 실생활에서 99.99%이상의 사용빈도를 나타내므로[6] 여기에 속하지 않는 한글 음절이 나올 확률은 지극히 희박하므로 대상에서 제외한다.

한글조합형 코드인 KSC 5601-1982는 2 바이트로 구성되는 한글 음절을 상위 1비트를 영문/한글 구분용으로 사용하고, 초성, 중성, 종성에 각각 5비트씩을 배정하여 만든 코드이다[7]. 따라서 조합형 코드를 사용할 경우 자소 단위의 비트 조합에 의해 한글의 음절 생성이 쉽게 이루어질 수 있다.

위의 알고리즘에 의해 초성, 중성, 종성이 대체된 음절을 생성할 때, 한 음절에서 철자오류는 단 하나의 자소에서 일어난다는 가정에서 출발한다. 마찬가지로 한 어절에서도 단 하나의 자소에서만 오류가 발생한다는 가정에서 출발한다. 이는 한 어절에서 두 개이상의 철자 오류가 극히 드물게 발생한다는 통계 자료에 근거하며, 이 가정을 함으로써 계산의 양이 훨씬 줄어들게 된다.

다음의 절차에 의하여 각 어절에 대해 형태소 분석을 행한다.

1. 첫 음절의 모든 후보를 각각 포함하는 어절을 생성하고, 어절에 대하여 형태소 분석을 행한다. 나머지 모든 음절은 원래의 음절을 그대로 사용한다. 성공한 경우는 이 어절을 오류 어절에 대한 후보어절로 간주하고 실패한 경우는 무시한다.
2. 그 다음 음절의 모든 후보를 포함하는 어절을 생성하고, 어절에 대해 형태소 분석을 행한다. 1과 같이 나머지 모든 음절은 원래의 음절을 그대로 사용한다. 성공한 경우에 대해서만 후보 어절로 간주한다. 그 다음 음절이 있으면 계속 반복한다.
3. 형태소 분석을 행하기 전에 생성된 음절의 개수가 많을 경우는 〈표 4〉의 음절 위치별 빈도순에 의하여 발생 빈도가 높은 음절에 대해서만 후보로 채택한다[8, 9].

〈표 5〉 한글의 음절별 출현빈도순위
〈Table 5〉 Frequency by Characters

순서무관	1번째 음절	2번째 음절	3번째 음절
이	이	다	이
다	있	이	하
의	대	는	에
는	것	해	을
에	지	고	의
을	전	리	다
고	수	국	한
지	한	한	로
로	정	정	는
한	사	으	를
가	등	기	가
하	그	시	은
대	경	어	적
기	가	제	고
은	국	지	해
서	기	라	했
사	일	상	지
정	보	일	시
도	미	도	기
를	주	을	자

여기서 또 하나 고려할 수 있는 대상 중의 하나는 자소별 빈도이다. 〈표 6〉과 〈표 7〉은 제의 말 뭉치에서 구한 자료인데, 이에 관한 연구 결과가 발표되기도 했다[10, 11].

〈표 6〉 첫 음절에서의 자소별 빈도순
〈Table 6〉 Frequency by Jaso in the 1st character

초성	중성	종성
ㅇ	아	null
ㄱ	오	ㄹ
ㅂ	어	ㄴ
ㅅ	이	ㅇ
ㅁ	으	ㅁ
ㄴ	우	ㄱ

〈표 7〉 임의의 음절에서의 자소별 빈도순
〈Table 7〉 Frequency by Jaso in any characters

초성	중성	종성
ㅇ	아	null
ㄱ	이	ㄴ
ㄹ	으	ㄹ
ㄷ	오	ㅁ
ㅅ	어	ㄱ
ㄴ	우	ㅇ

〈표 7〉에 근거하여 하나의 자소에 대하여 후보 자소가 많을 경우, 빈도순에 의거하여 후보를 줄일 수 있다.

여기서 또 하나 고려할 사항은 하나의 자소가 추가되었거나 누락된 경우이다. 오류 유형의 분석결과 다음과 같이 분류할 수 있다.

1. 종성이 없는 음절에서 종성이 추가된 경우
2. 종성이 있는 음절에서 종성이 누락된 경우
3. 단자음 종성의 음절에서 자음이 하나 추가되어 복자음으로 된 경우
4. 복자음 종성의 음절에서 자음이 하나 누락되어 단자음으로 된 경우
5. 단모음의 음절에서 모음이 추가되어 복모음으로 된 경우
6. 복모음의 음절에서 한 모음이 누락되어 단모음으로 된 경우

1이나 2는, 종성이 없는 음절에서는 종성을 추가하여, 종성이 있는 음절에서는 종성을 삭제하여 형태소 분석을 행함으로써 처리될 수 있다. 3이나 4의 경우는 〈표 3〉의 후보 리스트에 몇몇 개를 추가하여 처리할 수 있다. 가령 종성 '가'의 경우, 'ㄳ'을, 'ㄴ'인 경우에는 'ㄵ'이나 'ㄶ'을 추가하여 4를 해결할 수 있다. 같은 식으로 'ㄺ'인 경우 'ㄱ'이나 'ㄹ'을, 'ㅄ'인 경우 'ㅂ'이나 'ㅅ'을 추가하여 3을 해결할 수 있다. 이를 정리하면 〈표 8〉과 같이된다.

〈표 8〉종성에서 자음 추가로 인해 생성 가능한 복자음
〈Table 8〉 Generating Double Consonant by adding Consonant in the Final Consonant

단자음	복자음
ㄱ	ㄳ, ㄺ
ㄴ	ㄵ, ㄶ
ㄹ	ㄺ, ㄻ, ㄼ, , ㅀ, ㄾ,
ㅁ	ㄻ
ㅂ	ㄼ, ㅄ
ㅅ	ㄳ, , ㅄ
ㅈ	ㄵ
ㅌ	ㄾ
ㅍ	ㄿ
ㅎ	ㄶ

〈표 9〉 복종성에서 자음 탈락으로 인해 생성 가능한 단자음
〈Table 9〉 Generating Single Consonant by Skipping Consonant in Final Double Consonant

단자음	복자음
ㄳ	ㄱ, ㅅ
ㄵ	ㄴ, ㅈ
ㄶ	ㄴ, ㅎ
ㄺ	ㄱ, ㄹ
ㄻ	ㄹ, ㅁ
ㄼ	ㄹ, ㅂ
	ㄹ, ㅅ
ㄾ	ㄹ, ㅌ
ㅀ	ㄹ, ㅍ
	ㄹ, ㅎ
ㅄ	ㅂ, ㅅ

5나 6의 경우도 3과 4와 마찬가지로 〈표 3〉의 후보 리스트에 몇몇 개를 추가하여 처리할 수가 있다. 가령 'ㅜ'의 경우, 'ㅝ'나 'ㅟ'를, 'ㅡ'인 경우에는 'ㅢ'를 추가하여 6을 해결할 수 있다. 같은 식으로 'ㅝ'나 'ㅟ'인 경우 'ㅜ'를, 'ㅢ'인 경우 'ㅡ'나 ' ㅣ'를 추가하여 5를 해결할 수 있다. 이를 정리하면 〈표 10〉과 같이된다.

〈표 10〉 중성 추가로 인해 생성 가능한 복모음
〈Table 10〉 Generating Double Vowel by Adding Vowel

단모음	복모음
아	와
어	워
오	와, 왜, 외
우	워, 웨, 위
으	의
이	의

〈표 11〉 중성 누락으로 인해 생성 가능한 단모음
〈Table 11〉 Generating Single Vowel by Skipping Vowel

복모음	단모음
와	오, 아
왜	오, 애
외	오, 이
워	우, 어
웨	우, 에
위	우, 이
의	으, 이

그리고 오류의 유형 중 낮은 빈도 수를 보이는 경우이기는 하지만 다음의 예들도 나타난다. '둥으로'를 '등오로'로 입력한 경우에서와 같이 바로 인접하지 않은 자소로 대체된 경우이다. 이런 경우는 약 15%에 해당하는데, 이것을 해결하기 위해서는 해당 문자에 덜 인접한 문자들도 〈표 4〉에 삽입함으로써 해결할 수가 있으나 15%의 처리를 위해서 이를 확장했을 때 후보 어절의 개수 증가로 인하여 형태소 분석을 위한 처리 및 사전 검색을 위해 시간이 필요하므로, 적당한 선에서의 조정이 필요하다.

위의 알고리즘으로 IBM 486 PC 상에서 C언어로 구현하여 실험하였다. 자소가 대체된 경우만을 고려하여 실험한 결과, 각 음절에 대한 후보 음절의 평균 개수는 11.1개였으며, 생성된 후보 어절 개수는 어절의 음절 개수에 정비례하는 것으로 나타났다. 즉 어절의 길이가 세 음절이면, 후보어절의 개수는 33.3개였다. 33.3개의 후보 어절에 대한 형태소 분석의 시간은 IBM 486 PC (DX/4 100 MHZ)에서 0.5초 미만이었다. 형태소 분석을 행한 뒤의 최종 후보는 평균 3.1개였다. 이 중에서 맞는 어절은 62.1%가 포함되어져 있었다. 자소대체, 탈락 및 추가를 고려하여 실험한 결과, 최종 후보는 8개였으며, 이 중에서 맞는 어절은 84.1 %가 포함되어져 있었다.

형태소 분석을 위한 과정은 필자가 속한 연구팀이 개발했던 가나다 문서편집기의 일부 기능인 형태소 분석프로그램이 사용되었다[2].

4. 결 론

한국어 문서편집기에서 흔히 나타나는 여러 종류의 철자 오류를 수집하고 분류하여 철자 오류가 있는 어절에 대한 후보 어절을 제시하는데 있어서 방법을 제시하고 이를 구현하였다. 철자 오류의 유형에서 드러났듯이 자소 대체의 오류가 가장 빈번하였으며, 그 중에서도 해당 자소의 가장 인접한 위치에 자리한 자소가 잘못 입력된 경우가 가장 많았다. 이러한 통계 자료에 근거하여 시스템을 구축함으로써 신뢰성 있는 시스템이 만들어질 수 있었다. 오류가 있는 어절에 대해 평균 3.1 어절에서 8 어절이 후보로서 제공되었으며, 제시된 후보 어절 중 올바른 어절이 포함된 경우가 62.1 %에서 84.1%였다.

지금까지의 연구는 자연언어처리의 가장 기초단계인 형태소 분석을 이용하여 철자 교정의 방법과 구현을 제시하였지만, 궁극적으로는 통사분석, 의미분석 등의 연구도 당연히 계속되어야 한다. 그러나 철자 교정의 경우, 이는 주로 실시간 처리의 기반 위에서 이루어져야하기 때문에 통사분석이나 의미분석을 행하더라도 후보어절의 제시 등이 빠른 시간 내에 이루어져야 한다.

참 고 문 헌

[1] 남기심, 고영근, "표준 국어문법론", 탑출판사, 1994

[2] 강승식, "음절 정보와 복수어 단위 정보를 이용한 한국어 형태소 분석", 박사 학위논문, 1993

[3] 권혁철, "한글 및 한국어 정보 처리의 현황", 정보과학회지, 제12권 제8호, pp.3-16, 1994

[4] 김영택, "자연 언어 처리", 교학사, 1994

[5] 미승우, "새 맞춤법과 교정의 실제", 어문각, 1994

[6] 박동순, "컴퓨터 한글코드의 사용과 표준화", 한국정보과학회지, 제6권, 제1호, 1988

[7] 변정용, "훈민정음 원리의 공학화에 기반한 한글 부호계의 발전 방향", 정보과학회지, 제12권 제 8호, pp.72-88, 1994

[8] 권혁철, "단기강좌 : 문자인식기술", pp.213-244, 1992

[9] 이주근, 최홍문, "한국어 음절의 엔트로피에 관한 연구", 대한전자공학회지, Vol. 9,

No. 4, pp.197-204, 1972
[10] 이재홍, 오상현, "한글 음절의 초성, 중성, 종성 단위의 발생 확률, 엔트로피 및 평균 상호 정보량", 대한전자공학회지, Vol. 26, No. 9, pp.1-9, 1989
[11] 이재영, 조영일, "마코프 정보원을 이용한 한글 정보원의 근사적 해석", 정보과학회논문지, 제22권 제6호, pp.965-970, 1995
[12] 남궁건, "한글 낱말의 발생 빈도 분포와 Entropy에 관한 연구", 석사학위논문, 1979
[13] Dongyul Ra, "A Word Sense Disambiguation Method with a Semantic Network", 인지과학, 제3권 제2호, pp.225-247, 1992
[14] H. Takahashi, "A Spelling Correction Method and its Application to an OCR System", Pattern Society, Vol 23, No. 3/4, pp.363-377, 1990
[15] 伊東伸泰, "OCR入力された日本語文の誤り檢出と自動訂正", 情報處理學會論文誌, 第33卷 第5號, pp.664-670, 1992

임 한 규

1981년 경북대학교 전자공학과 졸업(학사)
1984년 연세대학교 산업대학원 전산전공 졸업(공학석사)
1993년~현재 성균관대학교 정보공학과 박사과정 재학중
1981년~82년 대한주택공사 전산실
1982년~86년 한국전자통신연구소 연구원
1986년~94년 한국아이비엠 소프트웨어연구소 선임연구원
1994년~현재 한서대학교 전산정보학과 전임강사
관심분야 : 자연언어처리, 문자인식, DB

김 응 모

1981년 성균관대학교 수학과 학사
1986년 Old Dominion Univ. 전산학 석사
1990년 Northwestern Univ. 전산학 박사
1990년~현재 성균관대학교 정보공학과 부교수
관심분야 : 데이타베이스, 정보검색