언어 정보 획득을 위한 한국어 코퍼스 분석 도구

이°호, 김 진 동, 임 해 창
고려대학교 전산과학과

A Korean Corpus Analysis Tool for Language Information Acquisition

Ho Lee, Jin-dong Kim, Hae-chang Rim
Department of Computer Science, Korea University

요 약

코퍼스는 기계 가독형으로 저장되어 있는 실제 사용 언어의 집합으로 자연어 처리에 필요한 여러 가지 언어 정보를 내재하고 있다. 이들 정보는 코퍼스 분석기를 이용하여 획득할 수 있으며 용례와 각종 통계 정보 및 확률 정보, 연어 목록 등은 코퍼스에서 추출할 수 있는 대표적인 언어 정보들이다. 그러나 기존의 한국어 코퍼스 분석 도구들은 용례 추출 기능만을 보유하여 활용 범위가 제한되어 있었다. 이에 본 논문에서는 대량의 한국어 코퍼스를 분석하여 용례뿐만 아니라 자연어 처리의 제분야에서 필요한 언어 정보들을 추출하는 방법에 대해 연구하였으며 이의 검증을 위해 KCAT(Korean Corpus Analysis Tool)를 구현하였다.

KCAT는 코퍼스 색인, 용례 추출, 통계 정보 추출, 연어 추출 부분으로 구성되어 있다. 용례 색인을 위해서는 여러 가지 사전과 용례 색인 구조가 필요한데 KCAT에서는 가변 차수 B-Tree 구조를 이용하여 사전을 구성하며 용례 색인을 위해 버킷 단위의 역 화일 구조를 이용한다. 질 좋은 용례의 추출을 위해 KCAT는 다양한 용례 연산 및 정렬 기능을 제공한다. 또한 통계적 방법의 자연어 처리 분야를 위해 어휘 확률, 상태 전이 확률, 관측 심볼 확률, 상호 정보, T-score 등을 제공하며, 기계 번역 분야에서 필요한 연어를 추출한다.

제 1 장 서론

1.1 연구 배경

코퍼스는 기계 가독형으로 저장되어 있는 실제 사용 언어의 집합이다[11]. 코퍼스는 소수의 사람에 의해 만들어진 것이 아니라 다수의 일반 사람들에 의해 실제로 사용되고 있는 언어의 집합이기 때문에 객관적이며 시대성을 담고 있다[6]. 그리고 대량의 균형 코퍼스는 그 언어가 실제로 사용되는 형태를 대부분 포함하기 때문에 코퍼스로부터 언어 정보를 획득하는 방법은 기존의 방법이 가지고 있는 단점을 보완할 수 있다.

코퍼스에서 추출할 수 있는 언어 정보에는 용례, 형태소 및 어절의 출현 빈도, 태그 bigram, 태그 trigram, 상호 정보, 연어, 오류 어절의 용례 등이 있으며 이들 언어 정보는 사전 편찬, 자동 태깅, 기계 번역, 철자 검사 및 교정, 형태소 분석, 음성 인식 및 합성, 문자 인식, 언어 학습 등 자연어 처리와 연관된 여러 분야에서 이용된다[11][8]. 그러나 언어 정보를 얻기 위해 방대한 양의 코퍼스를 수작업으로 분석하는 것은 너무 많은 시간과 노력이 필요할 뿐만 아니라 분석자의 주관이 개입되어 잘 못된 정보를 산출할 수도 있기 때문에 대부분의 코퍼스 분석은 코퍼스 분석 도구에 의해 이루어진다.

1.2 기존의 코퍼스 분석 도구

영어권의 경우 일찍부터 코퍼스에 대한 연구가 있었던 이유로 용례와 통계 정보를 추출해 주는 OCP(Oxford Concordancer Program), TACT(Text Analysis Computing Tools)와 연어를 추출해 주는 Xtract와 같은 다양한 코퍼스 분석 도구를 보유하고 있다[13][14][15].

그러나 국내에서는 코퍼스를 기반으로 하는 자연어 처리 기법이 늦게 도입된 이유로 코퍼스 분석 도구에 대한 개발이 부족한 실정이다. 현재 국내에 개발되어 있는 코퍼스 분석 도구는 모두 용례 추출 기능만을 지원하고 있으며 용례 추출에 있어서도 한국어의 특성을 제대로 반영하고 있지 못한다. KOCP(KAIST Oxford Concordancer Program)[3]는 OCP에 한글 처리 기능과 사용자 정의 테이블을 추가하여 만들어진 용례 추출 도구인데 한국어와 영어의 언어적 차이로 인해 용언의 활용 형태나 어미, 조사와 같은 형식 형태소에 대한 용례를 추출할 수 없다는 단점을 지닌다[3]. KIRT(Korean

Information Retrieval Tool)는 형태소 분석 결과에 나타나는 모든 형태소에 대해 색인을 하여 OCP의 단점을 보완하였으나 용례 연산이나 정렬 기능을 지원하지 않는다[5]. 이창덕 시스템[6]은 대량의 코퍼스를 위한 용례 추출기로서 용례 추출, 연산 및 정렬 기능을 가지고 있다. 그러나 앞 뒤 어절에 의한 용례 연산이나 정렬만이 가능하며 용례 정렬을 위해 색인 결과를 미리 정렬해 두기 때문에 새로운 문서에 대한 색인을 할 수 없는 단점이 있다[6].

제 2 장 코퍼스 분석기의 기능

지금까지 살펴본 한국어를 위한 코퍼스 분석 도구들은 용례 추출 기능만을 가지고 있다. 그러나 다양한 언어 정보의 추출을 위해 코퍼스 분석 도구는 다음과 같은 기능을 제공해야 한다.

2.1 용례 추출 기능

용례 추출기는 방대한 양의 코퍼스에서 용례를 추출함에 있어서 수작업을 최소화하는데 목적이 있다. 이를 위해 용례 추출기는 주어진 언어 요소에 대한 용례를 찾아 주는 용례 탐색 기능과 찾아진 용례에서 원하는 용례만을 선별하는 용례 연산 기능, 유사한 형태의 용례끼리 묶어주는 용례 정렬 기능 등을 지원해야 한다.

2.1.1 용례 탐색 기능

용례 탐색 기능은 형태소나 어절, 품사 등과 같은 언어 요소에 대한 용례를 탐색하는 기능이다. 형태소는 한국어 처리에 있어서 가장 기본이 되는 요소이기 때문에 형태소에 대한 용례 탐색 기능이 필수적이다. 한편 한국어의 표기가 어절 단위의 띄어쓰기를 따르고 있으며 어절이 연어의 구성 요소로 사용되는 경우가 많기 때문에 어절 단위의 용례 탐색 기능이 필요하다. 품사의 용례 탐색은 품사 제약에 의한 용례 연산 기능을 지원하기 위해 필요하다.

2.1.2 용례 연산 기능

풍부한 용례를 얻기 위해서는 대량의 코퍼스를 이용해야 하지만 태깅이 되지 않은 대량의 코퍼스에서 용례를 추출할 경우 원하지 않는 용례가 많이 발생한다. 따라서 질 좋은 용례를 얻기 위해서는 용례를 탐색한 다음 용례 연산을 이용하여 용례를 걸러내어야 한다. 용례 연산을 위해서는 위치 제한을 포함한 부울린 연산이 필요하다.

예를 들어 '감기는'이라는 어절은 '감:*명사*'의 곡용이나 '감기-:*자동사*'와 '감-:*타동사*'의 활용으로 생성될 수 있다. 그러므로 '감기는'이라는 어절의 용례 중에서 '감-:*타동사*'의 용례만을 선별하기 위해서는 '감:*명사*'의 용례와 '감기-:*자동사*'의 용례를 제거시켜야 한다. 이 때 '감-:*타동사*'와 '감기-:*자동사*'는 목적어의 유무에 의해 판별이 가능하므로 '감기는'의 앞 문맥에서 목적격 조사가 사용된 경우의 용례만을 선택하면 '감기-:*자동사*'의 용례를 제거할 수 있다[1].

2.1.3 용례 정렬 기능

대량의 코퍼스에서 용례를 추출했을 경우 용례 연산을 거치더라도 많은 용례가 얻어진다. 이 경우 용례가 일정한 규칙에 의해 정렬되어 있으면 용례를 분석하는 작업이 용이하다. 예를 들어 용례를 이용하여 '먹다:*타동사*'의 의미를 분류할 경우에는 찾아진 용례를 목적어에 따라 정렬하면 된다.

2.2 통계 정보의 추출

코퍼스에서 얻어 내는 언어 정보들 중에서 용례가 사전 편찬이나 언어 현상 파악에 도움을 주는 반면 통계 정보는 통계적 방법에 기반한 자연어 처리 기법에 주로 이용된다. 이 때 대부분의 통계 정보는 언이 요소의 출현 빈도와 동시 발생 빈도에 의해 구해질 수 있다.

예를 들어 HMM을 이용한 자동 태깅 시스템에서는 상태 전이 확률(STP:state transition probability)과 관측 심볼 확률(OSP:observation symbol probability)를 이용하는데 이들은 각각 [식 1]과 [식 2]에 의해 구해진다.

$$STP = \frac{freq(t_{i-2},t_{i-1},t_i)}{freq(t_{i-2},t_{i-1})} \quad \text{[식 1]}$$

$$OSP = \frac{freq(w_i)}{freq(t_i)} \quad \text{[식 2]}$$

[식 1]에서 $freq(t_{i-2},t_{i-1},t_i)$와 $freq(t_{i-2},t_{i-1})$는 각각 태그 trigram과 태그 bigram의 출현 빈도이며, [식 2]에서 $freq(w_i)$와 $freq(t_i)$는 어절과 태그 unigram의 출현 빈도이다.

그리고 상호 정보(mutual information)이나 T-score는 [식 3]과 [식 4]에 의해서 구해진다.

$$MI = \log \frac{p(x,y)}{p(x)p(y)} \quad \text{[식 3]}$$

$$T\text{-}score = \frac{p(x|z)-p(y|z)}{\sqrt{p(x|z)^2+p(y|z)^2}} \quad \text{[식 4]}$$

[식 3]에서 $p(x)$와 $p(y)$는 단어의 출현 빈도, $p(x,y)$는 두 단어의 동시 발생 빈도이며 [식 4]에서 $p(x|y)$는 Bayes' formula에 의해 단어의 출현 빈도와 동시 발생 빈도를 이용하여 구할 수 있다.

2.3 연어 추출

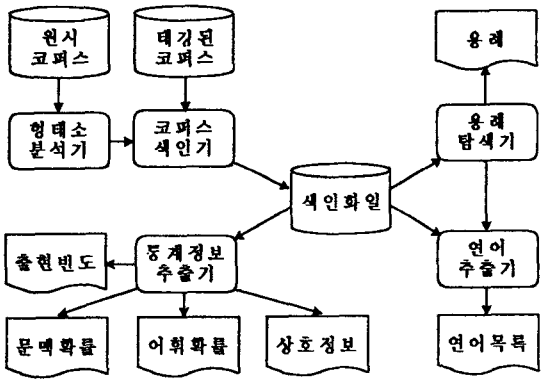
연어란 숙어나 관용구처럼 두 개 이상의 단어가 일정한 형태로 빈번히 사용되는 것을 의미한다. 일반적으로 연어는 단

1 이 때 목적어 유무를 정확히 판별하기 위해서는 구문 정보가 필요하지만 구문 정보를 얻을 수 없는 경우에는 앞에 있는 몇 어절 이내에 목적격 조사가 있는지 살펴봄으로써 유사한 결과를 얻을 수 있다. 그리고 목적어에서 목적격 조사가 생략되는 경우가 있으나 이는 전체 용례의 수에 비해 작으므로 무시할 수 있다.

어 개개의 의미와는 전혀 다른 새로운 의미를 지니게 되는 경우가 많기 때문에 기계 번역에서는 연어 사전이 필수적이다.

제 3 장 KCAT의 구성

이상에서 열거한 기능들을 고려하여 본 논문에서는 [그림 1]과 같은 구조를 지니는 코퍼스 분석 도구인 KCAT(Korean Corpus Analysis Tool)를 구현하였다. KCAT는 코퍼스 색인



[그림 1] KCAT의 시스템 구성

기, 용례 추출기, 통계 정보 추출기, 연어 추출기로 구성된다.

KCAT는 원시 코퍼스나 태깅된 코퍼스를 입력으로 받는다. 원시 코퍼스에 대해서는 형태소 분석 작업을 수행하며 태깅된 코퍼스가 입력되는 경우에는 태깅 정보를 바로 이용한다. 그 결과로 나온 각 어절과 형태소는 문서 내에서의 위치 정보와 함께 코퍼스 색인기로 전달되며 코퍼스 색인기에서는 용례 추출을 지원하기 위해 형태소, 어절, 태그 unigram의 용례 위치를 색인하며 빈도 정보와 통계 정보 추출을 위해 형태소, 어절, 태그 unigram, 태그 bigram, 태그 trigram의 빈도 정보를 갱신한다. 그 결과 언어 정보를 추출할 수 있는 색인 화일이 생성된다.

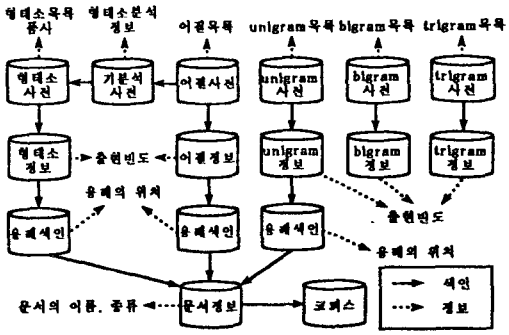
용례 탐색기에서는 형태소, 어절, 태그 unigram의 용례 탐색을 지원하며 탐색된 용례에 대해서는 OR, WITHIN, DIFFERRENCE 연산을 제공한다. 그리고 지정된 위치의 어절을 중심으로 하는 용례 정렬 기능과 함께 특정 문장 성분을 중심으로 한 용례 정렬 기능도 함께 지원한다.

통계 정보 추출기는 언어 요소의 출현 빈도 직접 구해 주며 용례 연산 결과를 통해 동시 출현 빈도를 제공한다. 또한 이들 정보를 이용하여 상태 전이 확률, 관측 심볼 확률, 어휘 확률(lexical probability), 상호 정보, T-score 등과 같은 다양한 통계 정보를 지원해 준다.

연어 추출기는 Xtract에서 이용한 방법을 한국어에 적용시켜 주어진 언어 요소에 대한 연어를 찾아준다. 이 때 언어 요소는 형태소, 어절, 태그 unigram이나 이들의 조합을 가리킨다.

3.1 코퍼스 색인

KCAT에서는 용례 탐색과 통계 정보 추출을 위해서 [그림 2]와 같은 구조로 코퍼스를 색인하였다.



[그림 2] 코퍼스 색인 구조

[그림 2]과 같이 KCAT의 코퍼스 색인 구조는 사전 화일, 정보 화일, 색인 화일로 구성되어 있다. 각 언어 요소의 출현 빈도와 색인 정보에 대한 인덱스는 정보 화일들에 저장되며 이를 참조하기 위해서는 사전이 필요하다. 그리고 형태소, 어절, 태그 unigram에 대한 용례는 용례 색인 화일에 들어 있다. 그리고 형태소 분석 정보 화일은 각 어절에 대한 형태소 분석 정보가 들어 있어 이미 등록되어 있는 어절에 대해서는 중복된 형태소 분석을 피하도록 하였다. 그리고 문서 정보 화일은 코퍼스의 각 문서 화일에 대한 형식적인 정보가 포함되어 있다.

[그림 2]에서 태그 bigram이나 태그 trigram에 대한 용례 색인을 별도로 하지 않는 이유는 태깅되지 않은 대량의 코퍼스를 분석할 경우 bigram과 trigram이 너무 많이 발생하기 때문이다. 예를 들어 연속된 세 어절이 각각 5 개, 8 개, 3 개의 중의성을 가진다고 하면 조합 가능한 태그 trigram의 갯수가 5×8×3=120 개가 되어 한 어절의 처리에 있어서 120 개의 태그 trigram 용례를 색인해야 한다. 따라서 이들을 모두 색인하는 것 보다는 unigram의 용례만을 색인한 후 탐색시 unigram의 용례에 대한 연산을 통해 bigram 및 trigram의 용례를 찾는 것이 타당하다.

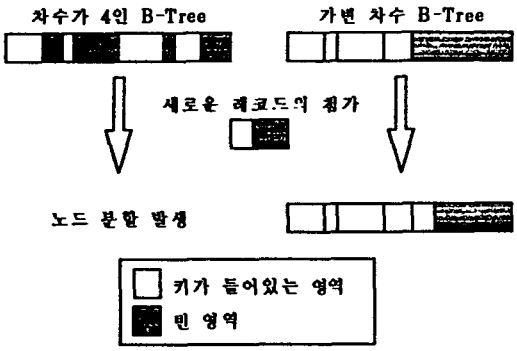
[그림 2]에 나타난 코퍼스 색인 화일의 내부 구조를 사전 화일, 정보 화일, 용례 색인 화일로 나누어서 살펴보면 다음과 같다.

3.1.1 사전 구조

코퍼스 분석 도구에서 형태소 사전은 탐색 작업만이, 그 밖의 사전은 탐색 작업과 삽입 작업이 필요하다. 그러나 한국어 정보 처리 분야에서 사용되는 기존의 사전 구조는 대부분 탐색 효율과 사전의 크기만을 고려해서 설계되었다. 그 결과 사전에 대한 표제어 삽입이 불가능하거나[1][2][7], 삽입할 때마다 사전 전체의 내용을 다시 기록해야 하는 단점이 있어[4]

코퍼스 분석 도구를 위한 사전 구조로는 부적절하다. 이에 KCAT에서는 코퍼스 분석 도구에 적합한 새로운 사전 구조를 설계하였다.

KCAT에서는 코퍼스 분석을 위한 사전의 구조로 변형된 가변 차수 B-Tree(variable order B-Tree)구조를 이용한다. 일반적인 B-Tree구조가 모든 레코드가 고정된 길이를 가지는 반면 가변 차수 B-Tree는 각 노드의 크기를 일정하게 하고 키의 크기에 레코드의 크기가 다르도록 하여 한 노드에 들어가는 레코드의 갯수(차수)를 노드마다 다르게 하였다[12][10]. [그림 3]에서는 가변적인 길이의 키를 가지는 경우에 대해 일반적인 B-Tree와 가변 차수 B-Tree를 비교하였다.



[그림 3] B-Tree와 가변 차수 B-Tree의 비교

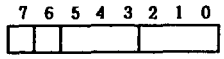
[그림 3]에서 나타나는 것과 같이 이 구조는 가변적인 길이의 키에 대한 색인 구조에서 B-Tree보다 기억 장소를 적게 차지한다. 그리고 B-Tree와 기능이 동일하기 때문에 삽입 및 삭제가 용이하며 디스크 위주의 색인 구조에서 빠른 탐색 속도를 나타내며 참조 국부성이 크기 때문에 형태소 분석을 위한 사전 구조로도 적합하다.

앞에서 언급한 가변 차수 B-Tree는 빠른 탐색 속도와 삽입의 용이성으로 인해 코퍼스 분석 도구를 위한 사전 구조에 적합하다. 그러나 키들이 그대로 노드에 저장되고 하위 노드를 가리키기 위한 포인터가 많이 필요하기 때문에 기억 장소의 측면에서 트라이 사전 구조에 비해 비효율적이다. 이에 KCAT에서는 전위 압축 기법(front compression)과 자식 노드에 대한 포인터의 제거 기법을 이용하여 기억 장소의 효율성을 극대화하였다. 또한 가상 B-Tree 기법을 이용하여 디스크에 위치한 사전의 탐색 속도를 개선시켰다.

전위 압축 기법은 코드가 유사한 키들이 연속적으로 저장되어 있을 경우 각 키에서 앞의 키와 유사한 부분을 제거시켜 기억 장소의 크기를 줄이는 방법이다[9].

전위 압축 기법을 사용하였을 경우 키는 다음과 같이 저장된다. 각 노드 내의 첫 번째 키는 그대로 저장한다. 그리고 두 번째 키부터는 한 바이트로 된 압축 필드와 키의 나머지 부분만이 저장된다. [그림4]는 압축 필드의 구조를 보여준다.

[그림 4]에서 overlap flag는 앞 키와 일치하는 부분이 있는지의 여부를, remain flag는 일치하지 않는 부분의 길이가

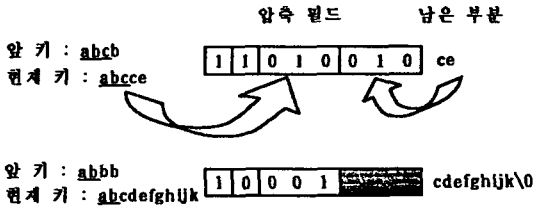


bit(s)	이름
0-2	remain size
3-5	overlap size
6	remain flag
7	overlap flag

[그림 4] 압축 필드

$8(=2^3)$보다 작은지를 나타낸다. 그리고 일치하는 부분의 길이는 overlap size에, 나머지 부분의 길이는 remain size에 각각 들어간다.

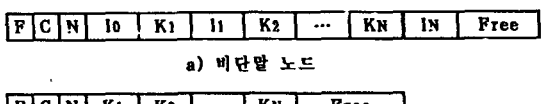
[그림 5]는 전위 압축을 이용한 예를 보여준다.



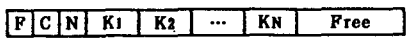
[그림 5] 전위 압축의 예

전위 압축은 키를 압축하거나 복구하는 작업이 간단하므로 시간의 소모가 적으며 압축으로 인해 차수가 증가하기 때문에 사전 전체의 깊이(depth)가 감소하여 오히려 탐색 시간까지도 줄어들게 하는 장점이 있다.

한편 일반적인 B-Tree에서는 모든 노드 내에 자식 노드를 가리키기 위한 (레코드의 갯수 + 1) 개의 포인터를 둔다. 그런데 차수가 큰 B-Tree일 경우 대부분의 노드는 단말 노드이며 단말 노드들은 자식 노드가 없으므로 모두 null 값을 가지고 있다. 따라서 단말 노드들은 자식 노드를 가리키는 포인터가 필요하지 않으므로 이를 제거할 수 있다. 일반적인 B-Tree의 경우 노드 내에 있는 각 레코드의 크기가 일정하므로 자식 노드에 대한 포인터를 제거시킬 수 없다. 그러나 가변 차수 B-Tree의 경우 레코드의 크기가 가변적이므로 자식 노드에 대한 포인터를 제거해도 무방하다. [그림 6]은 가변 차수 B-Tree구조에 자식 노드에 대한 포인터 제거 기법을 적용한 형태소 사전의 노드 구조를 나타낸다.



a) 비단말 노드



b) 단말 노드

F : 빈 영역의 크기
T : 단말 노드 표시
N : 레코드 수
K : 키(형태소)
I : 자식 노드에 대한 포인터
Free : 빈 영역

[그림 6] 형태소 사전의 구조

앞에서 설명되었던 전위 압축이나 단말 노드에서 자식 노드에 대한 포인터를 제거하는 기법은 사전 저장 공간의 크기를 줄이기 위한 것이었다. 반면에 가상 B-Tree 기법은 디스크 중심 사전의 탐색 속도를 향상시키기 위해 사용된 기법이다. 가상 B-Tree 기법은 B-Tree 구조로 되어 있는 인덱스 화일이 디스크에 있을 경우 일부 노드만을 메모리에 가져다 놓고 이용하는 방법이다[10]. 이 기법은 가상 메모리(virtual memory) 시스템에서의 페이징(paging) 기법과 유사하며 각 노드가 하나의 페이지(page)에 해당된다. 일반적으로 B-Tree에 대해 LRU 대체 알고리즘을 이용하면 탐색 시간을 상당히 줄일 수 있다[10]. KCAT에서는 사전 탐색 특성을 고려해서 B-Tree의 각 단계에 대해 하나의 노드만을 메모리에 올리도록 설계하였다. 일반적인 B-Tree의 경우 이 방법이 별로 효과적이지 않겠지만, 형태소 분석을 위한 사전의 경우에는 형태소 분석시 유사한 키를 연속으로 탐색하는 경우가 주로 발생하므로 메모리에 있는 노드의 갯수와 적더라도 탐색 속도는 거의 저하되지 않는다.

3.1.2 정보 화일 구조

KCAT에서 이용하는 정보 화일에는 [그림 2]에서 나타내어진 것처럼 형태소 정보 화일, 어절 정보 화일, 형태소 분석 정보화일(기분석 사전), unigram 정보 화일, bigram 정보 화일, trigram 정보 화일, 문서 정보 화일이 있다[2]. 이들에 대한 접근은 사전 화일을 통해서 이루어지며[3] 모든 정보 화일은 순차 화일 구조로 구성된다.

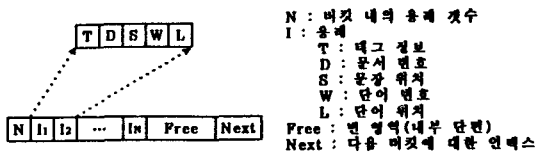
3.1.3 용례 색인 화일 구조

KCAT에서 이용하는 용례 색인 화일에는 형태소 용례 색인 화일, 어절 용례 색인 화일, unigram 용례 색인 화일이 있으며 이들은 모두 같은 형식의 정보를 지닌다. 용례 색인 화일의 각 레코드는 언어 요소의 위치에 대한 정보를 표현하기 위해 문서 번호, 용례가 들어 있는 문장의 시작 위치, 용례가 있는 어절의 번호 및 위치 정보를 가지고 있다. 이 때 용례가 들어 있는 어절의 번호는 용례의 위치 제한 연산에서 사용되며, 태그 정보를 위한 필드를 별도로 가진다.

KCAT는 역 화일(inverted file) 구조를 이용하여 용례 색인 화일을 구성되는데 삽입 및 탐색의 효율을 고려하여 버킷(bucket) 단위의 연결 리스트를 사용한다. [그림 7]은 버킷 단위의 연결 리스트 구조를 이용한 용례 색인 화일의 구조이다.

3.2 용례 추출

이상에서 설명된 코퍼스 색인 구조를 이용하여 KCAT는



[그림 7] 용례 색인 화일의 구조

다음과 같은 용례 추출 기능을 지닌다.

3.2.1 용례 탐색

KCAT는 용례 색인 화일을 참조하여 형태소, 어절, 태그 unirgam에 대한 용례 탐색을 지원한다. KCAT는 형태소 분석 정보나 태깅 정보를 이용하여 각각의 어절을 구성하는 형태소를 구한 다음 이들 형태소들에 대해 용례를 색인하기 때문에 실질 형태소 뿐만 아니라 형식 형태소의 용례도 모두 찾을 수 있다. 어절 및 태그 unigram의 경우에는 전위 절단(prefix truncation), 후위 절단(suffix truncation), 중위 절단(infix truncation), 양쪽 절단(prefix&suffix truncation) 등과 같은 형식의 단어에 대한 용례 탐색을 지원하여 부분 문자열이나 품사(열)에 대한 용례 탐색까지도 지원한다.

3.2.2 용례 연산

KCAT에서는 OR, WITHIN, DIFFERENCE 연산을 지원한다.

OR 연산은 용례 탐색 결과들을 병합하는 연산이다. 용례 색인 화일이 역 화일 구조로 되어 있으므로 OR 연산을 수행하기 위해서는 두 용례 리스트를 병합하는 작업이 필요하다.

목이다. 한 사를 약을 못 [먹어] 죽는 일이야 없겠지만 상황
내보내고, 하루 벌어 하루 [먹고] 살기도 어려운 이들은 어렐
리의 관심은 누가 얼마를 [먹었는가] 등 시시콜콜한 데 있지
음지언정, 명예와 사기를 [먹고] 사는 대다수 선량한 국군장
있다. 그것은 모처럼 큰맘 [먹고] 내린 명단 전면공개라는 파
찬으로 '김'을 가장 즐겨 [먹는] 것으로 나타났다.
심할 때에는 반드시 약을 [먹어야] 한다. </text> <text
통에 빈진 검두에는 약을 [먹어야] 하고 6 개월 이상 끈질기
대 피부과 박윤기 교수는 ["먹는] 두즘약 가운데 이번에 문
보사부의 부작용 경고로 [먹는] 두즘약에 대한 불안이 크다
은 약제를</head> 최근 [먹는] 두즘약인 '그리세오 퓸빈'

[그림 8] '먹-:*동사*'의 용례

WITHIN 연산은 AND 연산과 위치 제한 연산을 동시에 적용하는 것으로 두 용례 요소가 같은 문장 내에서 지정된 거리만큼 떨어져서 나타나는 경우에만 용례로 추출하는 연산이다. 예를 들어 '먹-:*동사*'에 대해 형태소 단위의 용례 탐색을 수행하였을 경우 [그림 8]과 같이 '먹-:*자동사*'의 용례와 '먹-:*타동사*'의 용례가 혼합되어 나타난다. 태깅된 코퍼스의 경우 이들 사이의 구분은 코퍼스 내에 있는 태깅 정보를 이용하면 가능하지만 태깅되지 않은 코퍼스에 대해 용례 탐색 작업을 수행했을 경우에는 태깅 정보가 없으므로 용례 연산을 이용하여 이들을 분리하여야 한다. '먹-:*타동사*'는 대부분의 경우

2 이들 중에서 태그 unigram 정보 화일에는 빈도 정보와 용례에 대한 인덱스가 들어가는데, 실제 구현에서는 정보 화일을 별도로 두지 않고 태그 unigram 사전 화일에 이 정보를 같이 두도록 하였다. 그리고 태그 bigram 정보 화일과 태그 trigram 정보 화일에는 들어갈 정보 또한 빈도 정보 밖에 없으므로 사전 화일이 정보를 모두 수록하였다.

3 문서 정보 화일은 사전 화일이 없이 문서 번호로 바로 접근한다.

목적어를 가진다. 따라서 목적격 조사인 '르', '을', '를' 등이 '먹-:*동사*'의 앞에서 사용된 경우만을 추출하면 전체 용례에서 '먹-:*타동사*'의 용례를 찾을 수 있다. [그림 9]는 '먹-:*동사*'와 <목적격 조사>의 용례를 위치 조건을 1~5로 하여 WITHIN 연산을 수행한 결과이다.

```
특이다. 한 사를 악을 못 [먹어] 죽는 일이아 없겠지만 상황
리의 관심은 누가 얼마를 [먹었는가] 등 시시콜콜한 데 있지
을지언정, 명예와 사기를 [먹고] 사는 대다수 선량한 국군장
찬으로 '김'을 가장 즐겨 [먹는] 것으로 나타났다.</p> <p>
심할 때에는 반드시 약을 [먹어야] 한다.</p> </text> <text
통에 번진 경우에는 약을 [먹어야] 하고 6 개월 이상 끈질기
```

[그림 9] '먹-:*타동사*'의 용례

DIFFERENCE 연산은 용례에 대해 차집합을 구하는 연산으로 주어진 두 용례 리스트에서 일치하는 용례들을 제거시키는 연산이다. [그림 10]은 '먹-:동사'의 용례와 '먹-:타동사'의 용례에 대해 DIFFERENCE 연산을 수행하여 '먹-:자동사'의 용례를 구한 결과의 일부분이다.

```
내보내고, 하루 벌어 하루 [먹고] 살기도 어려운 이들은 어떨
있다. 그것은 모처럼 큰맘 [먹고] 내린 영단 전면공개라는 파
대 피부과 박윤기 고수는 ['먹는] 무좀약 가운데 이번에 문
보사부의 부작용 경고로 [먹는] 무좀약에 대한 불안이 크다
은 각제를</head> <p>최근 [먹는] 무좀약인 '그리세오 풀빈'
```

[그림 10] '먹-:자동사'의 용례

3.2.3 용례 정렬

KCAT는 지정된 위치의 어절을 중심으로 한 용례 정렬 기능과 함께 용례 연산에 사용된 어절을 중심으로 한 용례 정렬 기능을 제공한다.

[그림 11]은 '먹-:타동사'의 용례를 용례 연산 어절—목적격 조사가 들어 있는 어절—을 중심으로 정렬한 결과의 일부분이다. 이 경우 각 문장의 용례 연산 어절은 '간을', '개를', '겁을', '겨자를' 등이다. 처음 세 개의 용례는 '간'을 목적어 취하는 경우의 용례이며, 다섯 번째부터 일곱 번째까지의 용례는 '겁'을 목적어로 가지는 경우의 용례이다. 그리고 가장 아래에 있는 세 개의 용례는 '마음'을 목적어로 취하는 경우의 용례이다. 각각의 경우 '먹-:타동사'의 의미를 보면 목적어에 따라 의미가 결정되는 것을 알 수 있다.

3.3 통계 정보 추출

KCAT는 코퍼스 색인 작업을 할 때 각각의 사전에 출현 빈도가 들어가게 되므로 형태소, 어절, 태그 unigram, 태그 bigram, 태그 trigram의 출현 빈도는 사전에서 직접 추출해 낼 수 있다. 그러나 형태소나 어절의 동시 출현 빈도는 테이블로 유지하기에는 너무 큰 기억 장소를 요구하기 때문에 용례 연산 결과로 얻을 수 있도록 구현하였다.

또한 KCAT에서는 기본적인 언어들의 출현 빈도와 동시 출현 빈도를 이용하여 상태 전이 확률, 관측 심볼 확률, 어휘 확률, 상호 정보, T-score 등을 산출한다.

```
두려워하지 않았다. 간을 [먹기만] 하면 자기의 겨드랑이에
다. 그는 으스대면서 간을 [먹었다.] 어머니의 말대로 곧 겨
를 때려눕히고 간을 꺼내 [먹는] 죽 사람의 겨드랑이에 날개
린이만 혼자 용시 두 개를 [먹었다.] 어색한 침묵의 시간이
미 입술을 본 봉순이 겁을 [먹으면서] 잡힌 팔을 뿌리친다.</
님." </p> <p>소년은 겁을 [먹은]. 소리를 냈으나 매양 당하기
다 해도 백형 자네가 겁을 [먹을] 필요는 조금도 없는 일이어
의 주민들은 울며 겨자를 [먹을] 분이었다.</p> <p>간척사업
다.</p> <p>혜숙은 나이를 [먹을] 줄 모르는 여자였다. 그녀
우습다. 나이를 한 살 더 [먹고도] 이렇게 태연할 수 있다니
. 하준은 나이를 한 살 더 [먹어] 열여덟 살이 되었다.</p> <
시와 매운탕 한 냄비를 다 [먹었고,] 소주 두 병을 비웠다.</
'우리 형제가 다같이 녹을 [먹으면서] 이런 영업을 하면 도리
친지들에 업혀 눈칫밥을 [먹고] 살다가 안동으로 장가들었
여타 하인들은 든을 떠어 [먹는] 것으로 알고 불신하게 된
른 기했다. 하준은 라면을 [먹으며] 문병 계획을 세웠다.</p>
것인가 말 것인가? 라면을 [먹을] 것인가 굶을 것인가? 대학
다. 하준은 마음을 독하게 [먹었다.] </p><p>"너 몇 학년이나
만들겠느냐고 마음을 들려 [먹었다.] </p><p>혜숙은 그녀의
날 어부는 그렇게 마음을 [먹고] 강가에 나갔더니 그토록 많
```

[그림 11] '먹-:타동사' 용례의 정렬

3.4 연어 추출

KCAT에서는 영어권의 연어 추출 도구인 Xtract에서 사용한 방법을 한국어에 적합하게 변형하여 연어를 추출하였다[4]. 이 때 연어의 주 단어(head word)로는 형태소, 어절, 태그 unigram 및 이들을 조합한 형태가 될 수 있도록 하였으며 연어 추출시 형태소와 어절을 분리해서 취급하였다.

연어 추출 과정은 다음과 같다. 연어의 주 단어(head word)를 입력 받으면 주 단어에 대한 용례 추출을 한 다음 주 단어와 같은 문맥에 나오는 단어들의 목록을 만들고 각 단어의 위치별 빈도를 조사한다. 그 다음 과정에서는 동시 발생 단어들 중에서 연어로 사용된 단어만을 선별하기 위해서 *strength*와 *spread*를 조사한 다음 기준치를 넘는 단어들에 대해서 연어 위치를 알아내기 위해서 정점의 위치를 탐색한다. [식 5]는 연어를 추출하는데 사용되는 식이다. 이 식에서 KCAT에서는 *strength*와 *spread*가 각각 1과 10 이상인 단어에 대해 k_1이 5인 경우를 만족하는 위치 j만을 연어로 선택하였다.

$$strength = \frac{freq_i - \overline{freq}}{\sigma}$$

($freq_i$: 단어 i의 발생 빈도,
$\overline{freq}$: 동시 발생 빈도의 평균,
σ : 동시 발생 빈도의 표준 편차)

$$spread = \frac{\sum_{i=1}^{10}(p_i^j - \overline{p_i})^2}{10} \qquad \text{[식 5]}$$

(p_i^j : 단어 i가 위치 j에 출현한 확률,
$\overline{p_i}$: 단어 i의 위치별 출현 빈도의 평균)

$$p_i^j \geq \overline{p_i} + (k_1 \times \sqrt{U_i})$$

(k_1 : 기준치, U_i : 단어 i의 *spread*)

4 Xtract에서 연어를 추출하는 단계는 세 단계로 구성되어 있는데, 이 중에서 세번째 단계에서는 부분 구문 정보를 이용한다. 그러나 KCAT에서는 원시 코퍼스에서 연어를 추출하기 때문에 세번째 단계는 적용하지 않았다.

[표 1]은 '뿐:*의존 명사*'를 주 단어로 하여 형태소 및 어절 단위의 연어를 추출한 결과이다[5].

언어 요소	위치
니라(종결형 어미)	뒤 1 어절
라(종결형 어미)	뒤 1 어절
아(명사)	뒤 1 어절
아니(형용사)	뒤 1 어절
알(타동사)	뒤 1 어절
을(관형사형 전성 어미)	앞 1 어절
아니라(어절)	뒤 1 어절
있을(어절)	앞 1 어절

[표 1] '뿐:의존 명사'에 의해 추출된 연어

제 4 장 실험 및 평가

이 장에서는 지금까지 제안된 방법에 따라 KCAT를 구현하고 실험한 결과를 사전 구조, 용례 색인 구조, 언어 정보 획득 기능의 측면으로 나누어 소개하고 평가한다.

4.1 사전 구조에 대한 실험 및 평가

본 논문에서는 코퍼스 분석 도구를 위한 사전 구조로 가변 차수 B-Tree의 변형 구조를 제안하였다. 약 84,307 개의 형태소를 가진 사전을 이 구조로 구축한 다음 코퍼스에서 추출한 21,301 어절에 대해 형태소 분석을 한 결과가 [표 2]에 나타나 있다. 이 경우 형태소의 탐색 횟수는 총 80,017 회이다.

사전의 위치	디스크	메모리	가상 B-Tree
디스크 접근 횟수(회)	239,601	-	61,899
평균 디스크 접근 횟수 (회/형태소)	2.99	-	0.77
소요시간(초)	642.08	48.72	53.71
평균 탐색 시간(초/형태소)	8.02 E-03	6.09 E-04	6.71 E-04
초당 탐색 횟수(형태소/초)	124.62	1642.39	1489.80
사전을 위한 메모리 사용량 (바이트)	-	685,056	1,536

[표 2] 형태소 사전의 탐색 속도

[표 2]에서 나타난 것처럼 가상 B-Tree 기법을 사용하였을 경우 사전이 메모리에 모두 들어 있을 경우와 비슷한 탐색 속도를 유지하면서도 사전을 위한 메모리의 양을 크게 줄일 수 있음을 알 수 있다. 사전 전체의 크기 685,056 바이트 중에서 정보 화일에 대한 인덱스 필드를 제외한 나머지 부분(키와 자식 노드에 대한 포인터)의 크기는 347,828 바이트로 텍스트로 키를 저장할 경우와 비교해서 약 35.5%의 압축이 이루어졌음을 알 수 있다.

[표 3]은 형태소 사전에 대한 순차 접근 시간을 나타낸다. 순차 접근은 절단된 단어의 탐색시 필요하며 [표 3]에서 보면 가상 B-Tree 기법을 사용하였을 경우 순차 접근을 빠르게 할 수 있음을 알 수 있다.

가상 B-Tree 기법 이용 여부	×	○
순차 접근 시간 (초)	661.52	11.59
디스크 접근 횟수(회)	252,927	1,336

[표 3] 형태소 사전의 순차 접근 시간

[표 4]에서는 제안된 가변 차수 B-Tree의 변형 구조를 다른 사전 구조와 실험 결과를 바탕으로 비교 하였다.

사용 기법	해싱	음절 단위 트라이 (연결 리스트)	음절 단위 트라이 (배열)	가변 차수 B-Tree
속도	빠름	빠름	매우 빠름	빠름
사전의 크기	큼	조금 작음	작음	매우 작음
삽입 삭제	부분적 지원	불가능	가능 (비효율적)	가능 (빠름)
순차 탐색	불가능	가능	가능	가능

[표 4] 형태소 사전의 특성 비교

4.2 용례 색인 구조의 평가

KCAT에서는 버킷 단위의 연결 리스트를 이용한 역 화일 구조를 용례 색인 구조로 사용하였다. 약 70만 어절에 대해 용례 색인을 한 결과 형태소 용례 색인만을 위해 약 28M 바이트가 소요되었다. 실제로 용례 하나를 색인하는 데는 11바이트가 사용되지만 삽입 및 탐색의 효율을 위하여 버킷 단위의 연결 리스트를 이용하였기 때문에 내부 단편이 발생하여 색인 화일의 구조가 커진 것이다[6]. 그러나 이것은 색인된 코퍼스의 크기가 작기 때문에 발생한 결과이며 코퍼스의 크기가 커지면 내부 단편에 의한 기억 장소 낭비가 줄어들게 된다.

한편 이 구조를 이용하여 용례를 색인할 경우에는 한 번의 디스크 읽기와 한 번의 디스크 쓰기만이 필요하므로 효율적이다. 또한 용례 탐색시에는 총

$$\frac{\text{탐색하는 용례의 갯수}}{\text{버킷당 용례의 갯수}(=47)}$$

번의 디스크 읽기를 수행하므로 빠른 용례 탐색을 지원할 수 있다.

5 [표 1]은 태깅되지 않은 코퍼스에 대해 연어를 추출한 결과이다. 때문에 '아니라'라는 어절이 중의적인 분석을 발생시켜 '아:*명사*, '알:*타동사*, '니라:*종결형 어미*'까지도 연어로 추출되었다.

6 약 70만 어절이 색인되어 있는 현재 하나의 버킷에 들어갈 수 있는 용례의 갯수의 절반인 24개보다 많이 출현한 형태소의 수는 4,476 개에 불과하기 때문에 나머지 형태소에 할당된 버킷은 반 이상이 내부 단편으로 낭비되고 있다.

4.3 기능적 측면의 평가

[표 5]에서는 KCAT의 기능을 기존의 한국어 코퍼스 도구들과 비교하여 나타내었다.

	KOCP	KIRT	이창덕 시스템	KCAT
형태소 용례 탐색	제한적 지원	가능	가능	가능
어절 용례 탐색	가능	불가능	제한적 지원	가능
unigram 용례 탐색	불가능	불가능	불가능	가능
용례 연산	가능	불가능	가능	가능
용례 정렬	불가능	불가능	제한적 지원	다양한 지원
통계 정보	제한적 지원	제한적 지원	제한적 지원	다양한 지원
연어 추출	불가능	불가능	불가능	가능

[표 5] 한국어 코퍼스 분석 도구의 기능 비교

제 5 장 결론 및 향후 연구 과제

본 논문에서는 언어 정보 획득을 위한 한국어 코퍼스 분석 도구에 대해 연구하였으며 이의 검증을 위해 KCAT를 구현하였다. 본 논문에서는 언어 정보 분석 도구의 요구에 맞는 새로운 사전 구조를 제시하였으며 용례 색인 구조로 버킷 연결 기법을 이용한 역 화일 구조를 이용하였다. 이는 용례 색인 작업시 삽입이 용이하며 탐색 속도가 빠른 장점이 있었으나 색인된 코퍼스의 양이 적을 경우 내부 단편에 의해 기억 장소가 낭비되는 문제가 있었다. 그러나 이 현상은 색인되는 코퍼스의 크기가 커짐에 따라 자연적으로 해결되는 문제이다. 그리고 기능적인 측면에서는 기존의 코퍼스 분석 도구보다 용례 연산과 정렬이 다양하며 새롭게 연어 및 통계 정보 추출이 가능하도록 설계되었다.

그러나 지금까지의 연구는 태깅되지 않은 코퍼스를 기반으로 하였기 때문에 중의성으로 인해 정확한 언어 정보를 얻을 수 없는 경우가 많았다. 따라서 앞으로는 코퍼스 분석 도구를 자동 태깅 시스템과 결합하는 과제가 남아 있으며 계층화된 코퍼스 관리를 통하여 분야별로 별도의 언어 정보를 추출하는 것이 필요할 것이다. 그리고 연어 추출에 있어서는 한국어에 적합한 연어 추출 방법에 대한 연구가 있어야 할 것이며 코퍼스 분석시 효율 향상을 위해 새로운 색인 구조에 대한 연구도 아울러 진행되어야 할 것이다.

참고 문헌

[1] 권혁철, 채영숙, 이영식, "한글 철자 검사기에서의 사전", 국어학회 학술논문집, 1992

[2] 박찬모, 이종혁, 왕광식, 한국어 기계 사전을 위한 개발 지원 환경의 연구, 최종 연구보고서, 포항공과대학, 1992

[3] 임현재, 한국어 및 영어에 대한 용어색인시스템의 설계 및 구현, 한국과학기술원 석사학위논문, 1990

[4] 이승선, 송주원, 황규영, 최기선, "TRIE 구조를 이용한 한국어 전자 사전을 위한 데이타베이스 인덱스 구조," 한국정보과학회 봄 학술발표논문집, Vol. 21, No. 1, 1994

[5] 이은철, 김성천, 황영섭, 이종혁, "한국어 텍스트 검색 시스템 KIRT의 구현," 한국정보과학회 봄 학술발표논문집, Vol. 19, No. 1, 1992

[6] 이창덕, 김경서, 송만석, "대량의 말뭉치에서의 용례 색인기의 구현," 한국정보과학과 봄 학술발표논문집, Vol 21, No. 1, 1994

[7] 최기선, 비정형 문서 정보검색 엔진 개발, 1차년도 중간 보고서, 시스템 공학연구소, 1993

[8] Kenneth W. Church, Robert L. Mercer, "Introduction to the Special Issue on Computational Linguistics Using Large Corpora," Computational Linguistics, Vol. 19, No. 3, 1993

[9] C. J. Date, An Introduction to Datavase System, Third Edition, Addison-Wesley, 1982

[10] Michael Folk, Bill Zoellick, File Structures, Second Edition, Addison-Wesley, 1992

[11] Geoffrey Leech, Steven Fligelstone, "Computers and Corpus Analysis," Computers and Writtern Texts, 1992

[12] E. McCreight, "Pagination of B* trees with variable length records," Communications of the ACM, Vol. 20, No. 9, 1977

[13] Oxford Computing Service, "MICRO-OCP," Oxford University Press, 1988

[14] Frank Smadja, "Macrocoding the Lexicon with Co-occurrence Knoledge," Lexical Acquisition, 1991

[15] Frank Smadja, "Retrieving Collocations from Text: Xtract," Computational Linguistics, Vol. 19, No. 1, 1993