DER 테이블을 이용한 고속 키워드 탐색

정규철, 장혜숙, 이진관, 박기홍
군산대학교 컴퓨터정보과학과
e-mail: kcjung@kunsan.ac.kr

A Study about Fast Keyword Retrieval using AC DER table

Kyu-Cheol Jung, Hae-Suk Jang, Jin Kwan Lee, Kihong Park
Dept of Computer Science, Kunsan National University

요 약

효율적인 키워드 추출이 정보검색 시스템에서 매우 중요한 일임에도 불구하고 원하는 목적의 적당한 키워드를 결정하는 것은 매우 어렵다. 왜냐하면 많은 복합어를 가지고 있기 때문이다. 기존 방법에서는 AC 머신의 경우 단일 키워드를 가지고 복합 키워드를 검색하지 못한다. 이러한 문제를 해결한 DER 구조의 경우에는 많은 검색시간이 걸리는 문제점을 가지고 있다. 따라서 본 논문에서는 이러한 문제점들을 해결하기 위해 이들을 기반으로 한 DERtable (DER 구조의 검색방법을 가지고 테이블로 구성)구조를 제안한다.

Ⅰ. 서 론

효율적인 키워드 추출은 문서 관리 시스템, 인터넷 정보검색에서 매우 중요한 일이다. 그러므로 많은 연구들이 이루어지고 있고, 여러 분야에 적용되고 있다. 키워드 추출 시스템에서 많은 복합어들은 자유롭게 키워드 후보들을 만들어낸다[1]. 이러한 키워드 추출에서 키워드 구조는 두 가지 형태를 가진다. 하나는 개별적인 단어의 선택이고 다른 하나는 개별 단어의 순서를 선택하는 것이다[1]. 개별적인 단어의 키워드 구성은 "단일 키워드(Single Component keyword)"라 부르고, 단어순서의 키워드 구성을 "복합 키워드(Long Component keyword)"라고 부른다[1].

단일 키워드 처리에서 복합어의 짧은 길이는 우선순위가 주어지고 추출 매칭에 의해 그들을 결정하기 쉽다. 그러나 그들은 복합 키워드에 의해 정확한 의미를 잃어버린다. 예를 들면, 단일 키워드 "자연"은 복합 키워드 "자연언어처리"의 의미를 표현 할 수 없다. 이에 비해 복합 키워드 "자연언어처리"는 단일 키워드 "자연"을 원소로 포함하지만, 단일 키워드 검색시 복합 키워드를 추출 매칭에서 검색할 수 없다[2].

AC(Aho-Corasik) 머신에서의 키워드 검색은 키워드 "자연"을 검색했을 때, "자연"만을 검색한다. 그리고 키워드 "자연언어처리"를 검색했을 때에만 "자언언어처리"와 "자연"의 검색이 가능하다. 즉, AC 머신에서는 "자연"을 가지고, "자연언어처리"를 검색할 수 없다.

DER(Delayed Extraction and Retrieval) 구조에서는 "자연"을 검색할 때, "자연언어처리"와 "자연"의 검색이 가능하다. 즉 단일 키워드를 검색할 때, 복합 키워드까지 검색이 가능하다[2]. 그러나 이러한 검색을 위해서는 reverse 정보를 이용하여, 각 상태들을 추적하면서, 검색을 해야 하므로 많은 검색시간이 걸리는 문제점이 있다.

본 논문에서는 위와 같은 문제점을 없애기 위해 테이블을 이용한 DER 구조를 제안한다. 그리고 테이블을 이용한 DER 구조를 DERtable 구조로 표현한다. 본 논문의 2장에서는 AC 머신에 대해 설명한다. 3장은 DER 구조에 대해 설명한다. 4장은 DERtable 구조에 대해 설명한다. 5장에서는 실험 및 평가를 한다. 그리고 마지막으로 6장에서 결론을 맺는다.

2. AC 머신

2.1 AC 머신

AC 머신은 단일 패스에서 텍스트 스트링에 포함된 모든 키워드의 위치를 결정할 수 있는 머신이다[2]~[5].

AC 머신은 텍스트 스트링을 입력으로 하고, 출력으로 텍스트 스트링에서 서브스트링에 해당되는 키워드(K_SET의 원소)의 위치를 찾아내는 프로그램이다. 입력 심벌의 집합 I에서 AC 머신의 동작을 다음 3개의 함수로 정의할 수 가 있다[2][5].

goto function g : S × I → S ∪ {fail},

failure function f : S → S,

output function Output : S → A,

K_SET의 부분집합.

K_SET은 AC 머신에 의해 검색될 수 있는 키워드의 전체집합이다. S는 상태들의 유한 집합이고, 출력함수에 의해 검색된 키워드의 집합을 A라 한다. AC 머신은 상태들의 집합으로 구성된다. 각 상태는 숫자에 의해 표현된다.

goto 함수 g는 상태 si에서 입력심벌 i를 보고, 다음 상태 sj로 이동하는 함수이다. 다음 상태로 가는 연결선이 없을 때, fail이 발생한다. fail이 발생하면 failure 함수 f가 사용된다. failure 함수 f는 상태 si에서 다음 상태 sj로 이동하는 함수이다. 출력함수 Output은 K_SET의 부분집합 Ai를 출력한다.

2.2 AC 머신 상에서의 키 검색

키워드의 집합 K_SET={"국립", "국립동식물", "국립동식물원", "동식물", "동식물원", "식물", "식물원", "식물학", "관엽식물", "원", "학"}에 대한 AC 머신에서 사용되어진 함수를 보여준다. 여기서, ¬{ 국, 동, 식, 관, 원, 학 }은 '국', '동', '식', '관', '원'과 '학'이외의 모든 입력 기호를 표시한다. 예를 들면, 상태1에서 상태2까지의 전이 라벨 국은 g(1,국)=2로 표시한다. 연결선이 없는 것은 fail를 나타낸다. AC 머신은 모든 입력 심벌 s에 대해 g(1,s)≠fail 이라는 속성을 가지고 있다. 출력함수 Output에서 검색 가능한 키워드들을 보여준다. Output(3)={국립}이라는 것은 상태3에서 검색이 가능한 키워드가 "국립"이라는 것을 의미한다. Output(6)={국립동식물, 동식물, 국립, 식물}이라는 것은 상태6에서 검색이 가능한 키워드가 "국립동식물", "동식물", "국립", "식물"이라는 것을 나타낸다.

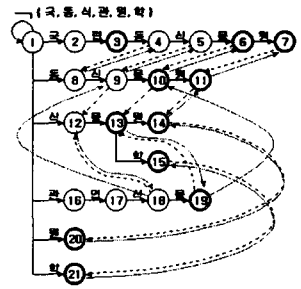
3. DER 구조

3.1 DER 구조

확장된 AC 머신인 DER 구조의 경우, 키에 대한 레코드는 파일번호나 문장번호 등과 같은 포스팅(posting)이라고 불리는 참조 정보를 가지고 있다. g(1,x)=s 와 Output(s)∋x와 같은 키 x에 대한 포스팅을 P(s)라 하자. failure 함수에 의한 반대 방향으로의 상태 전이를 표현하기 위해 아래 집합을 소개한다[2]. 아래의 집합은 reverse 함수 f-1(s)에 의해 구해진다.

$$f^{-1}(s)=\{s' \mid f(s')=s\}$$

[그림 1]는 DER 구조에서 사용되어진 goto 함수, failure 함수, reverse 함수에 의해 구성된 상태들의 관계를 보여준다. 상태1에서 상태2로 goto 함수에 의해 상태 전이가 가능하고, failure 함수에 의해 상태4에서 상태8로 상태 전이가 가능하다. 또한 상태8에서 상태4로 reverse 함수에 의해 상태 전이가 가능하다.



[그림 1] DER 구조의 goto, failure와 reverse 함수

[그림 2]은 향상된 출력함수인 Output1에 [그림 1](b)에 적용 시켜 나온 Output1 함수의 예를 나타낸 것이다.

[그림 1]과 [그림 2]에서 x="국립동식물"에서 Output1(6)={P(3), P(6), P(10), P(13)}은 g(1,x)=6과 같은 상태6에서 P(3), P(6), P(10), P(13)이 검색 될 수 있다. x="식물"에 대해 Output1(13)={P(13)}은 g(1,x)=13과 같은 상태13에서 P(13)이 검색될 수 있다. 그러나 Output1(13)에서는 "식물원", "동식물원", "동식물"과 같은 단일 키워드에 대한 포스팅을 검색하기는 불가능하다.

```
Output1(3)={P(3)}
Output1(6)={P(3), P(6), P(10), P(13)}
Output1(7)={P(3), P(6), P(7), P(10), P(11),
            P(13), P(20)}
Output1(10)={P(10), P(13)}
Output1(11)={P(10), P(11), P(13), P(14), P(20)}
Output1(13)={P(13)}
Output1(14)={P(13), P(14), P(20)}
Output1(15)={P(13), P(15), P(21)}
Output1(19)={P(13), P(19)}
Output1(20)={P(20)}
Output1(21)={P(21)}
```

[그림 2] 향상된 출력함수

3.2 DER 구조의 검색 기법

y와 z를 공백이 아닌 문자열이라고 할 때, 아래 4개의 검색 방법을 정의한다[2].

① EXACT 검색 - EXACT(x)

EXACT(s)={ P(s) | g(1,x)=s 이고 x∈Output(s) }

② PREFIX 검색 - PREFIX(s)

PREFIX(s)={ P(t) | g(1,x)=s 이고 x∈Output(s) 을 만족하는 각 s에 대해, g(s, y)=t 이고 xy∈ Output(t)을 만족하는 상태들 }

③ SUFFIX 검색 - SUFFIX(s)

SUFFIX(s)={ P(t)| g(1,x)=s 이고 x∈Output(s) 인 각 s에 대해, 만약 상태t가 f^{-1}(s)에 포함되고, *Output(t) ≠empty*이면, Output(t)는 *g(1,yx)=t*를 만족하는 yx를 갖는다. }

④ PROPER-SUB 검색 - PROPER(s)

*PROPER(s)={ P(r) | g(1,x)=s 이고 x∈ Output(s)*을 만족하는 각 s에 대해, 만약 f^{-1}(s)가 g(t,y)=r을 만족하는 상태t를 갖는다면, P(r)∈ Output(r)이다. }

4. DERtable

4.1 DERtable 구조

기존 DER 구조의 검색 방법은 failure 정보에 대한 reverse 정보를 이용하여 각 상태를 추적하면서 검색하므로 많은 시간이 걸리는 문제가 있다. 본 논문에서 제안한 "테이블을 이용한 DER 구조(DERtable)"는 DER의 검색방법을 이용하여 테이블 구조를 만들고, 검색 시에는 테이블을 이용하여 빠른 검색이 가능하다. 그러므로 DER 구조의 검색시간이 많이 걸리는 문제를 해결할 수 있다.

4.2 DERtable 구조의 구성

3.2절의 DER 구조의 검색기법을 이용하여 검색될 수 있는 모든 키워드를 테이블을 이용하여 정렬하므로써 테이블을 이용한 빠른 검색이 가능하도록 하였다. 이 같이 하여 만들어진 테이블은 PREFIX 테이블, SUFFIX 테이블, PROPER 테이블이다. EXACT 테이블은 PREFIX 테이블에 포함되므로 EXACT 테이블은 만들지 않고, 검색시 EXACT 인덱스는 PREFIX 인덱스 테이블을 이용한다.

4.2 DERtable 구조에서의 키워드 검색

[그림 3](a)는 검색 테이블의 인덱스(IND)를 나타낸다. (b)는 PREFIX 함수에 의해 구해진 PREFIX 검색 테이블이고. (c)는 SUFFIX 함수에 의해 구해진 SUFFIX 검색 테이블이다. (d)는 PROPER 함수에 의해 구해진 PROPER-SUB 검색 테이블이다.

START는 ARY의 시작 인덱스를 나타내고, OFFSET은 검색 엔트리의 수이고, IND는 상태번호부터 START의 인덱스로 맵핑한다. 그리고 t는 상태번호를 나타낸다.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
E-IND			1			2	3			4	5		6	7	8				9	10	11
P-IND										2	5		1	4						3	6
S-IND										2			1								
PS-IND																					

(a) 검색 테이블의 인덱스

	1	2	3	4	5	6	7	8	9	10	11
P-ARY	P(3)	P(6)	P(7)	P(10)	P(11)	P(13)	P(14)	P(15)	P(19)	P(20)	P(21)

	1	2	3	4	5	6	7	8	9	10	11
P-START	1	2	3	4	5	6	7	8	9	10	11
P-OFFSET	3	2	1	2	1	3	1	1	1	1	1

(b) PREFIX 검색 테이블

	1	2	3	4	5	6	7
S-ARY	P(19)	P(10)	P(6)	P(14)	P(11)	P(7)	P(15)

	1	2	3	4	5	6
S-START	1	3	4	5	6	7
S-OFFSET	3	1	3	2	1	1

(c) SUFFIX 검색 테이블

	1	2
PS-ARY	P(10)	P(6)

	1	2
PS-START	1	2
PS-OFFSET	2	1

(d) PROPER-SUB 검색 테이블

[그림 3] 검색 테이블

5. 실험 및 평가

5.1 검색 시스템 환경

검색 시스템은 사용되는 문서에서 키워드를 선택하여 인덱싱을 해야한다. 그러나 수작업을 통해 소규모로 구성하면 사전 구성시간이 많이 걸리는 문제점이 있다[6][7]. 본 시스템에서는 인덱싱을 위해 문서에서 키워드를 선택하지 않고, 코퍼스에서 체언을 추출한 후, DERtable 구조를 이용하여 키워드 사전을 구성하였다. 그리고 DERtable 구조로 구성된 키워드 사전에 있는 키워드로 문서들에 대한 포스팅 정보를 구성하였다.

사용한 코퍼스는 KAIST에서 구성한 100만 어절의 품사 부착 코퍼스와 1만 문장의 구문 구조 부착 코퍼스로 구성되며, 키워드 사전은 품사부착 코퍼스만을 사용하여 구성하였다[6][8].

5.2 실험 환경

제안한 방법의 평가를 위해 CD 데이터로 제공된 한

국정보처리학회 13회 춘계학술발표 논문집과 한국정보과학회 27회 추계 학술발표논문집 중 500개의 파일을 사용하여 포스팅한 결과, 1,722 복합 키워드가 생성되었다. 사용한 파일의 개수는 500개로 제공된 CD의 문서 600개에서 키워드사전과 관련된 주제의 문서로 선별하였다.

본 평가는 펜티엄 III 800MHz, 128MByte 메모리의 PC에서, 윈도우98, Visual C6.0에서 구현 및 실험을 하였다.

5.3 AC, DER와 DERtable 구조의 비교평가

[표 1]은 복합 키워드의 수에 따른 다양한 시뮬레이션 결과를 보여준다.

[표 1] 실험 결과

		K1	K2	K3	K4	K5
전체키워드 수		350	1,249	1,849	2,747	3,347
복합 키워드 수		0	350	597	932	1,022
전체 포스팅 수	AC	6,128	8,451	9,917	11,383	12,186
	DER와 DER_{table}	742	1,560	2,115	2,645	2,759
사전 크기 (Kbyte)	AC	23.4	32.2	37.8	43.4	46.5
	DER	31.2	42.9	50.4	57.8	62
	DER_{table}	18.5	34.5	45.7	61.4	70.6
검색 시간 (sec)	DER	0.1	0.2	0.2	0.4	0.6
	DER_{table}	0.1	0.1	0.1	0.2	0.2

* 검색시간의 비교평가는 AC와 DER에서 검색되는 키워드가 다르고 DER와 DERtable에서는 같은 키워드를 검색한다. 그러므로 DER와 DERtable의 검색시간 만을 평가한다.

이 결과로 DER 구조, DERtable 구조가 AC보다 사전 크기에서는 증가하지만, 복합 키워드들을 검색할 수 있고, DERtable 구조가 DER구조에 Table이 추가되지만, 검색시간에 총 키워드의 수가 증가할수록 빠른 검색 시간을 보여주고 있다는 것을 확인할 수 있다. 사전의 크기가 증가하는 이유는 사전의 구조가 변경되었기 때문이다. AC는 base, check와 failure의 세 배열로 이루어졌있다.[3] DER는 base, check, failure 배열에 reverse 배열이 추가된 구조로 reverse 배열 크기 만큼의 사전이 커지게 된다. DERtable은 base와 check 배열과 검색테이블로 이루어지게 된다. failure와 reverse 배열은 검색 테이블을 만들때만 필요하고, 검색시에는 필요없는 부분이므로 base와 check 배열과 검색테이블로 재구성된다. DERtable은 DER보다 사전의 크기가 반으로 줄지만 검색테이블이 추가되므로 비슷한 크기를 유지하는 것을 확인할 수 있다. 검색테이블의 크기는 키워드의 수와 관계가 있다. 키워드의 수가 늘어날수록 사전의 크기 증가하는 것을 확인할 수 있다.

6. 결 론

정보검색에서 사용하는 키워드는 의미정보가 내포된 체언을 주로 사용한다. 체언은 사물의 실체를 가리키는 말로서 문장에서 조사의 도움을 받아서 주어, 목적어, 보어의 구실을 하는 말이다. 모든 단어를 가지고 키워드 사전을 구현할 때는 사전의 크기가 커지고, 검색시간이 많이 걸리기 때문에 모든 품사의 단어를 키워드 사전으로 구성하는 것은 비효율적이다. 그래서 키워드 사전은 KAIST에서 구성한 100만 어절의 품사 부착 코퍼스에서 체언만을 추출하여 구성하였다.

또한 [표 1]에서 보는 바와 같이 총 키워드의 수가 증가할수록 검색시간이 빠름을 보여주고 있고, 포스팅의 수에서도 AC보다 DER와 DERtable 구조가 월등히 감소한 것을 확인할 수 있었다. 그러므로 제안한 방법은 컴퓨터 시뮬레이션 결과 효과적임을 확인하였다.

향후 연구과제에서 제안한 방법에 키워드별 문서분류 코드를 주어, 자동문서 분류 시스템을 구성하는 것을 목표로 하고 있다.

참 고 문 헌

[1] Ogawa, Y., Mochinushi, M. and Bessho, A. "A compound keyword assignment method for Japaness texts", IPS Japan SIG Notes. 93-NL-97-15, pp.103-110, 1993.

[2] Makoto Okada, Kazuaki Ando, Kazuhro Morita, Jun-ichi Aoe, "An Efficient Determination of Keywords for Compound Words", Proceedings of 18th ICCPOL, Vol 1, pp317-320, March 1999.

[3] Kazuaki Ando, Toshiharu Kinoshita, Masami Shishibori, Jun-ichi Aoe, "An improvement of the Aho-Corasick machine", International Journal of Information Sciences, Vol 3, pp139-151, 1998.

[4] A. V. Aho, M. J. Corasick, "Efficient string matching: an aid to bibliographic search", Comm. ACM, Vol.18, No.6, pp.333-340, 1975.

[5] 정민수, "코퍼스로부터 구문분석을 위한 사전 구성", 군산대학교 컴퓨터과학과 석사학위논문, 1999.

[6] 김덕봉, 최기선, "효율적인 한국어 형태소 해석 방법", 한국과학기술원 전산학과 석사학위논문, 1994.