접속정보를 이용한 한글 철자 및 띄어쓰기 검사기의 설계 및 구현

강재우°, 송춘환, 김연배, 최기선, 권용래, 김길창
한국과학기술원 전산학과

A Design and Implementation of Hangul Spelling and Word-spacing Checker using Connectivity Information

J.W. Kang°, C.H. Song, Y.B. Kim, K.S. Choi, Y.R. Kwon and G.C. Kim
Dept. of Computer Science, KAIST
E-mail:jwkang@cslabl.kaist.ac.kr

요 약

본 논문은 UNIX™ 환경에서의 한글 텍스트에 대해 일괄 처리 방식으로 한글 철자 및 띄어쓰기를 검사하는 시스템을 설계 및 구현하였다. 본 시스템은 접속 정보를 이용한 최단일치법을 사용하여 한 어절에 대해 형태론적인 분석을 하여 입력된 화일 내의 철자 및 띄어쓰기 오류를 찾아낸다.

I 서 론

컴퓨터 기술의 급격한 발달과 컴퓨터의 광범위한 보급으로 인하여 문서 편집기(word processor)를 이용하여 문서를 작성하는 일이 많아졌다. 특히, 많은 한글 문서 편집기가 개발, 보급되어 한글로 문서를 작성하는 사람들이 많아지고 있다. 또한, 컴퓨터에 의한 자연어 처리 시스템이나 자연어 질의 응답 시스템 등의 연구가 활발히 진행되고 있으며, 출판 문화의 발달로 신문사 또는 개인이 컴퓨터를 이용하여 기사나 원고를 편집 및 작성하고 있다. 이와 더불어 입력된 문서의 철자 및 띄어쓰기 오류의 검사 및 교정을 자동으로 처리하는 시스템의 필요성이 절실히 요구되고 있다. 자연어 처리 분야에서도 기계 번역을 위한 형태소 분석의 입력으로 한글 철자 및 띄어쓰기가 올바른 문장을 가정하고 만들어지는 경우가 대부분이다.[1,3,6]

영어 문화권에서는 영어 철자 검사 및 교정기(spelling checker)가 1960년대부터 연구되어 실용적인 시스템이 많이 개발되어 있다.[12,13,14] 한글 시스템으로는 한글 철자 교정 시스템[2]이 발표되기는 하였지만, 특정 시스템(NEC/S100)의 워드 프로세서에서만 사용 가능하고, 한글에 대해 제한된 처리만을 하여 실용성이 없었다.

본 논문은 한글 문서에 대해 일괄(batch) 처리를 하는 한글 철자 및 띄어쓰기 검사기를 설계 및 구현하였다. 본 시스템은 한 어절 내에서 형태소들의 접속 형태를 나타내는 좌우 접속정보표, 형태소와 이에 대한 좌우 접속정보로 구성된 사전 그리고 사전 편집기로 구성되어 있다. 철자 및 띄어쓰기 검사는 하나의 어절에 대해 형태소 분석을 통하여, 이 어절이 가능한 형태소들로 접속될 수 있는가 검사한다. 본 시스템은 UNIX 환경에서 C[11] 언어로 구현되었다.

UNIX™ is a trade mark of Bell Laboratories.

II 시스템 개요

본 시스템은 접속정보표와 사전, 사전 편집기, 그리고 한글 철자 및 띄어쓰기 검사 부분(hspell)으로 구성된다. (그림 1 참조)

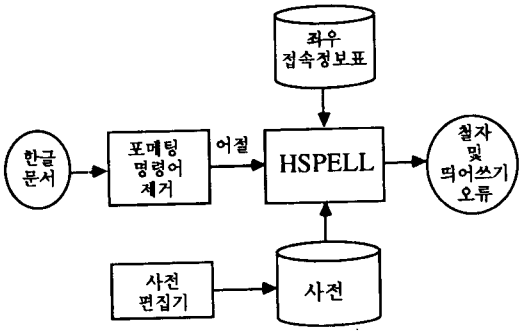

그림 1. 시스템의 개괄적인 block diagram

다음은 시스템의 각 부분에 대한 개략적인 설명이다.

1) 접속정보표

한 어절 내에서 형태소들의 접속 형태에 따라 그들을 범주화하여 좌우 접속 관계를 나타내는 표이다.

2) 사전

형태소와 이에 대한 좌우 접속 정보 쌍으로 구성되어 있다. 하나의 형태소가 여러개의 접속 범주에 속할 수 있으므로 여러개의 좌우 접속 정보의 쌍이 있을 수 있다.

3) 사전 편집기

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

4) hspell

사전과 좌우 접속 정보를 적재(load)한 후, 주어진 입력 문서를 어절 단위로 형태소 분석을 하여 철자 및 띄어쓰기를 검사하는 부분이다.

III 한국어에 대한 고찰과 오류의 종류

자연 언어의 기계적인 처리를 하기 위해서는 대상 언어에 대한 체계적인 연구가 필요하다. 특히, 한글 철자 및 띄어쓰기 검사기를 구현하는데 필요한 한국어의 특성 및 어절의 형태에 대해 살펴보고, 그 오류의 종류를 제시한다.

3.1 한국어의 특성

컴퓨터 처리시 고려해야 할 한국어의 특성은 다음과 같다.[1,4,5,8]

- 한글은 모아쓰기 특징을 가진다. 한글은 본질적으로 자음과 모음으로 구성된 언어이다. 이 자모가 조합되어 단음절을 형성한다. 따라서 컴퓨터 처리시 한글 코드가 조합형이 바람직하다.
- 띄어쓰기가 일정하지 않고 복잡하다.
- 한국어는 우랄 알타이 계통의 교착어(첨가어)로서 각 낱말의 어미 변화에 의해 문장의 성분을 결정하며, 첨용과 활용이 자유롭다.
- 조사와 주어의 생략이 자유롭고, 불규칙 현상 및 음운 현상이 발달하였다.
- 한글과 한문 및 영문을 혼용하여 사용하는 경우가 많다.

3.2 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 띄어쓰기가 이루어지므로, 한 어절내에서 형태소를 파악하여 오류를 찾는 것이 철자 및 띄어쓰기의 과제이다.[10]

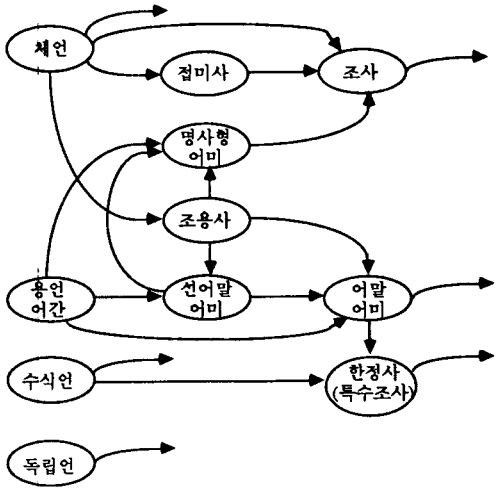

그림 2. 한국어 어절의 형태에 대한 network

형태소들의 접속 양상을 알기 위해서는 한국어 어절의 구조를 파악하여야 한다. 한국어 어절의 개략적인 구조는 그림 2와 같이 나타낼 수 있다.[2,7,18]

화살표(->)는 화살표의 좌측의 범주와 우측의 범주가 접속 가능함을 의미한다.

3.3 오류의 종류

철자 및 띄어쓰기 검색시에 발생할 수 있는 오류로서는 검사기가 가지는 시스템 오류와 이 시스템에 들어오는 입력문이 가지는 철자 및 띄어쓰기 오류인 입력문 오류이다.

시스템이 가지는 오류로는

(1) 시스템이 철자 및 띄어쓰기가 맞는 어절을 틀렸다고 취급하는 오류 (제1형 오류)

(2) 시스템이 철자 및 띄어쓰기가 틀린 어절을 맞았다고 취급하는 오류 (제2형 오류)

등이 있다.

첫번째 오류는 어절 중 미등록어가 있는 경우에 발생될 수 있고, 두번째 오류는 사전에 정보가 잘못된 경우에 있을 수 있다. 시스템 오류를 모두 다 줄여야 하겠지만, 특히 두번째 오류를 줄이는 것이 실용적인 시스템을 만들기 위해 중요하다.

입력문의 오류로는 철자 오류와 띄어쓰기 오류가 있다. 철자 오류는 올바른 맞춤법을 알지 못하고 잘못 쓴 경우, 또는 타자가 잘못된 경우가 있다. 맞춤법을 바로 알지 못하고 쓴 경우는 일관되게 잘못 쓰는 일이 많다. 타자가 잘못된 경우는 일관되게 틀리지는 않지만, 많은 부분을 차지할 것으로 생각된다. 띄어쓰기 오류는 가장 많이 틀리는 것이다.[4] 한국어는 영어나 일본어와는 달리 복잡한 띄어쓰기 규칙을 가지고 있다. 띄어쓰기 오류의 종류를 다음과 같이 두 종류로 정의한다.

띄불 오류 : 띄어 써야 할 것을 붙여 쓰는 오류.

불띄 오류 : 붙여 써야 할 것을 띄어 쓰는 오류.

우선, 띄불 오류는 하나의 어절을 형성하기 때문에 그 어절의 분석시 곧바로 밝혀질 수 있다. 따라서 그 어절 내에서 교정을 볼 수 있다. 그러나 불띄 오류는 하나의 어절을 2개 이상의 어절로 띄어 쓰는 오류이다. 따라서 한 문장 안에 있는 모든 어절에 대해서 붙여 써야 하는지 검사해야 한다.

IV 기존의 시스템 및 관련 연구

한글 철자 및 띄어쓰기 검사기는 단지 하나의 어절에 대해 형태론적인 분석만을 한다. 따라서 이와 관련있는 기존의 형태소 분석기에 대해서 살펴 보자. 또한 기존의 영어 및 한글 철자 교정기에 대해서도 살펴 보자.

4.1 형태소 분석 시스템

기존의 한국어 형태소 해석기들은 주로 최장일치법이나 Head-Tail 구분법을 이용하여 형태소 해석을 하였다. 이 방법 외에 Tabular Parsing 방법이 있다.

1) 최장일치법(Longest Match Strategy)

한 어절 내에서 분절이 가능한 여러 형태소 중에서 가장 긴 형태소를 선택하는 방법이다.[1,3,10] 예를 들면 '감기는'에 대해 앞 부분에서 최장일치법을 이용한 경우 '감기(cold)+는'을, 뒷 부분에서 적용한 경우 '감(wind)+기는'을 추출한다.

2) Head-Tail 구분법

어절을 변형하지 않는 부분(Head)과 변형되는 부분(Tail)으로 구분하여 어절의 뒷 부분에서부터 가능한 모든 Tail을 찾아 가면서 Tail 테이블을 구성한다. 또한, 어절의 앞 부분에서부터는 Head를 찾아 내어 그 Head와 각 Tail들과의 접속 가능성을 Tail 테이블에 들어 있는 정보를 이용해서 검사한다. 이러한 방법으로 접속 가능한 Head를 계속 찾는 방법이다.[9]

3) *Tabular parsing* 방법

CYK(Cocke-Younger-Kasami) 알고리즘의 변형으로 각 열(raw)에 대해서 그 열의 첫 자소로 시작되는 문자열을 한 번의 탐색을 거치는 동안 모두 찾는다. 다음으로 T(i,k)와 T(i+k,j-k)가 접속 가능하면 T(i,j)를 만든다. 여기서 T(i,j)는 *i*번째 자소로부터 시작하여 *j*개의 자소로 구성된 형태소이다. 길이가 n인 어절일 경우 T(1,n)을 찾는 것으로서 동적 프로그래밍 기법을 이용한다.[1]

4.2 철자 교정 시스템

영어 철자 교정 시스템은 1962년 최초로 항공기 승객 이름의 오류를 수정하는 방법이 제시된 후, 많은 연구가 진행되었다.[12,13,14,15] 영어는 단어 단위로 띄어 쓰기 때문에 띄어쓰기 검사가 필요없다. 또한 한 단어의 글절이 심하지 않으므로 간단한 방법으로 어휘 분석(lexical analysis)을 할 수 있다. 영어에서는 철자 오류는 간단히 검사할 수 있으므로 대부분 철자 교정에 대한 연구를 많이 하였다.

한글 철자 검사 및 교정에 관한 연구는 최근에 시작되었고, 기존의 시스템으로는 한글 철자법 교정 시스템[2]이 있다. 이 시스템은 문서 편집 중에 철자를 교정하는 대화식 시스템으로서 COBOL 언어로 NEC/S100 상에서 구현되었다. 이 시스템은 입력 문서가 한글 맞춤법에 맞게 띄어 쓰여야 하는 등 많은 가정을 하고 있다. 조사/어미 테이블을 이용하여 어절의 뒤에서 조사/어미를 잘라내고 나머지를 사전에서 찾아 없으면 오류로 간주하여 사용자에게 교정 여부 및 사전 등록 여부를 묻는다. 교정된 어절은 교정 단어 사전에 등록하여 다시 오류가 발생할 때 자동으로 교정한다. 이 시스템은 한글의 특성을 충분히 반영하지 못하여 실용성이 없을 뿐아니라 batch 처리가 불가능하다.

V 시스템의 설계 및 구현

5.1 접속 정보표

한국어 어절은 앞에서 본 바와 같이 형태소들이 여러가지 유형으로 접속된다. 어절은 기본적으로 홀로 사용되는 자립어 하나와 홀로 사용되지 못하는 부속어들로 구성된다. 부속어들도 그 붙는 자립어의 종성의 종류에 따라 몇 가지로 나누어진다. 이와 같이 형태소들의 접속 유형을 파악하여 표로 구성한 것이 접속정보표이다.[10] 본 시스템은 기존의 형태소 해석기를 위한 좌우 접속정보표[7]를 이용하였다. 그러나 위 해석기는 불규칙 활용이나 음운 현상을 procedural하게 처리하므로 접속정보표에는 이러한 현상을 나타내지 않았다. 또한 분류된 범주가 세분되지 않아 모든 형태의 어절을 처리할 수 없는 단점이 있다. 본 시스템은 접속 정보표에 불규칙 활용을 처리할 수 있고, 모든 어절을 처리할 수 있도록 수정, 보완하였다.

5.2 사전

(1) 사전 항목

사전 내의 항목의 수가 많으면 많을수록 제1형 오류가 적어진다. 그러나 가능하면 사전의 크기를 줄이는 것이 중요하다. 사전을 최소로 줄일 수 있는 방법은 형태소를 표제어로 하는 것이다. 따라서 사전 등록어는 형태소를 기본 단위로 한다. 사전은 형태소와 그 좌우 접속정보의 쌍으로 구성된다. 사전의 구조는 일반적인 편집기(editor)로 편집이 용이하게 텍스트 화일로 되어 있으며, 한 줄에 하나의 항목과 그에 대한 정보를 가지도록 한다. 하나의 항목은 처음에 형태소가 나오고, 다음에 좌우 접속 정보의 쌍들로 구성된다. 사전을 공식적(formal)으로 기술하면 다음과 같다.

$M_1, CL_{11}, CR_{11}, \ldots, CL_{1n}, CR_{1n_1}$
$M_2, CL_{21}, CR_{21}, \ldots, CL_{2n}, CR_{2n_2}$
.....
$M_i, CL_{i1}, CR_{i1}, \ldots, CL_{in}, CR_{in_i}$
.....
$M_N, CL_{N1}, CR_{N1}, \ldots, CL_{Nn}, CR_{Nn_N}$

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 $1<=i<=N$에 대하여 $1<=n_i<=10$이다. 예를 들면, 형태소 '가장'에 대한 사전 항목은 다음과 같다.

가장 8 18 37 81 1 2

(2) 사전의 종류

사전의 종류는 크기와 사용도에 따라서 공통적인 형태소로 구성된 상용사전과 전문 용어 사전, 그리고 나머지 모든 가능한 사전로 구성될 수 있다[14]. 그러나 본 시스템에서는 마스터 사전과 사용자 정의 사전으로 구성하였다. 마스터 사전은 기본적인 형태소, 즉 조사, 연결 어미, 종결 어미, 전성 어미, 보조 어간, 접사, 관형사, 대명사, 수사, 부사, 용언 어간 등으로 구성하고, 사용자 정의 사전은 명사만으로 이루어지며 전문 용어 사전에 대응될 것이다. 사전의 한 항목은 접속 정보를 가져야

하기 때문에 사용자 정의 사전에는 접속 정보가 간단한 명사로 한정하고 접속 정보는 사용자가 넣지 않고 시스템이 만든다.

5.3 알고리즘

본 시스템에서는 접속정보를 이용한 최단일치법(Shortest Match Strategy)을 사용한다. 철자 검사기는 하나의 어절에 대해 단지 형태론적인 분석만을 하기 때문에 구문론적 또는 의미론적인 오류는 고려 대상이 아니다. 따라서 한 어절 내에서 의미론적으로 애매성이 있는 경우에, 형태소 해석기에서와 같이 모든 가능한 형태소 결합을 찾는 것이 아니라, 단지 하나의 가능한 형태소 결합이 발견되면 올바른 철자로 간주한다. 예를 들면, 어절 '감기는'에 대해 형태소 해석기는 '감기(cold)+는'과 '감(wind)+기는', '감(wind)+기+는'을 모두 찾지만, 본 시스템은 '감+기+는'만을 찾고 끝난다.

본 시스템에서 사용하는 자료 구조는 다음과 같다.

1) 인덱스 트리(index tree)

사전 검색을 용이하게 하기 위하여 사전을 적재하여 TRIE 구조를 만든다.

2) 스택(stack)

형태소 분석을 하는 과정에서 발견된 형태소를 저장하기 위해 스택을 사용한다. 스택에 들어 있는 인접한 형태소들은 서로 접속 가능함을 의미한다.

3) 접속 행렬(connection matrix):CM

행(raw)은 좌접속 번호를, 열(column)은 우접속 번호를 나타내는 테이블이다. CM[i][j]가 true면 좌접속 번호 i와 우접속 번호 j가 접속 가능함을 의미하고, false면 불가능함을 의미한다. 이 접속 행렬은 좌우 접속정보표(Left and Right Connectivity Information Table; LRCIT)를 적재하여 초기화된다.

본 시스템에 입력되는 한글 텍스트의 코드는 N 바이트 조합형으로 내부적으로 풀어쓰기를 한 형태이다. hspell의 개략적인 알고리즘은 다음과 같다.

```
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 3. hspell의 개략적인 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 (stand-alonable) 않다면, 현 어절 E의 분석을 위해 *Shortest_Match_Strategy*라는 프로시듀어(procedure)를 부른다. 이 때, 접속행렬 CM과 사전의 TRIE 구조를 이용한다.

본 시스템은 어절 단위로 처리함을 밝혔다. 따라서 어절이 있는 한 계속 처리를 한다. 하나의 어절을 읽어 이 어절이 홀로 쓰일 수 있는가를 검사한다. 만약 홀로 쓰일 수 있으면 이 어절은 오류가 없으므로 다음 어절을 읽어 처리한다. 만약 홀로 쓰일 수 없으면 형태소 분석을 통하여 오류의 유무를 가린다.

다음은 최단일치법에 대한 자세한 알고리즘이다.

```
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, RC는 각각 M의 좌접속,
            LC = {l1, ..., l10}, /* 우접속 번호의 집합이며, 그 번호
            RC = {r1, ..., r10}; /* 10개로 제한되어 있다고 가정 */
      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,Cj);
              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,Rj);   /* S's top morpheme */
            set p to i;
            retry to find another longer morpheme;
      endwhile;  /* while p <= n */
    end;
```

그림 4. 최단일치법에 대한 자세한 알고리즘

스택 S에는 최단일치법에 의하여 발견된 형태소 정보가 저장된다. 저장되는 정보의 상태는 (형태소 시작 자소번호, 가능한 접속번호의 쌍들, 형태소 종료 자소번호)로 된다. 즉, 위에서 *(p, C, j)*는 어절 E중 형태소 $E_p\ E_{p+1}\ \cdots\ E_j$를 뜻하며, C는 이 형태소의 우 접속번호와 스택의 top의 좌 접속정보의 접속가능 집합이다. 형태소들이 접속가능하다는 것은 스택 S의 top에 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 것이므로, 그 보다 긴 형태소를 찾아 다시 접속가능한 형태소를 찾는다.

VI 결 론

본 논문은 접속 정보를 이용한 한글 철자 및 띄어쓰기 검사기를 최단일치법을 사용하여 설계 및 구현하였다. 본 시스템은 SUN 3/60의 UNIX 4.2 BSD 상에서 C 언어로 구현되었다. 한글 입력 코드는 N 바이트 조합형이다. 본 시스템을 몇가지 측면에서 기존의 형태소 해석 시스템이나 철자 교정 시스템과 비교하면 표 1과 같다.

표 1. 기존의 시스템과의 비교

비교 시스템	최장일치법	H-T 구분법	Tabular Parsing	철자교정 시스템[2]	본 시스템
접근 방법	top-down	top-down	bottom-up	top-down	top-down
형태소추출	제한	제한	약간 제한	제한	약간 제한
어절의 분석	좌에서 우	좌우, 우좌	좌우	우좌	좌우
시간 복잡도	낮다	중간	높다	중간	낮다
기억 공간	적다	적다	크다	중간	크다
현상인식	예 / 아니오	아니오	예	아니오	예
사전의 종류	각 품사별	각 품사별	단일 사전	연관	마스터
	연관 "	연관 "		품사별	명사
사전의 크기	크다	크다	적다	크다	중간
정보일관성	어렵다	어렵다	쉽다	어렵다	쉽다
역순list, index	필요	필요	불필요	필요	불필요
미등록어처리	아니오	아니오	예	예	아니오
애매성 해소	아니오	약간	예	아니오	아니오

본 시스템은 접속 정보표의 정확성에 의존도가 높고, 사전의 구성이 힘들다는 단점이 있다. 그러나 불규칙 현상이나 음운 현상을 접속 정보표를 이용하여 일관된 처리를 할 수 있다. 또한 사전을 미리 메모리에 적재해야 하므로 기억장치가 커야 한다. 현재 사전은 약 5,500개의 항목으로 되어 있다. 앞으로의 과제로는 실용성있는 시스템을 만들기 위해 적절한 크기의 사전을 구축하는 것이다. 또한 철자 검사기를 지원하는 교정용 편집기를 만드는 것이다. 그리고 자동 철자 교정을 위한 연구도 앞으로의 과제이다.

참 고 문 헌

[1] 김성용, Tabular Parsing 방법과 접속 정보를 이용한 한국어 형태소 분석기, 한국과학기술원 석사학위 논문, 1987.

[2] 김영웅, 한글 철자법 교정 시스템, 한국과학기술원, 석사학위논문, 1984.

[3] 김효준, 박재득, 김길창, 이식성이 있는 한국어 형태소 해석기 개발 연구, 87 특정연구과제 결과 발표 논문집, 1988.

[4] 미승우, *새 맞춤법과 교정의 실제*, 어문각, 1988.

[5] 성균관 대학교 대동 문화 연구원, 고등학교 문법, 대한교과서 주식회사, 1988.

[6] 손우형, 한국어 기계번역을 위한 형태소 분석에 관한 연구, 한국과학기술원 석사학위 논문, 1986.

[7] 이종혁, 김성용, 좌우 접속 정보표, 한국과학기술원, CS Lab. Memo, 1987.

[8] 조규빈, 하이라이트 고교문법, 지학사, 1988.

[9] 최형석, 이주근, "자연어 어절 처리 알고리즘," 한국정보과학회 가을학술발표논문집, 11권, 2호, 1984.

[10] 한국과학기술원 전산학과 컴퓨터시스템 연구실, "한국어 처리 시스템 개발 환경의 연구," 과기처 보고서, 1988.

[11] Brian W.Kernighan & Dennis M.Ritche, The C Programming Language, Prentice-Hall, 1978.

[12] Charles R. Blair, "A program for correcting spelling errors," Information and control 3, pp. 66-67, 1960.

[13] Fred J. Damerau, "A Technique for computer Detection and Correction of Spelling Errors," CACM, Vol. 7, No. 3, March, pp. 171-176, 1964.

[14] James L. Peterson, "Computer Programs for Detecting and Correcting Spelling Errors," CACM, Vol. 23, No. 12, December, pp. 676-687, 1980.

[15] Leon Daridson, "Retrieval of Misspelled Names in an Airlines passenger System," CACM, Vol. 5, No. 3, March, pp. 169-171, 1962.

<<부 록>>

(hspell의 메뉴얼)

HSPELL(1) USER COMMANDS HSPELL(1)

NAME

hspell, addnoun, dedt - find Hangul spelling and word-spacing errors

SYNOPSIS

hspell [-l] [-c] [-w] [-u user-dict] [-lcwu user-dict] [files]

addnoun [file]

dedt

DESCRIPTION

Hspell collects Hangul words(eojeol) from the named documents, and looks them up in a Hangul spelling dictionary. First, eojeol is breaked into morphemes (it will be nouns, adjectives, prefixes or endings ect.) that is in a dictioanry. Then, it is checked whether all morphemes are connectable or not. Eojeols that are not connectable between it's all possibl morphemes are printed on the standard output. In this method, Hangul word-spacing errors are detectable. If no files are named, words are collected from the standard input.

Hspell ignores most troff(1), tbl(1), and

Master dictionary consist of all morphemes except nouns. Each morpheme has a line with it's connection information that is left, right number pairs. Connection information is at most 10 number pairs. Noun dictionary contains only nouns that hasn't connection information. Each noun has a line. Nouns are non-hada-noun (used only noun) or hada-noun (used both as noun and stem of verb) that has a hyphen (-) at the end of the morpheme.
User-defined dictionary must be a noun dictionary.

Two morpheme's connectablily checking is performed as follow first morpheme's right connection information i and second morpheme's left connection information j is checked whether CM[i,j] (connection matrix) set.
if it is set, two morphemes are connectable. otherwise it's not connectable.

CM is initialize from LRCIT (Left and Right Connection Information Table) before loading dictionary.

OPTIONS

-l Print all erroneus eojeols with line number.

-c Print all erroneus eojeols with correcting erroneus word-spacings.

-w Only check Hangul wellformedness.

-u Load the files as user-defined noun dictionary.

FILES

/usr/local/hspeller/Master.dict - Hangule master dictionary
/usr/local/hspeller/Noun.dict - Hangul noun dictionary
/usr/local/hspeller/LRCIT - connection information table
/usr/local/hspell

SEE ALSO

deroff(1), sort(1V), uniq(1)

BUGS

The morpheme's connection information in the master dictionary is lack and inaccurate. The spelling list's coverage is uneven.

Sun Release 3.5 Last change: 25 Augest 1989

(수행 예)

Script started on Tue Sep 19 22:17:27 1989

[SUN:DEMO 101] cat sample
국민 교육 헌장

우리는 민족 중흥의 역사적 사명을 띠고 이 땅에 태어났다.
조상의 빈난 얼을 오늘에 되살려, 안으로 자주 독립의
자세를 확립하고, 밖으로 인류 공영에 이바지할 때다.
이에 우리의 나아갈 바를 밝혀 교육의 지표로 삼는다.
성실한 마음과 튼튼한 몸으로, 학문과 기술을 배우고 익히 며,
타고난 저마다의 소질을 계발하고, 우리의 처지를 약진의
발판으로 삼아, 창조의 힘과 개척의 정신을 기른다.
공익과 질서를 앞세우며 능률과 실질을 숭사하고,
경애와 신의에 뿌리바근 상부 상조의 전통를 이어받아,
명랑하고 따뜻한 협동 정신을 북도둔다.
우리의 창의과 협ㄹ력을 바탕으로 나라가 밝전하며,
나라의 융성이 나의 발전의 근본임을 깨달아,
자유와 권리에 따르는 책임과 의무를 다하며,
스스로 국가 건설에 참여하고 봉사하는 국민 정신을 드높인다.
반공 민주 정신에 투철한 애국 애족이 우리의 삶의 길이며,
자유세계의 이상을 실현하는 기반이다.
길이 후손에 물려줄 영광된 통일 조국의 앞날을 내다보며,
신념과 긍지를 지닌 근면한 국민으로서,
민족의 슬기 를 모아 줄기찬 노력으로, 새역사를 창조하자.
[SUN:DEMO 1] hspell sample
빈난
발켜
익히
며,
숭사하고,
뿌리바근
전통를
북도둔다.
협ㄹ력을
밝전하며,
를
[SUN:DEMO 2] hspell -l sample
3: 빈난
5: 발켜
6: 익히
6: 며,
8: 숭사하고,
8: 뿌리바근
8: 전통를
9: 북도둔다.
9: 협ㄹ력을
9: 밝전하며,
14: 를
[SUN:DEMO 3] hspell -c sample
빈난
발켜
익히
며,
==> 익히며
숭사하고,
뿌리바근
전통를
북도둔다.
협ㄹ력을
밝전하며,
를
==> 슬기를
[SUN:DEMO 4] hspell -w sample
협ㄹ력을
[SUN:DEMO 5] hspell -lc sample
3: 빈난
5: 발켜
6: 익히
6: 며,
==> 익히며
8: 숭사하고,
8: 뿌리바근
8: 전통를
9: 북도둔다.
9: 협ㄹ력을
9: 밝전하며,
14: 를
==> 슬기를
[SUN:DEMO 6] exit
script done on Tue Sep 19 22:18:51 1989