한글 문장의 자동 띄어쓰기를 위한 어절 블록 양방향 알고리즘

(*Eojeol*-Block Bidirectional Algorithm for Automatic Word Spacing of Hangul Sentences)

강 승 식 †
(Seung-Shik Kang)

요 약 자동 띄어쓰기는 띄어쓰기가 무시된 한글 문서의 자동색인이나 문자인식 시스템에서 줄바꿈 문자에 대한 공백 삽입 문제 등을 해결하는데 필요하다. 이러한 문서에서 공백이 삽입될 위치를 자동으로 찾아주는 자동 띄어쓰기 알고리즘으로 문장 분할 기법과 양방향 최장일치법을 이용한 어절 인식 방법을 제안한다. 문장 분할은 한글의 음절 특성을 이용하여 어절 경계가 비교적 명확한 어절 블록을 추출하는 것이며, 형태소 분석기를 이용한 양방향 최장일치법에 의해 어절 블록에 나타난 각 어절들을 인식한다. 4,500여 어절로 구성된 두 가지 유형의 문장 집합에 대하여 제안한 방법의 띄어쓰기 정확도를 평가한 결과 '공백 재현율'이 97.3%, '어절 재현율'이 93.2%로 나타났다.

Abstract Automatic word spacing is needed to solve the automatic indexing problem of the non-spaced documents and the space-insertion problem of the character recognition system at the end of a line. We propose a word spacing algorithm that automatically finds out word spacing positions. It is based on the recognition of Eojeol components by using the sentence partition and bidirectional longest-match algorithm. The sentence partition utilizes an extraction of Eojeol-block where the Eojeol boundary is relatively clear, and a Korean morphological analyzer is applied bidirectionally to the recognition of Eojeol components. We tested the algorithm on two sentence groups of about 4,500 Eojeols. The space-level recall ratio was 97.3% and the Eojeol-level recall ratio was 93.2%.

1. 서 론

띄어쓰기를 하지 않는 언어의 분석은 문장에서 단어 경계를 인식하는 작업이 선행되어야 하기 때문에 중국어의 형태소 분석은 입력 문장으로부터 단어들을 분리하는 문제가 가장 중요시 된다[1]. 일본어는 히라가나, 가다가나, 한자를 혼용하기 때문에 최장일치법(longest-segment method), 최소어절(least-*bunsetsu*) 인식법, 문자 유형에 따라 인식하는 방법 등을 이용하여 단어를 분리하고 있다[2]. 한국어는 어절 단위로 띄어쓰기를 하기 때문에 문장 단위의 자동 띄어쓰기의 필요성이 제기되지 않았으며, 정보검색이나 기계번역 등 일부 응용 분야에서 활용하기 위해 복합명사를 분해하는 알고리즘이 개발되었다[3,4,5].

최근에는 문자인식에 의해 대량의 정보자료를 입력할 때 줄바꿈 위치에서 공백을 삽입해야 하는지를 판단하는 문제가 발생하고 있다. 줄바꿈 위치의 띄어쓰기 문제는 전자출판 시스템에서 편집된 자료를 재편집하거나 문서편집기로 작성된 문서를 아스키 형태로 저장한 경우에도 필요하다. 특히, 일부 데이타베이스에서는 정보자료를 저장할 때 어떤 항목을 공백없이 저장한 경우가 있는데, 이러한 문서를 자동색인할 때 자동 띄어쓰기가 필수적이다. 이외에도 자동 띄어쓰기는 한글을 입력할 때 띄어쓰기 문제를 자동으로 처리하거나 철자 검사기에서 띄어쓰기 오류를 발견하는데 활용될 수 있다. 뿐만 아니라, 차세대 사용자 인터페이스로서 연속 어절 음성인식 기능이 상용화되면 그 필요성은 매우 증가할 것으로 예상된다.

† 종신회원 : 한성대학교 정보전산학부 교수
sskang@hansung.ac.kr
논문접수 : 1999년 5월 24일
심사완료 : 2000년 1월 21일

이러한 필요성에 따라 한글 문장의 띄어쓰기 문제를 '공백 인식 접근법'과 '어절 인식 접근법'이라는 두 가지 관점에서 정의하고, 문장을 어절 블록으로 분할하여 어절 경계를 인식하는 띄어쓰기 알고리즘을 제안한다.

2. 기존 연구 및 접근 방법

2.1 기존의 연구

한글 문서의 자동 띄어쓰기는 두 어절을 붙여썼을 때 자동으로 공백을 삽입하는 문제와 문장 전체 또는 여러 어절들을 붙여썼을 때 각 어절 경계에 공백을 삽입하는 자동 띄어쓰기 문제로 구분된다. 두 어절 사이의 띄어쓰기는 맞춤법 검사기에서 띄어쓰기 오류를 교정하는데 활용되고 있으며 오류 유형을 중심으로 처리하는 방법을 취한다. 맞춤법 검사기에서 붙띄오류와 띄붙오류를 자동으로 교정하는 방법으로 최재혁(1997)은 오류어 유형에 따라 처리하는 방법을 제안하였다[6].

문장 전체나 여러 어절에 대한 자동 띄어쓰기는 심광섭(1996)이 처음으로 시도하였으며, 말뭉치에서 획득한 이웃한 음절 사이의 띄어쓰기-붙여쓰기 *bigram* 음절특성을 이용한 통계적 기법과 형태소 분석기를 이용하고 있다[7]. 이와 유사한 방법으로 신중호(1997)는 음절 *bigram* 및 단어 *bigram* 정보를 이용하여 동적 프로그래밍에 기반한 어절 인식 알고리즘을 제안하였다[8]. 김계성(1998)은 음절 정보와 경험 규칙을 이용하여 어절을 분리한 후에 형태소 분석기와 어절 재결합에 의한 띄어쓰기 알고리즘을 제안하였다[9].

통계적 방법은 *bigram* 음절 정보를 획득할 때 사용한 말뭉치의 문서 유형에 따라 유사 분야의 문장들에 대한 성능이 매우 우수하지만 다른 유형의 문서들에 대해서는 정확도가 달라질 수 있으며, 규칙기반 방법에서는 띄어쓰기 오류가 발생했을 때 지속적으로 오류를 수정하는 번거로움이 있다. 기존의 연구에서 자동 띄어쓰기 문제를 접근하는 방법을 정리하면 이웃한 두 음절 사이에 공백 삽입 여부에 의한 '공백 삽입 접근법'과 어절들을 인식함으로써 어절 경계에 공백을 삽입하는 '어절 인식 접근법'으로 분류된다.

2.2 접근 방법

2.2.1 공백 삽입 접근법

심광섭(1996)은 n 음절로 구성된 문장을 $s_1s_2\cdots s_{i-1}s_is_{i+1}\cdots s_{n-1}s_n$라고 할 때 자동 띄어쓰기 문제를 음절과 음절 사이에 공백을 삽입하는 문제로 정의하였다[1]. 공백은 음절과 음절 사이에 삽입될 수 있으므로 공백 삽입이 가능한 위치는 모두 $(n-1)$개이다. $(n-1)$ 공백위치 중에서 i개의 공백을 삽입하는 경우의 수는 ${}_{n-1}C_i$이므로 문장에 0, 1, 2, ..., $n-1$개의 공백이 삽입되는 모든 가능성을 계산하면 $\sum_{i=0}^{n-1} {}_{n-1}C_i = 2^{n-1}$ 이고, 자동 띄어쓰기는 이 중에서 옳은 답을 찾는 탐색 문제(search problem)로 정의된다.

띄어쓰기가 옳은 답을 찾는 방법으로는 음절 *bigram* 및 어절 *bigram* 특성을 이용하는 방법이 사용되고 있다. 심광섭(1996)은 말뭉치로부터 임의의 두 음절 사이에 공백이 삽입된 빈도수를 이용하여 공백 삽입 확률을 구하고 임계치를 초과한 음절 경계에 공백을 삽입하는 *bigram* 음절 특성을 이용하고 있다. 또한, 형태소 분석기를 이용하여 인식된 어절을 확인하여 띄어쓰기 정확도를 높이는 효율적인 방법을 제안하였다.

2.2.2 어절 인식 접근법

어절 인식을 기반으로 한 자동 띄어쓰기는 문장내에 출현한 어절을 인식하여 어절 경계에 공백을 삽입하는 접근 방식이다. 즉, 문장을 구성하고 있는 어절이 인식되면 어절 경계에 공백을 삽입할 수 있으며, 세 어절로 구성된 문장의 경우에 두 번째 어절이 인식되었다면 그 어절의 양쪽 경계에 공백을 삽입한다.

이 접근법에서 어절들을 인식하기 위해 모든 부분문자열(substring)에 대해 형태소 분석기를 호출하고 이로부터 문장을 구성하는 조합을 발견할 수 있다. 그러나 모든 부분문자열에 대해 형태소 분석을 하는 것은 비효율적이므로 문장의 한쪽 끝에서부터 순차적으로 다음 어절들을 인식하는 방법이 가능하다.

3. 자동 띄어쓰기 알고리즘

어절 인식 접근법에 의한 띄어쓰기 알고리즘으로는 순방향, 역방향, 양방향 알고리즘이 가능하다.

3.1 순방향-역방향 알고리즘

어절 인식을 위한 순방향(forward) 알고리즘은 입력 문장을 전진 방향(좌에서 우로)으로 진행하면서 형태소 분석기를 이용하여 순차적으로 어절을 인식하는 방법이고, 역방향(backward) 알고리즘은 진행 방향으로 반대로 하여 문장끝에서부터 후진 방향으로 어절을 인식해 나가는 방법이다.

n개의 음절로 이루어진 문장 $s_1s_2\cdots s_{i-1}s_is_{i+1}\cdots s_{n-1}s_n$에서 음절 s_i부터 시작되는 어절의 인식은 1음절어 s_i, 2음절어 s_is_{i+1}, 3음절어 $s_is_{i+1}s_{i+2}$ 등 두 가지 이상의 중의성이 발생하는 경우가 발생한다. 이러한 중의성 문제는 길

1) 실제 문장에서는 문장부호와 숫자, 영문자 등 한글 이외의 문자가 포함되기도 한다.

이가 긴 어절을 우선으로 취하는 최장일치법을 적용하고 다음 어절을 인식할 때 문제가 발생했을 때 순서대로 짧은 어절을 취하는 퇴각검색 기법을 이용하여 해결이 가능하다.

그런데 어떤 경우에는 어절 인식 중의성으로 인해 앞 어절 인식 오류가 다음 어절을 인식하는데 영향을 주게 되고 다음 어절들이 계속해서 오인식되는 전파오류(triggered errors)가 발생하기도 한다. 대부분의 일반적인 문장에서는 조사나 어미가 오류가 전파되는 것을 차단해 주기 때문에 퇴각검색에 의해 전파오류가 자동적으로 해결되기도 한다.

그러나 드물긴 하지만 아래 예와 같이 '기초지방자치단체' 등 일부 복합명사에서 전파 오류가 심각한 문제를 야기할 수도 있다.

기초지 방자 치단 체 행 정전 산화[2)]

특히, 어절 인식기(형태소 분석기)가 1음절어를 옳은 어절로 인식하는 것을 허용했을 때 전파오류 문제는 매우 심각하다. 국어사전 표제어에 1,700여개의 음절이 사용되고, 사용빈도가 매우 낮은 음절들을 제외할 때 한글 문서에서 주로 사용되는 음절수는 1,300여개로 이중에서 1음절 어휘형태소 개수가 1,000여개를 차지한다. 즉, 형태소 분석기 대부분의 1음절어를 옳은 어절로 인식하게 된다. 따라서 일단 최장 첫 어절이 인식되고 나면 퇴각검색에 의해 옳은 어절을 인식할 가능성은 매우 낮다. 왜냐하면, 다음 어절로 2음절어 이상이 인식되지 않더라도 1음절어를 인식한 후에 다음으로 진행하기 때문이다.

이와 같이 1음절어 인식을 허용하면 대부분의 문장에서 전파오류가 빈번하게 발생할 뿐만 아니라 미등록어를 1음절어 혹은 2음절어로 분해하게 되어 정확도를 저하시키는 요인이 된다. 이러한 문제점을 예방하려면 형태소 분석기로 하여금 1음절어, 특히 1음절 명사를 인식하지 못하도록 해야 한다. 그러나 '이/그/저', '한/두/세/네', '때/등/중'과 같은 관형사와 의존명사, 1음절 용언 등을 인식하지 못하게 되므로 여전히 문제가 남아 있다.

3.2 양방향 최장일치법

양방향 최장일치법은 순방향 혹은 역방향 알고리즘에서 문제점으로 지적되는 전파 오류와 미등록어로 인한 어절 인식 오류, 1음절어 인식 오류 문제를 최소화하기 위하여 문장의 양쪽끝에서 동시에 진행하는 방법이다[6,10]. 즉, 순방향 알고리즘과 역방향 알고리즘을 조합하여 문장의 첫부분 혹은 끝부분부터 어절을 인식하다가 더 이상 어절이 인식되지 않으면 다른쪽 끝에서부터 반대쪽으로 어절을 인식한다.

이 방법은 양쪽끝에서 어절 인식이 동시에 진행되므로 전파 오류로 인한 정확도 감소를 줄일 수 있으며, 미등록어의 어절 경계를 인식하기가 용이하다. 또한, 어절을 인식할 때 양쪽 끝에서 최장일치 어절을 우선적으로 취함으로써 어절 인식 오류를 감소시키는 효과가 있다. 그러나 이 알고리즘에서도 전파 오류가 발생할 수 있으며, 미등록어가 두 개 이상 포함된 문장을 처리하기가 어렵다.

이러한 전파 오류와 미등록어로 인한 오류는 문장의 길이가 긴 경우에 더 자주 발생하기 때문에 띄어쓰기의 단위를 문장 전체가 아니라 어절 블록으로 분할하여 처리함으로써 오류 발생 가능성을 줄이는 방법을 취한다.

3.3 어절 블록 양방향 알고리즘

길이가 짧은 문장에 대한 띄어쓰기는 양방향 최장일치법만 이용하더라도 어절 인식 중의성이나 전파 오류의 발생 확률이 높지 않기 때문에 비교적 정확하게 어절을 인식하는 것이 가능하다. 그러나 긴 문장에 대해서는 어절 인식 중의성과 전파 오류에 의해 오인식된 어절이 다수 발견되어 정확도가 낮아지게 된다.

어절블록 양방향 알고리즘은 어절 경계가 비교적 명확한 어절 블록을 인식한 후에 어절 블록내에서는 양방향 최장일치법을 적용하는 방법이다. 즉, 어절 블록은 어절을 잘못 인식함으로 인하여 발생하는 전파 오류를 차단하는 역할을 한다. 그리고 블록내에서는 양방향 최장일치법을 적용한다.

한 문장을 몇 개의 어절 블록으로 분할할 때 블록과 블록 사이의 경계는 어절 경계와 일치하는 것이 바람직하다. 그러나 특정 어절이 두 블록으로 분할되지 않도록 완벽하게 블록 경계를 설정하기는 쉽지 않다. 따라서 가급적 블록 경계가 어절 경계와 일치하도록 문장을 분할하여 어절 블록을 인식하고, 어절 블록과 어절 경계가 일치되지 않은 경우는 어절 인식 오류를 수정하는 단계에서 처리하는 방법을 취한다.

```
Algorithm word_spacing(SYLLABLE s[])
{
   int i=0; /* 어절블록 시작 인덱스 */
   int j=0; /* 어절블록 끝 인덱스 */
   int k, n=strlen(s);
   int sp[SENTSIZE] = { 0 };

   while (j < n) {
```

2) '기초지'는 '기초'명사+'이'서술격조사+'지'어미로 분석되고, '전산화'명사는 사전에 수록되지 않은 단어이다.

```
        j = wblock_end_index(s, i, n);
        i = set_spaces(s, i, j, sp);
    }
    adjust_word_spaces(s, sp);
}
```

그림 1 어절 블록 양방향 알고리즘

어절 블록 양방향 알고리즘은 그림 1과 같다. 입력 문장 s에 대해 어절 블록의 시작 인덱스 i의 초기값을 0으로 하고 함수 wblock_end_index에 의해 어절 블록의 끝 인덱스 j를 구한다. 어절 블록 s[i]~s[j-1]에 대해 양방향 최장일치법(함수 set_spaces)을 이용하여 어절들을 인식하고 공백이 삽입될 위치를 배열 sp[]에 0 또는 1로 표시한다[3].

어절 블록의 인식이 정확하다면 다음 어절 블록의 시작 인덱스 i는 앞 어절 블록의 끝 인덱스가 된다. 그런데 어절 블록 경계가 어절 경계와 일치하지 않은 경우가 있으므로 어절 블록의 끝 어절이 형태소 분석에 실패한 불완전 어절이면 함수 set_spaces는 끝 어절의 시작 위치를 반환하여 다음 어절 블록의 시작 인덱스로 한다. 즉, 다음 어절 블록의 시작 인덱스는 j 또는 앞 어절 블록의 끝 어절의 시작 위치가 되며, 이는 함수 set_spaces의 반환값이다.

어절 블록의 끝부분이 어절로 인식되더라도 어절 블록의 끝이 어절 경계가 아닌 경우에 한 어절이 두 블록에 걸치는 경우도 있다. 이 경우는 모든 어절 블록에 대한 어절 인식이 끝난 후에 후처리 과정(4.3절)에서 설명하는 함수 adjust_word_spaces에 의해 오류를 교정한다. 자동 띄어쓰기 결과를 출력할 때는 입력문장의 각 음절 s[i]에 대해 sp[i]=1이면 s[i] 앞에 공백을 삽입한다.

4. 어절 블록 및 어절 인식

조사 '는'과 '를'은 체언 어절의 끝에 사용되므로 어절 경계일 가능성이 매우 높다. 특히, 많은 어절이 조사나 어미를 동반하고 있으며, 조사/어미로 사용되는 음절은 어휘형태소의 첫부분으로 사용될 가능성이 매우 낮다. 이처럼 조사/어미의 음절 특성에 의해 비교적 어절의 끝일 가능성이 높은 곳을 추정함으로써 어절 블록의 경계를 인식할 수 있다.

통계적 기법을 도입하여 *bigram* 음절 특성을 이용하고 임계치를 조절함으로써 어절 블록의 경계를 인식하는 방법이 가능하다. 그러나 통계적 기법은 상대적으로 더 많은 기억공간이 필요하므로 본 논문에서는 이와 유사한 성능으로 어절 블록을 인식할 수 있는 조사/어미 음절 특성을 이용하는 방법을 사용한다.

4.1 어절 블록 인식

조사로 사용되는 음절수는 70여개, 어미로 사용되는 음절수가 130여개로 한글 문서에서 사용되는 음절수 1,300여개에 비해 그 수가 매우 적으므로 문장내에 나타난 조사/어미의 음절 특성을 이용하여 어절 블록의 경계를 인식할 수 있다. 조사/어미의 음절 특성과 조사/어미 사전을 이용하여 문장내에서 조사와 어미가 발견된 위치를 표시한다.

구문분석과j_1j_2의j_1미분석이j_1j_3라$j_1e_1e_2$는j_1e_1어e_1려e_1운문제
구문분석과/의/미분석이/라/는/어/려/운문제

ㅇ 예문은 조사와 어미의 인식결과와 어절 블록 가능 위치를 '/' 기호로 표시한 것이다. '과'로부터 시작되는 조사는 1음절 '-과'와 2음절 '-과의'가 가능하므로 첫음절 '과'에 j_1j_2라고 표시하였다. 에서 각 음절이 조사/어미의 첫음절로 사용되는지에 따라 음절 특성을 부여하는데, 조사/어미의 음절 길이에 따라 i-음절 조사의 첫음절은 j_i, i-음절 어미의 첫음절은 e_i로 구분한다[4]. 이처럼 조사/어미의 인식결과에 의해 조사/어미의 시작 및 끝 위치를 알 수 있으며 이를 기준으로 어절 블록의 경계를 추정한다.

조사/어미의 인식결과에 의해 어절 블록의 경계를 추정할 때 '과/의'와 '이/라/는/어/려'와 같이 조사/어미의 끝음절들이 2개 이상 연속해서 나타났을 때 어절 블록이 잘못 설정되어 한 어절이 두 어절 블록으로 분할된다. 최장 조사/어미를 우선으로 하더라도 [구문분석과의][미분석...]으로 구분되는 문제가 발생한다. 어절 블록 인식 오류를 최소화하기 위하여 강승식(1995)에 따라 빈도가 높은 조사만으로 조사/어미의 빈도별 특성에 따라 아래와 같이 3가지 경우에 대해 실험하였다[11].

① 조사/어미를 모두 사용한 경우
② 저빈도 조사/어미를 제외한 경우
③ 고빈도 조사만 사용한 경우

3) sp[i]가 1이면 문자 s[i] 앞에 공백이 삽입되고, 0이면 공백이 삽입되지 않는다. sp[i]의 값은 공백 삽입 여부에 따라 0, 1을 가지게 할 수도 있고 공백 삽입 여부를 점수로 수록하여 5단계 혹은 10단계와 같이 세분화하여 사용할 수도 있다.

4) 어미 'ㄴ/ㄹ/ㅁ/ㅂ'은 정보를 부가하지 않으며, '은/는', '이/가' 등이 체언과 결합 제약을 위반하면 조사/어미 특성을 부여하지 않는다.

그 결과로 '고빈도 조사만 사용한 경우'가 가장 효과가 좋았으며, 고빈도 조사의 인식 결과에 의해 어절 블록의 끝 위치를 찾는 알고리즘은 다음과 같다.

```
int wblock_end_index(s, i, n)
SYLLABLE s[]; int i, n;
{
  int j;
  for (j = i+2; i < n; i++)
    if (high_freq_josa(s, j, n)
      return j + n_syl_josa(s, j);
  return n;
}
```

그림 2 어절 블록의 끝 인덱스 알고리즘

이 알고리즘에서 인자 i는 어절 블록의 시작 음절이고 n은 입력 문장의 끝 위치이다. 변수 j의 초기값을 i+2로 한 이유는 어절 블록 크기가 최소한 2음절 이상이 되도록 하기 위한 것이다. 즉, 어절 블록의 처음 2음절은 고빈도 조사라 할지라도 3번째 음절 이후의 고빈도 조사를 블록의 끝 위치로 한다. 그 이유는 '휴가를갔다'에서 '휴가'와 '를갔다'로 분할했을 때처럼 두 번째 음절이 고빈도 조사일 때 한 어절이 두 블록으로 분할되는 경우가 있기 때문이다.

함수 high_freq_josa(s, j, n)는 s의 j번째 음절부터 끝 음절 n까지 좌에서 우로 조사를 검사하여 고빈도 조사를 발견한다. 이 때 1음절 조사 '는/를'이 발견되면 우선적으로 추출하고, 기타 1음절 조사들은 그 위치에서 2음절 이상의 조사가 발견되지 않은 경우에만 어절 블록의 끝으로 간주한다. 또한, 고빈도 조사가 2개 연속되거나 한 음절 건너에 고빈도 조사가 있는 경우에는 2번째 조사를 블록의 끝으로 간주하며, 고빈도 조사지만 '가/고'는 어절의 첫음절 사용빈도가 높기 때문에 어절 블록의 끝을 인식할 때 우선순위를 낮춘다.

4.2 어절 블록내의 어절 인식

어절 블록내에서 어절들을 인식하는 방법은 형태소 분석을 이용한 양방향 최장일치법을 이용한다. 먼저, 조사 부분을 제외한 어절 블록의 음절수가 4음절 이하인 경우는 음절수에 따라 처리한다. 어절 블록 전체가 형태소 분석 성공이면 하나의 어절로 간주하고, 3음절이면 2+1 또는 1+2, 4음절인 경우는 2+2, 1+3, 3+1로 분할하여 검사한다(그림 3).

조사 부분을 제외하고 5음절 이상인 어절 블록은 역방향 분석을 먼저 시도한다. 순방향보다 역방향 분석을 우선으로 한 이유는 어절 블록의 끝부분이 조사로 끝나기 때문에 형태소 분석에 성공할 가능성이 많기 때문이다. 역방향 분석은 5/4/3/2 음절에 대해 순서대로 형태소 분석을 시도하여 최장일치 어절을 발견한다. 형태소 분석에 의해 어절이 인식되면 인식된 어절의 첫음절 위치가 i일 경우 sp[i]=1로 한다.

일반적으로 4음절 이상 어휘형태소는 2음절 혹은 3음절 어휘형태소로 구성된 복합어인 경우가 많으므로 1음절어보다는 2음절어나 3음절어를 우선으로 한다. 형태소 분석 성공 어절이 더 이상 발견되지 않으면, 순방향 분석을 같은 방법으로 시도한다. 어절 인식이 끝난 후에도 인식되지 않은 어절은 미등록어로 간주한다.

```
int set_spaces(s, i, j, sp)
SYLLABLE s[];
int i, j;  /* 어절블록 시작-끝 인덱스 */
int sp[]; /* 공백 표시 */
{
  if (is_word(s, i, j)) return 0;

  if (j-i == 3음절)
    /* 2+1, 1+2 유형 인식 */
  else if (j-i == 4음절)
    /* 2+2, 1+3, 3+1 유형 인식 */
  else
    /* 양방향 최장일치 어절 인식 */

  if (끝어절 == 불완전어절)
    return 끝어절시작인덱스;
  else return j;
}
```

그림 3 어절 인식 및 공백 표시 알고리즘

4.3 어절 인식 오류 교정

그림 1의 후처리 함수 adjust_word_spaces는 경험규칙에 의해 case-by-case로 구현되므로 알고리즘으로 기술하기가 곤란하며 그 내용을 설명하면 다음과 같다.

어절 인식 오류에는 블록 경계 오류와 블록내에서 어절 인식 오류가 있다. 블록 경계 오류 중에서 경계 어절(어절 블록의 끝어절)에 대한 형태소 분석이 실패한 것은 다음 어절 블록에 결합하여 어절 인식을 시도함으로써 해결된다. 그러나 경계 어절(어절블록의 마지막 어절)에 대한 형태소 분석이 성공한 것은 옳은 어절로 간주되므로 다음 어절 블록에 대한 처리가 끝난 후에 전 블록의 마지막 어절과 후 블록의 처음 블록에 대해 재

분석을 시도하는 방법으로 오류를 교정한다.

어절 블록 인식 오류로 인해 발생하는 어절 인식 오류 유형은 다음과 같다. 앞 어절 블록의 끝어절과 현재 어절 블록의 첫어절에 대한 형태소 분석 결과에 따라

- 둘 다 형태소 분석 실패
- 둘 다 형태소 분석 성공
- 한 쪽만 형태소 분석 실패

로 구분된다. 또한, 형태소 분석에 성공했다 하더라도 끝어절이나 첫어절이 1음절어인 경우에는 두 어절에 대한 어절 인식 결과를 교정해 준다. 각 경우에 대한 교정 방법은 경험규칙에 의해 case-by-case로 구현되며, 아래 1음절어 오류 및 띄어쓰기 중의성 오류를 처리하는 방법과 같은 방법에 의해 처리한다.

어절 블록내에서 어절 인식 오류는 주로 1음절어로 인하여 발생한다. 순방향 분석과 역방향 분석이 끝난 후 1음절이 남으면 1음절어가 인식되지만 그렇지 않은 경우도 발생한다. 인식되지 못한 1음절어 중에서 비교적 빈도가 높은 '이/그/저/한/두/세/네'와 '-을수있다'의 '수'는 독립 어절로 간주하는 것이 좋으나 기타 1음절어들을 인식하는 문제는 더 많은 연구가 요구된다.

어절 인식 오류 중 많은 경우가 띄어쓰기 중의성 때문에 발생한다. 예를 들어, '수정하여만들었다'를 '수정하여만 들었다'로 분해하는 오류는 최장일치 어절을 우선으로 하기 때문에 발생한다. 이러한 유형의 오류들은 이웃한 두 어절을 비교하여 경험규칙에 의해 교정이 가능하다. 즉, 두 어절이 모두 옳은 어절로 인식되었다고 하더라도 1음절씩 좌우로 이동하여 분석했을 때 어휘형태소의 길이가 1인 것보다는 2인 것을 우선으로 한다. 이와 더불어 숫자나 영문자 뒤에 오는 조사/어미는 별도로 인식하여야 하며, 쉼표와 괄호 등 문장부호 또한 별도의 처리 과정이 필요하다.

5. 실 험

본 논문에서 제안한 알고리즘을 구현하여 두 가지 유형의 문장 집합에 대해 실험하였다. 문장 집합 1은 여러 가지 유형의 문서에서 임의로 추출한 문장들이고, 문장 집합 2는 본 논문에서 수식과 표 등 한글 이외의 문자들로 구성된 문장들을 제외한 것이다. 두 실험 데이타의 크기는 표 1과 같다.

표 1에서 line수는 실험문장들의 줄수로 긴 문장은 2줄 이상으로 분할된 것도 있다. 공백위치수는 공백이 삽입될 수 있는 위치의 수로 n 음절로 구성된 문장의 경우에 공백위치수는 n-1개이다. 띄어쓰기 실험 결과를 두 가지 측정 기준에 따라 재현율을 계산하였다. '공백 재현율'은 공백이 삽입될 수 있는 모든 음절(또는 아스키 문자) 위치(공백 위치수)에 대하여 띄어써야 할 위치에 공백을 삽입한 것과 붙여써야 할 위치에 공백을 삽입하지 않은 것을 백분율로 계산한 것이다. '어절 재현율'은 문장 집합에 나타난 모든 어절수에 대하여 띄붙오류와 붙띄오류를 제외한 어절수를 계산하였다[5].

표 1 실험 데이타의 크기

	line수	어절수	공백위치수
문장집합1	150	1,882	5,790
문장집합2	285	2,599	7,928

표 1의 두 가지 문장집합에 대한 띄어쓰기 실험 결과는 표 2와 같다. '공백 재현율'과 '어절 재현율'은 비교 대상이 되는 모범 답안에 따라 오류의 수가 2배 이상 차이가 날 수 있다. 예를 들어 문장집합2에서 '정보전산학부', '자동색인', '문자인식'의 띄어쓰기 결과는 각각 '정보 전산 학부', '자동 색인', '문자 인식'이다. 이와 같이 띄어쓰기와 붙여쓰기가 모두 허용되는 것은 옳은 것으로 간주하였다.

표 2 자동 띄어쓰기 실험 결과

	공백 재현율	어절 재현율
문장집합1	97.2%	92.9%
문장집합2	97.4%	93.5%
평 균	97.3%	93.2%

6. 결 론

본 논문에서는 띄어쓰기 문제를 해결하기 위하여 '어절 인식 접근법'을 취했으며, 알고리즘의 복잡도를 줄이고 오류를 최소화하기 위하여 어절 블록을 추출하는 문장 분할 기법을 도입하였다. 또한, 어절을 인식할 때 발생하는 오류를 줄이기 위하여 어절 블록내에서 양방향 최장일치법에 의해 어절을 인식하는 방법을 이용하였다. 어절 블록내 어절 인식결과로 어절 블록의 경계가 잘못 설정된 것은 후처리에서 오류를 교정하여 정확도를 높였다.

5) 심광섭(1996)은 '공백 재현율'을 '음절 단위 정확도', '어절 재현율'을 '어절 단위 정확도'라 하였다.

본 논문에서 제안한 방법의 띄어쓰기 정확도를 평가한 결과 '공백 재현율'이 97.3%, '어절 재현율'이 93.2%였다. 이 방법은 문서나 문장 유형에 무관하게 적용되기 때문에 유형에 따른 정확도 편차가 적을 뿐만 아니라 어절 블록 인식을 위해 사용되는 조사/어미의 음절 특성은 형태소 분석시에 사용되는 정보를 그대로 활용하므로 기억 공간이 적고 실행 효율이 뛰어나다.

참 고 문 헌

[1] Chen K. J. and Liu S. H., "Word Identification for Mandarin Chinese Sentences," Proceedings of the 14th International Conference on Computational Linguistics, pp.101-107, 1992.

[2] Nobesawa S., et. el, "Segmenting a Sentence into Morphemes using Statistic Information between Words," Proceedings of the 15th International Conference on Computational Linguistics, pp.227-233, 1994.

[3] 윤보현, 조민정, 임해창, "통계정보와 선호 규칙을 이용한 한국어 복합명사의 분해", 정보과학회 논문지(B), 24권 8호, pp.900-909, 1997.

[4] 심광섭, "합성된 상호정보를 이용한 복합명사 분리", 정보과학회 논문지(B), 24권 11호, pp.1307-1317, 1997.

[5] 강승식, "한국어 복합명사 분해 알고리즘", 정보과학회 논문지(B), 25권 1호, pp.172-182, 1998.

[6] 최재혁, "양방향 최장일치법을 이용한 한국어 띄어쓰기 자동 교정 시스템", 한글 및 한국어 정보처리 학술발표 논문집, pp.145-151, 1997.

[7] 심광섭, "음절간 상호정보를 이용한 한국어 자동 띄어쓰기", 정보과학회 논문지(B), 23권 9호, pp.991-1000, 1996.

[8] 신중호, 박혁로, "음절단위 bigram 정보를 이용한 한국어 단어 인식 모델", 한글 및 한국어 정보처리 학술발표 논문집, pp.255-260, 1997.

[9] 김계성, 이현수, 이상조, "연속 음절 문장에 대한 3단계 한국어 띄어쓰기 시스템", 정보과학회 논문지(B), 25권 12호, pp.1838-1844, 1998.

[10] 최재혁, 이상조, "양방향 최장일치법에 의한 한국어 형태소 분석에서의 사전 검색 횟수 감소 방안", 정보과학회 논문지, 20권 10호, pp.1497-1507, 1993.

[11] 강승식, "상대적 출현빈도를 이용한 조사/어미 사전의 구성", 한글 및 한국어 정보처리 학술발표 논문집, pp.188-194, 1995.

강 승 식

1986년 서울대학교 컴퓨터공학과 학사. 1988년 서울대학교 대학원 컴퓨터공학과 석사. 1993년 서울대학교 대학원 컴퓨터공학과 박사. 1992년 한국 IBM 소프트웨어 연구소 연구원. 1993년 서울대 컴퓨터 신기술 공동연구소 연구원. 1994년 ~ 현재 한성대학교 정보전산학부 조교수. 관심분야는 형태소분석, 자동색인, 한국어정보처리