메타문자를 사용한 한국어 사전 탐색 앱

권홍석○, 김재훈
한국해양대학교 컴퓨터공학과
hong8c@naver.com, jhoon@hhu.ac.kr

Korean Word Search App Using Meta-characters

Hong-Seok Kwon○, Jae-Hoon Kim
Department of Computer Engineering, Korea Maritime University

요 약

스마트 폰의 보급이 대중화됨에 따라 다양한 앱들이 사용되고 있으나 효율적인 사전 탐색에 관한 앱은 그다지 많지 않다. 현재 공개된 한국어 사전 탐색 앱은 완전한 단어이거나 단어의 부분 문자열을 질의로 사용한다. 이 경우 완전한 단어를 기억하지 못하거나 한국어 정보처리를 위한 여러 형태의 음운 정보를 쉽게 탐색할 수 없다. 이러한 문제를 개선하기 위해 본 논문에서는 메타문자를 사용하여 효율적으로 단어를 탐색할 수 있는 앱을 개발한다. 본 논문에서 사용하는 메타문자는 임의의 음절을 표현하는 '*'와 '?'과 종성을 표현하는 ':'를 사용하며 사전구조는 자소 단위의 트라이를 사용한다. 또한 음절은 물론이고 자소(초성, 중성, 종성)로 구성된 질의를 탐색할 수 있다. 더구나 음절과 자소가 혼합된 질의도 사용할 수 있도록 하여 사용자의 편의를 크게 도모하였다.

주제어: 메타문자, 트라이, 단어 검색, 스마트 폰

1. 서론

무선이동통신의 발전과 함께 스마트폰의 보급이 널리 확산되고 기존의 전자사전의 기능들을 스마트폰이 수행하면서 사전 앱(app, application)의 사용이 점차 빈번해지고 있다. 현재 사용되는 대부분의 한국어 사전 앱은 질의에 있어서 찾고자 하는 단어 또는 그 단어의 부분 문자열을 질의로 사용한다. 본 논문에서는 기존의 사전 검색에서 더 나아가 메타문자(meta-characters)를 사용하여 고급검색이 가능하도록 하였다. <표 1>은 본 한국어 사전 탐색 앱에서 사용할 수 있는 질의의 예를 보여준다.

<표 1> 제안된 한국어 사전 탐색 앱에서 사용할 수 있는 질의의 예

번호	질의	탐색된 단어의 예
1	가*	가위, 가지, 가시나무, 가늠쇠...
2	*위	가위, 지위, 한가위...
3	부*고	부산고, 부산진고, 부산여자고...
4	가?	가위, 가지, 가슴...
5	가??	가늠쇠, 가랑이, 가오리...
6	?위	가위, 지위, 하위...
7	??위	한가위, 아래위, 공수위...
8	부?고	부산고, 부천고, 부성고...
9	부??고	부산진고, 부산여고...
10	ㄱㅇ	가위, 거위, 고의...
11	:ㄴ자	완자, 관자, 환자, 친자...
12	고ㅏ:ㄴ	고까신, 고각단, 고차원...
13	ㅏ:ㄴㅓ	가면적, 단순성, 악관절...

<표 1>에서 볼 수 있듯이 임의문자로 '*'과 '?'를 사용하며 '*'는 모든 부분문자열을 의미하고, '?'는 한 음절을 의미한다. 또한 초성과 종성을 구별하기 위해 임의문자 ':'도 사용한다. 본 논문에서 제안된 앱에서는 메타문자의 종류 및 위치에 따라 크게 3가지의 유형으로 나누어 처리되며 자세한 설명은 2.3절에서 자세하게 기술한다.

본 논문은 다음과 같이 구성된다. 2장에서는 메타문자를 사용한 한국어 사전 탐색 앱을 소개하고, 3장에서는 기존 사전 검색 시스템과의 비교해 본다. 마지막 4장에서는 본 연구의 결론 및 향후 과제에 대해 설명하고 끝을 맺는다.

2. 메타문자를 사용한 한국어 사전 탐색 시스템

2.1 전체 시스템 구성

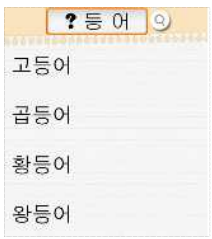
메타문자를 사용한 한국어 사전 탐색 시스템의 전체 흐름은 (그림 1)과 같다. 질의문을 입력받으면 그 질의문이 유효한지 여부를 조사한다. 유효하다면 질의문의 탐색유형을 결정한다. 여기서 말한 탐색유형은 2.3절에서 설명할 3가지 유형 중 하나를 의미한다. 그리고 질의문의 내용을 자소 단위로 분해하고, 그 결과를 이용해 트라이 탐색구조로 구축된 사전에서 적합한 단어들을 추출한다. 추출되어진 단어들은 자소 단위로 분해되어 있으므로 다시 재조립하는 과정을 거친 후, 조립된 단어의 음절의 수가 적은 순에서 많은 순으로 정렬되어 그 목록이 사용자에게 보여지게 된다. 사용자가 목록 중에서 최종 찾고자 하는 단어를 선택하게 되면 이벤트를 후킹하여 해당 단어의 뜻이 보여진 다음 모바일 국어사전 페이지[1]를 보여주게 된다.

1) http://m.krdic.naver.com/

(그림 1) 전체 시스템 구성도

2.2 인터페이스 설계

(그림 2)는 본 논문에서 제안된 앱의 인터페이스이며 기존의 사전 앱과의 큰 차이 없게 배치하여 사용자들로 하여금 낯설지 않게 하였다. 질의문 작성은 (그림 2)에서 질의문 작성 박스에 기입하여 작성할 수 있으며 본 앱에서 사용할 수 없는 질의문을 작성 시 오류 메시지를 보여주도록 하였다. 검색되어져 보인 결과는 (그림 2)에서 질의문 작성 박스 아래의 리스트 뷰를 통해서 보여지는데 조건에 만족하는 단어들 중에서 음절의 수가 적은 순에서 많은 순으로 정렬되어 보인다. 검색되어진 단어들 중에서 사용자가 원하는 단어를 터치하면 다음 모바일 웹 페이지에 연결되어 검색된 결과가 나타나는 방식으로 설계하였다.

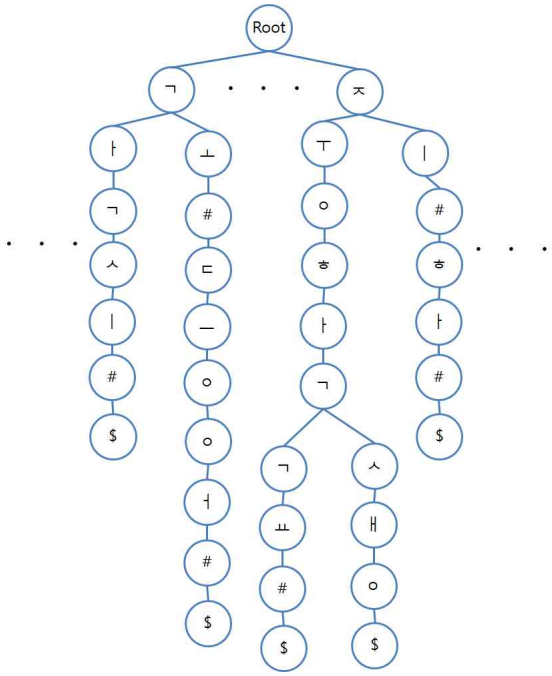


(그림 2) 한국어 사전 탐색 시스템의 구성

2.3 트라이를 이용한 사전구축 및 탐색

본 논문에서 제안된 시스템은 자소단위의 트라이 탐색 구조를 이용했다. 즉 저장 및 검색의 기본 단위가 자소이며, 하나의 음절을 저장하기 위해서는 3개의 자소로 분리하여 분리된 각각의 자소를 하나의 노드에 저장하는 방법이다[1]. 자소단위 트라이 구조의 장점으로는 사전 검색시 어절의 첫 음소부터 차례대로 읽어 그에 해당하는 트라이 노드들을 음소의 수만큼만 링크를 따라가며 노드를 검색하므로 검색 속도가 상당히 빠르다[2]. 단점으로는 하나의 음절을 3개의 자소 단위인 초성, 중성, 종성으로 분리하여 저장해야 하므로 트라이를 구성하는데 전체 노드수가 최대 3배 가까이 증가해 음소별 트라이 구조에 비해서 주 기억 장치의 사용 효율이 저하되며 낭비가 생긴다[3].

본 논문에서 효율적인 탐색을 위해서 전방 색인 사전과 후방 색인 사전을 사용하며, (그림 3)은 전방 색인 사전의 일부이다.



(그림 3) 전방 색인 사전

(그림 3)에서 볼 수 있듯이 단어를 자소 단위로 분해하여 트라이 구조에 넣되 모든 음절은 초성, 중성, 종성이 있다고 가정하고 종성이 없는 음절의 경우에는 그 노드에는 문자 'X'을 삽입했다. 그리고 단어의 끝을 알리기 위해 마지막에는 '$'를 삽입했다. 그리고 하나의 음절에 균일하게 3개의 노드를 할당함으로써 기능 구현에 있어 편의를 취하는 반면 기억공간의 손실이 발생했다. 후방 색인 사전의 경우 자소 단위로 분해된 단어들을 역순으로 변환한 후 트라이 구조에 삽입했으며 전방 색인 사전과의 기본 법칙은 동일하다.

탐색방법은 세 가지의 유형으로 나눈다. 첫 번째 유형으로는 메타문자 '*'을 질의문에 사용하는 경우로 <표 1>에서의 번호 1, 2, 3이 이에 해당된다.

- '*'가 질의문의 뒤에 오는 경우 탐색하기 위한 사전으로는 전방 색인 사전을 사용한다. 예를 들어 "사*"였다면 먼저 질의문 "사*"에 대하여 메타문자를 제외한 완성형한글에 대해 자소단위로 분해한다. 자소분해시 종성이 없는 경우에는 문자'X'를 삽입했다. 분해된 'ㅅ'+'ㅏ'+'X'를 루트에서부터 차례대로 탐색하여 해당하는 트라이 노드를 기억하고 해당 노드의 아래 구간트리를 순차적으로 순회하고 저장하면서 단어의 끝을 알리는 '$'만나면 하나의 단어로 인식하고 이를 반환한다. 이 과정은 구간트리를 모두 순회할 때까지 계속된다.
- '*'가 질의문의 앞에 오는 경우 탐색을 위한 사전으로 후방 색인 사전을 사용하며 탐색 방법은 전방 색인 사전을 사용한 방법과 동일하다.
- '*'가 질의 문자열 중간에 삽입되는 경우 탐색을 위해 전방 색인 사전과 후방 색인 사전 모두를 사용한다. 예를

들어 질의문이 '부*고'였을 때 기호'*'을 기준으로 질의문을 '부*' 그리고 '*고'로 나눈 뒤 나눠진 각각에 대해 앞서 설명했던 탐색방법들을 적용하고 그 결과들을 교집합 취하여 최종 질의문에 적합한 단어들을 찾아낸다.

두 번째 유형으로는 메타문자 '?'을 질의문에 사용하는 경우로 <표 1>에서의 번호 4, 5, 6, 7이 이에 해당된다. 본 논문에서는 두 번째 유형과 세 번째 유형을 처리하기 위해 질의문에서 완성형한글은 자소분해하고 나머지 자모들은 다음과 같은 규칙을 사용하여 정규화하였다.

S → 초 | 중 | 종 | ? | λ

C → ㄱ | ㄲ | ㄴ | ㄷ | ㄸ | ㄹ | ㅁ | ㅂ | ㅃ | ㅅ | ㅆ | ㅇ | ㅈ | ㅉ | ㅊ | ㅋ | ㅌ | ㅍ | ㅎ

J → ㅏ | ㅐ | ㅑ | ㅒ | ㅓ | ㅔ | ㅕ | ㅖ | ㅗ | ㅙ | ㅚ | ㅛ | ㅜ | ㅝ | ㅞ | ㅟ | ㅠ | ㅡ | ㅢ | ㅣ

Z → ㄲ | ㄳ | ㄵ | ㄶ | ㄺ | ㄻ | ㄼ | ㄽ | ㄾ | ㄿ | ㅀ | ㅄ | ㅆ | ㅇ | ㅈ | ㅊ | ㅋ | ㅌ | ㅍ | ㅎ

초 → COO초 | COO중 | COO종 | λ

중 → OJO초 | OJO중 | OJO종 | λ

종 → OOZ초 | OOZ중 | OOZ종 | λ

? → OOO

O → C | J | Z

위의 규칙에서 'O'는 사전탐색 시 어떤 하나의 트라이 노드와 매치가 가능하다.

- '?'가 질의문 뒤에 오는 경우 탐색을 위한 사전으로는 전방 색인 사전을 사용한다. 예를 들어 질의문이 '사ㄱ?'라고 가정하면 위의 규칙을 적용하면 'ㅅ'+'ㅏ'+'X'+'ㄱ'+'O'+'O'+'O'+ 'O'+'O'로 분해되고 (그림 4) 탐색 알고리즘을 통해 질의문에 적합한 단어를 찾아낸다. 지금부터 탐색 알고리즘에 대해서 설명한다.

탐색 알고리즘은 기본적으로 질의문의 길이만큼 반복 호출되는 구조이다. 루트에서부터 질의문의 첫 인덱스를 시작으로 탐색해나간다. 탐색 중에 문자 'O'를 만나면 그 상위노드의 바로 아래 자식들을 모두 순회해야 하므로 해당 상위노드를 시작으로 반복 호출하여 탐색된다. 최종 적합한 단어의 선택은 반복 호출되는 순간마다 질의문의 길이가 1씩 감소함에 따라 질의문의 길이가 0이 되고 최고 말단 노드가 단어의 끝을 알리는 '$'을 만나면 최종 결과 리스트에 더해진다.

마지막 유형으로는 특수 문자 기호 '*' 그리고 '?'를 사용하지 않고 초성, 중성, 종성으로만 구성된 질의문의 경우이다. 탐색을 위한 사전으로는 기본적으로 전방 색인 사전을 사용하여 탐색하며 만약 질의문의 첫 문자가 'O'가 아닌 경우 전방 색인 사전을 사용하고 질의문의 마지막 문자가 'O'인 경우 후방 색인 사전을 사용한다. 탐색방법은 두 번째 유형과 동일한 탐색방법을 사용하여 적합한 단어를 찾아낸다.

```
String query;     // 자소분해된 질의문
int pos = 0;      // 질의문의 인덱스
String res;       // 매칭된 단어
List<String> list// 적합한 단어들의 리스트
int leng;         // 질의문의 길이
int m;            // 탐색유형
root.matching(query, pos, res, list, leng, m) {
   if (m == 3 and leng == 0) or (pos == leng-1 ))
      list.add;
   if (c ==’$’ and leng == 0)
      list.add;
   if (s == null or child == null or pos >=s.leng-1)
      return;
   TN n = child.get(s[pos]);
   if (n == null) return;
   if(pos == s.leng-1)
       n.matching(s, pos+1, res+s[pos], list, leng-1, m);
   else
       n.matching(s, pos+1,   res+s[pos+1], list, leng-1, m);
   if(s[pos] == 'O') {
       set<char> keyset = child.keySet();
       if (n == null) return;
       if(pos == s.leng-1) {
          n.matching(s, pos+1, res+s[pos],
                     list, leng-1, m);
       else
          n.matching(s, pos+1, res+s[pos+1],
                     list, leng-1, m);
}
```

(그림 4) 탐색 알고리즘

3. 기존 사전 검색 시스템과의 비교

본 논문에서는 자소 단위의 트라이 사전을 구축함으로서 기존 사전과는 다르게 특정 종성으로 시작하는 단어 그리고 자음, 모음의 조합으로도 검색이 가능하다. 이와 같은 검색방법은 자연언어처리 분야에 도움이 될 것이라고 기대한다. 그러나 자소 단위의 트라이 구조 특성상 기호 '*', '?'들이 임의의 음절들 앞, 뒤에 동시에 쓰이거나 음절들 사이사이 2개 이상 존재할 경우 예를 들어 "*지*", "?고?", "한*양*교" 등의 검색은 불가능하다는 단점이 있다.

4. 결론

본 논문에서는 트라이 탐색구조 기반의 사전탐색에서 임의 문자 기호를 사용하여 다양한 조건의 검색이 가능한 시스템을 제안하였다.

우리는 임의 문자 기호 및 초성 검색을 위해 총 2개의 사전을 구축하여 해결했다. 그러나 경우에 따라 사전을 두 번 탐색해야 하는 문제점과 색인 구축 시 큰 용량의 주기억 메모리를 필요로 하는 문제점들이 발생하였다. 이러한 문제들을 해결하기 위해 색인 구축 시 압축의 기술이 필요하다[4].

제안시스템은 향후 트라이를 개선하여 색인 구축 시 발생하는 비용을 줄여볼 계획이다.

참고문헌

[1] 길철수, “이중 배열 트라이 구조를 이용한 한국어 전자 사전의 구축”, 정보과학회논문지, 제23권, 제1호, pp. 85-94, 1996.

[2] 이승선, “Compact TRIE Index(CompTI) : 한국어 전자 사전을 위한 데이터베이스 색인 구조”, 한국정보과학회논문지, 제22권, 제1호, pp. 3-12, 1995.

[3] 이승선, “TRIE 구조를 이용한 한국어 전자 사전을 위한 데이터베이스 인덱스 구조”, 한국정보과학회논문집, 제21권, 제1호, pp. 849-852, 1994.

[4] C. D. Manning, P. Raghavan, and H. Shütze, Introduction to Information Retrieval, Cambridge University Press, 2010.