키팩트 색인텀에 기반한 정보검색 시스템

박의규°, 나동열, *변성찬, †정경택, 박세영
연세대학교 전산학과
*LG 종합기술원
†서치캐스트 주식회사
{ekpark,dyra}@magics.yonsei.ac.kr, *schbyun@lgcit.com, †{ktchong,sypark}@searchcast.net

An Information Retrieval System Based on Keyfact Index Term

Eui-Kyu Park, Dong-Yul Ra, *Seong-Chan Byun, †Kyung-Taek Chung, Se-Young Park
Dept. of Computer Science, Yonsei University
*LG Corporate Institute of Technology
†Search Cast Co., Ltd.

요약

지금까지의 정보검색 시스템은 소위 키워드 기반 정보검색 시스템으로서 색인이 단일 단어(single word) 즉 키워드의 집합으로 나타내어 진다. 그러나 이 방법은 문서의 내용을 정확히 표현하는 데 한계가 있다. 따라서 최근에는 단어 이상의 구문 단위인 구(phrase)를 이용하여 색인과 검색을 하도록 하는 시스템을 개발하고자 하는 추세에 있다. 따라서, 본 논문에서는 키워드보다는 의미를 좀더 잘 나타내고 일반적인 구보다는 정형화된 형태의 색인 단위인 키팩트를 색인어로 하는 정보검색시스템을 개발하고 이의 성능을 살펴보았다.

1. 서론

정보검색 시스템에서 정보의 내용을 대표하는 색인어의 추출은 정보가 기하급수적으로 늘어가는 요즘 중요한 부분으로 인식되고 있다. 정보검색 시스템에서 정보의 내용을 대표하는 색인어를 추출하는 방법으로는 우선 전문화된 사람에 의한 색인어 추출을 들 수 있다. 그러나 정보가 기하급수적으로 늘어나고 있는 현실을 감안한다면 사람에 의한 색인어 추출 방법은 경제적이라 할 수 없다. 또한 사람에 의한 색인어 추출 방법은 색인어의 일관성 결여라는 문제가 상존하며 색인하고 자 하는 문서의 양이 점차 방대하여 지므로 컴퓨터에 의해 자동적으로 색인을 해야 한다는 것이 당연시 되고 있다[9].

지금까지의 정보검색 시스템은 소위 키워드 기반 정보검색 시스템으로서 색인이 단일 단어(single word) 즉, 키워드의 집합으로 나타내어 진다. 그러나 이 방법은 문서의 내용을 정확히 표현하는 데 한계가 있다. 따라서 최근에는 단어 이상의 구문 단위인 구(phrase)를 이용하여 색인과 검색을 하도록 하는 시스템을 개발하고자 하는 추세에 있다. 특히 구를 알아 내기 위해서는 자연어 처리 기술이 필수적이므로 정보검색 시스템에 자연어 처리 기술을 도입하는 것이 큰 추세의 하나가 되고 있다[5][1].

구 기반 정보검색 시스템에서의 문제점은 구를 단위로 색인을 하는 것이 쉽지 않다는 것이다. 우선 구의 크기 및 모양이 정형화 되어 있지 않기 때문에 구를 이루는 단어의 수가 매우 가변적이라는 것이다. 또한 같은 의미를 표현하는 데에 있어서 여러 가지의 형태의 구가 가능하다는 것이다[3].

따라서, 키워드보다는 의미를 좀더 잘 나타내고 구보다는 정형화된 형태의 색인어인 키팩트[2]를 기반으로 하는 정보검색 시스템의 필요성이 제안되었다[8]. 우리는 이와 같은 키팩트 기반 색인을 영어 문서 정보검색 시스템의 개발에 적용하여 보았다. 전체 시스템의 구성은 [그림 1]과 같다. 본 논문에서는 이 시스템의 설계, 구현 및 실험 결과에 대하여 설명하고 자 한다.

2. 색인 단위로서의 키팩트

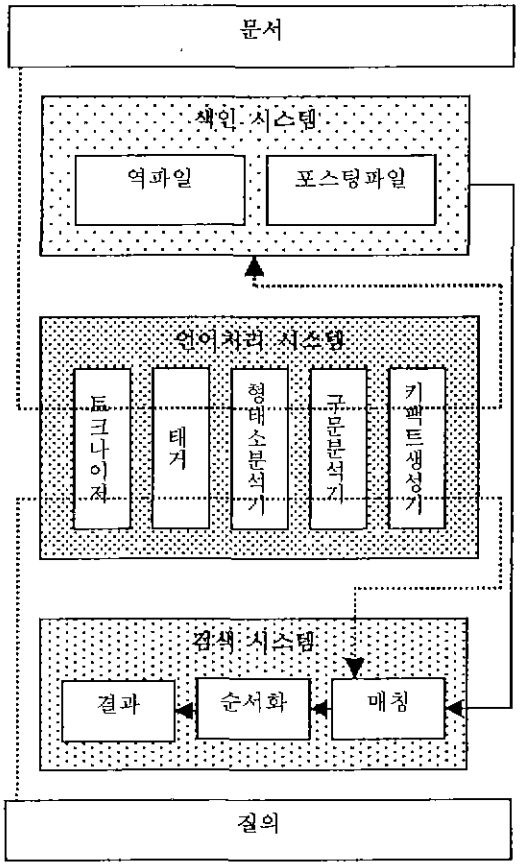
지금까지의 정보검색 시스템은 소위 키워드 기반 정보검색 시스템으로서 색인이 단일 단어(single

[1] 현재까지는 자연어 기술을 이용한 정보검색 시스템이 생각처럼 그렇게 큰 성공을 거두고 있지는 못하다. 그럼에도 많은 사람들은 그 필요성에 대해서는 부인하지 않고 있다.

[2] 키팩트는 정보검색 분야 문헌에서 많이 이야기되고 있는 head-modifier 쌍과 거의 같은 개념으로 볼 수 있다.

word) 즉, 키워드의 집합으로 나타내어 진다. 키워드 기반 정보검색 시스템에서의 문제점은 키워드를 이용하여 문서를 정확히 나타낼 수가 없다는 것이다. 단일 단어를 이용할 경우 단일 단어 두개가 어떤 의미를 나타내는 경우 이 의미를 나타낼 수 없기 때문이다. 예를 들어 "high school"의 경우 두개의 단어가 합쳐져서 "고등학교"라는 하나의 의미를 갖지만, 키워드 기반에서는 "high", "school"이 나뉘어서 처리되기 때문에 이러한 의미를 정확히 나타내기가 어렵다.

키팩트는 객체와 속성을 나타내는 두개의 단어로 구성된 오브젝트(object)이다. 예를 들어, "high school"의 경우 키팩트로 "[school,high]"가 생성되며, 여기에서 "school"은 객체이며, "high"는 속성이 된다. 이것의 의미는 "school"이 속성 "high"를 갖는다는 것이며, 이로 인해 "고등학교"라는 의미를 정확히 나타낼 수 있게 된다.



[그림 1] 전체 시스템 구성도

따라서, 본 논문에서는 키워드보다는 의미를 좀더 잘 나타내고 구보다는 정형화된 형태의 색인어인 키팩트를 추출하는 시스템을 제안한다.

키팩트 기반의 정보검색 시스템은 키워드 기반의 정보검색 시스템보다 정확률을 향상 시킬 수 있다. 예를 들어,

D_1="…. church chimney ….."

D_2= "…church…… chimney……."

두 개의 문서에서 추출될 수 있는 키워드 기반의 색인텀들은 church, chimney 이고, 키팩트 기반의 색인텀들은 [chimney,church], [chimney,], [church,] 이다. 이러한 상황에서 다음과 같은 질의를 사용하여 검색을 수행하면 키워드 기반의 정보검색 시스템과 키팩트 기반의 정보검색 시스템의 차이를 알 수 있다.

질의 1: chimney

질의 2: church

질의 3: church chimney

질의 1 이나 질의 2 모두 키워드 기반이나 키팩트 기반에서 문서 D_1 과 D_2 가 동일한 순위를 갖는다. 그러나 질의 3 은 키워드 기반에서는 문서 D_1 과 D_2 가 동일한 순위를 갖을 수 밖에 없으나 키팩트 기반에서는 "church chimney"를 포함한 문서 D_1 이 그렇지 않은 D_2 보다 더 높은 순위를 갖는다.

우리의 키팩트는 다음과 같은 형태를 갖는다 :

[명사, 명사], [명사, 형용사], [명사, 동사].

또한, 키팩트는 키팩트를 구성하는 명사, 동사, 형용사들을 stemming 하여 얻어진 원형을 이용하여 생성된다. 예를 들면 다음과 같다.

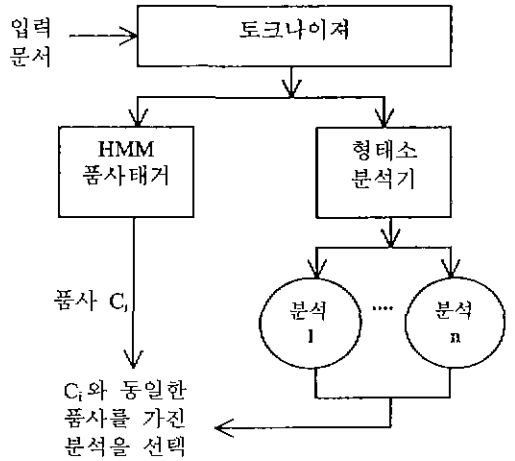
studied the factory

⇒ [factory, study]

information retrieved by ....,
information retrieval,
retrieved information

⇒ [inform, retrieve]

3. 형태소분석기



[그림 2] 형태소분석기의 구성도

3.1 토크나이저

토크나이저는 입력문서의 내용을 토큰(token)으로 나눈다. 토큰으로 나눌 때 고려하여야 할 문자는 공백문자, new-line 문자, 마침표(.), 쉼표(,), 따옴표(",')로 나누어지는 단위이다.

토큰 's 는 소유격을 나타내는 하나의 형태소로서 사전에 등록되어 있다. 쉼표나 마침표인 경우는 이를 분리하여 하나의 토큰으로 출력하여야 한다. 특히 마침표가 문제를 복잡하게 만드는 경우가 많다. 생략용으로 사용된 마침표의 경우는 토큰의 일부로 간주하여야 한다. 그러나 어떤 마침표는 생략으로도 사용되고 동시에 문장의 마침을 나타내는 기호로도 사용된다.

단어의 끝에 붙어 있는 기호 . 가 마침표인지 생략기호인지 아니면 둘 다인지를 판단하는 것은 쉽지않다. 본 논문에서는 기호 . 가 마침표인지 생략기호인지를 판단하는 몇몇 휴리스틱을 적용하였다.

3.2 HMM 품사 태거

태거는 문서 내의 어휘 중의성을 해결할 수 있도록 품사 태깅을 한다. 품사 태깅은 좀더 정확한 어휘 정규화를 이끌어 낼 수 있다. 본 논문에서 제안한 시스템에서 사용한 태거는 *HMM(Hidden* Markov Model) 기반 태거로서 Penn Treebank 데이타를 이용하여 훈련되었다. 이용하는 품사집합은 Penn Treebank 의 품사집합과 동일한 것이다[2].

품사 태깅에 있어서 문제가 되는 미등록어에 대한 처리는 몇몇 휴리스틱을 이용하여 처리하였다. (휴리스틱을 적용할 수 없는 경우는 일반명사로 태깅되도록 하였다.)

이렇게 개발되어 본 시스템에서 사용하는 태거의 정확률은 약 96% 정도이다.

3.3 형태소분석

형태소분석은 각 단어들을 stemming 하여 원형을 복원한다. 단어의 변화된 형태가 입력으로 들어오더라도 원형을 복원하여 이용하므로, 원형이 같은 단어는 같은 형태의 *키퍽트로* 생성된다. 이를 정보검색에 이용하면 재현률을 높일 수 있다.

형태소분석의 원리는 기본형만 사전에 저장하고, 변화형은 불규칙 처리 규칙에 의하여 원형으로 복원한다. 우리가 사용하는 방법은 기본적으로 Kimmo 의 2 층 구조와 같은 기법이다[4]. 단, 완전 불규칙의 경우는 사전 검색을 이용한다.

4. 구문분석

영어 구문분석에 대한 문법은 문맥자유문법(CFG: context-free grammar)을 이용한다. 형태소분석기를 거친 하나의 문장은 영어 CFG 구문분석기에게 전달된다. 그러면 이 구문분석기는 CFG 에 기반하여 문장을 분석한다.

4.1 영어 문법

문법은 문맥자유문법에 의하여 작성된다. 문법의 일부 규칙의 모습은 다음과 같다.

- 기본 명사구: 명사구에서 가장 핵심을 이루는 기본적인 구조

```
NP → PN                          // PN 은 고유명사
NP → DET N        // DET 는 관사, 지시관사, 수사
NP → DET ADJP N              // ADJP 는 형용사구
NP → N
NP → ADJP N
NP → NP N
ADJP → ADJ                       // ADJ 는 형용사
ADJP → ADJP ADJ
```

- 전치사구를 포함하는 명사구

```
PP → PR NP          // PP 는 전치사구, PR 는 전치사
NP → NP PP
```

위와 같은 규칙에 피처(feature)를 추가함으로써 필요한 사항을 테스트하거나 생성되는 구(phrase)에게 어떤 특성 즉 피처를 배달아 놓을 수 있다. 예를 들어 다음과 같은 규칙을 보자:

AJP → Vb [ing_PT]

이 규칙에서 [ing_PT]는 심볼 Vb 가 가지고 있어야만 하는 피처를 나타낸다. 따라서 이 Vb 에 해당하는 구문구조(이 경우는 Vb 가 terminal symbol 이므로 대응되는 입력단어)가 피처로 ing_PT 를 가지고 있어야 한다는 것이다. 즉, Vb 에 대응하는 단어가 현재진행형(ing_PT) 특성을 가지고 있어야 함을 의미한다.

생성되는 구문구조에 피처를 붙일 수도 있는데 이 에는 다음과 같은 규칙에서 볼 수 있다:

NP[smp] → AJP NC

이 규칙에서는 AJP 와 NC 구문구조를 합하여 NP 구문구조를 만들 수 있음을 나타낸다. 그런데 이렇게 만들어진 NP 구문구조가 피처 smp 를 가지게 하라는 것이다.

본 시스템의 영어 문법에서는 규칙의 오른편의 심볼 옆에 있는 피처는 그 심볼에 해당하는 구문구조가 그 피처를 가지고 있는가를 테스트하라는 의미로 사용하고, 규칙의 왼편의 심볼 옆에 있는 피처는 그 심볼에 대응하는 구문구조에 붙여 줄 피처를 나타낸다.

4.2 구문분석 기법

문법에 따라 문장을 분석할 수 있는 구문분석기가 키팩트 생성기의 핵심이 된다. 본 시스템에서는 문장 전체를 완전히 분석하는 것이 주된 목표가 아니고 분석이 가능한 부분에 대한 구문구조를 알아내는 것이다.

문법이 전체 문장을 완전히 포용하는 것을 보장할 수 없으므로 본 시스템에서는 순수 상향식 기법을 사용한다. 순수 상향식 기법은 문장 전체에 대한 분석을 얻을 수 없는 경우에도 계속 진행하여 나갈 수 있으며 결과적으로 분석이 가능한 모든 부분 분석 결과를 내준다. 본 과제에서는 순수 상향식 파싱의 대표적인 방법인 left-corner 파싱 기법을 사용한다[2].

구문분석 알고리즘:

(1) agenda = NULL; /* 초기화. Make it an empty list */
    i = 1; /* i 는 다음 읽을 단어의 인덱스 */
(2) loop
        if agenda == NULL then
        begin
            if i = n+1 then
                stop;
            if A is category of ai then
                add A(i-1, i) into agenda;
            i = i + 1;
        end;
        Remove an item from agenda & let it be B(k,j);
        /* Do starting operation */
(2-1)   Add item [$A \rightarrow B \bullet \beta, j, k$] for every 문법규칙 A→Bβ; (단, B 가 테스트할 피처를 가지고 있으며 그 피처를 B(k,j)가 가지고 있는 경우에만 이 item 이 add 됨.)
        /* Do extending operations */
(2-2)   For every existing item [$A \rightarrow \alpha \bullet B \beta, l, k$] add [$A \rightarrow \alpha B \bullet \beta, l, j$]; (단, B 가 테스트할 피처를 가지고 있으면 그 피처를 B(k,j)가 가지고 있는 경우에만 이 item 이 add 됨.)
        /* Do completion operations */
(2-3)   For every item of the form [$A \rightarrow \alpha B \bullet, l, j$] added in steps (2-1) & (2-2), insert A(l,j) into agenda;
    end loop;

5. 키팩트 생성

5.1 명사 소분류

명사구에 대한 키팩트를 생성하는데 있어서 중요하게 이용되는 정보는 명사에 대한 소분류 정보이다. 정확한 키팩트를 생성하기 위해서는 명사의 의미 정보를 이용하여야 하나 현재로서 이것은 불가능하며 단시일 내에 해결될 수 있는 문제가 아니다. 따라서 본 시스템에서는 자세한 의미 분류 대신 다소 거칠기는 하지만 다음과 같이 명사를 소분류하여 이 소분류 정보를 의미 정보대신 이용하였다.

(1) NQ: 고유명사 (예: Northrop, Washington, ...)
(2) NH: 동작명사 (예: exploration, development, ...)
(3) NM: 상태명사 (예: redness, calmness, ...)
(4) NV: 무의미명사 (예: thing, fact, matter, ...)
(5) NB: 일반명사(위 1 에서 4 까지의 아무 것에도 속하지 않는 명사) (예: apple, house, president, ...)

소분류 정보는 명사의 자질정보로 사전에 기록되어 있다.

5.2 키팩트규칙

키팩트규칙의 패턴부 추출기에 대하여 이해하기 위해서는 먼저 키팩트규칙에 대하여 알아야 한다. 구문분석기가 발견한 구에 키팩트 생성규칙의 패턴부가 존재하면 키팩트를 생성하게 된다. 키팩트 생성 규칙의 구성은 다음과 같다:

$$[PE_1][PE_2]\ldots[PE_m] \Rightarrow [KF_1]\ldots[KF_n]$$

위의 형태에서 왼편은 패턴부로서 구문구조내에 존재하는지 검사할 패턴을 나타낸다. 여기에서 PE 는 패턴부의 원소인 Pattern Element 로서 명사구를 구성하는 하나의 단어에 대응된다:

PE: 한단어의 스트링 또는 품사 또는 자질을 주므로서 표시된다.

오른편은 왼편의 패턴부가 존재하는 것으로 판단되는 경우 생성하여야 할 키팩트들을 열거한다.

KF: [i,j]의 형태로서 i, j 는 왼쪽의 PE 의 인덱스이다. 생성되는 키팩트는 [PE_i 의 단어, PE_j 의 단어] 이다.

(예) NB NH NB ⇒ [1,2] [3,2]
이 규칙에서 왼쪽의 PE_1 에 해당하는 NB 는 명사의 소분류를 나타내는 자질이다. 즉, 이 규칙의 패턴부는 명사구 내에 소분류가 NB, NH, NB 인 세 개의 연속된 단어가 나타나는지를 검사하라는 것이다. 예를 들어 "automobile development plan"이라는 연속된 명사가 나타나는 명사구가 있다 하자. 그러면 [automobile, development], [plan, development] 라는 두 개의 키팩트를 생성하게 된다. 다음 표는 키팩트 규칙들의 일부를 보인 것이다.

NB ⇒ [1 0]	NB of NV ⇒ [1 0]
NH ⇒ [1 0]	NH of NH ⇒ [3 1]
NQ ⇒ [1 0]	NB on NB ⇒ [1 3]
N Vb ⇒ [1 2]	NB PREP NB ⇒ [1 3]
Vb N ⇒ [2 1]	NB PREP NQ ⇒ [1 3]
AJ N ⇒ [2 1]	....
NB NB ⇒ [2 1] [2 0] [1 0]	NH of NB ⇒ [3 1]
NB NH ⇒ [1 2] [1 0]	NH of NQ ⇒ [3 1]
NB NM ⇒ [1 2]	NH to NQ ⇒ [3 1]
NB NV ⇒ [1 0]	NQ of NQ ⇒ [1 3]
NB N ⇒ [1 2]	NV in NV ⇒ [0 0]

....	NB of NB ⇒ [1 1]
NB for NB ⇒ [1 3]	NB of NQ ⇒ [1 3]
NB for NH ⇒ [1 3]	NV of NB ⇒ [3 0]
NB NB NB ⇒ [3 2] [3 1] [2 1]	NB NB NQ NQ ⇒ [2 1] [4 3] [4 2]
NB NH NB ⇒ [3 2] [1 2]	NQ NQ NQ NQ ⇒ [4 3] [3 2] [2 1]
....	

5.3 키팩트규칙의 패턴부 추출 방법

문장이 분석되는 과정에서 위의 키팩트 규칙에 적용될 수 있는 패턴이 나타나면 이를 추출하여야 한다. 이것을 "키팩트규칙 패턴부"라 부르자. 추출된 키팩트규칙 패턴부는 키팩트 생성기에 의한 패턴매칭 작업을 통하여 매칭되는 규칙을 알아 내고 이의 액션부에 표시된 대로 키팩트들을 생성하게 된다.

패턴부 추출은 컴파일러에서 많이 사용하는 syntax-directed translation 기법을 사용하였다[1]. 이는 구문트리의 각 노드에 속성(attribute)을 두며 한 노드의 속성의 값은 자식 노드의 속성의 값을 이용하여 계산하는 기법이다 (이러한 속성을 synthesized attribute 라 한다). 속성의 계산은 각 CFG 규칙마다 의미규칙(semantic rule)을 부착하여 정의한다.

[표 1] 의미규칙

CFG 규칙	의미 규칙
NC → N	NC.hd = N; NC.t = N
NC → NC N	NC.hd = N; NC.t = NC1.t ‖ N
NP → DT NC	NP.sd = NC.t; NP.hd = NC.hd
NP → NC	NP.sd = NC.t; NP.hd = NC.hd

여기에서 이용된 각 속성을 소개하면 다음과 같다.

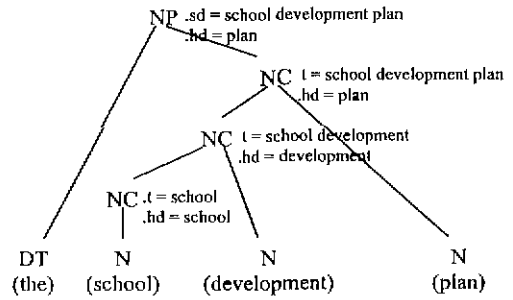
hd: 헤드 명사를 가진다.

t: 연속되는 명사열을 가진다.

sd: 키팩트생성기로 보내져야(send) 할 단어열을 가진다.

(주: 의미규칙에서 ‖는 concatenation 작업(오른쪽에 붙임)을 나타내는 연산자이다.)

예를 들어 "the school development plan"에 대한 구문트리 및 의미규칙에 의하여 각 노드에서 계산되는 속성값을 나타내면 그림 3 과 같다.



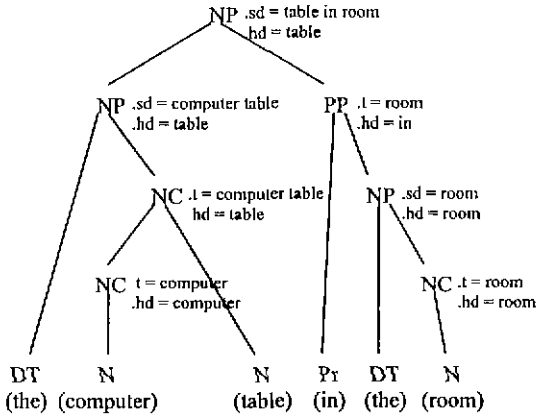
[그림 3] 구문트리 노드의 속성값

위 트리와 같이 속성값이 계산되었다 하자. 그러면 나중에 트리의 각 노드를 depth-first 방식으로 방문하여 sd 속성에 저장된 값을 키팩트생성기 쪽으로 보낸다.

전치사구를 가진 명사구의 분석과 관련한 규칙들과 이들의 의미규칙을 생각하여 보자.

CFG 규칙	의미 규칙
PP → Pr NP	PP.hd = Pr; PP.t = NP.hd
NP → NP_1 PP	NP.hd = NP_1.hd; NP.sd = NP_1.hd ‖ PP.hd ‖ PP.t

명사구 "the computer table in the room"에 대한 구문트리 및 각 노드에서의 속성의 값은 다음과 같다.



[그림 4] 진치사구 포함 명사구에 대한 구문트리 및 속성값

위와 같이 분석이 완성된 NP 노드들은 NPS 라는 연결리스트에 연결되어 있게 된다. 문장의 구문분석이 완료된 후 NPS 에 연결되어 있는 각 NP 노드에 대해서 그 안의 각 노드를 방문하며 sd 속성에 있는 값을 키팩트생성기에게 전달한다. 위의 구문트리에 대해서는 다음 3 개의 단어열들이 보내진다:

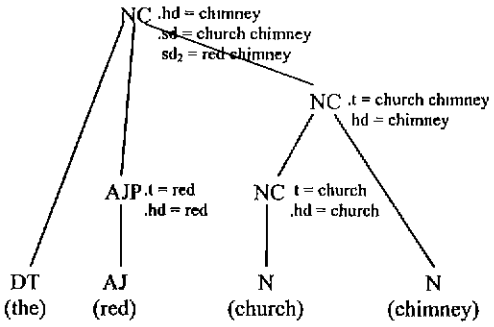
computer table
room
table in room

키팩트생성기는 각 단어열을 받아서 패턴부가 매칭되는 규칙을 탐색하여 해당 액션부에서 지시하는 대로 키팩트를 생성한다.

수식어를 가진 명사구의 처리와 관계가 되는 규칙 및 의미 규칙들은 다음과 같다.

CFG 규칙	의미 규칙
AJP → AJ	AJP.hd = AJ
AJP → AJP AJ	AJP.hd = AJ
AJP → N POSS	AJP.hd = N
AJP → Vb [ing_PT]	AJP.hd = Vb
NP → DT AJP NC	NP.sd = NC.t; $NP.sd_2$ = AJP.hd ‖ NC.hd

위에서 sd_2 속성은 키팩트생성기에 보내야 할 또 하나의 단어열을 담고 있다. 위의 규칙에 따라 "the red church chimney"를 분석한 모습은 다음과 같다.



[그림 5] 두 개의 패턴부열을 가진 노드

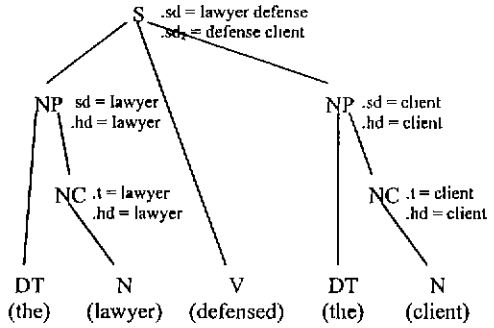
결국 그림 5 의 구문트리를 depth-first 방식으로 각 노드를 방문하면서 sd 나 sd_2 속성이 있으면 그 내용을 키팩트 생성기에 전달한다. 이 구문트리로부터 키팩트생성기에게 전달할 단어열은 다음과 같다:

church chimney
red chimney

본 시스템의 특징은 키팩트에 동사를 포함함으로써 문서의 내용을 더욱 잘 나타내게 된 점이다. 이와 관련된 규칙의 일부는 다음과 같다.

CFG 규칙	의미 규칙
S → NP_1 Vb [base] NP_2	S.sd = NP_1.hd ‖ Vb; S.sd_2 = Vb ‖ NP_2.hd

"the lawyer defensed his client"에 대한 구문트리 및 관련된 속성의 값을 보면 다음과 같다.



[그림 6] 동사를 포함한 질에서의 키팩트의 생성을 위한 구문트리

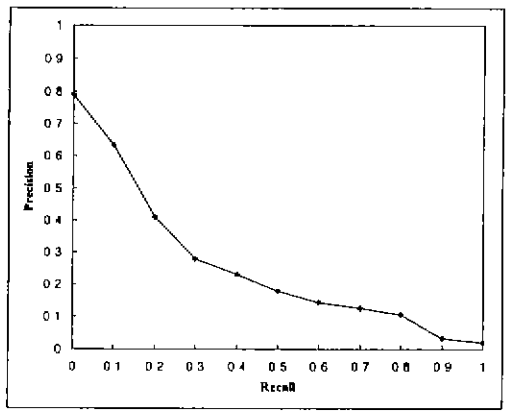
5.4 키팩트생성기

키팩트생성기에 대한 입력은 키팩트규칙 패턴부로서 추출되어 전달되는 단어열이다. 각 단어는 스트링, 품사, 자질정보들을 함께 가지고 있다. 키팩트규칙의 모습은 5.2 키팩트규칙에서 소개하였다. 패턴부의 각 PE 는 단어스트링, 품사, 소분류를 포함한 자질 중의 어느것 하나로도 표시할 수 있다.

단어열에 대하여 키팩트생성기는 키팩트규칙 베이스 안의 규칙을 차례로 하나씩 택하여 규칙의 패턴부의 명세를 주어진 단어열이 만족하는지 검사한다. 만약 검사를 만족하면 이 규칙의 액션부에 지정된 대로 키팩트들을 출력하고 즉시 종료한다. 만족되지 않으면 계속 다음 규칙에 대하여 검사한다.

6. 실험 및 평가

색인 단위로 키팩트를 이용하는 것의 효과를 알아보기 위하여 키팩트를 이용한 정보검색 시스템을 구축하여 실험하였다. 구축된 정보검색 시스템은 벡터 공간 모델을 사용하였으며, 테스트 데이터로는 TREC-6 의 20000 문서를 추출하여 테스트를 하였다. 질의로는 description 만을 사용하였다. 우리 시스템의 성능은 [그림 7] 과 같다. 외국의 정보검색 시스템들 중 동일한 데이타에 대한 실험 결과가 좋은 순서로 상위 8 개 시스템의 성능은 [그림 8]와 같다. 이 두 그래프로 부터 키팩트를 이용한 정보검색 시스템이 보다 좋은 결과를 얻었음을 알 수 있다.



[그림 7] 키팩트를 이용한 정보검색 시스템의 재현률/정확률 그래프

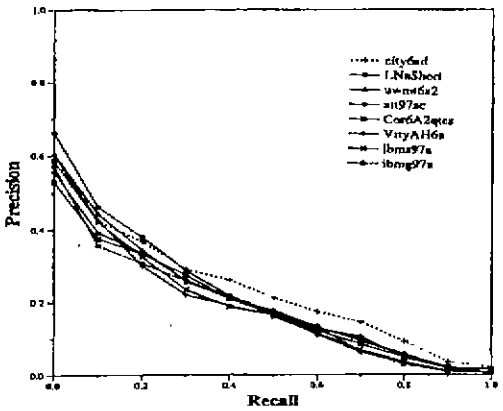
7. 결론

키워드보다 의미가 강하면서 구 보다는 정형화된 형태의 키팩트를 사용하므로써 우리는 시스템의 성능이 향상되는 것을 관찰하였다. 따라서 키팩트가 문서의 정보를 표현하는 보다 나은 색인어로 사용될 수 있다는 것을 확인한 것이다.

그러나 우리가 사용한 구문 분석기는 개선의

여지가 매우 큰 현재로서는 불완전한 것임에 틀림이 없다. 좀 더 넓은 범위의 영어를 완전하게 분석할 수 있는 구문 분석기를 사용한다면 보다 좋은 결과가 예상된다.[3]

제시된 시스템이 갖추지 못한 주요한 기능 중의 하나는 복합명사 분석에 대한 것이다. 이 기능을 추가하므로써 시스템의 성능 향상을 기대할 수 있다.



[그림 8] TREC-6 에 참여한 시스템 가운데 상위 8 개 외국시스템의 재현률/정확률 그래프

참고문헌

[1] Alfred V. Aho and Jeffrey D. Ullman, *The theory of parsing, translation, and compiling*, Englewood Cliffs, NJ· Prentice Hall, 1972.

[2] James Allen, *Natural language understanding*, Second edition, The Benjamin/Cummings Publishing Company, Inc. Menlo Park, CA, 1994.

[3] A.T. Arampatzis, T. Tsoris, C.H.A. Koster and Th.P. van der Weide, "Phrase-based Information Retrieval," Technical Report CSI-R9809, Computer Science Institute, Univ. of Nijmegen, Nijmegen, The Netherlands, March 1998. Submitted to Information Processing & Management Journal, 1998.

[4] Kimmo Koskenniemi, "A general computational model for word-form recognition and production," Proceedings of Coling '84, Association for Computational Linguistics, pp. 178-181, 1984.

[5] J. Perez-Carballo, Tomek Strzalkowski, "Natural language information retrieval: progress report," Information Processing and Management 36, pp. 155-178, 2000.

[6] G. Salton, *Automatic Text Processing*, Addison Wesley, 1989.

[7] G. Salton, "On the application of syntatic methodologies in automatic text analysis," Information Processing and Management, 26-1, pp. 73-92, 1990.

[8] 한국진자통신연구원, 내용기반 멀티미디어 정보검색 기술개발, 정보통신부제출 연구보고서, 1997. 12.

[9] 최기선, "한국어 정보검색," 정보과학회지 제 12 권 제 8 호, pp. 24-32, 1994.

[3] 이 점은 현재로는 확인이 안된 것으로 정보검색 분야가 앞으로 이의 확인을 위해서 많은 노력이 필요한 방향이다.