한국어 수사어절의 유형 분류 및 정규화

강 승 식

한성대학교 정보전산학부
136-792 서울특별시 성북구 삼선동2가 389
sskang@hansung.ac.kr

Classification and Normalization of Korean Numerals

Kang, Seung-Shik
School of Information and Computer Engineering
Hansung University

요약

여러 가지 형태로 표현되는 수사어절을 아라비아 숫자로 구성된 표준형으로 변환하기 위하여 수사어절을 인식하는 알고리즘과 수사어절을 표준형으로 변환하는 수사어절 정규화 알고리즘을 제안한다. 띄어쓴 수사어절은 전처리 단계에서 수사어절 인식 알고리즘을 이용하여 한 어절로 결합한다.

1. 서론

일반 문서에서 수사어절은 출현빈도가 낮으므로 수사어절 자체를 형태소 분석의 단위로 하는 것이 일반적이다. 그런데 기계번역이나 한국어 분석 시스템에서 수사어절을 인식하거나 번역하려면 문서에 출현한 모든 유형의 수사어절에 대해 각각 처리해 주어야 한다. 이러한 문제는 한글문서에서 숫자와 수사 표현 방법의 다양성 때문에 발생하고 있다[1]. 수사어절은 여러 가지 형태로 표현되는데 그 예는 아래와 같다.

"230000원", "230,000원", "이십삼만원",
"이십 삼만원", "2십3만원", "2십 3만원" 등

특히, 전자거래 시스템에서는 상품 가격의 색인 및 검색이 매우 중요한 요소가 되며, 질의어에 의해 검색되는 용어는 가격 어절이 다수를 차지한다. 예를 들어, 질의문 "가격이 1백만원인 TV를 사고 싶다."에 의해 '1000000원'인 TV에 대한 정보를 검색한다. 이러한 가격 검색 기능이 가능하려면 가격 어절을 표준형으로 변환하는 기능이 필요하다. 또한, 일반적인 검색시스템에서 불용어로 간주되는 숫자 포함 어절을 색인어로 추출해야 한다.

수사어절 처리는 아라비아 숫자와 한글을 혼용하는 문제와 수사어절의 띄어쓰기 문제가 복합되어 있다. 즉, 아라비아 숫자로 표기되거나 한글로 표현된 것, 띄어쓰기를 한 것과 그렇지 않은 것을 모두 표준형으로 변환해야 한다. 수사어절의 처리는 임의의 어절이 수사어절인지 아닌지를 인식하는 문제, 수사어절을 정규화 하는 문제, 띄어쓴 수사어절을 한 어절로 결합하는 문제로 구분된다.

2. 수사어절의 유형

한글 문서에서 수사어절은 아라비아 숫자로만 표현된 경우, 한글로만 표현된 경우, 두 가지 유형의 문자가 혼합된 경우, 띄어쓰기 오류가 포함된 경우가 있다.

2350000원, 2,350,000원, 235만원, 2백35만원, 2백3십5만원, 이백삼십오만원, 2백 35만원, 2백 3십5만원, 2백3십 5만원, 2백 3십 5만원, 이백 삼십오만원, 이백삼십오만원, 이백 삼십 오만원

이처럼 다양한 수사어절들은 모두 아라비아 숫자로만 이루어진 대표형으로 변환되어야 한다. 숫자 변환의 편의성을 위해 수사어절의 유형을 어절 중간에 공백이 있는 것과 공백이 없는 것으로 구분한다. 형태소 분석기는 공백을 어절 분리 단위로 사용하므로 띄어쓴 수사어절을 하나로 결합해야 하기 때문이다.

<u>유형-A. 공백이 없는 유형</u>
① 23500000원
② 23,500,000원
③ 2350만원
④ 2,350만원
⑤ 2천3백50만원
⑥ 2천3백5십만원
⑦ 이천삼백오십만원

<u>유형-B. 공백이 삽입된 유형</u>
① 2천 3백 5십만원
② 2천 3백 50만원
③ 2천3백 5십만원
④ 2천5백3십7억 2천3백5십만원
⑤ 이천삼백 오십칠만원
⑥ 이천 삼백 오십 칠만원

자주 사용되는 유형-A와 유형-B를 우선적으로 처리한다. 유형-C는 숫자 표기 오류이므로 유형-A와 유

형-B에서 자동으로 처리될 수 있는 유형들에 대해서만 아라비아 숫자로 변환한다. 유형-A의 '23,000,000원'과 '2,350만원'은 ','를 삭제하는 전처리가 필요하다. 어절 중간에 삽입된 구분자 ','는 천 단위인 경우에만 삭제하고 그 위치가 잘못된 것은 오류어로 간주한다.

유형-C. 숫자-한글 혼용 오류 유형

① 2백 삼십오만원
② 2백 3십오만원
③ 2백삼십 5만원
④ 2백 삼십 5만원
⑤ 이백 삼십5만원
⑥ 이백3십오 만원
⑦ 이백 3십 5만원

유형-A와 유형-B는 모두 한글과 아라비아 숫자가 혼용된 경우가 존재하므로 숫자를 인식하는 알고리즘을 공유하는 방법과 독립적인 처리 방법이 모두 가능하다. 다만, 처리 방법을 단순화하고 정확한 분석을 위하여 유형-B의 경우에는 입력문장의 전처리 과정에서 띄어쓴 어절들을 한 어절로 결합하는 방식을 취한다.

3. 수사어절 인식 알고리즘

유형-A의 수사어절을 아라비아 숫자 스트링으로 변환하는 알고리즘은 입력 어절 어절인지를 판단하는 알고리즘과 수사어절을 아라비아 숫자 스트링으로 변환하는 알고리즘으로 구현된다. 입력 어절이 수사어절인지, 아닌지를 판단하기 위해서 그림 1과 같은 변수들을 이용한다.

```
int s1=1000;  /* '천/백/십' 기다림 */
int s2=1000;  /* '조/억/만' 기다림 */
int aflag=0;  /* '1-9' 발견? */
int hflag=0;  /* '일-구' 발견? */
int acount=0; /* '1-9'의 개수 */
int hcount=0; /* '일-구'의 개수 */
```

그림 1. 수사어절 판단을 위한 변수 정의

변수 s1은 다음 음절로서 '천/백/십'이 출현한 순서를 제어하기 위한 것이다. 즉, '2천3백5십'은 옳지만, '3백2천5십'은 오류 어절이므로 이 순서를 제어하기 위한 용도로 사용된다. 초기값 1000은 '삼억오천육백칠십팔만', '삼억육백칠십팔만', '삼억칠십팔만', '삼억팔만'과 같이 '천', '백', '십' 등 1000보다 작은 음절 중 하나가 올 수 있음을 의미한다. '천' 단위가 처리되면 s1=100으로 값을 바꿔 주고, 이것은 '백/십'이 올 수 있음을 뜻한다. 만일, s1=1000인 경우에 '천' 단위가 없이 '백' 단위가 오면 s1=10으로 값을 바꿔준다.

변수 s2의 용도는 s1과 유사하며 '조/억/만'이 출현한 순서를 제어하기 위한 것이다. 즉, '조/억/만'의 순서는 옳지만 '만/조/억' 순서는 틀린 어절이므로 이 순서를 제어하기 위한 용도로 사용된다. 초기값 1000의 의미는 '조/억/만' 중 아무거나 올 수 있음을 의미하고, 100은 '억/만' 중 하나가 올 수 있음을 의미한다.

변수 aflag와 hflag은 각각 '1'~'9', '일'~'구'에 대하여 이미 숫자 하나가 발견되었는지, 그렇지 않은지를 표시한다. 이 변수는 '육칠백'이나 '67백'과 같이 숫자 단위어 앞에 1개의 숫자만 오도록 제한함으로써 이 제약을 위반하면 오류어로 간주하기 위한 것이다.

acount와 hcount는 각각 입력 어절에 출현한 '1'~'9', '일'~'구'까지의 숫자 개수를 저장한다. 이 변수들은 정확하게 수사어절로 판단되지는 않았지만, 수사어절일 가능성이 높은 어절을 인식하기 위한 것이다.

```
Algorithm is_num_string(char *word)
{
   while (*word) {
     if (*word == '천/백/십')
        변수 s1의 값을 조절;
     else if (*word == '조/억/만')
        변수 s2의 값을 조절;
     else if (*word == '1-9' or '일-구')
        변수 aflag, hflag의 값을 조절;
     else if (*word == '원/달러/엔/...')
        return 2; /* 가격 명사 */
     else if (*word == '\0')
        return 1; /* 아라비아 숫자 */
     else {
        if (숫자어절 추정) return 3;
        else return 0; /* not 숫자어절 */
     }
     word++;      /* 다음 음절 또는 숫자 */
   }
}
```

그림 2. 수사어절 인식 알고리즘

그림 2는 입력 어절이 수사어절인지, 아닌지를 판단하는 알고리즘으로 반환값은 '순수한 수사어절', '가격에 관한 수사가 발견된 어절', '수사어절로 추정되는 어절', '수사어절이 아닌 어절' 중 하나이다. 알고리즘에서 인자 'word'는 한글인 경우에 2바이트씩 음절 단위로, 아스키 문자는 1바이트씩 증가시킨다.

수사어절을 인식할 때 아래 어절들은 일반적인 명사로 사용되고, 수사로 사용되는 경우는 드물기 때문에 예외로 간주한다.

백사, 천사, 만사, 조사, 영원,
일원, 이원, 삼원, 사원, 구원

또한, 아래 어휘 중에서 '원'과 같이 가격 연산자가 없는 것은 수사어절이 아닌 것으로 간주한다.

일조, 이조, 사조, 육조, 구조, 이만, 오만, 팔만

4. 수사어절 정규화 알고리즘

아라비아 숫자와 수사가 혼용된 문자열을 아라비아 숫자 문자열로 변환하는 함수 num_conv()를 구현하는 위해서 숫자를 한글로 표기하는 규칙을 이용한다. 즉, 영어에서는 10^3 단위로 끊어서 표기하지만 한글에서는 10^4 단위로 끊어서 표기하고 있다.

예1. 3천4백5십6억3천4백5십6만3천4백5십6
예2. 삼천사백오십육억삼천사백오십육만삼천사백오십육

예1과 예2는 '만'과 '억'을 기준으로 '천/백/십' 단위

수사가 나타난다. 한글과 숫자가 혼합된 유형은 먼저 104 단위인 '조/억/만'을 기준점으로 '천/백/십'으로 구성된 10단위 문자열이 반복되고 있다. 따라서 10단위 문자열을 숫자로 변환한 후에 각각 '만'과 '억' 단위를 곱해 주는 방법으로 처리할 수 있다.

```
Algorithm numeral2number(char *str)
{
  char *n1=str, *n2;

  while (*n1) {
    if (*n1 == '1/2/.../9')
        *n2++ = *n1++;
    else if (*n1 == '일/이/.../구')
        *n2++ = conv2arabic(*n1++);
    else if (is_10unit(*n1))
        *n2++ = *n1++;
        /* 십/백/천/만/억/조 */
    else if (*n1 == ',')
        ;
    else break;  /* error */
   }
   conv_numeral_unit(n2, str);
}
```

그림 3. 수사어절 변환 알고리즘-1

수서어절 변환 알고리즘은 그림 3과 같이 기술된다. 이 알고리즘은 두 단계 과정으로 구성된다. 첫 번째 단계에서는 '일/이/.../구'를 '1/2/.../9'로 변환하는 단순한 작업이다. 즉, 예2의 입력 어절에서 '일/이/.../구' 등 각 숫자들을 예1과 같이 아라비아 숫자로 변환한다.

입력 어절 n1에 대해 첫 번째 단계에서 모든 '일/이/.../구'를 '1/2/.../9'로 변환 결과인 n2는 예1과 같이 아라비아 숫자와 10단위 연산자 '십/백/천/만/억/조'로 구성된 스트링이다.

conv_numeral_unit()은 n2를 '345634563456'과 같이 변환하는 함수이며 그림 4와 같이 기술된다. 이 함수에서 숫자를 변환하는 과정은 다음과 같다. 입력 어절을 '조/억/만' 단위로 분할하여 처리한다. 변환된 결과는 네 문자씩 "0000"으로 초기화하여 각 위치에 '0' 대신 해당 10진수 문자로 대치한다. 알고리즘의 변수 pb와 pe는 '천/백/십/일'의 시작 및 끝 위치이다.

변수 man1과 man2는 '조/억/만' 단위를 구분한 결과를 저장하기 위한 용도로 사용된다. man1은 '조/억/만' 중에서 처음 출현한 단위에 대한 값이며 3보다 큰 수로 초기화한다. man2는 두 번째 출현한 단위에 대한 값으로서 0, 1(만 단위), 2(억 단위), 3(조 단위)을 가질 수 있다.

'억' 단위 없이 '조' 뒤에 '만'이 오는 것은 man1=1, man2=3이므로 이 경우에는 '억' 단위에 대해 "0000"을 추가해 준다. man1의 값이 man2보다 큰 경우는 '조...억...만'의 순서가 틀린 것이므로 오류이다.

변수 pb와 pe에 의해 발견된 '천/백/십/일' 문자열은 set_1000_unit()에 의하여 "0000" 위치에 해당 10진수 문자로 대치된다. 함수 set_1000_unit()은 10단위 수사 '천/백/십'을 인식하여 "0000"의 해당 위치의 '0'을 10단위 수사 앞에 오는 아라비아 숫자로 대치한다. 마지막으로 remove_leading_0s()에 의해 "0012"와 같이 '0'으로 시작되는 것은 "12"로 변환한다.

```
Algorithm conv_numeral_unit(char *in, *out)
{
  int i=0, man1=10, man2;
  char *pb=in, pe;

  while (*pb && (pe=get_MAN_unit(pb))) {
    man2 = is_numeral_10000x(*pe);
    if (man1-man2 == 2) { /* no 억단위 */
     out[i] = out[i+1] = '0';
     out[i+2] = out[i+3] = '0';
       i = i + 4;
     } else if (man1 <= man2)
       return; /* '조/억/만' 순서 오류 */

     man1 = man2;
     out[i] = out[i+1] = '0';
     out[i+2] = out[i+3] = '0';
     set_1000_unit(pb, pe, out, i);

     i = i + 4;
     pb = pe+1;
   }

   out[i+4] = '\0';
   remove_leading_0s(out);
}
```

그림 4. 수사어절 변환 알고리즘-2

5. 띄어쓴 수사어절의 결합

'2백 3십 5'와 같이 입력 문장에서 두 어절 이상으로 띄어쓴 수사어절은 전처리 과정에서 한 어절로 결합한다. 띄어쓴 어절들을 한 어절로 결합할 때는 앞뒤 어절이 수사어절인지를 검사하여 두 어절을 하나로 결합했을 때 역시 수사어절인 경우로 제한한다. 각 어절이 수사어절인지, 아닌지를 판단하는 수사어절 인식 알고리즘을 이용한다. 띄어쓴 수사어절을 직접 아라비아 숫자 문자열로 직접 변환하는 방법도 있으나, 수사어절 인식 알고리즘을 그대로 이용하는 방법을 취한다.

6. 결론

수사어절을 아라비아 숫자로만 표기된 경우, 한글로 표기된 경우, 아라비아 숫자와 한글이 혼용된 경우 등 유형별로 분류하였다. 한글로만 표현한 것, 한글과 아라비아 숫자가 혼합된 것 등 각 유형들에 대해 아라비아 숫자로 변환하는 알고리즘을 제안하였다. 한글과 숫자가 혼용된 수사어절을 띄어쓴 경우에는 전처리 과정에서 이를 인식하여 한 어절로 결합한다.

알고리즘을 구현하여 수사어절의 인식 및 정규화 알고리즘이 작동됨을 확인하였다. 이 과정에서 일반 명사를 아라비아로 숫자로 변환하는 오류를 발견하고 일반 명사로 사용되는 것들은 변환되지 않도록 예외처리 기능을 추가하였다.

참고문헌

[1] 김민정, 권혁철, "한국어 형태소 분석에서의 수사 처리", 제3회 한글 및 한국어 정보처리 학술발표 논문집, pp.178-187, 1991.

[2] 서정수, 현대 국어문법론, 한양대학교 출판원, 1996.