모듈화된 형태소 분석기의 구현

이운재, 김선배, 김길연, 최기선
한국과학기술원 전산학과/전문용어언어공학연구센터
{wjlee, yuntan, gykim, kschoi}@world.kaist.ac.kr

Implementation of Modularized Morphological Analyzer

Woon-Jae Lee, Sun-Bae Kim, Gil-Yeon Kim, Key-Sun Choi
Department of Computer Science KAIST/KORTERM

요 약

자연언어처리 분야에서 형태소 분석은 가장 기본적인 단계로서 응용 시스템의 목적에 따라 사용되는 형태소 분석기의 수준과 사용 정보가 달라진다. 기존의 형태소분석기의 기능을 다른 목적을 지닌 응용 시스템에서 사용하려 할 때, 분석수준과 사용정보의 이질성으로 인해 변경 또는 확장하는데 많은 어려움이 있다. 이러한 형태소 분석기의 변경과 확장에 대한 다양한 요구를 수용하기 위한 방법으로서 재사용가능한 모듈화된 형태소 분석기의 구현을 제안한다. 모듈화된 형태소 분석기는 구성 요소인 모듈들의 독립성과 재사용성을 보장하기 때문에 확장과 보수가 쉽고, 특정한 요구사항에 대하여 새로운 형태소 분석기를 구현하는데 기존의 모듈들을 사용함으로써 시스템의 개발 시간을 단축시킨다. 본 논문에서는 이러한 모듈들의 사용성을 보여주기 위해 전처리기, 형태소 분석기, 명사 추출기, 태거 등을 하나의 시스템 안에 모듈화된 개념으로 구현하였고, 형태소 분석기는 사전, 음운 변화 처리, 결합 검사, 분석 알고리즘 등을 모듈화하여 재사용할 수 있다는 것을 보여준다.

1. 서론[1)]

대부분의 자연언어처리 시스템들은 여러 개의 구성요소로 이루어져 있다. 예를 들면 기계번역기는 형태소 분석기, 구문/의미 분석기, 변환기(source language에서 target language로의 단어/구문 변환), 구문/형태소 생성기 등의 요소를 포함한다. 이러한 구성 요소들은 더 작은 단위로 분할될 수 있는데, 형태소 분석기의 경우는 사전, 결합규칙, 음운변화규칙, 분석 알고리즘과 내부 구조(chart)로 이루어진다.

이러한 자연언어처리 시스템은 하나의 목적을 위하여 만들어지기 때문에 각 시스템의 구성 요소들은 전체 시스템의 목표를 이루기 위한 가장 효율적인 방법으로 설계되는 것이 일반적이다. 그러나 한가지 목적을 위한 가장 효율적인 시스템의 설계가 다른 목적으로 사용되거나 확장 또는 변경되는 경우에는 비효율적인 설계가 될 수 있다. 예를 들면 색인을 위해 만들어진 형태소 분석기가 효율적인 색인을 위하여 명사를 중심으로 품사와 사전이 만들어지고 분석 알고리즘이 설계되었다면, 이 형태소 분석기를 태깅에 사용하기 위해서 확장하는 것이 거의 불가능한 경우가 있다. 반대로 태깅을 위해 만들어진 형태소 분석기는 색인에 사용하는 경우 분석의 과다함으로 오히려 색인어 추출의 정확도를 떨어뜨리게 되거나 형태소 분석기와 태거를 같이 사용하여 색인어를 추출하는 경우 속도가 느려지는 단점을 가지게 된다.

자연언어 처리 시스템 중에서 특히 형태소 분석기는 이미 실용화 단계를 넘어섰고 다양한 요구에 의한 다양한 시스템들이 만들어지고 있어서 활용성 향상과 확장에 대한 요구를 수용하는 방법이 필요한 시점이다. 결국 이러한 요구를 쉽게 처리하는 방법은 형태소 분석기를 모듈화하는 것이다. 모듈화는 형태소 분석 시스템의 각 구성 요소를 기능에 의해 개념적으로 분리하는 것뿐만 아니라 실제로 분리된 모듈로 설계/구현하는 것을 의미하며 각 모듈의 독립성을 보장하여 재사용이

1) 본 연구는 제 1회 형태소 분석기 및 품사 태거 대회(MATEC '99)의 일환으로 수행되었다.

가능하도록 하는 것이다.

본 논문에서는 자연언어처리의 다양한 응용 시스템 개발에 필요한 구성 요소의 재사용을 위하여 모듈화의 개념을 정리하여 제안하고, 모듈화된 형태소 분석기의 예를 통하여 모듈화의 장점을 설명하고자 한다.

2. 관련 연구 및 문제 정의

모듈화의 개념은 여러 분야에 사용되고 있고 자연언어처리 분야에서도 자연스럽게 모듈화의 개념이 도입되어 있으나, 자연언어처리 분야에서 모듈화 자체에 대한 연구는 많지 않다. 형태소 분석, 태깅, 구문 분석을 순차적으로 실행하는 pipelined architecture나 라이브러리 형태로 제공되는 HAM(Hangul Analysis Modulfunction)[2], MORAN-DCP TOOLKIT[3], LimaTK[4] 등은 모듈화의 방법들이다. 최근에 논의되고 있는 TagSet 및 corpus의 표준화와 matec99와 같은 평가 대회 역시 자연언어처리 시스템의 모듈화에 직간접적인 영향을 미치고 있다.

전산학 분야에서 모듈이라는 단어는 독자적인 기능을 가진 교환 가능한 구성 요소를 말한다. 기존의 자연언어처리 시스템들은 모두 '개념적으로' 모듈화 되어 있다고 할 수 있다. 어떠한 시스템도 설계시에 내부의 구성 요소들을 구분하지 않고 프로그램을 만드는 경우는 없기 때문에 대부분의 시스템에서 모든 구성 요소들은 독자적인 기능성을 가지고 있다. 그러나 구현된 모든 시스템들이 모듈을 분리해 재사용할 수 있는 것은 아니다. 따라서 모듈에 대한 정의는 재사용을 고려하여 다음과 같은 세 가지 관점으로 나누어 볼 수 있다.

1. 설계시의 모듈(design-time module)
   개념적으로 독자적인 기능성을 가진 구성 요소들
2. 구현시의 모듈(implementation-time module)
   재사용 가능한 프로그램의 구성 요소들
3. 실행시의 모듈(run-time module)
   하나의 시스템을 구성하는 교환 가능한 프로그램들과 데이터들

설계시의 개념적인 모듈화만 이루어진 시스템은 실제 구현시에는 여러 구성 요소들이 상호 밀접한 의존 관계를 가지는 경우가 많다. 이러한 복합적인 시스템의 구조는 상호 의존관계 때문에 한가지 구성 요소를 바꾸고자 할 때 다른 구성 요소들도 따라서 바뀌어야 하는 문제를 가지고 있다. 그러나 속도의 효율성을 높일 수 있고, 쉽게 구현이 가능하다는 장점 때문에 초기 설계에 주로 사용된다.

구현시의 모듈화를 고려한 시스템은 한 시스템의 목적에만 부합되는 특수한 구성요소가 아닌 일반성을 가지는 구성요소로 이루어진다. 예를 들면 형태소 분석기의 사전 구조는 잘 알려져 있는 dbm을 포함하여 대부분의 경우 라이브러리화된 경우가 많다. 이렇게 모듈화된 시스템은 확장과 보수가 쉽고 재사용이 가능하다는 장점을 가지고 있으나, 경우에 따라 속도가 느리거나 크기가 커지는 단점을 가질 수 있다.

전통적인 자연언어 처리 방식의 하나인 pipelined architecure는 실행시 모듈화의 한가지 방법이다. 또한 형태소 분석기 등에서 사전이나 음운 변화 규칙과 같은 데이터를 분석 엔진으로부터 분리하는 것도 실행시점의 모듈화 방법이라고 할 수 있다.

자연언어처리 시스템의 모듈화의 목적은 재사용에 있다. 따라서 설계시 모듈의 독자적인 기능성은 구현시 모듈의 독립성으로 나타나야 한다. 즉, 모듈의 재사용성을 보장하기 위해서는 시스템을 구성하는 모든 모듈들을 분리 가능한 실행 프로그램 혹은 데이터 혹은 라이브러리로 구현하여 모듈 상호간의 의존성을 배제해야 한다.

모듈이 독립성을 가지기 위해 고려해야 할 사항으로 첫째는 모듈이 단독으로 한가지 기능을 수행해야 한다는 것이다. 만일 두 개의 모듈 A와 B가 함께 한가지 기능을 수행한다면 이 두 모듈은 하나의 모듈로 묶어 주어야 한다. 또한 하나의 모듈이 두 가지 이상의 기능을 한다면 이 모듈은 각각의 기능으로 나누어주어야 한다.

둘째는 모듈간 데이터의 공유를 없애야 한다는 것이다. 아무리 잘 설계된 시스템이라 하더라도 데이터의 공유를 완전히 없애기는 어렵다. 데이터의 공유를 줄이기 위해서는 프로그램에서 여러 모듈이 공유하는 전역 변수의 사용을 피해야 한다. 또한 하나의 모듈이 사용하는 전역 변수라 하더라도 사용하지 않는 것이 모듈의 독립성을 유지하는 방법이다. 예를 들어 C 언어의 FILE이라는 데이터 구조는 파일을 다루는 함수들이 공유하는 전역 변수들을 묶어 지역 변수화 하는 방법이다. 이

---

2) http://ham.hansung.ac.kr/

3) http://www.ibase.co.kr/

4) 일본의 "[언어자원의 공유와 재이용] 심포지엄"에 발표된 山下達雄의 논문 "형태소 분석 시스템의 기능분할과 재이용을 향하여"에서 본 논문과 유사한 모듈화의 주장을 볼 수 있다.
http://cactus.aist-nara.ac.jp/~tatuo-y/ma/
http://www.etl.go.jp/etl/nl/sympo99/

렇게 함으로써 하나의 프로그램이 동시에 여러 개의 파일을 다룰 수 있게 된다.

셋째는 모듈의 입출력이 명확하고 단순해야 한다는 것이다. 일반적인 경우 모듈의 입력과 출력을 명확하다. 예를 들어 다음과 같은 음운 변화를 처리하는 함수가 있다고 가정할 수 있다.

p(input,output)

여기에서 input은 문서에 나타난 표층 문자열이고 output은 음운 복원에 의해 만들어진 심층 문자열이다. 이때 input과 output은 특정 시스템에서만 사용되는 특수한 구조를 가져서는 안 된다.

경우에 따라서는 이 함수가 특정 입력에 대하여 내부적으로 공유 데이터인 chart를 수정하는 일이 발생하는 경우가 있을 수 있다.

```
p(input)
{ .....
  update chart procedure
  .....
}
```

이런 경우는 최소한의 독립성을 유지하기 위해 다음과 같이 고쳐져야 한다.

```
prule(chart, input)
{
  p(input,output);
  update_chart(chart, output);
}
```

이렇게 함으로써 적어도 p() 함수는 독립성을 유지 할 수 있게 되는 것이다. 만일 음운 복원을 하는 함수가 1개의 결과를 내주는 것이 아니라 임의의 개수의 결과를 만들어 낸다면 함수의 인수로 결과를 받을 수는 없다. 따라서 다음과 같이 고쳐질 수도 있다.

```
p(chart, input)
{ ....
  update_chart(chart, output1);
  ....
  update_chart(chart, output2);
  ....
  update_chart(chart, output3);
}
```

이와 같이 데이터의 공유를 최소화하고 입출력을 명확히 함으로써 최소한의 독립성이 유지된다. 이 경우 완전히 라이브러리화되지는 못하겠지만 최소한의 수정으로 다른 시스템에 이식할 수 있는 재사용성은 보장된다.

모듈이 재사용성을 높이기 위해서는 모듈 설계시 다음과 같은 사항들이 고려되어야 한다.

1. 일반화

모듈의 기능이나 입출력 데이터가 어느 특정 용도에 특화되지 않고 다양한 응용에 이용 가능하도록 설계되어야 한다. 예를 들면 사전 구조는 한국어뿐만 아니라 영어나 일본어 등을 저장할 수 있는 구조여야 한다.

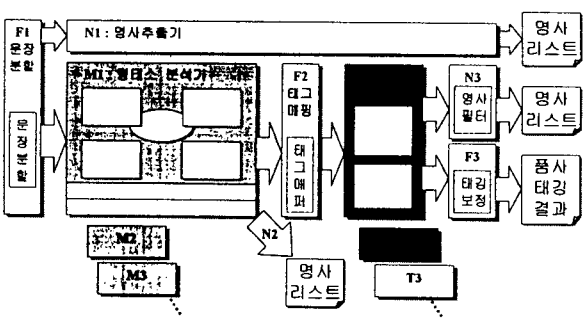
2. 단순화

모듈의 단순화는 프로그램의 단순화만을 의미하는 것이 아니라 모듈의 기능의 단순화를 의미한다. 하나의 모듈은 가장 작은 단위의 기능으로 나누어져야 일반성을 유지할 수 있을 뿐만 아니라 프로그램의 보수와 확장이 쉬워진다.

3. 문서화

모듈화의 목적 중 가장 중요한 것은 재사용이기 때문에 모듈을 참조하거나 변경하는 경우에 대비하여 반드시 문서화가 필요하다.

3. 모듈화된 형태소 분석기



[그림 1] 모듈화된 형태소 분석기와 태거의 구성

[그림 1]은 모듈화된 형태소 분석기와 태깅 시스템의 전체적인 구성도이다. 그림에서 각 상자들은 모듈을 의미한다. 다른 모듈에 포함되어 있지 않은 모듈들은 모두 실행시 교환가능한 'run-time module'이라고 할 수 있으며, 다른 모듈의 내

부에 포함된 모듈들은 구현시에 포함되는 재사용 가능한 'implementation-time module'들과 재사용 불가능한 몇 개의 'design-time module'들로 되어 있다.

M1은 단순화된 형태소 분석기의 구현이며 실험용으로 만들어 졌다. M2는 다단계 음운 변화 모델을 테스트하기 위하여 만들어 졌으며, M3는 색인용으로 빠른 사전 검색을 위해 만들어졌다.

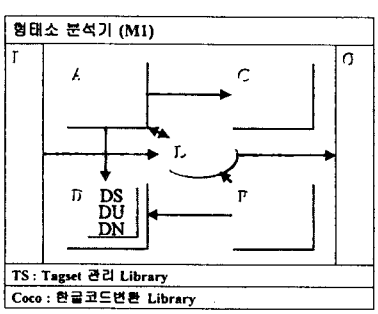
T1, T2, T3는 태거 모듈들이다. F1, F2, F3는 필터 모듈들로 전처리나, 후처리를 담당한다. N1, N3는 명사 추출기 모듈이고 N2는 형태소 분석기의 명사 추출 기능이다.

3.1. 태깅 시스템과 교환 가능한 sub system들

[그림 1]에서 태깅 시스템의 경우 F1-> M1-> F2-> T1-> F3의 순서로 진행되는 파이프라인 구성을 보여준다. 여기에서 M1과 T1은 태깅을 위한 필수적인 구성 요소이고, F1, F2 ,F3는 비필수적인 구성요소이다. M2와 M3는 M1을 대신할 수 있는 실행시 모듈이라고 할 수 있으며 T2와 T3는 T1을 대신할 수 있는 실행시 모듈이다.

T1은 어절 단위 bi-gram 정보를 이용하는 단순한 태깅 프로그램으로 230 line밖에 되지 않는다. T2는 최대 엔트로피 모델을 사용하여 구현하였으며, T3는 tri-gram을 사용하는 모델을 가정한 것으로 구현되지 않았다.

3.2. 형태소 분석기와 sub module의 재사용



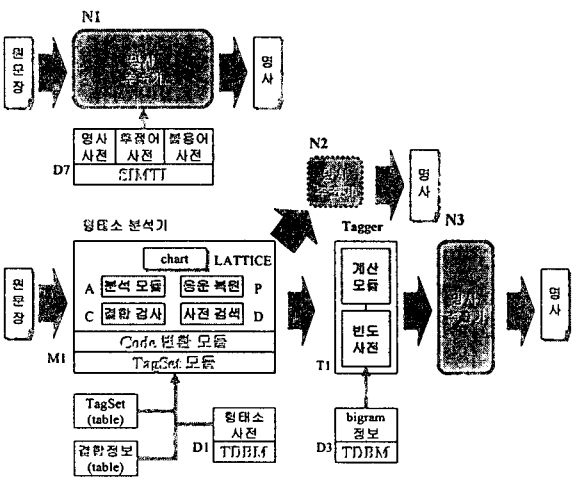
모듈	기능
I	입력
A	분석
D	사전검색
P	음운변화
C	결합정보
L	분석구조체
O	출력
DS	시스템사전
DU	사용자사전
DN	숫자사전

[그림 2] 형태소 분석기의 구성

[그림 1]의 M1, M2, M3는 모두 형태소 분석기로서 분석의 핵심이 되는 알고리즘은 다르지만 사전과 같은 데이터는 공유할 수 있다. [그림 2]에 나타난 M1의 D, C, TS, Coco 와 같은 모듈은 M2에서도 그대로 사용된다.

3.3. 명사 추출기

[그림 1]과 [그림 3]의 N1은 형태소 분석을 하지 않고 명사 사전만을 이용하여 명사를 추출하는 방법으로 속도가 빠르고 구현 방법이 단순하다. N2는 형태소 분석기의 기능으로서 최장일치, 미등록어 처리기능에 명사 출력 기능을 추가하여 얻어지는 명사추출기를 보여준다. N3는 형태소 분석과 태깅을 거쳐 보다 정확한 결과를 얻고자 하는 경우에 사용될 수 있다.

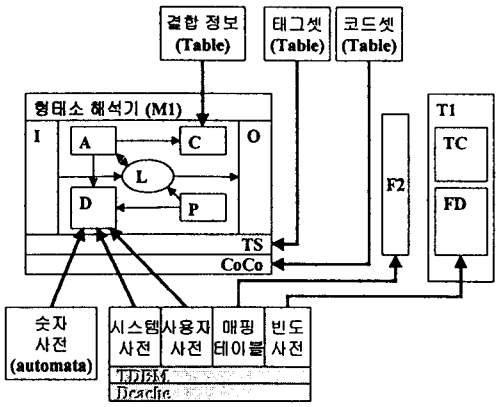


[그림 3] 명사 추출기의 구성

색인어를 추출하기 위한 명사 추출기의 경우 응용 시스템에 따라서 정확도를 요구하는 시스템과 속도를 요구하는 시스템으로 나누어 질 수 있는데 데이터의 양이 적고 정확한 결과가 필요한 논문 등의 색인에는 N3가 적합하고 데이터의 양이 많아서 정확한 결과보다는 빠른 속도가 요구되는 Web Page 검색과 같은 시스템에서는 N1이 효율적이라고 할 수 있다. N2는 N3보다 빠르고 N1보다는 정확한 결과를 얻기 원하는 경우에 사용될 수 있다.

3.4. 데이터와 저장구조

형태소 분석기에는 사전을 비롯하여 여러 가지 지식들이 데이터로서 사용된다. [그림 4]는 형태소 분석기에 사용되는 데이터와 그것을 다루는 모듈들에 대한 구성을 보여준다. 데이터가 프로그램 안에 포함되어 있지 않고 별도의 파일로 분리되어 있는 것은 프로그램의 일반화의 관점에서 중요하다. 예를 들어 코드변환기에서 코드셋(또는 코드 변환 테이블)이 분리되어 있지 않은 경우, 이 코드변환기는 이미 정해진 몇 개의 코드간의 변환만이 가능하다. 그러나 프로그램에서 코드셋이 분리되어 있는 경우 있는 경우 이 코드변환기는 코드셋 데이



[그림 4] 데이터의 분리와 공유

터 파일을 바꾸어 주면 새로운 코드변환기가 되는 것이다. (아래 그림에서의 코드셋은 실제로는 분리되어 있지 않다.) 마찬가지로 결합정보 태그셋 등이 분리되어 있음으로 해서 이 형태소 분석기는 태그셋의 변경이나 결합정보의 변경에 쉽게 적응 할 수 있으며 심지어는 데이터 파일들을 모두 바꿈으로써 다른 언어를 처리할 수 있는 시스템이 될 수도 있다.

M1의 사전 검색 모듈 D는 세 부분으로 이루어져 있다. 시스템 사전 검색과 사용자 사전 검색, 그리고 숫자 인식기이다. 이렇게 함으로써 숫자 처리를 사전 검색과 동일한 관점에서 처리 할 수 있으며 형태소 분석기의 기본 사전 내용은 변경하지 않고 사용자가 고유명사 사전 등을 쉽게 관리할 수 있다. 또한 이러한 사전 검색의 구조는 확장성을 보장한다. 예를 들면 숫자 사전과 유사한 automata에 의해 처리될 수 있는 특수문자 처리 모듈을 추가하거나 응용 분야에 따라 각 분야별 사전을 추가해 사용하는 것이 가능하다.

M1의 사전 구조로 사용된 라이브러리 TDBM[5](TRIE based DBM)과 DCACHE(disk cache module)는 F2의 매핑 테이블과 F3의 빈도 사전에도 그대로 사용된다. TDBM은 원래 형태소 분석을 위해 고안된 사전 구조로서 형태소 분석기를 위한 함수들과 일반적인 데이터를 다루는 함수들과 색인용 함수들로 나누어져 있다. 시스템 사전과 사용자 사전은 형태소 분석용 함수들에 의해 다루어지고 매핑 테이블과 빈도 사전은 일반 데이터를 처리하는 함수들에 의해 검색된다.

[그림 3]에 나타난 SIMTI[6](SIMple Trie Index) 구조는 N1의 사전(D7)과 [그림 1]의 M3의 사전의 구조로 사용되며, 형태소 분석기 내부 구조에서도 임시 저장 구조로 이용된다. ([그림 5])

3.5 모듈화의 장점

모듈화된 형태소 분석/태깅 시스템의 장점은 첫째 태깅 시스템과 같이 파이프라인 구조에서 구성요소들을 응용에 따라 선택적으로 실험하거나 상호 비교하거나, 또는 T1, T2, T3를 동시에 실행한 결과를 이용하여 그중 좋은 결과를 선택할 수 있다는 점이다. 둘째로 M1, M2, M3 형태소 분석기는 과거에 새로운 모델을 실험하기 위하여 형태소 분석기를 새로 만들거나 거의 새로운 형태소 분석기의 구현이 불가능한 것에 대하여 대부분의 모듈을 공유함으로써 새로운 알고리즘을 실험하기 위한 새로운 형태소 분석기의 구현이 단순해진다.

4. 모듈

4.1. 형태소 분석 알고리즘 -- M1-A 모듈

4.1.1. 분석을 위한 내부 저장 구조 -- chart

형태소를 효율적으로 분석하기 위해서는 형태소 분석의 중간과정과 최종 분석 결과가 저장되는 내부 저장 공간이 필요하다. PC-KIMMO와 같이 별도의 내부 저장 구조 없이 형태소 분석을 하는 경우는 backtracking을 하게 되고 실행 속도가 느려진다. 대부분의 형태소 분석기에서 부분 분석을 공유하는 chart를 이용하는데, [김성용 1987]의 CKY table, [김덕봉 1993]의 DDAG, [이상호 1994]의 격자구조가 모두 여기에 속한다.

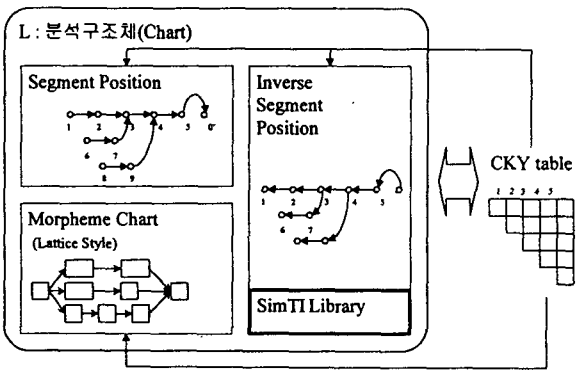
본 논문의 M1에 사용된 chart는 Lattice형태의 구조를 사용하는데 개념적으로는 [이상호 1994]의 격자 구조와 같다. Lattice 형태의 chart는 [김성용 1987, 이은철 1992]의 CKY table에서 index에 해당하는 부분을 segment position이라는 이름의 linked list 형태로 바꾸고 table의 element들을 lattice구조로 바꾸어 link로 연결한 형태이다.

[그림 5]에서 작은 원으로 표시된 것이 segment position의 linked list이고 작은 사각형으로 표시된 것이 lattice 구조를 이루는 morpheme들이다. segment position은 초기에 시작 위치로부터 끝 위치까지의 하나의 path만이 있었는데 음운복원을 통하여 새로운 path를 이루는 segment position들이

5) TDBM은 처음에 형태소 분석기를 위한 TRIE 구조의 사전으로 만들어졌으나 후에 일반적인 사전을 지원하기 위해 dbm 형식의 함수를 제공하면서 TDBM이란 이름을 정했다. dbm과 유사한 기능을 하면서 대용량의 데이터에 대해 빠른 검색 속도가 보장된다.

6) SIMTI는 TDBM의 단순화된 형태로 TDBM이 disk-base로 동작하는데 비해 SIMTI는 모두 memory로 올려져 실행되기 때문에 검색 속도가 빠르고 사용이 간편하다.

추가되었다. segment position들이 Tree의 역링크로 되어 있기 때문에 새로운 segment position을 추가하는 것이 쉽지 않다. Inverse Segment Position은 새로운 segment position을 추가하는 경우를 관리하기 위하여 사용된다.



[그림 5] 형태소 분석기의 부분 분석을 공유하기 위한 Lattice style chart

4.1.2. chart를 이용한 분석 알고리즘

입력된 하나의 어절을 형태소 분석하는 과정은 다음과 같이 설명할 수 있다.

- 주어진 어절에 대하여 chart 초기화
  length= init_chart(word);
- 형태소 분석
  Ma(1,length);
- 형태소 분석 알고리즘

```
Ma(from,to)=
    if(from > to) then "END"
    else for all i (from <= i <= to)
            Connect(Dict(from,i),Ma(i+1,to))
```

이때 chart 구조 안에서 index i는 segment position의 link상의 위치이고 from과 to 사이에 존재한다. 그러나 index i는 from에서 to로 가는 초기의 path상의 것만은 아니다. 음운 변화를 복원하는 과정에서 만들어지는 *from -> to* 의 새로운 path가 존재하기 때문에 i는 from과 to의 사이에서 음운 복원에 의해 만들어진 path상의 index이기도 하다.[7]

이 프로그램의 진행에서 chart의 구성 요소인 morpheme과 segment는 state를 가지고 있어서 한번 분석한 부분은 두 번째는 분석하지 않고 연결 가능성만을 검사한다.

실제로 구현된 형태소 분석 알고리즘은 다음과 같다.

```
- ma(word)
  // 주어진 word에 대하여 chart를 초기화한다.
      segment position을 만들고 link를 연결한다.
      segment[0]에는 morpheme[0]를 연결하고
      morpheme[0]에는 "iwg[8]"라는 품사를 저장한다.
  // 형태소 분석
      r=anal_chart(0)
      return r;

- anal_chart(i)
  // morpheme[i] 다음에 분석할 Segment Position
      from=morpheme[i].next_seg;
  segment[from]에서 시작되는 모든 문자열에 대하여
      사전 검색
      음운 복원과 사전 검색
      새로운 chart[x]를 segment[from]에 등록
  segment[from]에 연결된 모든 chart[x]에 대하여
      r=anal_chart(x)
      if(r>0)
            if(con_check(i,x) ==OK)
                  connect(i,x);
                  최장일치 처리;
  return morpheme[i].connection_count
```

4.1.3. 최장일치

최장일치 기능은 M1-A의 부가적인 기능이다. 초기에 주어진 옵션에 따라 1개의 형태소 분석이 성공한 경우 더 이상 분석을 하지 않으면 되는 것이다. 이때 분석의 순서는 단어의 길이가 긴 morpheme을 우선 분석하는 방식이어야 한다.

최장일치 옵션이 있는 경우 위의 알고리즘에서 최장일치 처리 부분에 다음과 같이 한 줄을 추가하면 된다.

```
    if(최장일치) return 1;
```

4.1.4. 미등록어 처리

7) [그림 5]에서 보여지는 segment position의 link는 어절의 뒤쪽을 root로 하고 link의 방향은 어절의 앞에서 뒤쪽으로 향하는 역링크 Tree 구조의 모습을 보인다.

8) iwg는 inter word gab이라는 의미로 단어의 시작과 끝을 나타내는 품사의 역할을 한다. 예를 들어 동사 어간은 어절의 처음에 사용될 수 있지만 어절의 끝에 사용될 수는 없다는 것을 다음과 같이 표시한다.

```
connection("iwg","pvg")=1
connection("pvg","iwg")=0
```

형태소 분석의 결과가 0이면 미등록어 분석을 시작한다.

- 미등록어 처리를 위한 초기화를 한다.
  // segment position을 따라가면서
  from=1;
  for(to=1;segment[to]!=0;to=segment[to].next)
      path(from,to)에 해당하는 문자열을 사전에서 찾은 것으로 하고 "unk"품사를 할당한다.
- 다시 형태소 분석을 실행한다.
  r=anal_chart(0)

4.2. 음운 변화 처리 - M1-P 모듈

4.2.1. interface와 기능

음운 변화 처리 모듈의 실행은 다음과 같이 함수를 호출하는 방식으로 이루어진다.

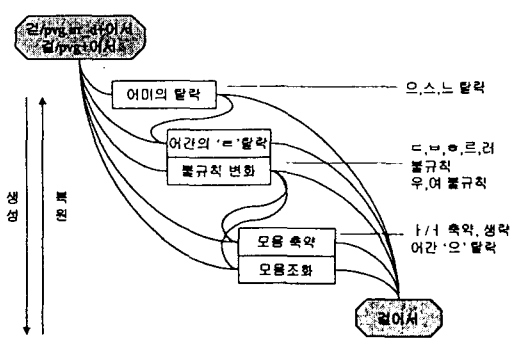
prule(int from, char *prev, char *str)

이 함수는 주어진 문자열 str에 대하여 음운 변화에 대한 복원 규칙을 적용하여 두 개의 문자열 front와 back으로 나누고 chart를 수정하는 함수를 호출한다. 이때 front는 사전 검색을 해야할 부분이고, back은 다시 형태소 분석을 해야 하는 부분이다.

phone_change(from, front, back, front_tag_type, back_tag_type, 불규칙_정보);

4.2.2. 음운 변화 처리 방식

음운 변화의 처리 형태는 생성의 역 과정을 거치게 된다. 크게 나눠 보면 어미의 탈락, 어간의 탈락과 같은 자동적 변화와, 불규칙 용언에 의한 불규칙 변화, 모음조화 및 축약과 같은 선택적 변화로 나눌 수 있다. 각 변화는 유사한 것들끼리 하나의 함수로 묶여, 독립적으로 일어나는 변화의 처리와 연쇄적으로 일어날 수 있는 변화에 대한 처리가 가능하게 하였다.



[그림 6] 음운 변화 모델

예를 들어, 어간에서 'ㄹ' 탈락이 일어나고 어미에서 '으' 탈락이 일어난다고 할 때, 'ㄹ' 탈락을 처리하고 있는 함수에서는 탈락이 일어났을 경우의 처리를 한 후, 어미의 '으' 탈락을 처리하는 함수를 불러 처리를 해주게된다. 또한, 이와는 독립적으로 원래 문자열에 대해, '으' 탈락이 일어났는지를 다시 검사하여, 최대한 가능한 모든 변화에 대한 처리를 시도하고 있다.

나니 --> 날+니
--> 날+으니
나니 --> 나+니
나니 --> 나+으니

음운 변화의 처리는 간단한 패턴매칭으로 처리 할 수 있도록 설계를 하고 있다. 사용되는 정보는 연속된 3개이하의 음운에 대한 정보만 사용한다. 가령 'ㄹ'탈락을 처리하는 경우를 보게 되면, 'ㄹ'이 {'ㄴ','ㅂ','ㅅ','오'}와 만나서 탈락을 하게 되는데, 이를 복원해 주기 위해, 모음의 뒤에, {'ㄴ','ㄹ','ㅂ','ㅅ','오'} 등이 오는 지를 검사하여 매치가 되면 종성 'ㄹ'을 삽입해 주는 것이다. 즉, '사니'와 같은 경우 '살+니'의 형태로 음운의 복원이 일어나게 된다.

3개의 음운만을 살펴보고 처리를 하는 경우, 'ㄹ'탈락의 경우를 보더라도, '갈으니'와 같은 경우 '갈을+니'의 형태로 복원을 하는 경우가 생기고, 어간에서의 'ㅡ'탈락, 'ㅏ/ㅓ'탈락과 같은 경우는 모음하나만을 보고 복원을 하기 때문에 'ㅏ','ㅓ'만 들어가면 복원을 해 주게 된다. 하지만 이런 과분석의 경우 거의 대부분을 형태소 분석기에서 결합정보와, 사전정보 등을 사용해 처리를 할 수 있어, 음운 정보의 처리는 간단한 규칙만을 사용하여 처리를 하고 있다.

음운 변화를 검출하기 위해 필요한 데이터들은 배열에 저장이 되어 사용이 되는데, 다음과 같은 형태로 저장되고 있다.

```
char *pset[MAXPSET][2] = {
/* ㄹ탈락*/
{"l11","ㅏㅑㅓㅕㅗㅛㅜㅠㅡㅣㅐㅒㅔㅖㅚㅟㅢㅓㅏㅓㅐㅔ"},
{"11","ㄴㄹㅁㅂ^ㄴ^ㅅ"},
{"r11",""},
/* ㅡ탈락*/
{"l12",""},
{"12","ㅏㅓ"},
{"r12",""},
/* ㅏ 탈락 */
```

{"l13",""},
{"13","ㅏ"},
{"r13",""},
...
{NULL,NULL}
}

위의 데이터에서 'ㄹ'탈락을 발견하기 위한 규칙에는, "l11"; "11", "r11"이 규칙의 이름으로 사용되는데, "11"이란 숫자는 규칙의 ID가 되고, l,r,{NULL}을 앞에 붙여 각각 현재 보고있는 음운의 왼쪽음운, 오른쪽음운, 현재음운이 속해야 하는 패턴 집합임을 나타내었다.

즉, 'ㄹ'탈락의 경우 'ㄴ','ㄹ','ㅁ','ㅂ',초성 'ㄴ', 초성 'ㅅ' 앞에 모음이 나타나게 되면, 이 규칙에 속하는 변화라는 것을 나타내며, ""과 같이 패턴 집합이 NULL인 경우는 DON'T CARE로 처리를 하도록 하고 있다.

그밖에 간단한 것들, 즉, 집합으로 표현되지 않고, 하나만 속해 있거나 한 경우들 중 일부는 배열에 집어넣지 않고 직접 비교를 하고, 모음조화의 처리를 위해, 양성모음과 음성모음 등도 따로 배열에 집어넣어 사용하고 있다.

현재는 positive 규칙만을 적용시켜 복원에 이용하고 있지만, 성능의 개선을 위해서는 치명적일 수 있는 잘못된 복원을 차단 할 수 있도록 negative 규칙을 추가해 주는 작업이 필요할 것이고, 보다 정확한 결과를 얻으려면, 보다 많은 음운을 보고 결정 할 수 있도록 규칙을 확장해 주는 것도 필요할 것이다.

그리고 생성의 역 과정이 복원에 사용되었듯이 생성 모듈을 사용해 복원된 음운 변화 형태소들을 복원했을 때 원래의 문자열이 나오는지를 검사한다면 올바른 음운변화의 복원이 일어났는지에 대한 검증이 가능할 것이다.

4.2.3. M1에서의 원형 복원 특성

'으'를 매개모음으로 볼 것인가 어미의 일부로 볼 것인가에 대하여는 여러 가지 견해가 있을 수 있으나 M1 형태소 분석기에서는 '으'를 어미의 일부로 인정하고 있다. 그 이유는 다음과 같은 현상 때문이다.

먹니 <-- 먹+니 (의문형)
먹으니 <-- 먹+으니 (연결형)

의문형 어미 '니'와 연결형 어미 '으니'는 표층 표현이 다르다. 즉 매개모음이 자동 삽입된 것이 아니다. 또한 다음과 같이 비슷한 형태의 음운 현상 때문에 '으', '스', '느' 탈락을 같은 맥락으로 보고 있다.

- 어미의 '으'탈락은 모음이나 'ㄹ'다음에 일어난다.
  날+으니 -->날+니 --> 나니
  날+니 --> 나니
  나+으니 --> 나니
  나+니 --> 나니

- 'ㅂ니다','ㅂ니까'의 기본형은 '습니다'와 '습니까'로 보고 모음이나 'ㄹ' 다음에 '스'가 탈락한 것으로 본다.
  돕+습니다 --> 돕습니다
  가+습니다 --> 갑니다
  갈+습니다 --> 갈+ㅂ니다 --> 갑니다.

- 'ㄴ다'의 경우는 '는다'를 기본형으로 보고 있으며 모음이나 'ㄹ' 다음에 'ㄴ'가 탈락하는 것으로 본다.
  먹+는다 --> 먹는다
  가+는다 --> 간다
  놀+는다 --> 놀+ㄴ다 --> 논다

어미의 '어/아'에 대한 기본형은 '어'로 보고 원형복원을 한다.

막아 <-- 막+어

4.3. 전처리기 - F1 모듈(문장 분할)

전처리기는 형태소 분석기와 tagger에서 처리하기 곤란한 문제를 미리 분할하기 위하여 사용된다. 전처리기에서 주로 하는 일은 문장구분이다. 즉, 입력 문서를 문장단위로 나누는 일이다. 이제까지 문장구분에 대해서는 주로 영어권에서 연구되었다. 특히 영어에서는 마침표(.)가 문장이 끝날 때 사용되는가 약어 등의 다른 의미로 사용되는가에 대한 연구가 많다 [Palmer 97]. 여기에서는 문장구분이 될 수 있는 특수기호의 좌우 문자열을 가지고서 문장 구분을 하는 간단한 전처리기를 구현한다.

문장구분이 될 수 있는 특수기호는 물음표(?), 느낌표(!), 마침표(.)이다. 각각에 대한 처리는 아래와 같다.

* 물음표나 느낌표가 나타났을 때
  - 특수문자와 붙어서 사용되지 않은 경우에 문장구분을 한다. 여기에서 말하는 특수문자는 아스키코드값을 가지는 문자를 말한다.

* 마침표가 나타났을 때
  - 뒤에 숫자와 붙어서 사용된 경우는 무시한다.
  - 마침표 바로 전문자가 영문이면 무시한다. 여기에서 영어 약어사전을 이용하여 약어인 경우에만 무시하면 더 정확한 결과를 얻을 수 있을 것이다.
  - 오른쪽에 또 다른 마침표가 나오면 무시한다. 이것은 말줄임표에 대한 처리이다.
  - 마침표 바로 앞단어의 길이가 2byte이하이면 마침표 앞 단어에서 문장구분을 하고, 그 줄의 나머지를 또 다른 한 문장으로 처리한다. 이것은 '가.', '나.' 등과 같은 말머리표에 대한 처리이다.
  - 그 이외에는 문장구분을 한다.

위의 모든 경우에 대해서 괄호나 따옴표에 대한 처리가 필요하다. 위의 규칙을 적용하여 문장구분을 할 때 괄호나 따옴표 쌍이 끊어지지 않도록 한다.

4.4. 후처리기 - F2 모듈(mapper)

형태소 분석의 결과가 다음 단계의 input과 같지 않은 경우 filter를 두어 그 문제를 해결할 수 있다.

- mapper 입력
  입력은 형태소 분석기의 결과이다.

- parsing
  형태소 분석기의 결과는 일정한 형식을 가지고 있기 때문에 쉽게 parsing이 가능하다. mapper가 인식할 수 있는 중간 형태의 데이터 구조로 만든다.

- mapping
  mapping은 보통 품사가 같지 않은 경우에 사용한다. 그러나 형태소의 구분에 대한 기준이 달라지는 경우도 있으며 형태소의 기본형이 다른 경우도 있다. F2 모듈은 다음과 같은 형태의 mapping table을 이용하여 매핑을 시도한다.
  <mapping의 형태>
    - input 품사 --> output 품사
    - input 단어/품사 --> output 단어/품사
    - input 단어/품사1+...+단어/품사n
      --> output 단어/품사1+...+단어/품사m

- mapper 출력
  mapper의 출력은 다음 프로그램의 입력에서 분석 가능한 형태이다. 즉, 다음 프로그램이 태거라면 태거의 입력 형식에 맞추어 출력한다.

[그림 1]에서 F2 모듈의 위치는 필요에 따라 T1의 다음이나 F3의 다음에 위치할 수도 있다 예를 들어 태거가 형태소 분석기와 같은 어휘/품사 체계를 사용하고 그 다음의 구문 분석기가 형태소 분석기와는 다른 품사 체계를 사용한다면 F2는 T2 다음에 위치해야 한다. 이때 위치를 바꾸는 것은 실행시 순서만 바꾸어 주면 된다.

4.5. 어절 기반 품사 태거 - T1 모듈

T1 모듈은 [신중호 1994]의 어절 태그를 기반으로 하는 간단한 품사 태거이다. 품사태깅은 문장 $W=W_0W_1...W_n$에 해당하는 어절 태그열 $T=T_0T_1...T_n$을 찾는 문제로 정의할 수 있다. 어절 태그열을 구하는 함수 ϕ를 확률식으로 표현하면 다음과 같다.

$$\phi(T) = \arg\max_{T} p(W|T)p(T)$$

여기에 마르코프 독립 가정과, 문장의 어절태그 발생 확률은 바로 앞 태그에 영향을 받는다는 가정에 의해 단순화시키면 다음과 같은 식으로 나타낼 수 있다.

$$P(T, W) \cong \prod_{i=1}^{n} \frac{P(T_i, W_i)\ P(T_i|T_{i-1})}{P(T_i)}$$

$P(W_i|T_i)$는 마르코프 독립 가정을 한 번 더 이용하여, 어절 내의 형태소-태그의 발생확률과 태그간의 bigram 전이확률로 표현한다.

$$P(T_i, W_i) \cong \prod_{j=1}^{n} \frac{P(t_{i,j}, w_{i,j})\ P(t_{i,j}|t_{i,j-1})}{P(t_{i,j})}$$

($w_{i,j}$: i번째 어절의 j번째 형태소,
$t_{i,j}$: i번째 어절의 j번째 품사 태그)

$P(T_i|T_{i-1})$를 구하기 위해, 어절 내 형태소의 품사 태그열로부터 [신중호 1994]에서 사용한 어절 태그로 표현하여, 형태소 태그열의 빈도로부터 어절 태그의 빈도를 구한다. 이 때, 어절 경계를 넘는 태그열의 빈도가 포함되지 않도록, 공백에도 별도의 형태소 태그를 부여하였다.

"나는 학교에 간다."라는 문장을 예로 들어보자. 이에 대한 형태소 분석 결과는 아래와 같다.

<나는>
나/npp+는/jxc

나/nqpa+는/jxc
나/ncn+는/jxc
나/pvg+는/etm
날/pvg+는/etm

<학교에>
학교/ncn+에/jca

<간다.>
가/pvg+는다/ef+./sf
갈/pvg+는다/ef+./sf

이렇게 형태소 분석 결과로 나온 "나/npp+는/jxc"에 대한 발생 확률을 기존의 코퍼스를 이용해서 구할 수 있다. 형태소 결과가 npp[9]일 경우 "나"라는 단어가 발생할 확률 ($P(나|npp)$), npp형태소 다음에 jxc 형태소가 출현할 확률 ($P(jxc|npp)$) 등을 이용해서 "나는" 어절이 "나/npp+는/jxc"으로 분석될 확률을 구한다.

어절 간 전이 확률에 대하여 "나는"이라는 어절을 "npp+jxc"로 분석한 후, 다음에 "학교에"라는 어절이 "ncn+jca" 형태로 출현할 확률을 구할 수 있다. 어절내에 형태소 결과들만 보고 다음 어절의 형태를 추론하는 것이다. 이런 식으로 어절 출현 확률, 어절간 전이확률을 이용해서 각 문장의 형태소 분석 결과를 태깅할 수 있다.

한 문장의 끝에 해당하는 기호는 전처리기(F1)에서 주며, 그 기호가 형태소 분석 결과에도 반영되어 태거는 문장의 끝을 알 수 있다. 문장의 끝을 발견하면, 그 문장을 이루는 어절들의 확률을 가지고 비터비 알고리즘을 실행한다. 비터비 알고리즘은 가장 발생확률이 높은 태깅 결과 문장을 구할 수 있도록 해 준다.

4.6. 최대 엔트로피 품사태거 - T2 모듈

T2 모듈 역시 앞에 형태소분석의 결과로 나온 형태소 후보들 중 가장 적합한 후보를 골라 태깅하는 모듈이다. 앞의 T1 모듈에서는 어절을 기반으로 한 간단한 태거를 만들었고, 여기서는 같은 기능을 하는 태거를 최대 엔트로피 모델을 써서 구현해 보았다.

최대 엔트로피를 사용한 품사 태거는 다음과 같은 확률식으로 정의된다.

$$P(t_1 \ldots t_n | w_1 \ldots w_n) = \prod_{i=1}^{n} P(t_i | h_i)$$

$$P(t|h) = \frac{1}{Z(h)} \prod_{j=1}^{k} \alpha_j^{f_j(h,t)}$$

$$\alpha_j \in \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$$
$$f_j \in \{f_1, f_2, \ldots, f_k\}$$
$$f_j(h,t) = \{0,1\}$$
$$h_i = \{m_i, t_{i-1}, t_{i-2}\}$$

위에서 두 번째 식이 바로 주어진 문맥이 h(history)일 때 태그가 t가 될 확률이다. 이식은 자질함수의 값이 1일 때의 파라미터 값 α에 의해 영향을 받는다. $Z(h)$는 전체 확률 값을 1로 만들기 위한 정규화 값이다. 두 번째 식을 문장 전체로 확장한 식이 첫 번째 식이다.

기존의 최대 엔트로피 품사 태거[강인호 1998]에서는 형태소 분석과 품사 태깅을 동시에 수행하는 반면 본 품사 태거에서는 형태소 분석과 품사 태깅을 분리하였다. 형태소 분석기는 앞에서 구현한 M2 형태소 분석기를 사용하였고, 품사 태깅에 대해서만 최대 엔트로피 모델을 적용하였다.

위의 식에서 보듯이 주어진 문맥에서 형태소의 태깅확률 $p(t|h)$는 문맥의 자질함수에 따라 결정된다. 본 시스템에서는 다음과 같이 bi-gram과 tri-gram 문맥정보만을 자질로 사용하였다.

템플릿	자질의 수	f(t,h) = 1
1(Bigram)	29	h = {ti-1=X} 이고 ti=t
2(Trigram)	29*29	h = {ti-2ti-1=XY} 이고 ti=t

태거는 perl로 구현하였으며 최대 엔트로피의 파라미터 값를 구하기 위해 노르웨이의 Hugo WL ter Doest가 만든 MaxEntropy.pm 라이브러리를 사용하였다. 본 시스템의 평균 재현률과 정확도는 다음과 같다.

	재현율	정확도
matec99 평균치	0.93	0.93
본 시스템	0.74	0.81

아직 어휘자질과 통사/의미 자질을 사용하지 않아 성능은 조금 떨어지지만 자질 추가 후 성능 개선의 여지가 충분하다. 대표적인 오류로 관형사형 전성어미(etm)의 오류와 보조용언

9) 품사 기호[이공주 1996] 예들 - jcs: 주격, jco: 목적격, jcc: 보격, jcm: 관형격, jcv:호격, jca: 부사격, jcj: 접속격, jct: 공동격, jcr: 인용격

(px)의 오류를 살펴보면 다음과 같다.

원 어절	시스템 결과	matec99 정답
발정한	발정/nc+하/xsv+은/etm	발정/nc+하/xsv+ㄴ/etm
기뻐했다	기쁘/pa+어/ec+하/px+었/ep+다/ef	기뻐하/pv+었/ep+다/ef

이는 태그셋과 형태소의 선정기준이 불일치해서 생긴 문제로 볼 수 있으며 오류의 대부분을 차지한다.

최대 엔트로피 모델의 가장 큰 장점은 여러 정보들이 자질함수로 통합될 수 있다는 점에 있다. 이제까지 규칙과 통계기반으로 구분되어 처리되던 정보들이 하나의 프레임웍에 포함될 수 있는 것이다. 따라서, 실제 어떠한 자질들이 성능 향상에 도움을 주는가를 평가하는 것이 중요하다. 현재, 이에 대한 연구가 진행되고 있다.

4.7. 명사 추출기 - N1 모듈

본 명사 추출기는 한국어 텍스트를 어절단위로 분할한 후, 각 어절을 입력으로 받아서 명사만을 결과로 출력하는 시스템이다. 예를 들어 '채식주의도 나름대로 의미가 있다." 라는 문장을 생각해 보자. 입력은 "채식주의도", "나름대로", "의미가", "있다" 의 4개의 어절이 되고 이에 대해 명사라고 추정되는 " 채식주의", "의미"를 출력한다.

입력으로 주어진 어절이 n음절이라고 하면 추출 가능한 명사 조합의 수는 $\sum_{k=1}^{n}(n-k+1)$이 되고 이는 그림과 같이 트리 구조의 탐색 공간으로 나타낼 수 있다. 따라서, 명사 추출의 문제는 트리 탐색 문제로 귀결된다.

군	군전	군전력	군전력은
	전	전력	전력은
		력	력은
			은

명사추출의 문제를 위와 같은 트리 탐색으로 정의할 때 발생하는 문제는 크게 세 가지로 나누어 볼 수 있다. 하나는 탐색 기법에 대한 문제(BFS, DFS, ...)이고, 둘째는 해당 노드가 명사인지 아닌지의 판단을 어떻게 할 것인가(Evaluation Function)의 문제이며, 셋째는 해답이 여러 가지가 있을 수 있다는 문제이다.

각 문제에 대해 해결책으로 사용한 방법론은 다음과 같다.

1. 탐색 기법 - 깊이 우선 탐색 (Depth-First Search) 을 수행하여 최장일치 명사를 답으로 출력한다. 예를 들어 "군전력은" 이라는 어절에 대해 깊이 탐색을 수행하면 "군전력"이라는 명사가 최장일치로 검색되므로 이 후에 '전력', '전력은' 과 같은 노드에 대해서는 탐색을 수행하지 않아도 된다.

2. 명사인지의 여부 판단 - 형태소나 구문, 의미 정보를 사용하지 않고 어휘정보만을 사용하여 명사여부를 판단한다. 형태소 분석을 시도하지 않으며 단지 명사사전, 조사 및 후절어 사전, 불용어절 사전의 세 가지 사전의 검색만을 통해 해당 명사를 추출한다. 따라서, 기존의 형태소 분석결과를 명사추출에 사용하던 방법론에 비해 속도면에서 경쟁력이 있다. 내부적인 비교 실험 결과 소요 시간면에서 약 75배의 성능향상이 있었다.

3. 여러 가지 해답의 처리 - 명사로 추정되는 후보가 연이어 나오는 경우에는 복합명사로 처리했다. 예를 들어 "사과나무에는" 과 같은 어절에 대해 "사과"와 "나무"가 명사로 추정되는데 이런 경우에는 "사과나무"라는 하나의 복합명사로 출력한다.

이와 같은 방법론을 사용하였을 때 본 명사추출기의 장점은 다음과 같다. 어휘 정보만을 사용하므로 사전관리에 의해 성능향상을 가져올 수 있다. 또, 재현률과 정확도가 높으면서도 처리 속도가 빠르다.

본 시스템의 알고리즘은 크게 3가지 부분으로 나눠볼 수 있으며 그림에 제시한다.

- 불용어절 차단
- 최장일치를 통한 명사 후보 추출
- 조사/어미 사전을 이용한 명사 추정

```
noun_extract(wp)
{
      if ( 불용어 사전 검색(wp) ) return □ ;
      명사 사전 검색(wp);
      noun = 최장일치명사;
      if (noun) {
         sfix = wp/noun;
         if ( 후절어(wp/noun) ) return noun;
         else return noun_extract(sfix);
      } else {
         sfix = 후절어사전 검색(wp);
         if (sfix) return wp/sfix;
      }
}
```

기타 고려한 사항으로는 한 글자 명사와 고유명사의 처리가

있다. 한 글자 명사는 전체 명사의 5~8%를 차지한다. 본 시스템에서는 이전의 두 형태소를 보고 한 글자 형태소가 명사인지 아닌지를 판단하게 하였다. (tri-gram 확률 모델이용) 한편, 고유명사의 처리를 위해 전화번호부에서 추출한 고유명사사전을 가공하여 사용하였다.

	재현율	정확도
학습 집합	0.91	0.97
실험 집합	0.86	0.88
matec99 평균치	0.85	0.80

정확도와 재현률은 표와 같으며 matec99에 제출된 명사 추출기의 평균을 상회하고 있다. 시간 복잡도 면에서는 42,618어절/초, 30MB/분 (인텔 펜티엄 II 400MHz, 리눅스)로 대단히 빠른 수행 속도를 보인다. 공간 복잡도 면에서도 프로그램 크기가 1kb가 넘지 않으며 사전크기는 2MB를 넘지 않는다.

본 시스템은 정확도와 재현률면에서 타 시스템에 뒤지지 않으면서도 수행 속도와 메모리 크기 면에서는 월등한 성능을 보이고 있다. 따라서, 대용량의 텍스트를 처리해야 하는 웹 문서 색인 등과 같은 실제적인 문제에 잘 적용될 수 있다.

5. 실험 및 평가

5.1. 시스템의 특성

- 형태소 분석기의 목적

모듈화된 형태소 분석기는 정보검색, 구문분석, 기계번역, 태깅 등 다목적 형태소 분석기를 구현하는 것을 목표로 한다. 그러나 다목적 형태소 분석기의 구현은 각 목적에 대하여 효율성을 보장할 수 없다. 따라서 본 논문의 주장은 각 목적에 맞는 형태소 분석기를 각각 만들기 위하여 형태소 분석기를 모듈화 하는 것이다.

- 표제어 선정 기준

본 형태소 분석기에서 사용하는 사전의 표제어 선정중 가장 큰 특성은 '으', '스', '느'로 시작하는 어미들이다. 이에 대하여 4.2.3절에 설명하였다.

그외의 세그멘테이션에 대한 원칙은 확실히 정해지지 않았다. 결국 이러한 문제는 한국어에서 단어와 형태소의 구분이 무었인지 또는 단어가 무엇인가하는 정의와 관련이 있는데 이에 대한 정의를 내리기는 매우 어렵다. 기본적으로는 가능하면 모두 분석하는 것을 원칙으로 하였으나 불필요한 과분석이 일어날 가능성이 높은 단어나 분석이 불가능한 단어는 묶어서 하나의 단어로 등록한다.

'선보이다' : "선보이/동사+다/어미"
'먹음직하다' : "먹음직하+다",

본 시스템의 형태소 분석기는 다어절 한 단위에 대하여 처리하지 않는다. 그러나 다어절 한 단위에 대한 처리는 반드시 필요한 것으로 보고 있으며 추후 시스템의 확장에 포함될 예정이다. 이렇게 의미적으로 한 단어인 어절들에 대하여는 어휘의 정보가 해결하는 열쇠가 될 것이다.

"ㄹ 수 있" : 의존 명사 '수'에 해결 정보 추가
"에 대하" : 동사 '대하'에 해결 정보 추가
"예술의 전당" : 고유명사 처리 + 고유명사 사전
"어두워질 때까지" : 명사 '때'에 해결 정보 추가

사전 구축을 위해 단어를 분석해 가다 보면 단어들이 품사에 따라 어느 정도는 공통성을 가지는 듯하지만, 결국은 모든 단어가 각기 다른 구문/형태적 특성을 가진다는 사실을 알게 된다. 이러한 문제는 현재의 수준으로는 어휘에 해결 정보를 가지게 하는 일종의 case-by-case에 의하여서만 해결이 가능한 것으로 본다. 그러나 단어들의 특성을 좀더 세밀한 특성들로 나눌 수 있다면 일반적인 기준 또는 원칙이 발견될 수 있을 것이다.

5.2. matec99 대회를 통한 시스템 평가

MATEC99 대회를 통하여 실험에 참가한 모듈은 형태소 분석기(M2→F2)와 명사 추출기 (N1)뿐이다. 실제 본 연구의 주가 되는 형태소 분석기 M1은 대회 기간중 완성되지 못하였기 때문에 실험에 참가하지 못했다.

- M2→F2 실험 결과

	어절수	재현률	정확도
뉴스	7443	0.46	0.54
비소설	7318	0.45	0.57
소설	5698	0.40	0.55
전체	20459	0.44	0.55

[형태소 분석기 과반수 지지식 평가 결과]

형태소 분석기의 평가 결과로 재현률과 정확도가 낮게 나온 이유는 형태소 분석기(M2)와 부속 모듈들에 대한 튜닝이 부족했기 때문이다. 약 10% 정도의 어절이 미등록어로 표시되

었고, 20% 정도의 어절에서 어미의 mapping을 제대로 하지 못한 문제가 발생했다.

결국 후처리기(F2)의 mapping 프로그램은 완성되었지만 후처리기의 data인 mapping table은 단순히 품사만은 바꾸는데 그쳤기 때문에 상당히 많은 어미와 동사 부분에서 형태소 분석의 결과가 정확히 일치하지 못 하였다. 다음에 보는 바와 같이 M2 형태소 분석의 결과는 '은/etm'을 결과로 출력하는데 정답 코퍼스는 'ㄴ/etm'으로 표시된다.

발정/nc+하/xsv+은/etm	발정/nc+하/xsv+ㄴ/etm

이러한 문제는 F2의 mapping table에 다음과 같은 표를 추가하고 mapping 프로그램의 일부를 수정하여야 해결 가능하다.

SOURCE	TARGET
#모음|#ㄹ종성/*+{은/etm}	ㄴ/etm
#모음|#ㄹ종성/p*+{습니다/ef}	ㅂ니다/ef
#모음|#ㄹ종성/p*+{는다/ef}	ㄴ다/ef

- N1 실험 결과

	재현률	정확도
뉴스	0.84	0.88
비소설	0.90	0.92
소설	0.83	0.84
전체	0.86	0.88

[명사 추출기 평가 결과]

명사 추출기 N1은 형태소 분석을 하지 않고 명사 사전과 후절어 사전만을 이용하여 빠르게 명사 추출을 한다. 따라서 N1의 실험 결과를 보면 명사 추출을 하는데에는 형태소 분석과 태깅을 거쳐 명사를 추출하는 것보다는 명사 추출 전용 프로그램을 만드는 것이 더욱 효율적이라고 볼 수 있을 것이다.

5.3 자체 평가

matec99 평가 대회 이후에 자체적인 평가는 하지 않았다. 처음 이 대회에 참가할 때부터 우리 팀은 새로운 형태소 분석기와 태거의 구현을 목표로 하고 있었고 대회에서 요구하는 기존의 시스템에 대한 성능 평가를 위한 사전 어휘 및 품사 mapping과 형태소 분석기/태거의 튜닝에 많은 시간을 투자하지는 못했다. 그러나 이번 대회를 통하여 모듈화된 형태소 분석기와 태거를 설계 구현하는 계기가 되었고 현재의 상태에서 어느정도의 비교를 통하여 우리팀의 시스템에 대한 객관적인 평가를 하는 계기가 되었다.

5.4. 장단점

본 논문을 통해 구현된 모듈화된 형태소 분석기의 장점은 첫째 새로운 시스템의 구현이 쉽다는 점이다. 기존의 형태소 분석기의 구성 요소들이 재사용 가능한 모듈로 되어 있기 때문에 새로운 시스템을 구현하기 위해서 기존의 모듈들을 사용함으로써 쉽게 구현이 된다. 둘째는 프로그램의 확장/변경이 용이하다는 것이다. 각 모듈들은 단순화되어 있기 때문에 프로그램을 확장하거나 변경하는 것이 쉽다. 셋째는 같은 조건에서 여러 개의 다른 모듈을 실험하는 것이 가능하다는 것이다. 예를 들어 사전, 결합정보, 분석 알고리즘, 내부구조가 동일하고 음운 복원이 다른 두 개의 형태소 분석기를 만들어 실험하는 것이 가능하다. 넷째는 프로그램의 보수가 쉽다는 것이다. 단순화된 모듈은 에러를 발견하기 쉽고, 각 모듈은 다른 모듈과 독립되어 있기 때문에 프로그램을 고치는 것도 쉽다. 그밖의 장점으로는 모듈들이 작은 크기로 단순화되어 있어서 공동작업자간의 이해가 쉽고, 교육 및 전수가 용이하다는 점이다.

반면 모듈화 하는데 있어서의 단점으로는 첫째, 초기 설계에 모듈화를 위해 많은 시간이 소요된다는 점이다. 재사용이 가능한 모듈을 구현하기 위해서는 설계 단계에 충분한 시간을 가지고 설계를 해야 한다. 둘째는 모듈화된 시스템은 일반화된 설계로 인하여 공간을 많이 차지하거나, 속도가 느려지는 경우가 있다. 그러나 컴퓨터의 발전속도를 고려해 볼 때, 속도와 공간의 문제는 중요하지 않다. 일반적으로 모듈화는 시간복잡도를 상수배 만큼만 증가시킬 뿐이다.

6. 결론

본 논문에서는 자연언어 처리 시스템 - 특히 형태소 분석기 -에서의 모듈화에 대한 중요성을 설명하고 형태소 분석기를 중심으로 하는 예에서 모듈화를 통하여 다양한 모델의 형태소 분석기와 태거를 구성한 결과를 보여준다.

모듈화된 형태소 분석기 및 태거는 새로운 이론은 적용하기 위한 시스템을 기존의 시스템으로부터 쉽게 만들어 내는 장점을 가지고 있기 때문에 개발 시간이 매우 짧고 시스템을 유지 보수 하기 쉽다. 또한 같은 조건에서 다른 부분만을 비교 검증하는 것이 가능하게된다.

앞으로 이러한 모듈들이 더욱 많이 개발되기 위해서는 자연 언어 처리 분야에서 사용되는 품사를 비롯하여 코퍼스, 사전 등이 표준화되고 모듈의 공유를 위한 연구가 진행되어야 할 것이다.

감사의 글

이 시스템 구현에 필요한 모듈들을 함께 구현해준 최용석, 이주호, 서충원에게 감사의 마음을 표한다.

참고문헌

[강인호 1998] 강인호, 김재훈, 김길창, "최대 엔트로피 모델을 이용한 한국어 품사 태깅", 제 10회 한글 및 한국어 정보처리 학술대회, 9-14쪽, 1998년.

[김덕봉 1993] 김덕봉, 최기선, "DDAG: 효율적인 한국어 형태소 해석 방법", 제 5회 한글 및 한국어 정보처리 학술 대회, 341-353쪽, 1993년.

[김성용 1987] 김성용, "TABULAR PARSING 방법과 접속정보를 이용한 한국어 형태소 분석기", 석사학위논문, 한국과학기술원, 1987

[신중호 1994] 신중호, 한영석, 박영찬, 최기선, "어절구조를 반영한 은닉 마르코프 모델을 이용한 한국어 품사태깅", 한글 및 한국어 정보처리 학술대회, 389-394쪽, 대전, 1994년 11월 18-19일

[이공주 1996] 이공주, 김재훈, 최기선, 김길창, "구문 트리 부착 코퍼스 구축을 위한 한국어 구문 태그", 인지과학, Vol. 7, No. 4, 7-24쪽, 1996년

[이상호 1994] 이상호, 김재훈, 조정미, 서정연, "부분 분석 결과를 공유하는 한국어 형태소 분석", 제 11회 음성통신 및 신호처리 워크샵 논문집, 75-79쪽, 1994년

[이은철 1992] 이은철, 이종혁, "계층적 기호 접속정보를 이용한 한국어 형태소 분석기의 구현", 제 4회 한글 및 한국어 정보처리 학술대회, 95-104쪽, 1992년

[ETRI 1999] 컴퓨터-소프트웨어 기술 연구소, 지식정보연구부, "품사 부착 말뭉치 구축 지침", 한국전자통신연구원, 1999.

[Palmer 1997] David D. Palmer, Marti A. Hearst, " Adaptive Multilingual Sentence Boundary Disambiguation", CL 1997, pp. 241~267, 1997