



실무에 활용하는 Elasticsearch 검색엔진 구축

3일차 : Elasticsearch 분석기

오늘의 아젠다



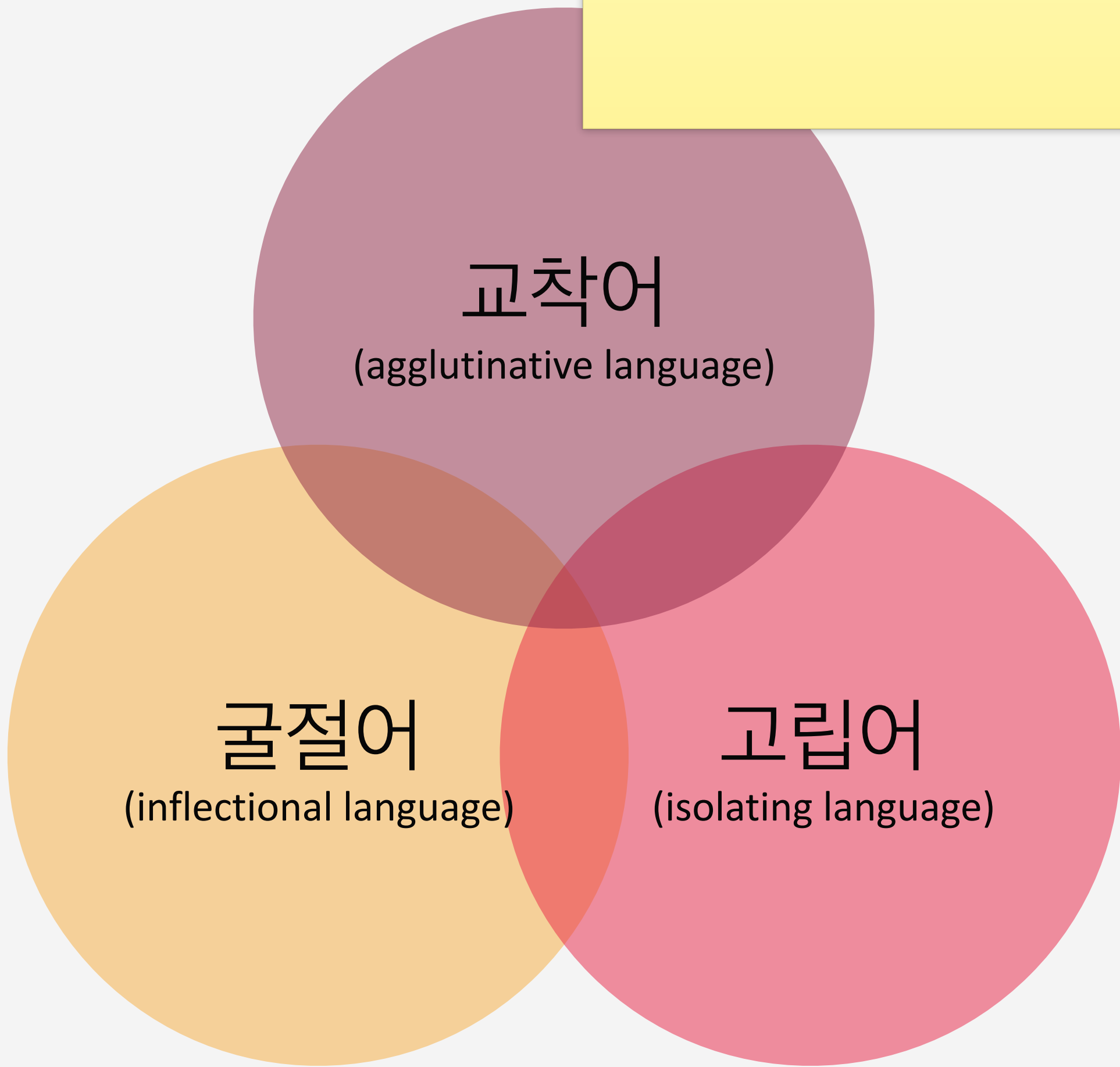
· Elasticsearch에서의 텍스트 데이터 처리 방법 및 분석기 설정

· 한글 형태소 분석기 설치 및 운영

· 형태소 분석기 플러그인 개발

먼저 한국어의 특징을 살펴봅시다.

한국어의 분류 (형태론적 관점)



교착어는 접사를 다양하게 붙일수 있으나
굴절어는 일부 어미가 변화는 하나
어간이 의미가 없다.

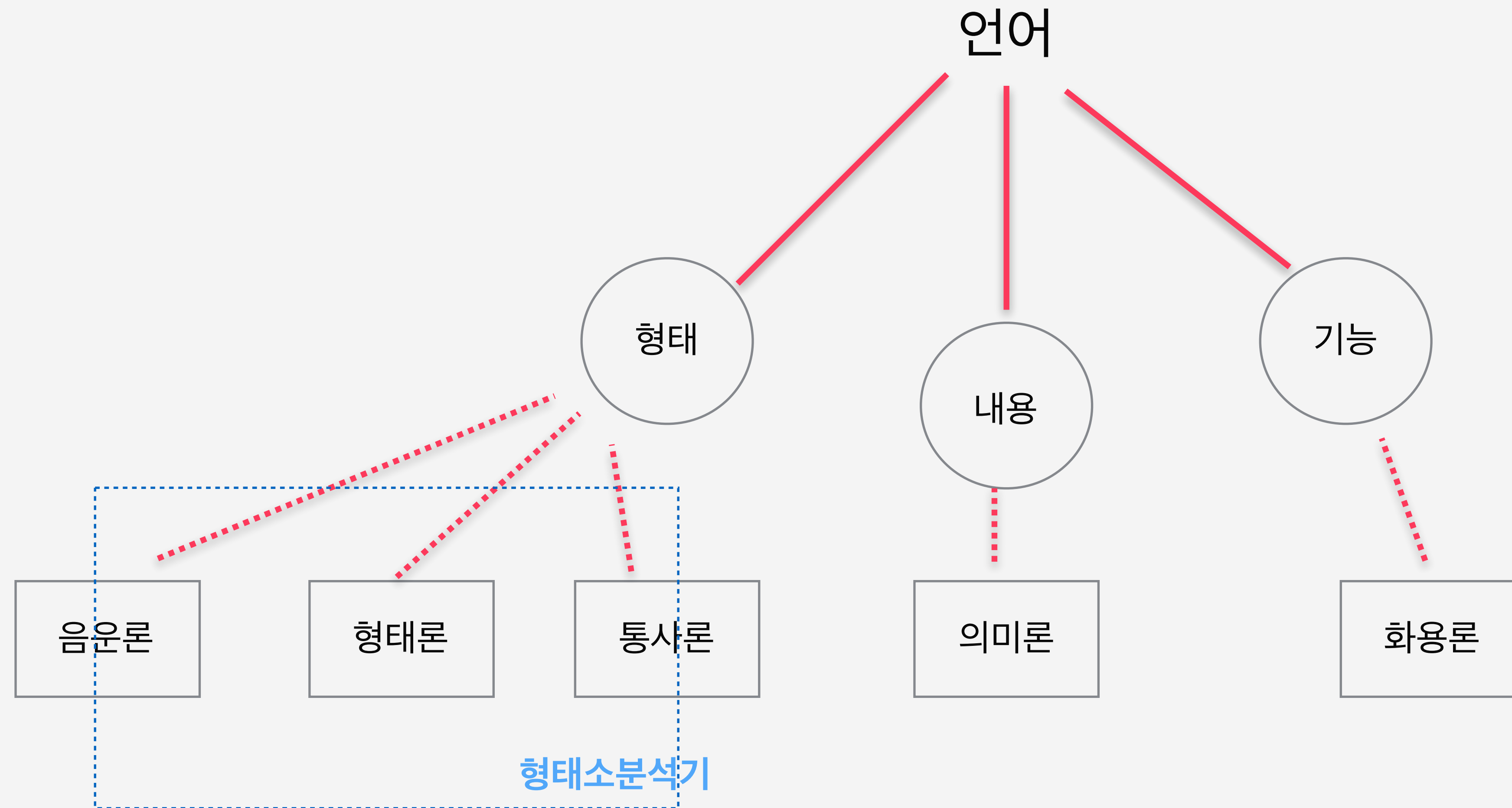
교착어 : 고립어와 굴절의 중간적 성격
어근에 접사가 결합되어 문장 내의 각 단어의 기능
어간에서의 어형 교체가 전혀 일어나지 않음.

굴절어 : 단어의 활용 형태가 단어 자체의 변형으로 나타나는 언어
어미가 여러가지로 변화하여 문법적 기능도 변화
beauty/beautiful

고립어 : 낱말이 그 어떤 형태상의 변화가 없이 글 가운데 나타나고
다른 말과의 문법적 관계는 어순에 의해 표시

한국어는 교착어에 가깝지만 굴절어의 특성도 나타나
(예) 나가 → 내가, 가+었다 → 갔다)
영어는 굴절어이기도 하지만 고립어의 특성이 강함(어순 종속)
중국어는 고립어이지만 교착어 특성도 있다.
(예)我的)

언어의 구성 요인



언어의 구성 요인

음운론 : 언어의 발음을 연구하는 분야.

형태론 : 품사(~사)

단어형성법(합성어,파생어) 을 연구하는 분야

통사론 : 문장성분을(~어) 연구하는 분야

문장의 표현 : 높임법,상대 높임법,객체높임법,부정문,긍정문,문장의 종결 표현/사동/피동/시제

의미론 : 언어의 내용에 초점 (단어의 의미, 문장의 의미)

반의어, 다의어,동음이의어,유의어

단어의 의미 => 계열적 의미 (동의,상하,대립), 결합적 의미(대등합성어, 혼성어,관용어,연어) 다의 관계, 동음관계

문장의 의미 => 동의성, 중의성, 잉여성

화용론 : 담화의 맥락이 화자청자 장면 직시에 따라 달라진다.

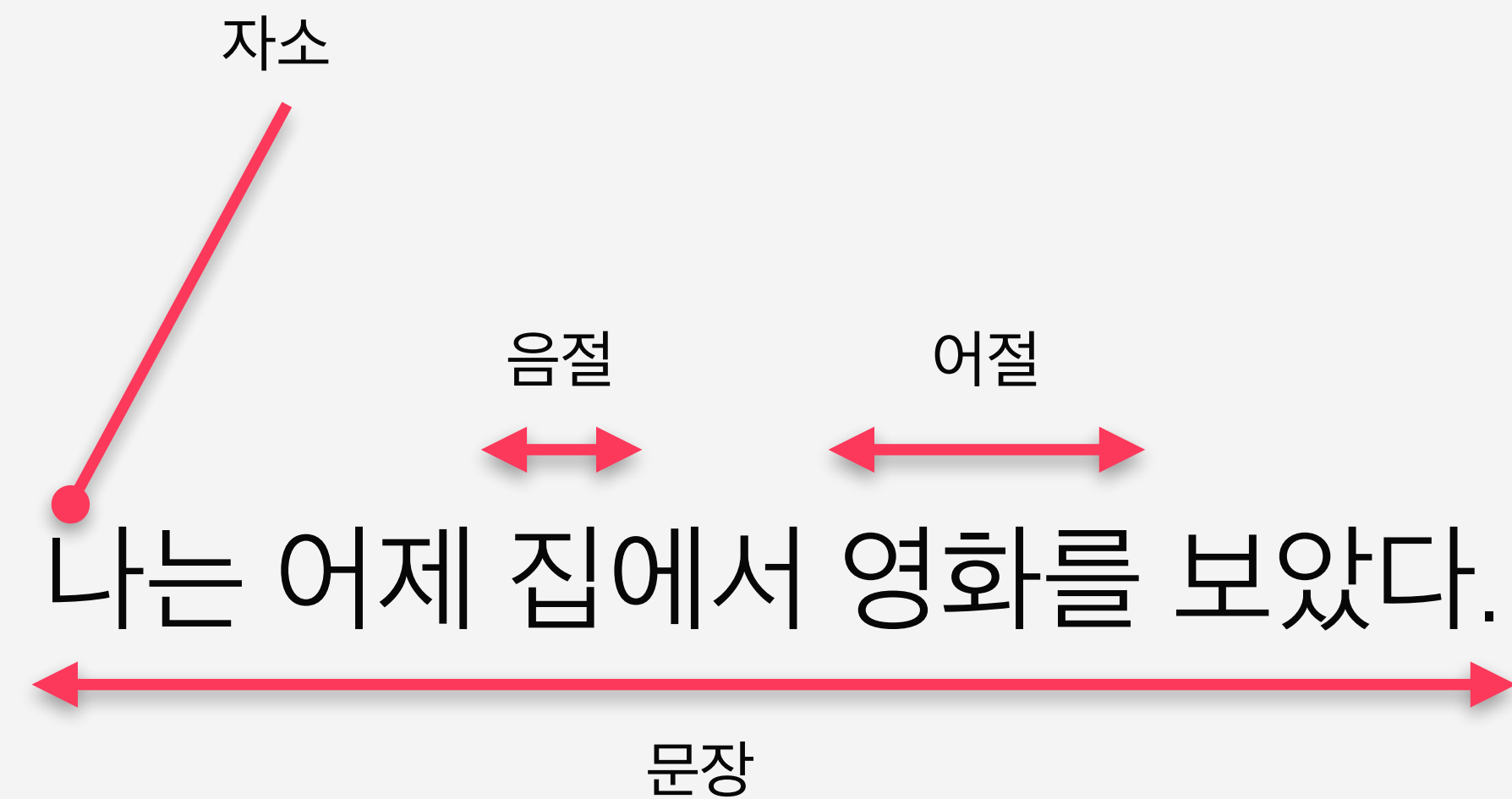
신정보와 구정보에 의한 발화의 흐름

대화상의 함축

발화행위(액면그대로의 의미, 이면의 의미)

한국어의 구성

- 자소 : 초성+중성+종성
- 음절 :
 - 자음과 모음이 생성하는 소리의 단위
 - 영어의 소리를 구성하는 기본 단위
 - 단어를 발음할 때 생기는 모음(a, e, i, o, u)의 소리
- 단어(=어절 word phase) : 형태소와 형태소가 결합
 - 실질형태소(어휘형태소) + 문법형태소
 - 실질형태소 + 실질형태소 = 복합어



한국어의 형태론적 특성

영어는 단어의 중의성이 큰 반면 어절 분리의 문제가 없다.

예) like (명사/동사/형용사/전차사/접속사)

한국어는 품사의 중의성은 적은 반면 어절 유형에 따른 형태소 분리의 중요성이 문제

예) '감기는'

감기(명사) + 는(조사)

감기(동사) + 는(어말어미)

감(동사) + 기 (명사형어미) + 는(조사)

(문장에서 용언의 어간에 붙어 명사와 같은 기능을 수행하게 하는 어미. '-음', '-기')

한국어의 구문론적 특징

- 영어

문장을 구 단어의 위치가 고정적인 구조적 언어성하는 어절 순서는 S-O-V

- 한국어

문법형태소(조사 등)이 발달하여 부분 자유 어순 언어
(장난 나랑 지금 하나?)

한국어는 이중 주어를 가질수 있음

예) 토끼가 앞발이 짧다.

주어를 생략할수 있음

밥을 먹고 학교에 가서 강의를 들었다.

유니코드에서의 한국어의 특징

유니코드

지구상의 존재하는 모든 이성적 글쓰기 시스템을 포함하는 단일 문자 집합

유니코드는 문자와 코드에 대한 사상(Mapping)

예) A->0100 0001

유니코드가 정의할수 있는 글자수 제한은 없음.

Hello => U+0048 U+0065 U+006C U+006C U+006F

인코딩

유니코드를 어떻게 컴퓨터가 알아 듣는 말로 부호화 할것인가..

UTF-8

: 국제 표준

0~127까지는 단일 바이트로 저장

128 부터는 2,3바이트로 시작해서 최대 6바이트까지 저장

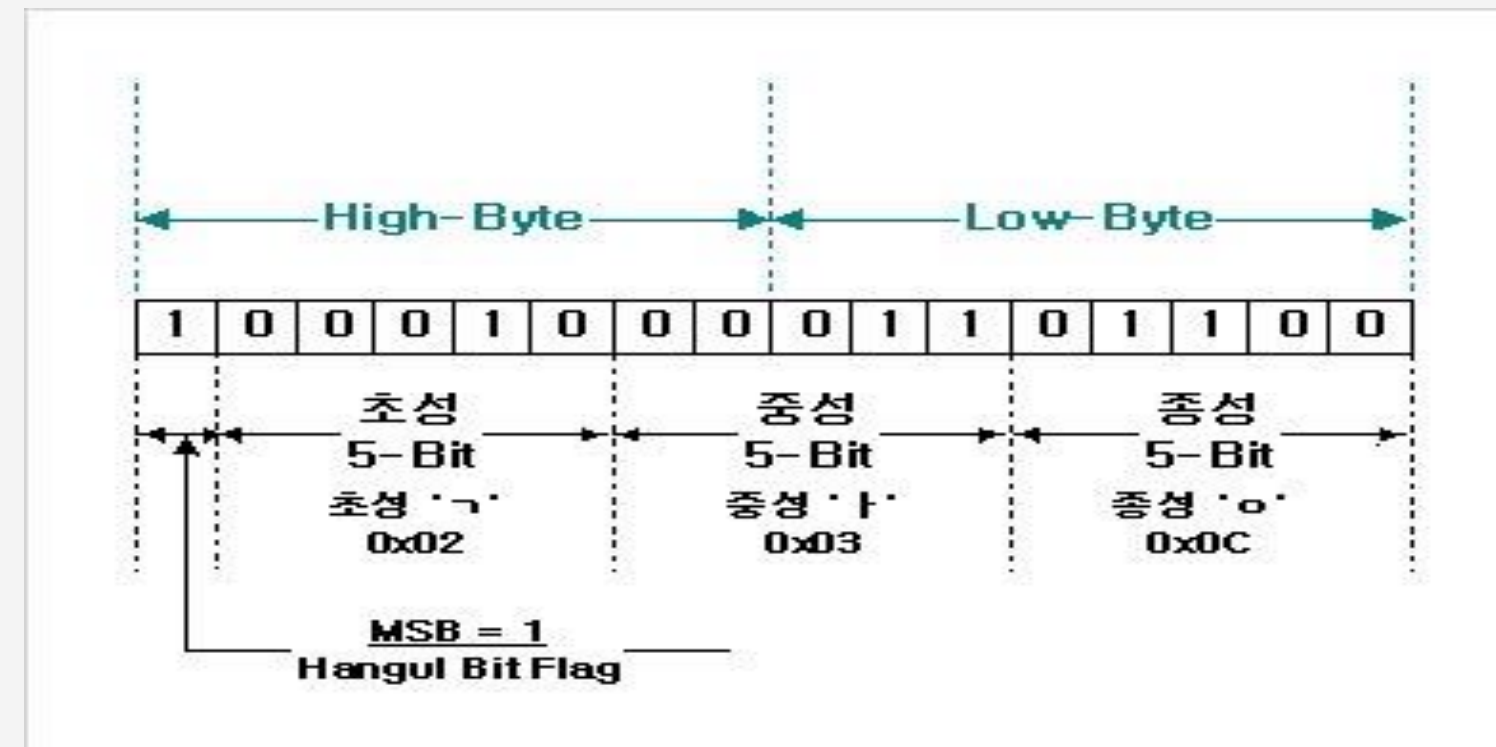
UTF-7, UTF-16 등등

유니코드에서의 한국어의 특징

1. KS 완성형 (KSC5601, ISO2022-KR,euc-kr)

- 0xA1~OxFE까지 94*94=8836 개의 코드 영역에 2350자의 한글,4888자의 한자 ,1128자의 특수문자 470자의 기타 문자를 정의함.
- 자소에 대한 정보가 없음. 자소 분해 불가능
- 완성형 코드 테이블

2. KSSM 조합형



3. 확장형 완성형 (CP 949 혹은 UHC)

- MS에서 KSC-5601로는 두 바이트로 표시 할수 없는 한글 문자를 가나다 순으로 배치

4. 유니코드 (UCS-2, UTF-8)

- ISO 10646국제 표준 Universal Character Set(UCS) vs 유니코드 프로젝트

유니코드에서의 한국어의 특징

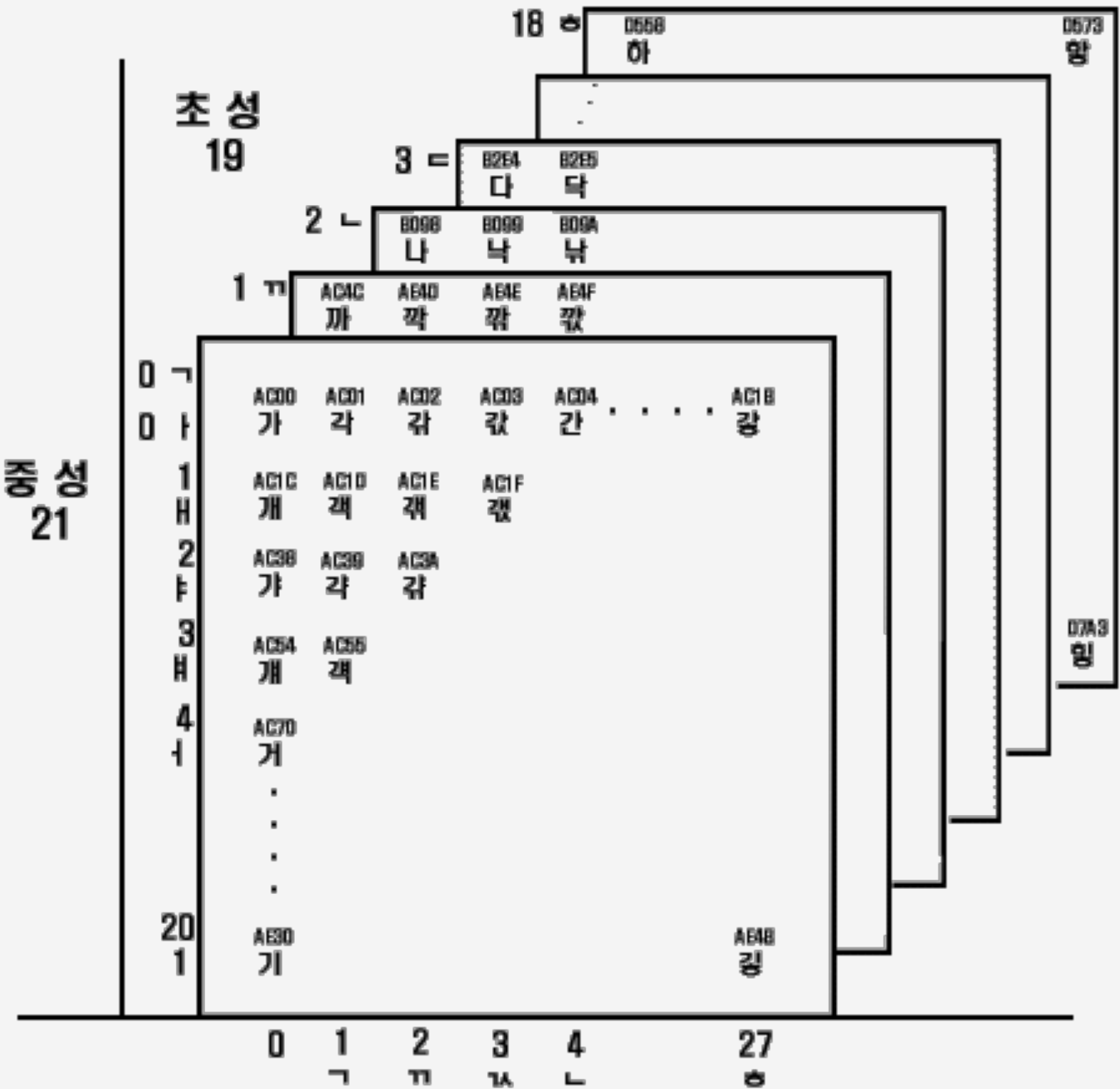
이름	처음	끝	개수
한글 자모 (Hangul Jamo)	1100	11FF	256
호환용 한글 자모 (Hangul Compatibility Jamo)	3130	318F	96
한글 자모 확장 A (Hangul Jamo Extended A)	A960	A97F	32
한글 소리 마디 (Hangul Syllables)	AC00	D7AF	11184
한글 자모 확장 B (Hangul Jamo Extended B)	D7B0	D7FF	80

종성= 유니코드값-0xAC00 을 28로 나눈 나머지 값이 0이면 종성없음
1이면 ㄱ27이면 ㅎ

초성 = 유니코드값-0xAC00 을 21X28=588로 나누었을 때의 몫이
0이면 ㄱ, 1이면 ㅋ ...18이면 ㅎ

중성 = 유니코드값-0xAC00 을 21X28=588을 나눈 나머지를 28로
나누었을 때 0이면 ‘ㅏ’ 1이면 ‘ㅑ’ ... 20이면 ‘ㅣ’

음절 = UNICODE - 0xAC00
초성 = hcode / 588
음절 = hcode % 588
중성 = hcode / 28
종성 = hcode % 28



유니코드에서의 한국어의 특징

역 계산

유니코드값 = $(0xAC00) + (\text{초성코드} * 21 * 28) + (\text{중성코드} * 28) + (\text{종성코드})$;

예) 삼성전자

ㅅ ㅊ ㅁ ㅅ ㅊ ㅇ ㅈ ㅊ ㄴ ㅈ ㅊ

1. PreFix 모드

ㅅ
사
삼
삼ㅅ
삼서
삼성
삼성ㅈ
삼성저
삼성전
삼성점
삼성전자

한국어 연구 성과

21세기 세종 계획은 1997년에 그 계획이 수립되어 1998년부터 2007년까지 10년 동안 시행

기초자료 (말뭉치)

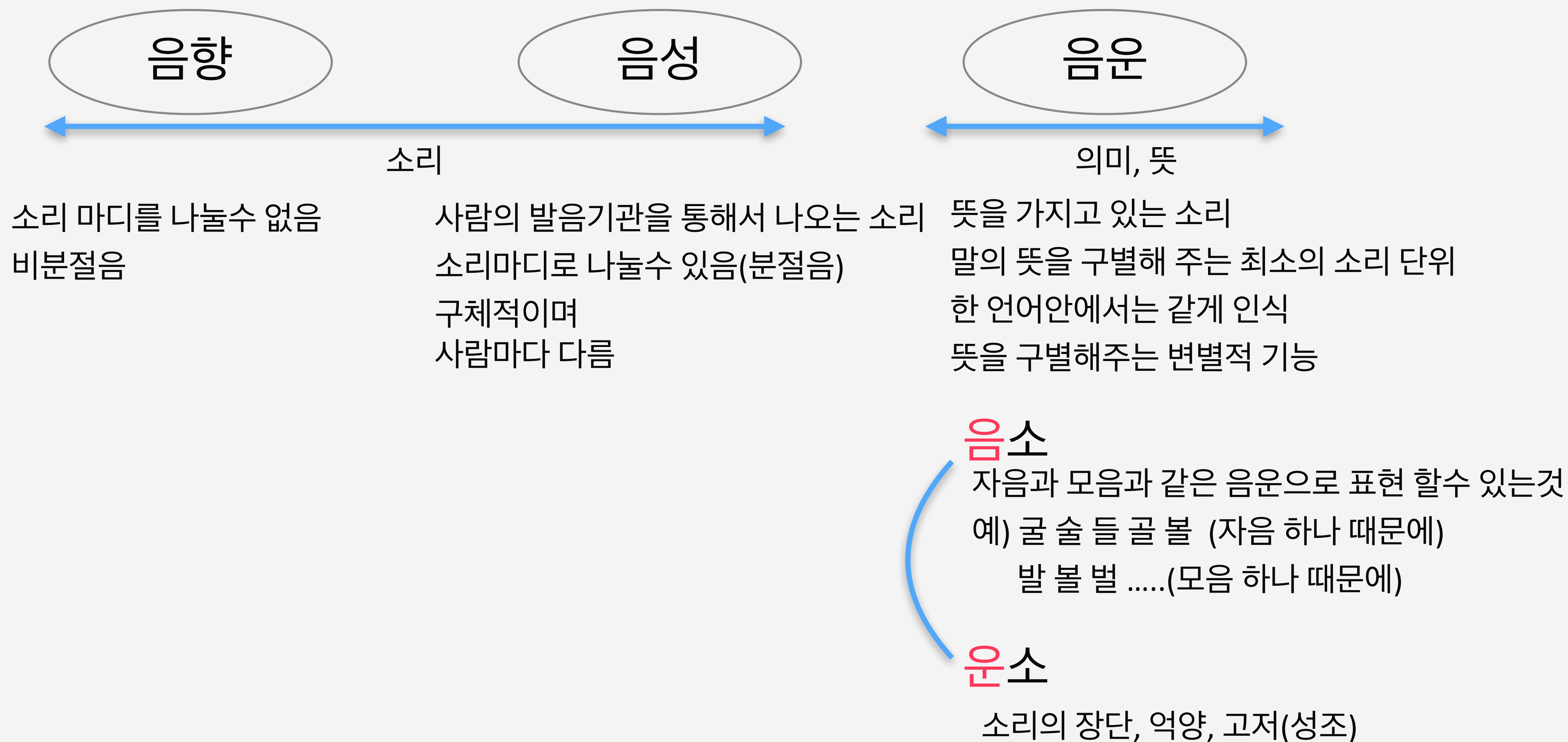
구분	계획	결과	
현대국어 말뭉치	5억 어절 (+7,000만)	9316만 어절(구어 466만 어절, 전문용어 200만 어절 포함) 9 + 7000만 어절	18.6%
현대국어 분석 말뭉치	7,900만 어절(+200만)	1,230만 어절	15.6%
북한/해외한국어 말뭉치	4,000만 어절	1,083만 어절	27%
옛문헌/방언/구비문학 말뭉치	2,600만 어절	637만 어절	24.5%
대역(한국어-외국어) 말뭉치	1,000만 어절	766만 어절	76.6%
계	6억 5500만 어절	2억 32만 어절	30%

전자사전

구분	목표	결과
기본 사전	80만 단어	60만 항목
전문용어 사전	35만 단어	
연어 정보 사전	170만 연어	
대역 사전	31만 단어	
개념 사전	30만 개념	

한국어의 문법 (음운론)

음운론



음운의 변동 (교체)

음운의 변동을 알아야 하는 이유는

음운의 변동이 발음 뿐만 아니라 표기에 영향을 미치는 경우가 있기 때문

음운의 변동은 교체, 축약, 탈락, 첨가 4가지 경우가 있다.

1) 음절의 끝에서 발음되는 자음은 “ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅇ” 이것 외의 자음이 오면 7개중 하나로 바뀜
예) 부엌

2) 절음 법칙 : 실질형태소 앞에 모음으로 시작하는 실질 형태소가 오면 바꾸고 올라온다.

옷안 => 온안 => [오단]

옷위 => 온위 => [오뒤]

3) 연음법칙 : 모음으로 시작하는 형식 형태소(조사)는 그대로 올라온다.

옷이 => [오시]

밖에=>[바께]

닭을 => [달글]

음운의 변동 (교체)

모음조화 :

양성 모음 (ㅏ,ㅑ)

음성모음(ㅓ,ㅕ,ㅡ)

중성모음(ㅣ)

끼리 끼리 어울리는 현상

깍아 깍아서 깍아도 깍아라 깍았다.

알록달록, 얼룩덜룩, 사랑사랑

현대 오면서 문란해짐. (아름다워, 오순도순(오손도손))

음운의 변동(축약)

자음 축약 :

ㄱ ㄷ ㅂ ㅈ + ㅎ => ㅋ ㅌ ㅍ ㅊ

좋다 => 조타

좋고 => 조코

잡히다 => 자피다.

먹히다 => 머키다.

모음축약 :

ㅣ + ㅓ => ㅕ

ㅏ + ㅓ => ㅗ

ㅓ + ㅓ => ㅖ

가리어 => 가려

오아서 => 와서

두었다. => 뒀다.

특별 예) 파이다

파이어 => 파여 , 패여서 (패여서 X)

음운의 변동(탈락)

“ㄹ” 탈락 :

1) 파생어와 합성어에서(명사) 형태소의 끝소리 ㄹ 이 ㄴ ㄷ ㅂ ㅈ (나도사자)앞에서 탈락

불나비 => 부나비

바늘질 =>바느질

딸님 => 따님

겨울내 => 겨우내

2)용언의 어간의 끝소리 “ㄹ” 이 'ㄴ ㅂ ㅅ 오 (나보소오)" 앞에 예외 없이 탈락

날다 => 나는, 납니다.

살다 => 사는, 삽니다.

놀다 => 노는 , 냅니다.

음운의 변동(탈락)

ㅎ 탈락 :

모음으로 시작되는 어미를 만나면 어간의 ㅎ 탈락

좋은 => 조은

낱은 => 나은

놓아 => 노아

쌍이다 => 싸이다

하 줄임말:

하 앞이 안울림 소리(ㄱ, ㄷ, ㅂ)면 하 전체가 떨어져 나감

울림소리면 ㅏ 만 떨어져 나감

예) 편안하지 않아 => 편안ㅎ지 않아 => 편안치 않아 ㄴ(울림)

섭섭하지 => 섭섭지 ㅂ(안울림)

팁 : ㅎ이 나오면 둘중 하나 축약/탈락

음운의 변동(탈락)

“ㅅ” 탈락 (표기도 적용)

ㅅ이 모음 어미 앞에서 탈락

잇+ 어 = 이어

긋+ 어 = 그어

벗 + 어 = 벗어 (불규칙)

모음탈락

동음 탈락 : 연결된 동음중 뒤 모음이 생략

가 + 아서 => 가서

서었다. => 섰다.

가아 => 가

음운의 변동(첨가)

사잇소리현상:

전제조건은

합성어(합성명사) 앞말이 울림소리면 (ㄱ ㄴ ㄹ ㅇ 만누라야)

합성어에 우리말이 하나라도 있어야함. (100% 한자는 안됨 시옷을 표기 할수 없음)

예) 장미 빛 = 장미뻗 = 장밋빛(표기) 사잇소리 현상

전세+집 = 전세찜 = 전셋집 사잇소리 현상

제사+날 =>제산날(뒷말이 ㄴ ㄹ 시작) 제삿날

이+몸 => 잇몸

기타 변동

두음법칙 :

리치 => 이치

량심 => 양심

력사 => 역사

ㄴ ㄹ + ㅣ ㅑ ㅓ ㅕ ㅠ = 탈락

ㄹ => ㄷ ㅌ ㄱ ㄴ ㄷ 앞에서 ㄴ으로 교체 될때 있음

낙원 내일 누각 노인

한국어의 문법 (형태론)

형태소란?

단어 : 자립해서 쓰이는 최소의 말
돌다리 , 맨발, 뛰었다.
+ 자립해서 쓸수 없는 말 , 쉽게 분리되는 말 (조사)

형태소 : (소) 원소의 소 (본질 바탕) 형태를 이루는 바탕
: 최소의 의미를 가진 말
일정한 뜻(어휘적 의미,문법적 의미) 을 가진 가장 작은 말의 단위
더 나누면 뜻이 없어지는 단위

예) 돌다리 는 하나의 단어
돌+다리 => 돌다리
단어는 1개 형태소는 2개

기준	종류	예시
자립성에 따라서	자립 형태소	명사,대명사,수사,관형사,부사,감탄사
	의존 형태소	조사,용언의 어간과 어미,접사
의미에 따라	실질 형태소 (어휘 형태소)	자립형태소 전부, 용언의 어간
	형식 형태소 (문법 형태소)	용언의 어미,조사,접사

한글의 품사체계

기능에 따른 분류	고유한 명칭에 따른 분류	형태의 변화 여부에 따라서
체언	명사	불변어(서술격 조사 예외) ~이다, ~이고
	대명사	
	수사	
관계언	조사	
수식언	관형사	
	부사	
용언	동사	가변어
	형용사	
독립언	감탄사	불변어

용어 정리

어간 :

활용할때 변하지 않는 부분 (오직 동사,형용사)

예)

자다

자고

자니

자서

어간 : 자-

날아가다

날아가고

날아가니

날아가서

어간 : 날아가-

어간이 다르면 다른 단어

예) 먹다, 먹이다

먹/ 먹이/ 다른 단어임 (파생단어)

어미:

용언 등의 어간에 붙어서 쓰임에 따라 변하는 부분 (어미가 변하는건 활용)

어근:

한 낱말에서 실질적인 의미를 나타내는 부분

말의 뿌리 (이 단어가 어떻게 생성되었는가?)

어간+어미 = 동사 아니면 형용사

어근이나 접사(접두사,접미사) = 합성어(단어의 형성법),파생어 파트

합성어 : **흑형**

파생어 : 욕심**쟁이**

접사

독립적으로 쓰이지 못하고 어근이나 어간에 붙어서 의미나 품사를 바꾸는 기능

1) 접두사 : 어근의 앞에 붙어서 그 의미를 한정

군소리,**헛**소리,**잔**소리,**덧**버선

2) 접미사 : 어근의 뒤에 붙어서 의미를 한정

멋**쟁이**,지우**개**,덮**개**

명사의 특징

수식을 받을 수 있다. (관형사의~)

격조사(자격,문장성분을 결정)를 취할 수 있다.

주격조사,목적격조사,보격조사,관형격조사,주사격조사,서술격조사,호격조사.

지훈이가 (주격)

지훈이를 (목적격)

지훈이가 강사가(보격조사) 되었다. (보어는 (되다,아니다 앞에 나오는 이/가))

지훈이의 (부사격)

지훈에게 (부사격)

지훈이다, (서술격조사)

훈이야 (호격조사)

격조사에 의해서 **문장의 성분**이 결정됨

고유명사 : 특정 하나의 개체를 다른 개체와 구별하기 위해 붙인 이름

복수접미사 **-들이** 붙으면 보통 명사가 됨

마크주커버거들이야...

보통명사 : 어떤 속성을 지닌 대상들에 두루 쓰이는 이름

명사의 특징(계속)

자립성 여부에 따라서

자립 명사: 혼자서 자립적으로 쓰일 수 있는 명사

의존 명사 : 의존명사 앞에 반드시 관형어로 꾸며줘야 쓸수 있는 말 (띄어 써야함)

예) 수 것 바 즐 터 만큼

네가 가져가야 할 것이 많다. (체언이기 때문에 조사가 붙을수 있음)

유의해야 할 것을 적었다.

네가 해야 할 것이다.(활용이 가능한 서술격 조사)

사람이나 짐승이나 다를 바가 있느냐?

그것이 내가 아는 바이다.

* 의존명사는 자립 형태소

서술성 의존 명사

예) 따름, 뿐, 리, 때문

그렇게 할 따름이다.

그렇게 되었을 뿐만 아니라

같이 있었기 때문이다.

너뿐만 아니라(체언+보조사)

주어성 의존 명사

예) 지 수 리

그것을 할 수가 없다.

내가 할 리가 있겠니?

어떻게 할지 모르겠다.(하다+지)

단위성 의존 명사

예) 평 개 마리 원 번 초 켄레

강아지 다섯 마리를 키운다.

대명사

명사의 특징을 가지지만 관형사의 수식을 받지 않음
(헌 그녀가, 새 너)

관형어의 수식은 된다. 먹는(관형어) 사람 / 먹는 그녀

지시 대명사 : 이것 그것 저것

인칭 대명사 : 1인칭 - 화자 / 2인칭 - 청자 / 3인칭 - 여기 없는 사람.

미지칭 : 이름이나 신분을 모를 때

의문문에 쓰이는

누구의 얼굴이 먼저 떠오르냐? 그것이 **무엇**인지 아니?

부정칭 : 특정 대상을 가르키지 않는 인칭 대명사 (아무 라는 말을 허용)

아무라도 응시 할수 있다. **누구**라도 상관없다. 먹을 것이라면 **무엇**이든 좋다.

재귀칭 : 한번 나온 명사를 다시 가르킬때 쓰이는 인칭대명사, 재귀 대명사

요즘 아이들은 **저**(3인칭 재귀대명사)만 안다니깐.

돌아가신 아버님은 소나무를 좋아하셨어 저 소나무도 **당신**께서 심으셨지

철수는 아직 어려서 자기만 알아

그는 선생님 앞에서 **자기** 멋대로 말했다.

수사

명사와 문법적 기능은 거의 같다.
조사와 결합 가능 (체언)
관형사의 수식을 받을 수 없다.
예외) 그 셋(수사)은 늘 붙어 다닌다.

수사의 종류

양수사(수량을 나타냄)

고유어 : 하나 둘 셋

한자어 : 일 이 삼 사

서수사(순서를 나타냄)

고유어 : 첫째, 둘째, 셋째

한자어 : 제일, 제이, 제삼 (계약서)

예외)

첫째가 사람(형제)일 때는 명사

하루, 이틀, 사흘, 나흘

날짜나 기간의 이름은 명사로 취급

조사

홀로 쓸수 없지만 이것 역시 단어 이자 형태소 - (어미는 단어는 X 형태소는 O)

조사는 생략이 가능

예) 공부 좀해라 책 줄 수 있니?

체언 용언 부사 뒤에 붙어 쓰임

가기(동사)도 잘(부사)도 간다.

너(체언)만(조사)을(격조사) 느끼며

체언뒤 : 학생은,비행기가

동사뒤 : 입어만(보조사) 보아라

형용사뒤 : 예쁘게도 생겼네

부사뒤 : 참 잘도 자는구나.

관형사 뒤에는 오지 못함.

- 조사와 어미의 차이는??

조사는 없어도 어절을 이룸

어미는 없으면 어절을 이룰수 없음.

예)

먹지도 못했다. (O)

먹지 못했다. (O)

먹 못했다. (X)

조사 (격조사)

조사의 종류는 (격조사, 보조사, 접속조사)

격조사 : 앞에 오는 체언이 문장 안에서 일정한 자격을 가지도록 하는 조사

1)주격조사 (이/가/께서/에서) 우리 학교에서 우승했다. (~에서 장소의 개념이 아님)

2)목적격 조사(을/를)

3)보격 조사(이/가)

4)서술격 조사(~이다) 나는 선생이고 너는 학생이다.

5)관형격 조사 : -의 (하나 밖에 없음)

나의 책 (소유 관계))

주권의 박탈 (목적어 서술어)

근대화^의 물결 (동격 관계) 내 마음^의 호수 (마음 = 호수)

6)부사격 조사 : -에서(처소),로(도구),으로(자격),에(원인),에게(상대),보다(비교),와(함께),처럼(비유)

철수는 학교에서(장소)) 공부한다.

코고는 소리에(에) 잠이 깼다.

동생이 형보다(비교) 낫다.

돌로(도구) 칼을 만들었다.

나는 영희와(함께) 도서관에 갔다.

7)호격조사 : 야 여 이시여 철수야, 신이시여.

조사(보조사)

보조사 : 선행하는 체언에 특별한 의미를 더해 주는 조사(속뜻을 가진)
보충을 하되 의미를 가지는 조사

빵을 먹어
=> **빵만** 먹어 (목적어 자리임)

-은/-는 (대조 및 화제 제시)
옷**은** 이쁘네.
오늘**은** 날씨가 맑아. (주격 조사는 은는 이가가 아님)

-만 (단독 only) : 그것만 해라
-도 (역시 also) : 그것도 좀 해라
-부터 (시작) : 그것부터 좀 해라
-마저,까지,조차 (극단) : 너마저 .. 너까지, 너조차
-만큼 (비슷한 정도) :하늘만큼 사랑해

접속 조사 : 두 단어를 같은 자격으로 이어 주는 구실하는 조사나

-와 -과 랑 하고
꽃게랑 새우랑
영희와 철수가 만났다. (접속 조사)
철수가 영희와 만났다. (부사격 조사)

높임 조사 : 높임 기능을 하는 조사

-요 : 제가 왔는데요. 이것좀 먹어요, ~하시지요, 먹어요,~해요 (생략이 가능)

친구가 아니요,(심표가 있으면 어미다.생략불가능) 형제입니다.
우리는 친구가 아니지요. 형제이지요
아니(어간) + 지(종결어미) + 요 (높임의 보조사)
우리 말에서 요는 종결 어미가 될수 없으면 종결 어미는 -오 다.

관형사

명사를 수식함
조사 결합 불가능
형태가 변하지 않는다.(활용이 안됨)

성상 관형사 : 성질 상태를 이야기함
옛 헌 새 온갖 모든

지시 관형사 : 어떤 대상을 가리키는
이 그 저 다른 여느 어느 이런 저런 그런

수 관형사 : 수량을 나타내는 관형사
한 두 세 네 다섯 엿 일곱

관형사와 다른 품사의 구별 (품사 통용)
1. 이 그 저 : 조사 결합이 가능하면 대명사, 불가능하면 지시 관형사

그보다 더 중요하다 (대명사)
그 사람 (관형사)
이 책에는 (관형사)
그와는 다른 사람이다 (대명사)
이보다 더 좋을 수 없다. (대명사)

2. 조사 결합 가능하면 수사 불가능하면 수 관형사
사람 일곱**이** 왔다. (수사)
일곱 사람 (관형사)
장미꽃 **한** 송이를 주세요(관형사)
다섯 사람이 왔다. (관형사)
사람 **다섯**이 왔다. (수사)

3. 다른 : 주어의 유무로 구별 (땀 을 넣어서 말이 되면)
개성이 **다른** 것 (형용사)
그는 구조가 **다른** 집을 짓는다. (형용사)
너 말고 **다른** 사람(관형사)
너는 **다른** 책을 가지고 오너라.

4.. 새 :

너무 바빠서 쉴 **새**도 없다. (명사)
새 책을 가지고 오너라.(관형사)

부사 (수식의 끝판왕)

용언을 수식
관형사도 수식
부사도 수식
활용은 안한다.
문장 전체를 수식하기도 함.

잘 잔다.
매우 잘 잔다.
참 잘하네
다 모여라
먼저 먹어라
가장 먼저 먹어라

-보조사와 결합 가능
잘도 잘다.
멀리도 가네

성상 부사

잘 바로 겨우 가장 먼저 더구나 매우 무척
상징 부사:의성어 의태어
꼬끼오, 꿈틀꿈틀

지시 부사

일찍이,장차,언제,아까

부정 부사

못 했다. 못했다.

문장 부사 : 문장 전체를 수식

과연, 분명히, 어찌 도리어 ,게다가 , 확실히

접속 부사

그러나 그러니까 하지만 더욱이 게다가 곧 즉 또 (수식 기능 없음)

동사

주어의 움직임이나 작용을 나타내는 단어

반드시 어간+어미 형태를 가진다.

- **먹어** **먹고** **먹으니** **먹게** **먹는** **먹자**

현재 시제를 나타내는 어미

-는다. / -ㄴ다 와 결합 가능 (형용사 안됨)

달린다. 간다.난다.늙는다.

명령형 청유형으로 활용 가능함.

던져라 ,던지자,날려라 날리자 , 가라 가자

동사의 종류

1. 자동사 : 움직임이 그 주어에만 관련된 동사 목적어가 필요 없음

아버지가 가신다. 그녀가 잔다.

2. 타동사 (영어에서도 마찬가지)

움직임이 다른 대상임

그녀가 영화를 본다.

아버지가 신문을 읽는다

형용사

주어의 성질이나 상태를 나타내는 단어의 부류
동사와 마찬가지로 어간+어미
현재 시제와 결합 X
명령 청유형 X

형용사의 종류

성상 형용사 : 성질을 나타내는
고요하다 예쁘다.

지시 형용사 : 지시성을 나타내는 형용사
그러하다.저러하다,어떠하다

보조용언

본용언에 문법적 뜻을 더하는 단어
혼자 쓰이지 못함.
반드시 띄어쓰기 함

먹어 **보아라**(보조용언)

먹어 **버려** (보조용언)

삶아 먹었다. (본용언 + 본용언)

~에서

~아서

를 넣어서 말이 되면 본용언+본용언

어미의 종류

- 어말어미

1. 종결어미

마침표, 느낌표, 물음표 앞에 오는것 (~다,~까,~구나)

2. 연결어미

쉼표 앞에 오는것 (대등하기(and but or) 너는 공부하고/ 공부했지만/ 공부하든지/
혹은 종속적으로 연결) 니가 공부하면/ 공부하려고/ 공부하니까/

3. 전성어미(성질을 바꾸어주는 어미) 품사는 변화 없지만 명사의 성질로(명사형)

먹기 먹음

관형사형, 읽는 먹는 가는

부사형으로 변경

- 선어말어미 : 어말어미 앞에 있는거 (무조건))

시제 선어말 (과거 “았었”, 현재 “는/ㄴ” 미래 “겠” 회상: “더”)

높임 선어말

추측 선어말

접미사의 종류

명사화 접미사

‘-이’, ‘-음/-ㅁ’ 이 붙어서 되는것
길이 , 깊이,높이,다듬이
걸음, 묶음, 믿음

용언화 접미사

‘-하다’, ‘-이다’
게임하다. (체언+용언화접미사)
딱하다
착하다
눅눅하다.

부사화 접미사

(나란하다) 나란히(넉넉하다) 넉넉히(무던하다) 무던히(속하다) 속히(뚜렷하다) 뚜렷이(버젓하다) 버젓이사

용언의 활용

- 1) 규칙 활용 : 어미가 붙었을때 어간 어미의 형태가 유지
달라진다해도 규칙으로 설명 가능

- ㄹ 탈락 (100% 적용)

살다 => 사니

울다 => 우는

- ㅡ 탈락 (100% 적용)

쓰다 => 써

모으다 => 모아

- 2) 불규칙 활용 : 어간 어미의 기본 형태가 바뀌고 규칙으로
설명할수 없는 경우

어간 불규칙

어미 불규칙

어간+어미 불규칙

ㅅ 불규칙 : ㅅ이 모음 어미 앞에서 탈락

짓다 => 지어

긋다 => 그어 ex)벗어, 웃어

ㄷ 불규칙 : ㄷ 이 모음 어미 앞에서 ㄹ로 변함

듣다 => 들어

걷다 => 걸어 ex) 얻어, 걷다

ㅂ 불규칙 : ㅂ이 모음어미 앞에서 오/우로 변함

돕다=> 도와

굽다 => 구워 ex)잡아,굽어

ㄹ 불규칙 => ㄹ가 모음 어미 앞에서 'ㄹ'탈락 + ㄹ 첨가

오르다 => 올라

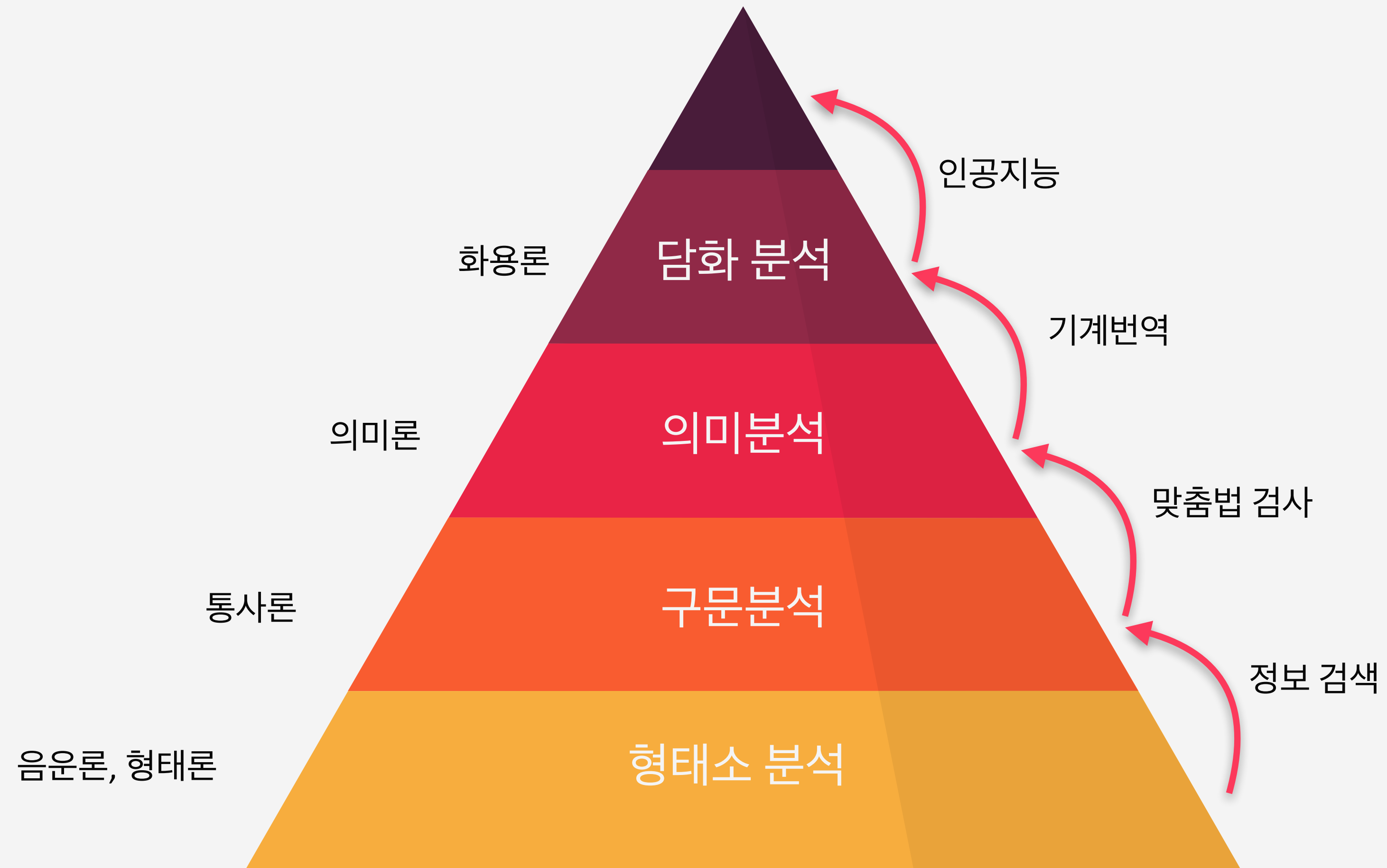
이르다 => 일러 ex)따르다 => 따라라

우 불규칙 => '-어'나 '-어'로 사직하는 어미를 만났을때 'ㄴ'가 탈락

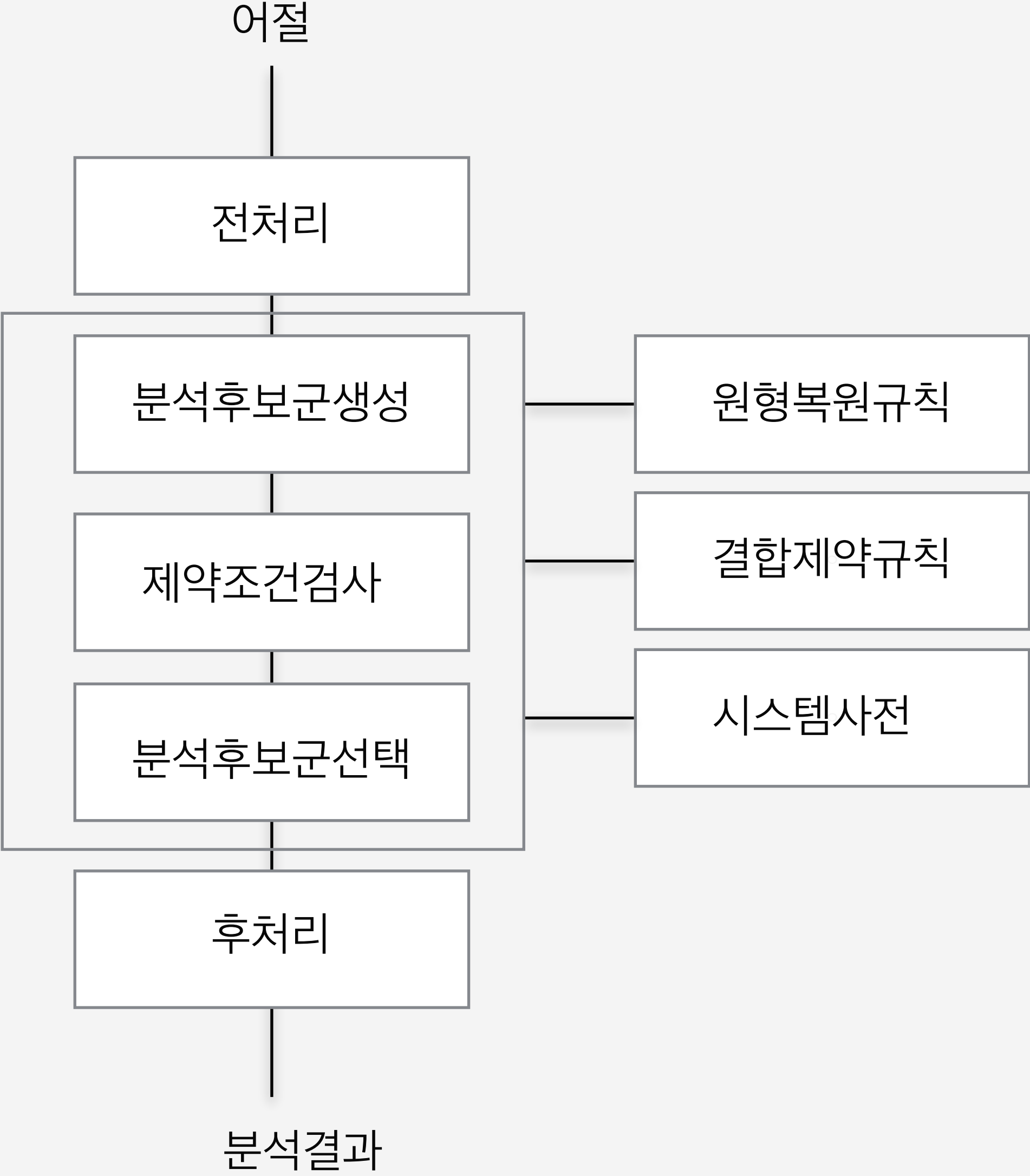
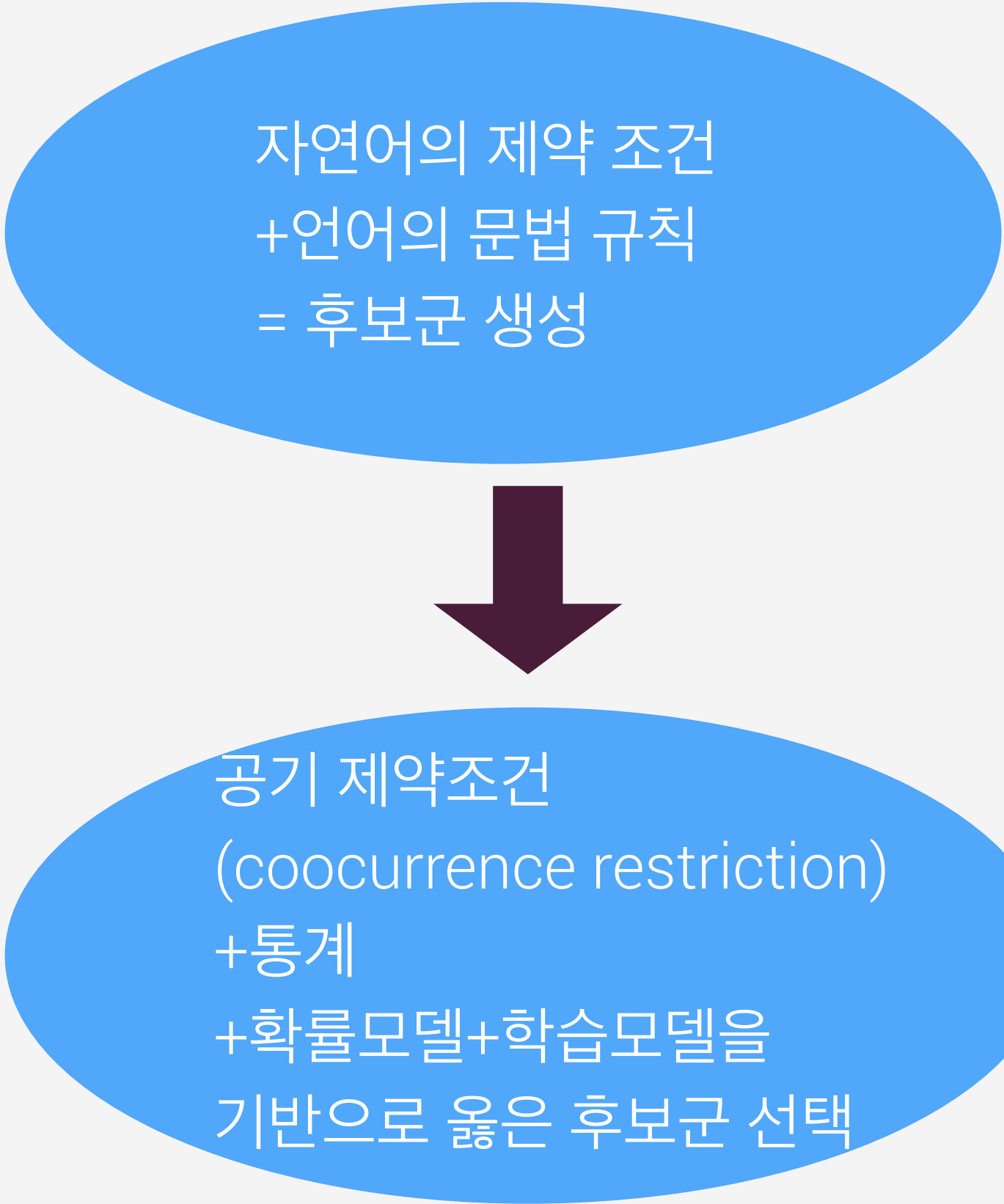
푸다 => 퍼 ex)이루다=>이루어

한국어 형태소 분석 기법

자연어 처리 기술의 계층도



형태소 분석의 기본 컨셉



어절의 형성 규칙

체언

체언+조사

체언+용언화접미사+어미 (사랑/하지/만)

체언+용언화접미사+'음/기' + 조사

체언+용언화접미사 +'아/어'+보조용언 + 어미

체언+'에서/에서부터' + '이' + 어미

용언 + 어미

용언+'음/기'+조사

용언+음/기+이+조사

용언+'아/어'+보조용언+어미

용언+'아/어'+보조용언+'음/기'+조사

부사,관형사,감탄사

부사+조사

체언+동사+어미

한글의 제약 조건

음운 제약 조건

‘-는’ 은 종성 자음을 가지지 않는 체언과 사용되는 조사이다.

먹는 -> 먹/NN/ + 는/JO/ (X)

감는 -> 감/NN/ + 는/JO/ (X)

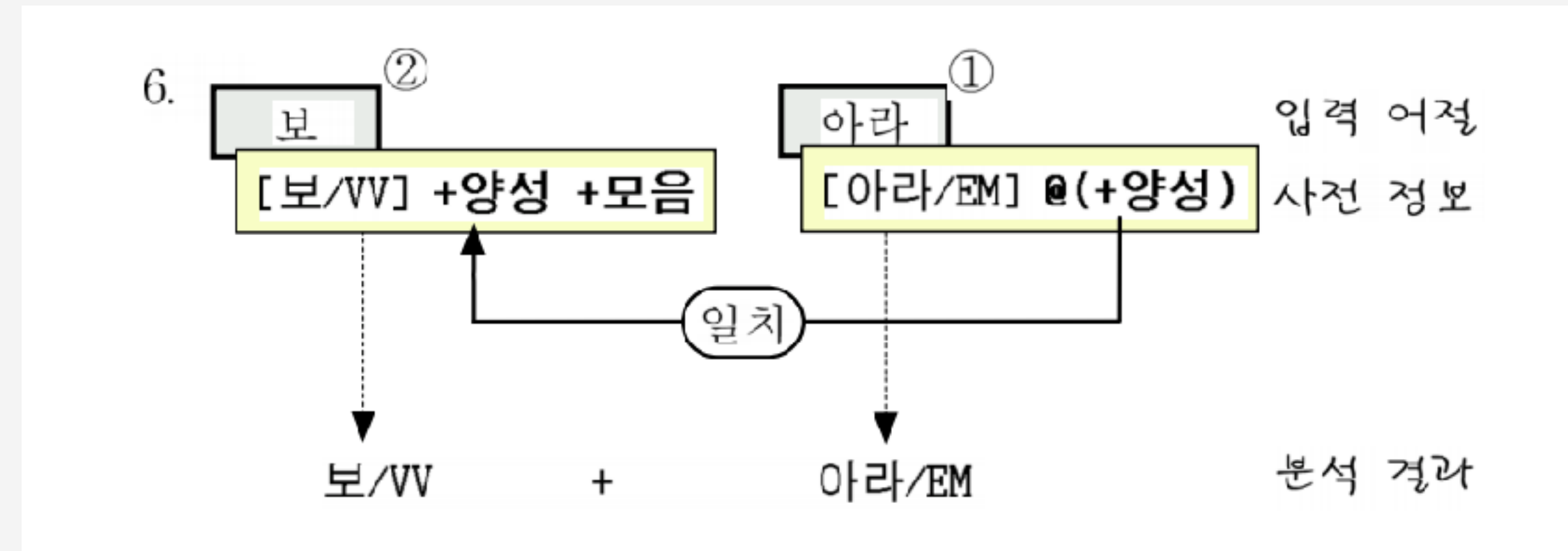
‘-는’ 은 모음으로 끝나는 단어만 쓸수 있다.

‘-은’ 은 자음으로 끝나는 단어만 쓸수 있다.

‘-아라’ 의 앞단어는 양성모음만 올 수 있다. (보아라(O),보어라(X))

‘-어라’ 의 앞단어는 음성모음만 올수 있다. (묵어라(O),묵아라(X))

이러한 모든 제약 조건을 사전에 수록



한글의 제약 조건

형태 제약 조건

게 : ([ㄱ게 /EM] @(+ㄱ))

예) 여기서 살^게

다 : ([다/EM] @(+어간 OR +ㅅ))

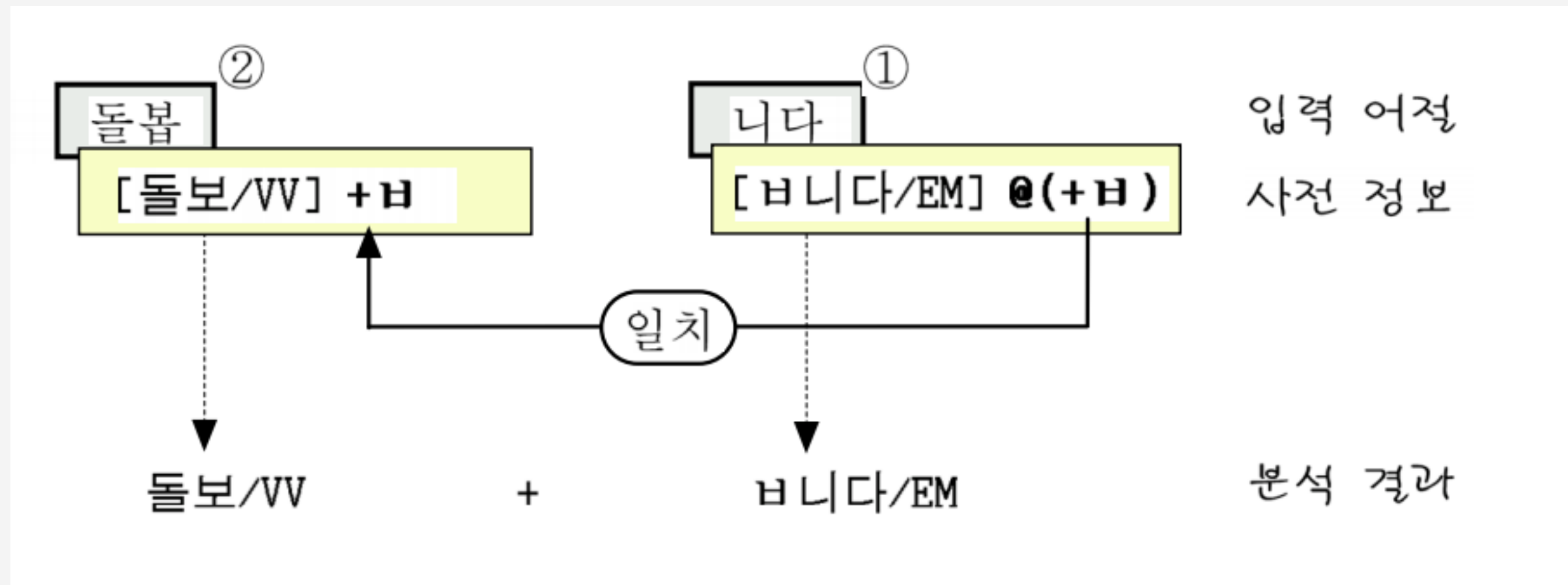
예) 내가 간^다. 내가 갔^{었다}.

([ㄴ다/EM] @(+ㄴ))

예) 나는 가져간^다.

세 : ((ㅁ세/EM] @(+ㅁ))

니다 : ([ㅂ니다/EM] @(+ㅂ))



한글의 제약 조건

품사 제약 조건

에서 : ([에서/JO] #(NN NP NU NX))
은 : ([은/JO] #(NN NP NU NX AD) @(+자음))
는 : ([는/JO] #(NN NP NU NX AD) @(+모음))
거나 : ([거나/EM] #(VV VX SV AJ AX SJ EP) @(+어간 ! +ㅅ))
는군요 : ([는군요/EM] #(VV VX SV EP) @(+어간))
군요 : ([군요/EM] #(AJ AX SJ EP) @(+어간 ! +ㅅ))

하 : ([하/VV] #(NN AD) +어간 +양성 +모음)
했 : ([하/VV + 았/EP] #(NN AD) +ㅅ +음성)
시 : ([시/EP] #(VV VX SV AJ AX SJ) @(+모음 +어간) +어간)
셨 : ([시/EP + 었/EP] #(VV VX SV AJ AX SJ) @(+모음 +어간) +ㅅ)

한글의 제약 조건

단일어 후보제약

1. 2음절 이상의 문법형태소가 분리된 어절은 단일어로 분석될 가능성이 낮다.

조사나 어미의 처음 두 음절이 자립형태소의 마지막 두 음절로 사용되는 어절이 존재하지 않거나 존재하더라도 이를 무시 할수 있다

‘-에서’,-부터’,-었다’ => 단일어 후보를 생성하지 않음

예외 (다시다, 시다)

2. ‘는/을/를/에’로 끝나는 어절은 단일어 혹은 독립언으로 분석될 가능성이 거의 없다.

문제가 되는 것은 자립형태소의 끝음과 같은 문법형태소

예)

국어 사전에서의 자립형태소중에 ‘는’,’를’로 끝나는 단어는 없다.

‘-을’의 경우 가을 고을 노을 마을 이 있지만 -을 결합 규칙을 위반한다.

예외는 ‘빌어먹을’,’젠장맛을’ 정도가 존재

한글의 제약 조건

문법 형태소 분리 제약

1. 2음절 이상의 조사(또는 어미)가 분리된 어절은 어미(또는 조사)가 분리될 가능성이 없다.

2. 문법 형태소를 분리할 때 최장 문법형태소가 분리된 분석 후보와
최장 문법형태소보다 1음절 짧은 문법형태소에 대한 후보만 생성한다.

(단일어 후보제약 1에 근거)

예) “학교에서만은” 분석

학교/에서만은

학교에/서만은 (에 앞에 누/멍/성/기/보/니/리 인 경우만 예외 처리)

학교에서/만은 (단일어일 가능성이 없다.)

학교에서만/은 (단일어일 가능성이 없다.)

한글의 제약 조건

어미 분리 조건

1. 규칙 용언과 규칙 어미

규칙 용언과 어간만 변하는 불규칙 요언의 어미부가 문법 형태소 사전에 어미로 등록되어 있으면 이를 어미로 추정

예) 죽다 => 죽고, 죽어,죽네

2. 'ㄴ/ㄹ/ㄱ/ㅁ'으로 시작되는 어미

어간부가 받침 'ㄴ/ㄹ/ㄱ/ㅁ'으로 끝나고 어미부가 문법 형태소 사전에 어미로 등록되어 있으면 어미로 추정

예)가다 => 가 + ㄴ 다

3. '하여/거라/너라' 불규칙 어미

어간부의 마지막 음절과 어미부의 첫부분이 각각

'하-'와 '-여-',

'가-'와 '-거라',

'오-'와 '-너라' 이면 어미부를 어미로 추정

한글의 제약 조건

4. '아/어/가' 탈락되는 어미

어간부 끝음절이 '아/어'가 아니고 ('아/어'+어미부)가 문법형태소 사전에 어미로 등록되어 있으면 '아/어'가 생략된 어미로 추정
예) 가+ 았다 => 갔다

5. '아/어'로 시작되는 어미의 변이체

어간부의 끝음절의 중성이 'ㅏ/ㅓ/ㅕ / ㅗ / ㅛ / ㅛ / ㅛ / ㅛ' 이고 ('아/어'+어미부)가 문법형태소 사전에 어미로 등록되어 있다면 어미로 추정
예) 주+었다 <=> 주었다 <=> 줬다
오+았다 <=> 오았다 <=> 왔다

6. '푸' 불규칙

어간부가 '퍼-'이고 ('아/어'+어미부)가 문법형태소 사전에 어미로 등록되어 있으면 '아/어'가 생략된 어미로 추정

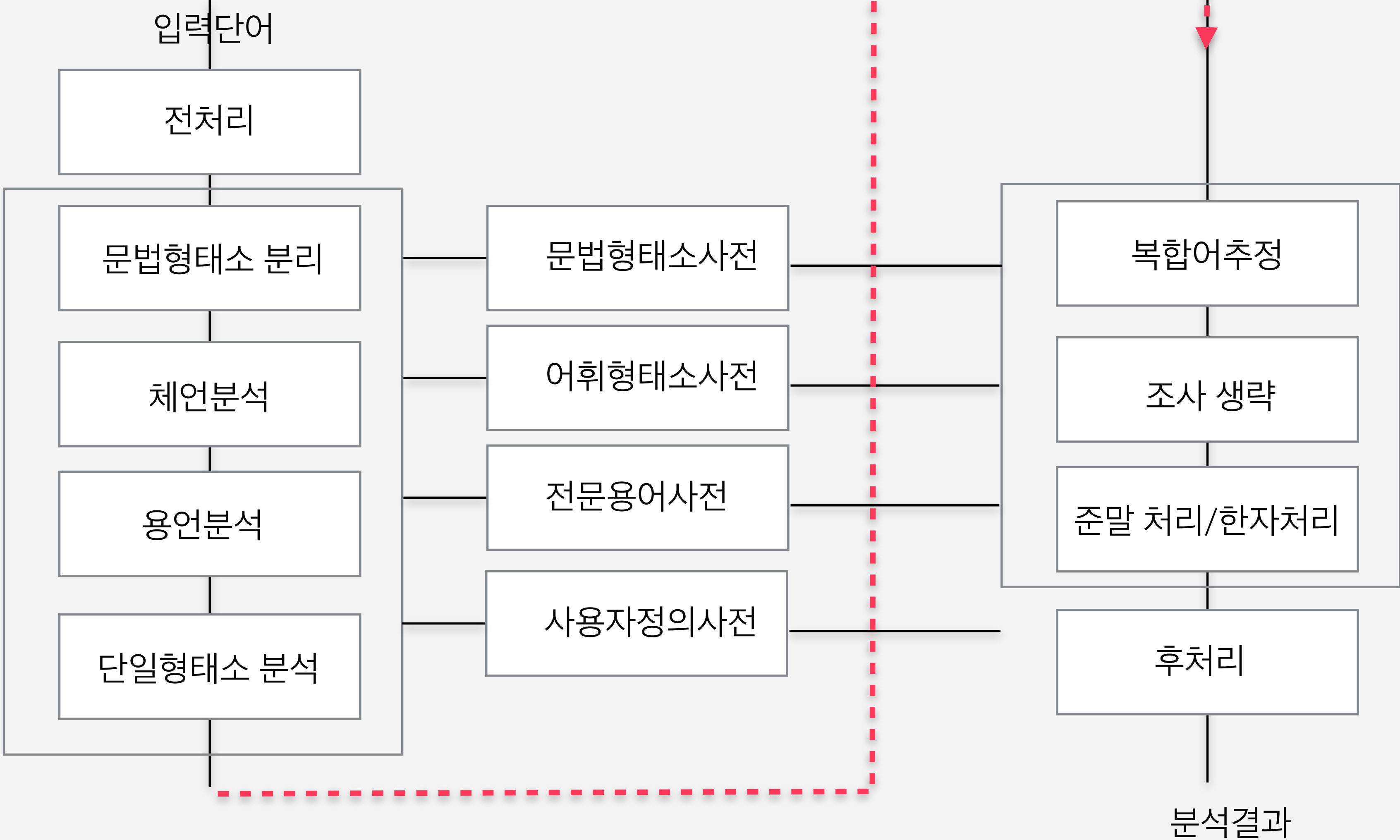
국어의 모든 불규칙이 깔끔히 정리되어 있음

<http://semanticweb.kaist.ac.kr/research/morph/oldDoc/irrword>

이 분리조건으로 용언 어절에 나타난 모든 어미 분리 가능

단 분리된 어간부는 선어말 어미를 포함하고 있음으로 선어말 어미를 다시 분리해야 함.

형태소 분석의 기본 컨셉



형태소 분석의 기본 컨셉

전처리 : 형태소 분석이 대상이 되지 않는 단어 분리하거나 제거
어절 단위로 토큰나이징

문법형태소 분리 : 단어 형성 규칙에 따라 문법형태소인 조사와 어미 분리

체언 분석 : 조사가 분리된 단어는 체언으로 추정하고 어근부가 체언이 아닌경우 접미사를 분리하고
다시 체언 분석시 명사형 전성어미가 발견되면 용언 분석을 실시

용언 분석 : 어미가 분리된 단어는 용언으로 추정되므로 어휘형태소 사전에서 이를 확인함
어근부가 용언이 아닌경우 용언화 접미사(-하다/-되다/-시키다)와 같은 접미사가 발견되면 체언 분석 시도

단일형태소 분석 : 조사나 어미가 발견되지 않는다면 부사어,감탄사,관형사인지 확인 한다.

복합어 추정 : 조사로 분리되었으나 어근부가 체언으로 분석되지 않으면 복합명사인지 확인

준말 처리 : 축약되거나 생략된 단어를 기분석 사전을 통해서 복원

분석 결과의 중의성의 유형

1. 어근의 중의성

‘자는’ => ‘자’(체언) + ‘는’(조사) / ‘자’(용언) + ‘는’(어미)

‘보고서’ => ‘보고서’(체언) / ‘보’(용언) + ‘고서’(어미)

2. 형태소 분리 단계에서의 중의성

‘녹음이’ => ‘녹음’(체언) + ‘이’(조사) / ‘녹’(용언) + ‘음’(어미) + ‘이’(조사)

‘가시는’ => ‘가시’(체언) + ‘는’(조사) / ‘가’(체언) + ‘시’(선어말어미) + ‘는’(어미)

3. 형태소 길이의 중의성

‘자라는’ => ‘자라’(용언) + ‘는’(어미) / ‘자’(용언) + ‘라는’(어미)

‘경기도’ => ‘경기도’(체언) / ‘경기’(체언) + ‘도’(조사)

4. 불규칙 어간의 중의성(분리 위치는 같으나 원형복원이 다름)

‘가는’ => ‘가’(용언) + ‘는’(어미) / ‘갈’(용언) + ‘는’(어미)

‘나는’ => ‘나’(용언) + ‘는’(어미) / ‘날’(용언) + ‘는’(어미)

5. 서술격 조사의 중의성

‘사과라고’ => ‘사과’(체언) + ‘라고’(조사)

/ 사과(체언) + 이(서술격조사) + 라고(어미)

6. ‘아/에’ 복원 중의성

‘가요’ => ‘가’(용언) + ‘요’(어미)

‘가’(용언) + ‘어요’(어미)

‘경찰서는’ => 경찰서(체언) + 는(조사)

경찰서(체언) + 에서는(조사)

7 품사 자체의 중의성

‘곳’ => 명사, 의존명사

중의성 해결 방법

1. 가중치 부여법

원칙1 **형태소 길이 비례 원칙**

원칙2 **형태소 개수 비례의 원칙**

원칙3 **원형 보존의 법칙**

일반적으로 축약 탈락된 결과의 선호도는 낮음

축약 탈락에 의한 형태소의 변이체가 포함된 분석의 가중치 낮춤

원칙4 **통계적 선호도의 원칙**

중의성 있는 단어에 대해 통계적으로 선택 빈도가 높은 것에 가중치

2. HMM (은닉마르코프 모델)

3. CRF(conditional random field)

품사 태거로서의 HMM

현재 품사의 발생은 바로 이전의 품사에만 의존 (가정1)

현재 어절의 발생은 현재의 품사에만 의존(가정2)

HMM이 성립될 요소 $\{S, V, A, B, \pi\}$

- S : 상태 (품사) - 관측할수 없는 것
- V : 각 상태에서 일어날 수 있는 서로 다른 관측 심볼 (단어)
- A : 상태 전이 확률
- B : 관측 확률
- π : 초기 상태 확률

품사 태거로서의 HMM

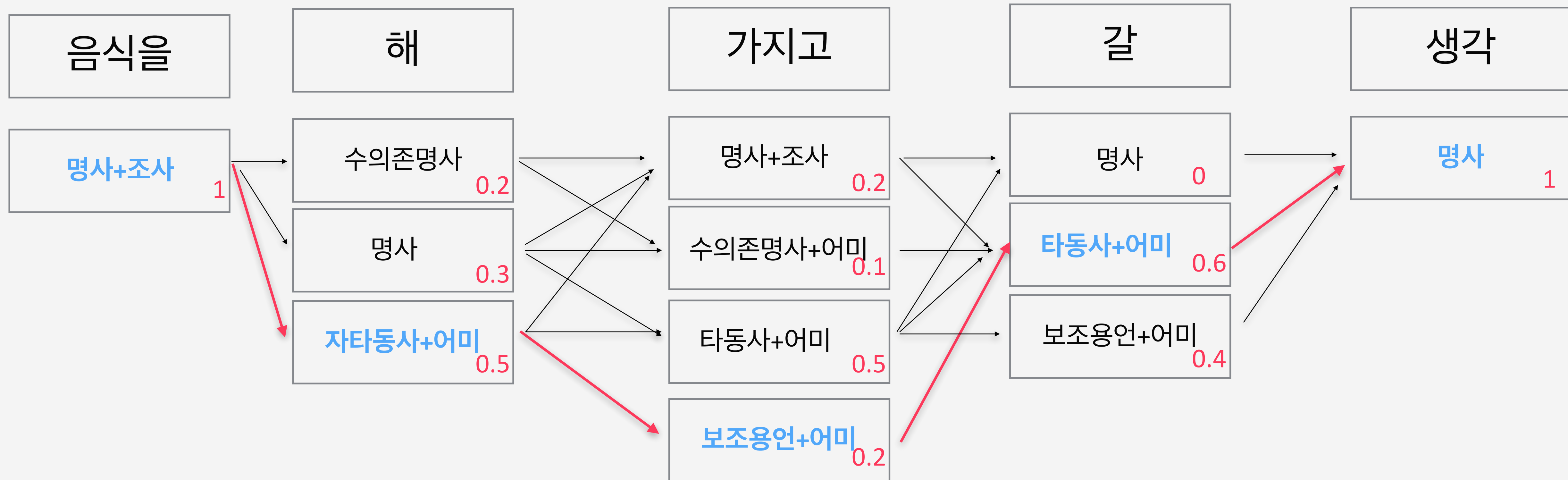
음식을	해	가지고	갈	생각
명사+조사 1	수의존명사 0.2	명사+조사 0.2	명사 0	명사 1
	명사 0.3	수의존명사+어미 0.1	타동사+어미 0.6	
	자타동사+어미 0.5	타동사+어미 0.5	보조용언+어미 0.4	
		보조용언+어미 0.2		

품사 태거로서의 HMM

앞 어절들과의 관계 중 가장 확률이 높은 것을 선택

$$S_{i+1a} = \operatorname{argmax}(S_{i-1} * P_i * P_{i-1a})$$

계산이 끝난후 Back Track 하면서 선택



CRF (Conditional Random Field)

시퀀스 데이터에서 하나의 시퀀스로 판단하는 것이 아니라 이전과 이후를 참조하여 지금의 상태를 결정하는 것

*HMM과의 차이

HMM은 이전 품사열 몇개만 고려 (멀리 떨어져 있는 단어의 의존 관계는 힘들다)

CRF는 좌우의 여러 특징을 고려하여 특징 함수(feature function)를 정의

예) “이 단어의 첫 글자가 대문자이면 명사일 확률이 1이다”

“앞 단어가 the이면 망사일 확률이 1이다.”

각 특징 함수들은 가중치를 가짐

학습 데이터를 통해서 그 가중치의 최적값을 찾아냄 (회귀 분석과 비슷한 면모)

복합명사 인식의 문제

단어 = 단일어 + 복합어 (파생어 + 합성어)
문장에서 나타나는 미등록어의 대부분은 복합명사

사전에서 추출한 복합명사의 구성

음절의길이	개수	백분율
4음절	13,308	73.1
5음절	3174	17.4
6음절	1124	6.2
7음절	403	2.2
8음절이상	195	1.1

복합명사 인식의 문제

4음절 복합명사 구조

복합명사구조	개수	백분율
2+2	12039	90.5
미등록+2	660	4.9
2음절+미등록	447	3.4
미등록+미등록	162	1.2

6 음절 복합명사 구조

복합명사구조	개수	백분율
2+2+2	429	38.2
3+3	164	14.6
2+4	74	6.6
4+2	62	5.5
미등록+미등록	395	35.1

5 음절 복합명사 구조

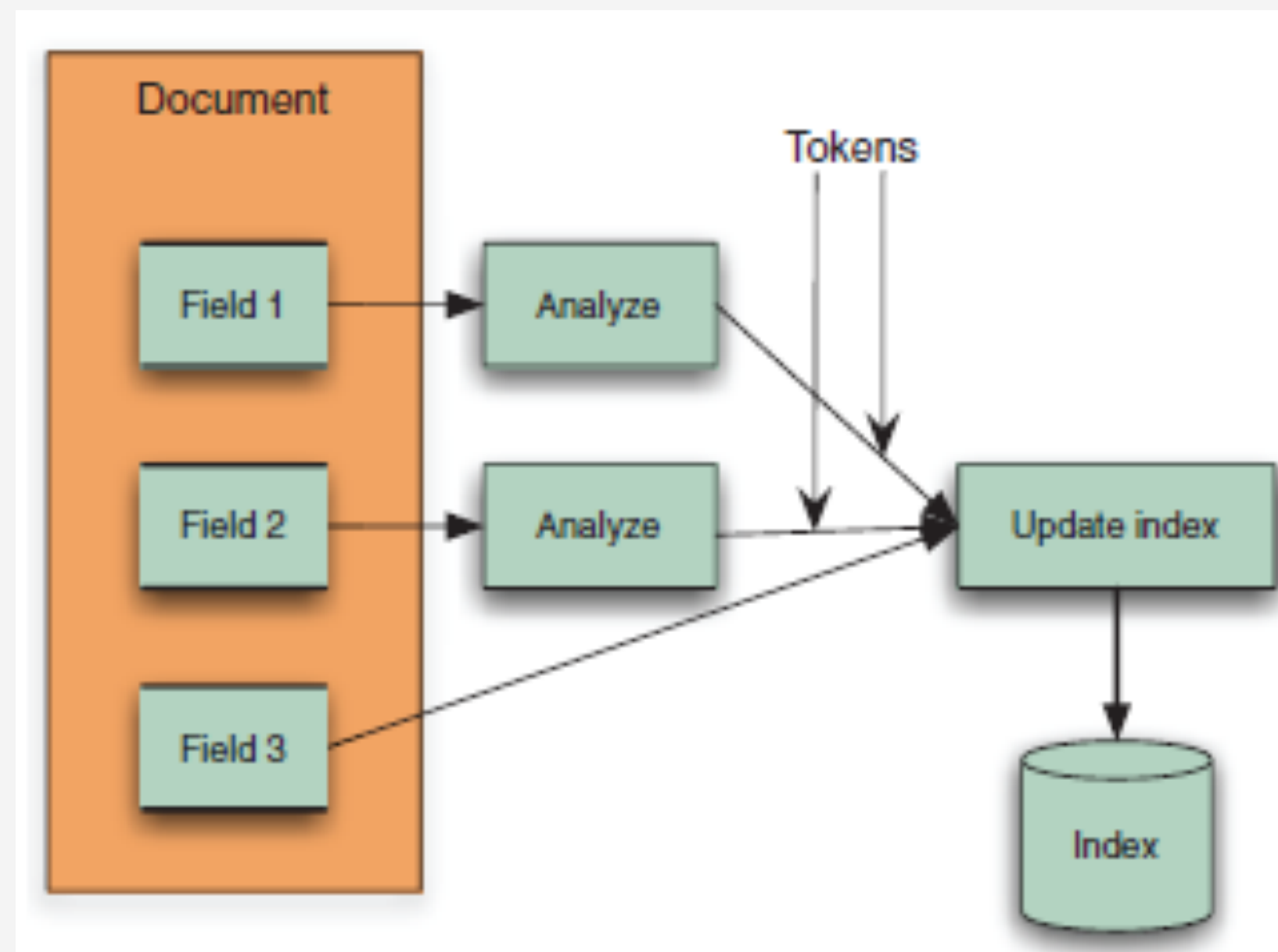
복합명사구조	개수	백분율
2+3	1242	39.1
3+2	734	23.1
2+미등록	838	26.4
미등록+2	221	7.0
미등록+2	88	2.8
미등록+미등록	51	1.6

통계적 빈도가 높은 구성 패턴을 우선 검사
6음절 이상의 복합명사는 좌방향 최장일치된 명사를 우선 생성

루씬에서의 형태소 분석기

루씬에서의 형태소 분석의 개요

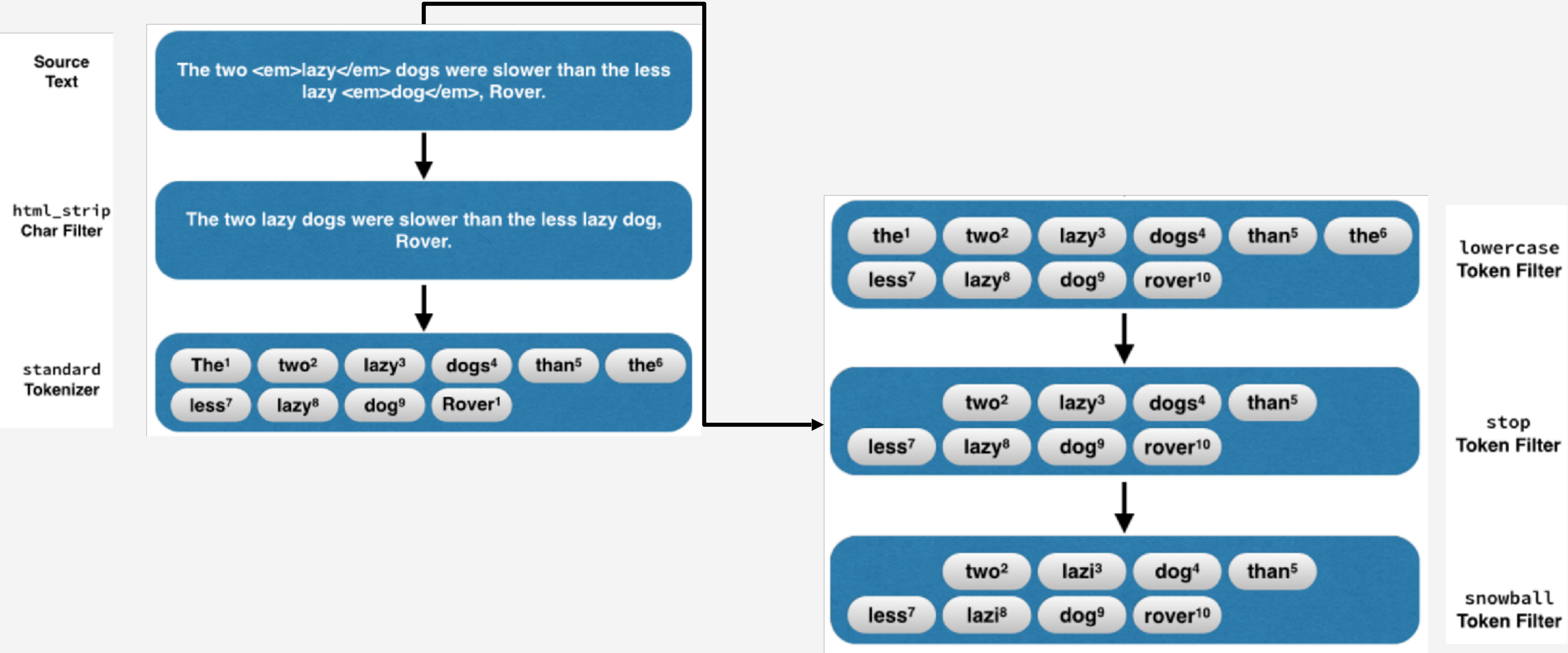
- Text Analysis는 색인어를 추출하는 과정
- Text Analysis는 문장을 Term들로 변환하는 것
- Term = “필드명 + Token”
- Token은 색인어 추출, 소문자 치환, 불용어 제거 등의 과정을 거쳐 텍스트로부터 쪼개진 조각들



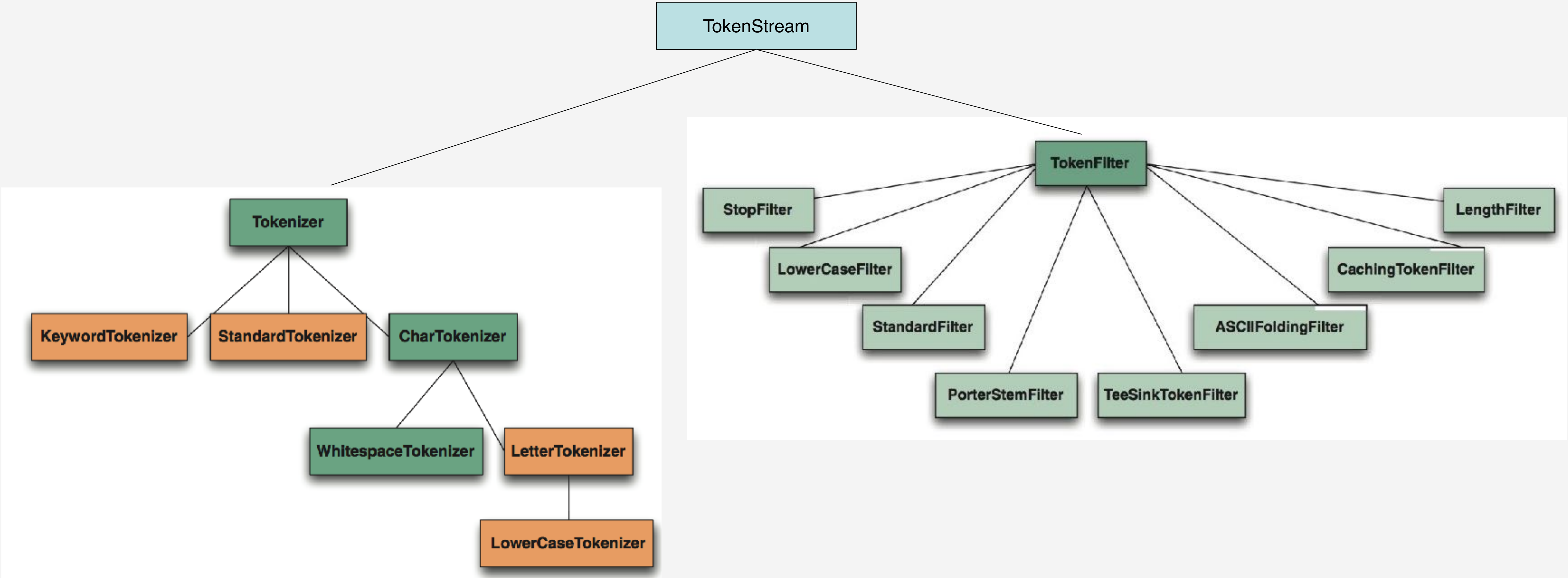
[Analyzer chain]

[Analysis Process]

Analyzer chain 흐름도




클래스 다이어그램



Solr와 E/S에서의 형태소 분석

Solr 관리화면의 Analysis 메뉴에서 색인어 추출 과정을 확인할 수 있음



Dashboard

Logging

Core Admin

Java Properties

Thread Dump

nas

Overview

Analysis

Dataimport

Field Value (Index)

The two lazy dogs are slow

Analyse Fieldname / FieldType: text_en

ST	The	two	em	lazy	em	dogs	are	slow
SF		two	em	lazy	em	dogs		slow
LCF		two	em	lazy	em	dogs		slow
EPF		two	em	lazy	em	dogs		slow
SKMF		two	em	lazy	em	dogs		slow
PSF		two	em	lazi	em	dog		slow

Field Value (Query)

The two lazy dogs are slow

☐ Verbose Output

Analyse Values

ST	The	two	em	lazy	em	dogs	are	slow
SF	The	two	em	lazy	em	dogs	are	slow
SF		two	em	lazy	em	dogs		slow
LCF		two	em	lazy	em	dogs		slow
EPF		two	em	lazy	em	dogs		slow
SKMF		two	em	lazy	em	dogs		slow
PSF		two	em	lazi	em	dog		slow

E/S는 REST API로 확인

← → ↻ ⓘ 192.168.0.165:49200/blog/_analyze?analyzer=korean&text=문재인%20대통령의%20

```
{
  "tokens" : [
    {
      "token" : "문재인/N",
      "start_offset" : 0,
      "end_offset" : 3,
      "type" : "N",
      "position" : 0
    },
    {
      "token" : "대통령/N",
      "start_offset" : 4,
      "end_offset" : 7,
      "type" : "N",
      "position" : 1
    },
    {
      "token" : "특사/N",
      "start_offset" : 9,
      "end_offset" : 11,
      "type" : "N",
      "position" : 2
    },
    {
      "token" : "일본/N",
      "start_offset" : 13,
      "end_offset" : 15,
      "type" : "N",
      "position" : 3
    },
    {
      "token" : "방문/N",
      "start_offset" : 17,
      "end_offset" : 19,
      "type" : "N",
      "position" : 4
    }
  ]
}
```



아리랑 형태소 분석기

2008년 : Lucene Korean Analyzer 공개 (<http://cafe.naver.com/korlucene>)

2009년 : 공개SW 개발자 대회 수상

2013년 : Apache Lucene Project 기부

<https://github.com/korlucene/>

 Lucene Core / LUCENE 4956 1 of 10
Return to search

the korean analyzer that has a korean morphological analyzer and dictionaries

Edit Comment Agile Board More Actions Resolve Issue Close Issue Workflow Share Views

Details

Type: New Feature
Priority: Major
Affects Version/s: 4.2
Component/s: modules/analysis
Labels: newbie
Lucene Fields: New

Status: Open
Resolution: Unresolved
Fix Version/s: None

People

Assignee: Christian Moon
Reporter: SoeMyung Lee
Vote (0) Watching (11)

Dates

Created: 25/Apr/13 04:56
Updated: 14/Aug/13 12:40

Agile

View on Board

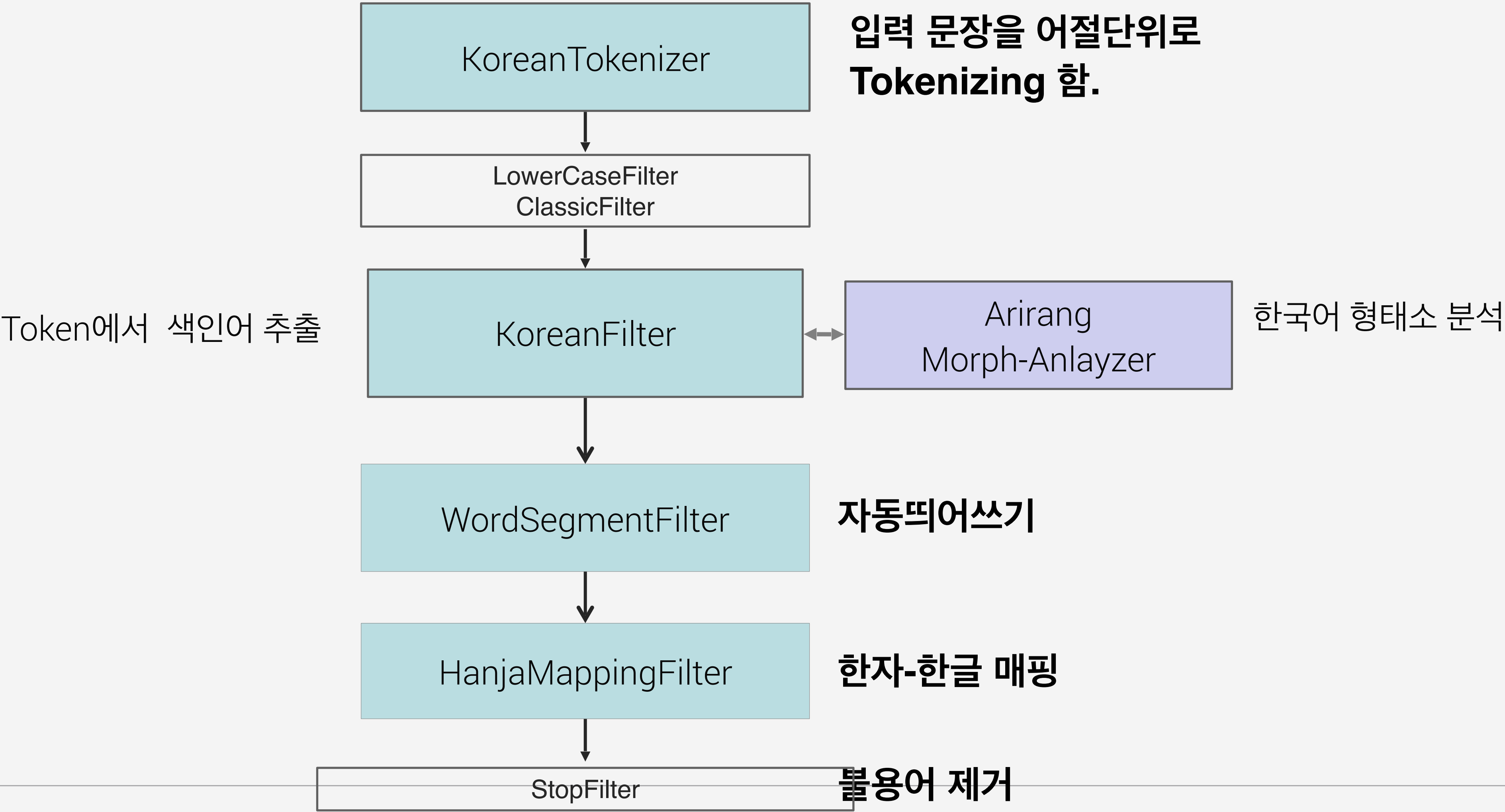
Description

Korean language has specific characteristic. When developing search service with lucene & solr in korean, there are some problems in searching and indexing. The korean analyzer solved the problems with a korean morphological analyzer. It consists of a korean morphological analyzer, dictionaries, a korean tokenizer and a korean filter. The korean analyzer is made for lucene and solr. If you develop a search service with lucene in korean, it is the best idea to choose the korean analyzer.

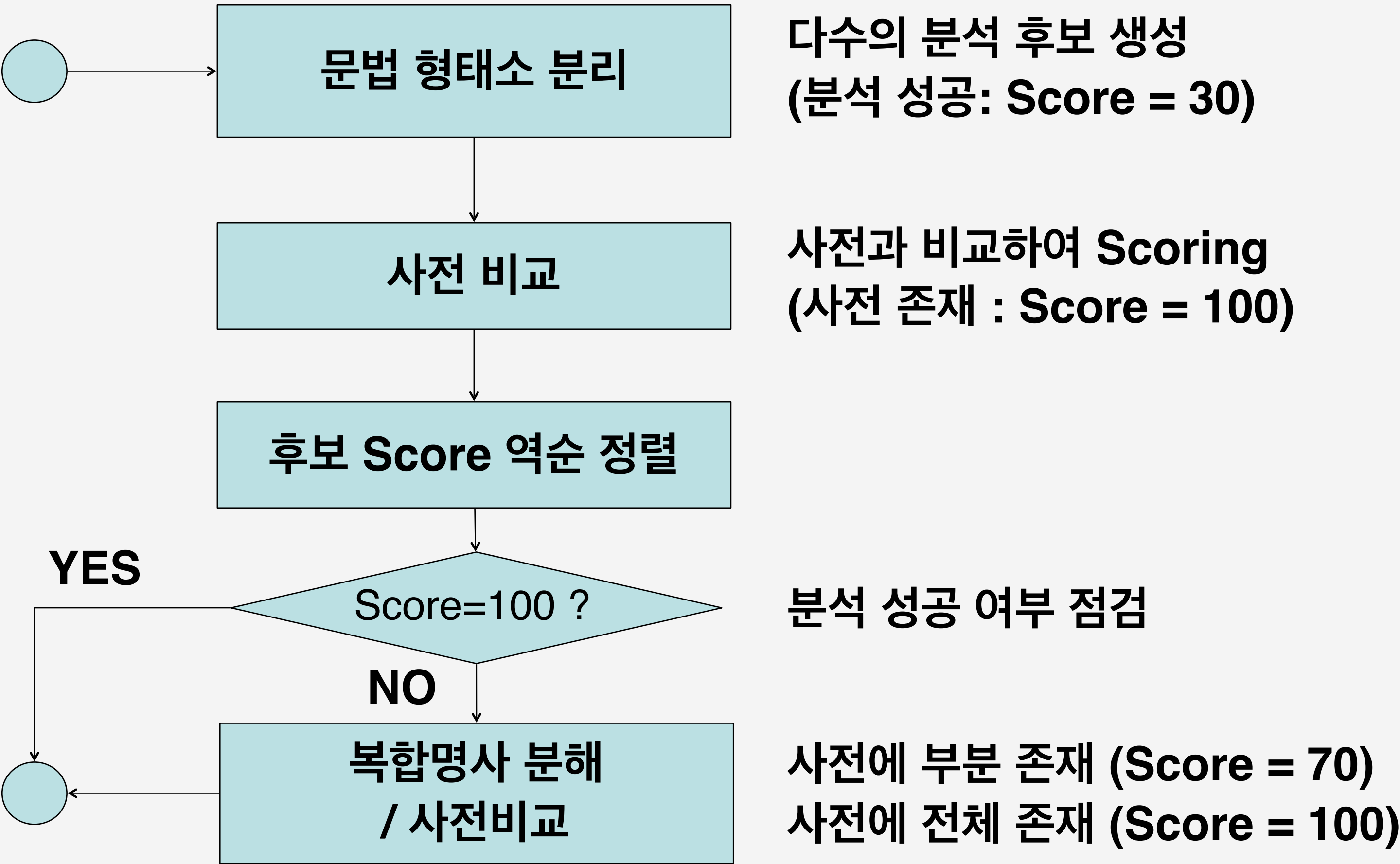
Attachments

kr-analyzer-4x.tar	2.12 MB	25/Apr/13 04:56
lucene4956.patch	25 KB	13/May/13 11:05
LUCENE 4956.patch	2.23 MB	14/Aug/13 09:05

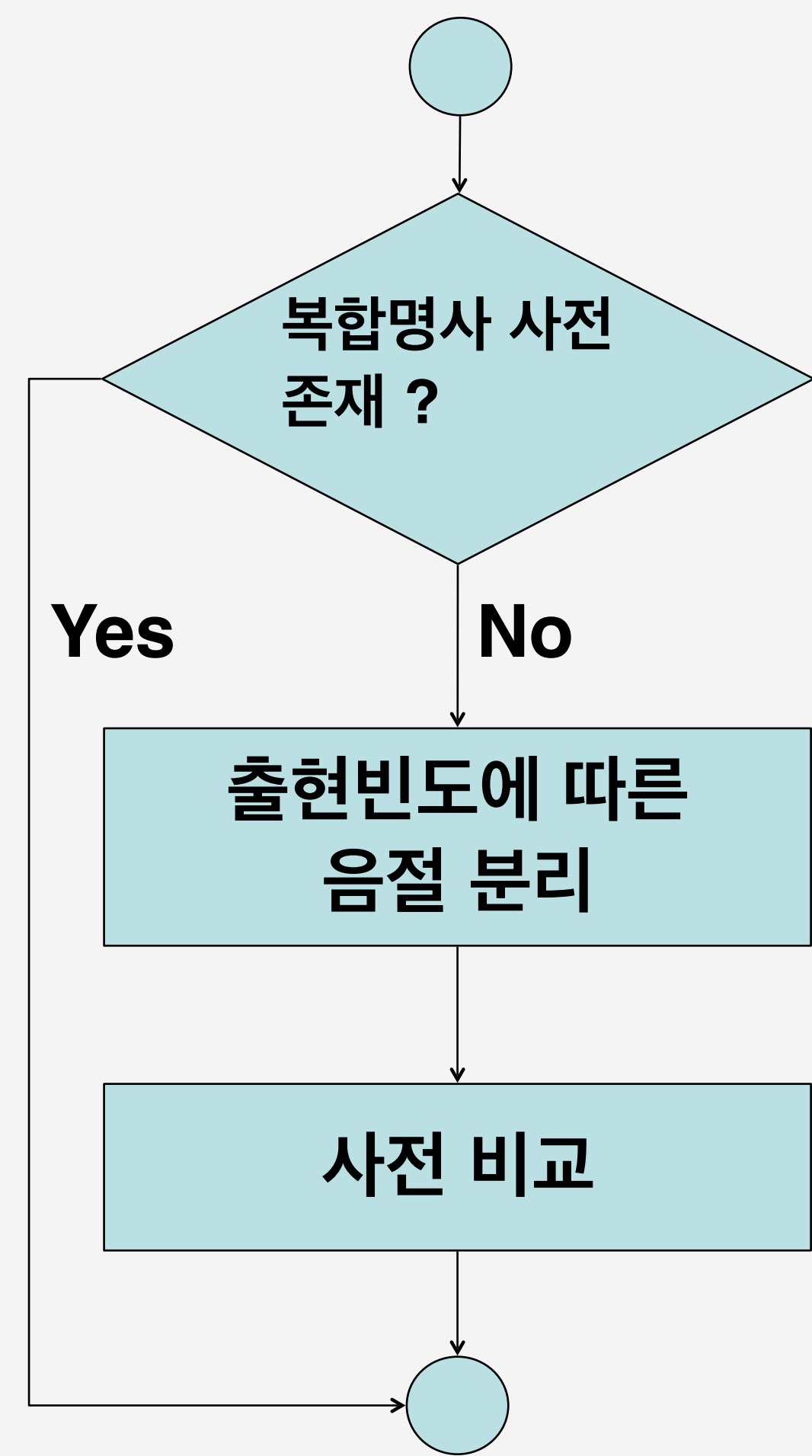
아리랑 형태소 분석기 구조



아리랑 형태소 분석기 형태소분석 흐름



아리랑 형태소 분석기 복합명사 분해



복합명사 음절분리 규칙

- 1. 3음절 복합명사
2음절 + 1음절 / 1음절 + 2음절
- 2. 4음절 복합명사
1음절 + 3음절 / 1음절 + 2음절 + 1음절 / 2음절 + 2음절
- 3. 5음절 복합명사
2음절 + 3음절 / 3음절 + 2음절 / 4음절 + 1음절 / 2음절 + 2음절 + 1음절 / 2음절 + 1음절 + 2음절
- 4. 6음절 이상 복합명사
최장 명사를 기준으로 좌.우를 분리하여 1~4번 적용

형태소분석 패턴

- 분석패턴은 AnalysisOutput 클래스에서 확인할 수 있다.

- 분석 패턴의 종류는 다음과 같다.

1)PTN_N = 1; /* 체언 : N/PN/NM/XN/CN/UN/AS/HJ/ET */

2)PTN_NJ = 2; /* 체언 + 조사 */

3)PTN_NSM = 3; /* 체언 + 용언화접미사 + 어미 */

4)PTN_NSMJ = 4; /* 체언 + 용언화접미사 + '음/기' + 조사 */

5)PTN_NSMXM = 5; /* 체언 + 용언화접미사 + '아/어' + 보조용언 + 어미 */

6)PTN_NJCM = 6; /* 체언 + '에서/부터/에서부터' + '이' + 어미 */

7)PTN_NSMXMJ = 7; /* 체언 + 용언화접미사 + '아/어' + 보조용언 + '음/기' + 조사 */

8)PTN_VM = 11; /* 용언 + 어미 */

9)PTN_VMJ = 12; /* 용언 + '음/기' + 조사 */

10)PTN_VMCM = 13; /* 용언 + '음/기' + '이' + 어미 */

11)PTN_VMXM = 14; /* 용언 + '아/어' + 보조용언 + 어미 */

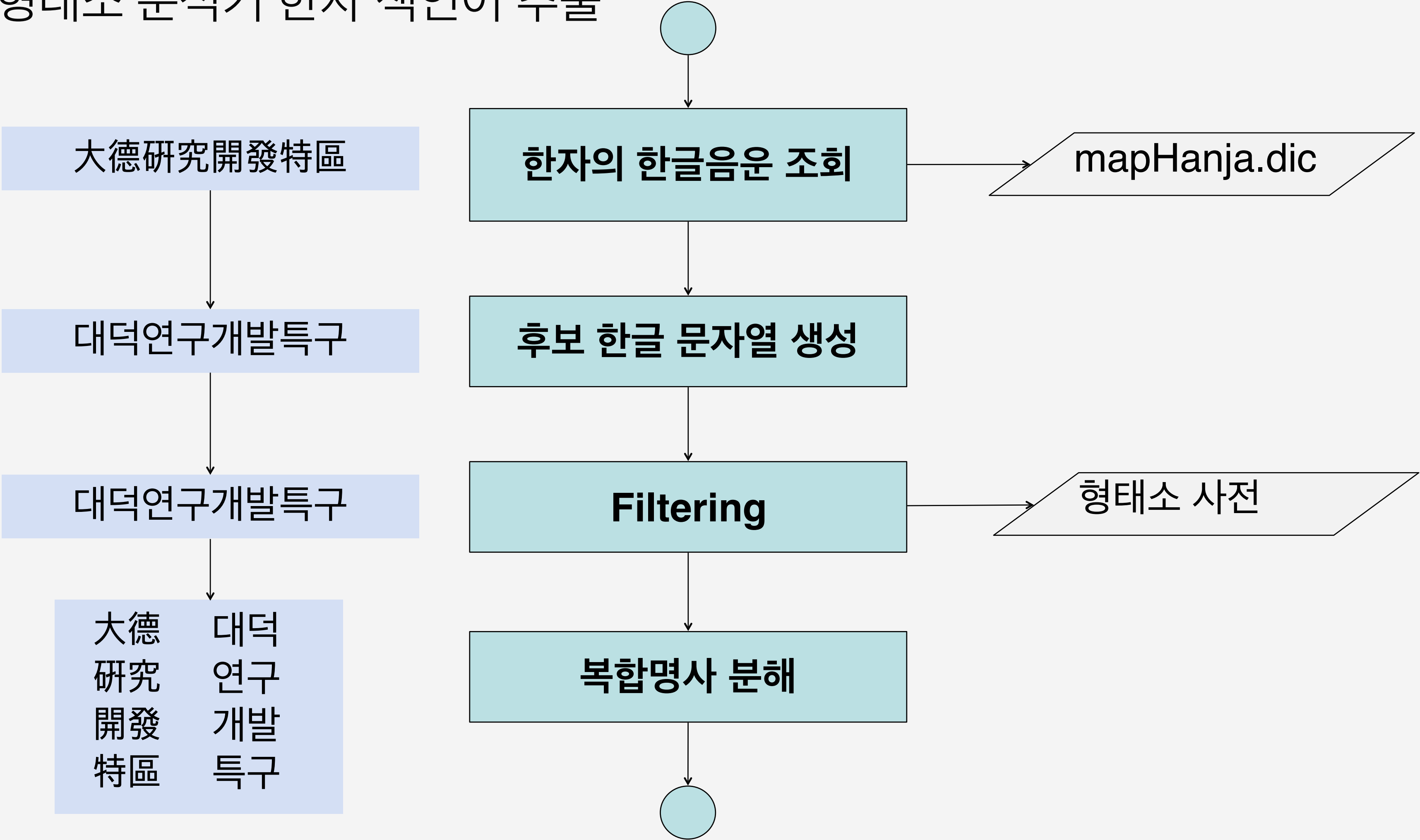
12)PTN_VMXMJ = 15; /* 용언 + '아/어' + 보조용언 + '음/기' + 조사 */

13)PTN_AID = 21; /* 단일어 : 부사, 관형사, 감탄사 */

형태소분석 결과 정렬

- 복수의 분석 결과 중에서 키워드 가능성이 높은 순으로 정렬
- 정렬은 다음의 순서에 따른다.
 - 1) Score 가 높은 분석 결과
 - 2) Score 100인 분석 결과 중에서는 체언 또는 독립언이 상위에 정렬
 - 3) 복합명사가 부분일치인 경우, 최장 단어의 길이가 큰 순으로 정렬
 - 4) 복합명사가 부분일치인 경우, 사전과 일치하는 길이가 큰 순으로 정렬
 - 5) 형태소 분석 패턴의 정의가 작은 순으로 정렬

아리랑 형태소 분석기 한자 색인어 추출



띄어쓰기

띄어쓰기가 가능한 곳인 조사, 어미 다음의 경우에 대한 처리
예외케이스인 “~할 수”를 뺀 나머지는 자동띄워쓰기 완성
입력된 문장의 전방에서 부터 조사와 어미가 될 가능성 조사
어미와 조사 규칙만으로 분석을 하면 오류 발생 가능성이 있음
분석된 결과를 도출 한 후 이전에 분석된 결과와 결합 할 수 없는 경우라면
기 분석된 결과를 제외한 나머지 후보로 다시 분석시도

아리랑 형태소 분석기 사용법

설치 방법 : 플러그인 형태로 설치

6.0 이전:

```
bin/plugin -install chanil1218/elasticsearch-analysis-korean/1.3.0
```

<https://github.com/korlucene/arirang-analyzer-es-plugin>

6.0 이후 :

```
$ bin/elasticsearch-plugin install --verbose {아리랑파일이 존재하는 서버 경로}/elasticsearch-analysis-arirang-6.1.3.zip
```

- 사전위치 : org.apache.lucene.analysis.kr.dic

- 사전 찾는 순서

Application Classpath 의 패키지 경로에 있는 사전

KoreanAnalyzer 의 Jar 파일에 내장된 사전

- 사전 Customizing 방법

extension.dic 를 Application classpath 에 위치 (추천)

total.dic 를 Application classpath 에 위치 (추천하지 않음)

아리랑 형태소 분석기 사전 구성

1. total.dic : 기본사전
2. extension.dic : 확장사전
3. josa.dic : 조사사전
4. eomi.dic : 어미사전
5. prefix.dic : 접두어 사전
6. suffix.dic : 접미어 사전
7. compounds.dic : 기분석 복합명사 사전
8. syllable.dic : 음절정보사전

아리랑 형태소 분석기 사전 구성

확장사전 정보 (extension.dic)

사랑,100100000X

coma(,)를 중심으로 좌측은 단어, 우측은 단어정보

1 2 3 4 5 6 7 8 9 10

체언 용언 기타 하여동사 되어동사 '내'가 붙을 수 있는 체언 reserved 불규칙변형

- 불규칙변형의 종류

B:ㅂ 불규칙, H:ㅎ 불규칙, L:ㄹ 불규칙, U:ㄴ 불규칙, S:ㅅ 불규칙, D:ㄷ 불규칙, R:ㄹ 불규칙, X:규칙

<http://cafe.naver.com/korlucene/135> 참조

복합명사 정보 (compounds.dic)

문법 :

분해전 단어:분해후단어1,분해후단어2,...,분해후단어N:DBXX

하여(다)동사(D)

되어(다)동사(B)

객관화:객관,화:1100 객관화하다, 객관화되다

사전 파일을 위한 CLASSPATH 설정

elasticsearch.in.sh 파일을 수정

```
ES_CLASSPATH="$ES_HOME/lib/elasticsearch-5.5.0.jar:$ES_HOME/lib/*:$ES_CONF_PATH/dictionary"
```

```
config/dictionary/org/apache/lucene/analysis/ko
```

```
config/dictionary/org/apache/lucene/analysis/ko/dic
```


아리랑 형태소 분석기 설정 사용법

queryMode	search time인지 여부, default to false index time에는 모든 키워드 후보를 token으로 추출하나, search time에는 1순위 키워드만 token으로 추출
decompound	복합명사를 분해할 지 여부, default to true
preserveVerb	용언(예쁘다)을 검색어로 추출할 지 여부, default to false 용언화접미사(예:사랑하다)가 덧붙은 단어는 체언이다.
exactMatch	복합명사를 분해할 때, 분해된 단어가 사전에 모두 존재해야 복합명사로 분해한다. default to false
preserveCNoun	복합명사 분해하기 전의 단어를 token으로 추출 여부, default to true
preserveOrigin	형태소분석이 안된 어절을 token으로 추출여부, default to false
bigrammable	형태소분석에 실패한 경우 Bigram을 token으로 추출할 지 여부, default to false
wordSegment	형태소분석에 실패한 경우 자동 띄어쓰기를 할 지 여부, default to false

아리랑 형태소 분석기 커스트 마이징

arirang morph : 아리랑 분석기의 핵심, 기본분석과 사전 정보로 구성

<https://github.com/korlucene/arirang.morph>

arirang analyzer : lucene의 analyzer를 상속받아 lucene에서 사용 할 수 있도록 구성

<https://github.com/korlucene/arirang-analyzer>

사용 결과

```
$ curl -X POST -H "Cache-Control: no-cache" -H "Postman-Token: 6d392d83-5816-71ad-556b-5cd6f92af634" -d '{
  "analyzer": "arirang_analyzer",
  "text": "[한국] 엘라스틱서치 사용자 그룹의 HENRY 입니다."
}' "http://localhost:9200/_analyze"
```

```
{
  "tokens": [
    {
      "token": "[",
      "start_offset": 0,
      "end_offset": 1,
      "type": "symbol",
      "position": 0
    },
    {
      "token": "한국",
      "start_offset": 1,
      "end_offset": 3,
      "type": "korean",
      "position": 1
    },
    {
      "token": "]",
      "start_offset": 3,
      "end_offset": 4,
      "type": "symbol",
      "position": 2
    },
  ],
  "
```

token : 톰의 내용

start_offset : 입력 문서에서 해당 톰의 시작위치

end_offset : 입력 문서에서 해당 톰의 끝위치

position : 입력 문서에서 몇번째 톰인가의 순서

동의어 처리 관계에서 중요한 역할

은전한뿔 형태소 분석기

은전한닢 개요

이용운, 유영호 두 명의 개발자 공동개발
아파치 라이선스

지원 OS : Linux, Windows

개발언어 : C++ / JAVA / Scala

사전형식 : csv

언어, 사전 코퍼스에 의존하지 않는 범용적인 설계

품사 독립적 설계

CRF 채용하여 HMM 보다 성능 향상 (Conditional Random Fields)

Double-Array TRIE

mecab-ko : 일본의 mecab 엔진을 한국어 사정에 맞게 커스트 마이징

mecab-java : C++로 되어 있는 mecab을 자바에서 사용하기 위함 JNI(Java Native Interface)

mecab-ko-dic : 한국어 말뭉치 학습 사전(세종계획)

mecab-ko-lucene-analyzer (elasticsearch-analysis-mecab-ko) lucene/solr /elastic 용 플러그인

은전한닢 설치방법

사전준비

```
$> yum install gcc-c++
```

mecab-ko

```
$> tar -zxvf mecab-0.996-ko-0.9.2.tar.gz
$> cd mecab-0.996-ko-0.9.2
$> ./configure --with-charset=utf-8
$> make
$> make check
$> make install
```

```
# /usr/local/bin -> 실행파일 경로
# /usr/local/lib -> 라이브러리 파일 경로
# /usr/local/etc -> 설정파일 경로
```

mecab-ko-dic

```
$> tar -zxvf mecab-ko-dic-2.0.1-20150920.tar.gz
$> cd mecab-ko-dic-2.0.1-20150920
$> ./configure --with-charset=utf-8
$> make
$> make install
```

```
# /usr/local/lib/mecab/dic/mecab-ko-dic 사전경로
```

mecab 테스트 (콘솔에서)

```
$> /usr/local/bin/mecab -d /usr/local/lib/mecab/dic/mecab-ko-dic
형태소 분석을 테스트하기 위한 문장을 넣습니다.
```

==분석 결과 값==

```
형태소 NNG,*F,형태소,Compound,**,형태/NNG/*+소/NNG/*
분석 NNG,*T,분석,**,*,*
을 JKO,*T,을,**,*,*
테스트 NNG,*F,테스트,**,*,*
하 XSV,*F,하,**,*,*
기 ETN,*F,기,**,*,*
위한 VV+ETM,*T,위한,Inflect,VV,ETM,위하/VV/*+L/ETM/*
문장 NNG,*T,문장,**,*,*
을 JKO,*T,을,**,*,*
넣 VV,*T,넣,**,*,*
습니다 EF,*F,습니다,**,*,*
. SF,**,*,*,*,*
EOS
```


은전한닢 설치방법

mecab-java 설치

```
$> tar -zxvf mecab-java-0.996.tar.gz
$> cd mecab-java-0.996
$> vi Makefile
아래의 INCLUDE에 jdk 경로를 지정해 준다.
INCLUDE=/usr/local/jdk1.8.0_121/include

아래의 $(CXX) -O3을 -O1로 변경한다.
(은전한닢 설명에는 OpenJDK를 사용할 경우 O1로 변경하라고 되어 있으나 OracleJDK에서도 변경해 줘야 된다.)
all:
    $(CXX) -O3 -c -fpic $(TARGET)_wrap.cxx $(INC)
    $(CXX) -shared $(TARGET)_wrap.o -o lib$(TARGET).so $(LIBS)
    $(JAVAC) $(PACKAGE)/*.java
    $(JAVAC) test.java
    $(JAR) cfv $(TARGET).jar $(PACKAGE)/*.class
변경 후 저장.

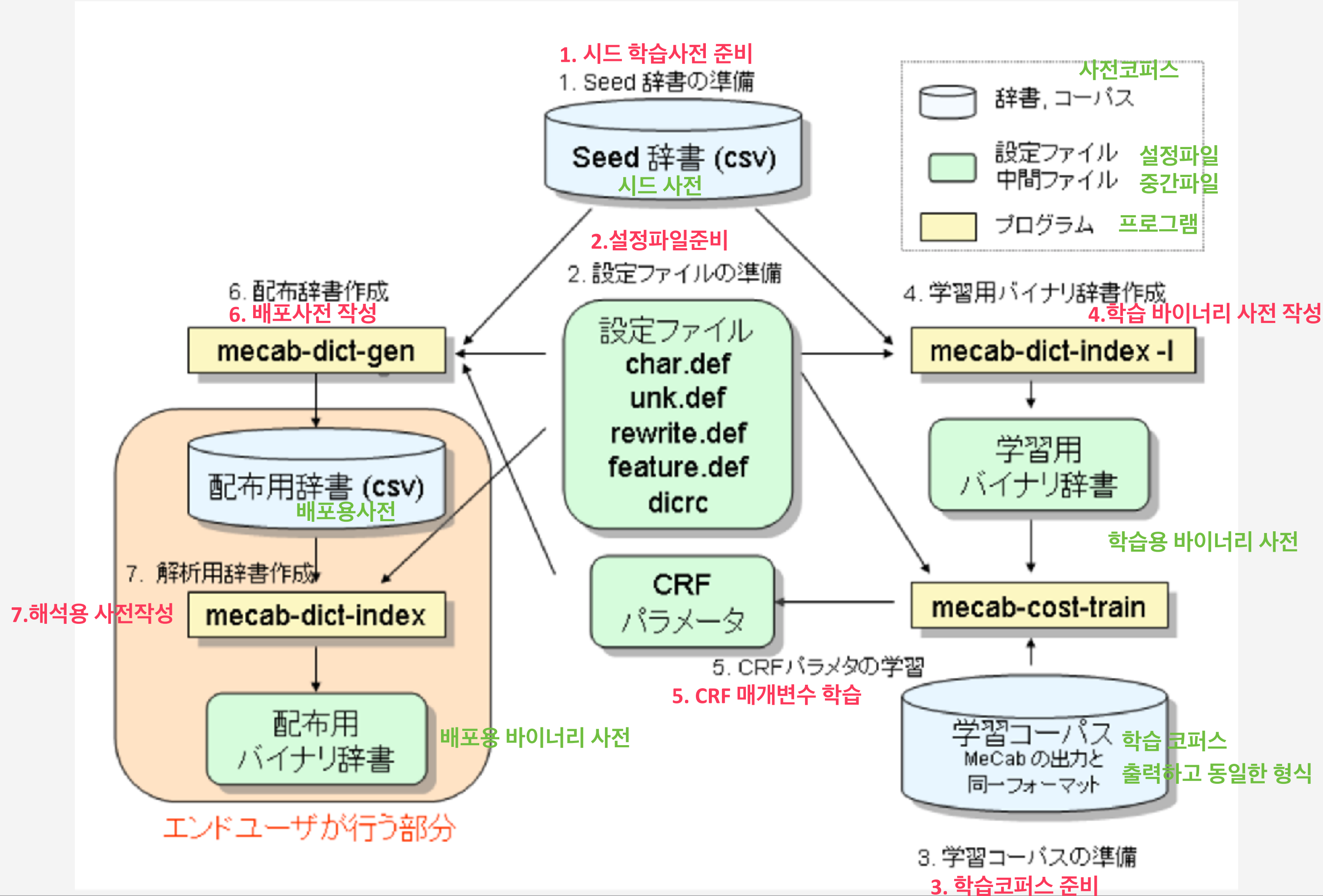
$> make
$> cp MeCab.jar [solr path]/server/lib/ext
$> cp libMeCab.so /usr/local/lib
```

```
{
  "tokens": [
    {
      "token": "문재인/N",
      "start_offset": 0,
      "end_offset": 3,
      "type": "N",
      "position": 0
    },
    {
      "token": "대통령/N",
      "start_offset": 4,
      "end_offset": 7,
      "type": "N",
      "position": 1
    },
    {
      "token": "특사/N",
      "start_offset": 9,
      "end_offset": 11,
      "type": "N",
      "position": 2
    }
  ],
}
```

ElasticSearch Plugin 설치

```
$ ./bin/plugin install https://bitbucket.org/eunjeon/mecab-ko-lucene-analyzer/downloads/elasticsearch-analysis-mecab-ko-x.x.x.x.zip
```

은전한닢(Mecab)에서의 시스템 사전 생성과정



은전한닢(Mecab)에서의 시스템 사전 생성과정

(1)

mecab-ko-dic

시드(Seed) 데이터로

말뭉치(세종계획) 과 참고 사전 파일을 준비 (좌우 문맥 ID 및 연접비용은 0)

feature.def 파일 (특성함수 정의)

rewrite.def 파일 (각 특성의 내부 상태에 대한 매핑 정의)

./build

(2)

mecab-ko-dic-2.0.1-20150920.tar.gz

사용자 배포용 사전 파일 생성

좌우 문맥 ID 및 연접비용이 학습을 마치고 확정됨

(3)

압축 풀고 make; make install

/usr/local/lib/mecab/dic/mecab-ko-dic/matrix.bin

mecab-ko에서

사용 가능한 바이너리 형태로 만듦

다시 빌드
make install

(4)

사용자 사전 변경

mecab-ko-dic/userdic

mecab-ko-dic/tools/add-userdic.sh

mecab-ko-dic/

—user-nnp.csv —user-person.csv —user-place.csv

빌드되지 않은 사용자 사전이 추가됨

(좌/우문맥ID와 비용이 계산되어 있음)

(2)번과 같은 상태

은전한닢(Mecab) 사용자 사전 변경

mecab-ko-dic/userdic 사전 디렉토리
userdic/

└── nnp.csv (일반명사 사전)

대우,,,,NNP*,F,대우*,*,*,*

구글,,,,NNP*,T,구글*,*,*,*

└── person.csv (인명사전)

까비,,,,NNP,인명,F,까비*,*,*,*

└── place.csv (지명사전)

세종,,,,NNP,지명,T,세종*,*,*,*

세종시,,,,NNP,지명,F,세종시,Compound,*,*,세종/NNP/지명+시/NNG/*

사전 컴파일

\$ mecab-ko-dic/tools/add-userdic.sh

mecab-ko-dic

└──

└── user-nnp.csv

└── user-person.csv

└── user-place.csv

\$make install

사전 형식 및 품사 태그표 참조

<https://docs.google.com/spreadsheets/d/1-9blXKjtjeKZqsf4NzHeYJCrr49-nXeRF6D80udfcwY/edit#gid=0>

은전한닢(Mecab) 키워드 분석의 예

원하는 결과가 도출되지 않을시에 비용 변경 방법

1) 단어의 비용을 강제로 변경

예) 네네 치킨 (bestn 실행결과)

#표현층,품사,의미부류,좌문맥ID,우문맥ID,낱말비용,연접비용,누적비용

네 NP,,1790,3549,4151,-1756,2395

네 XSN,,2469,3597,2239,-3891,743

치킨 NNG,,1784,3537,2056,-229,2570

EOS

네네 IC,,197,15,3427,-2203,1224

치킨 NNG,,1784,3537,2056,-710,2570

계산된 결과가 최종 누적비용은 같으나 네네/치킨을
올리고 싶다면 네네라는 단어 자체의 낱말 출현 비용을
강제로 낮추어야 함 (user-nnp.csv 파일을 **직접 편집** 후
컴파일)

2) 기분석 타입(Preanalysis) 혹은 복합명사 타입(Compound)을 이용

사전에서 Compound 타입과 Preanalysis타입은 인덱스 생성시 정답을 제공하기 위해 사용

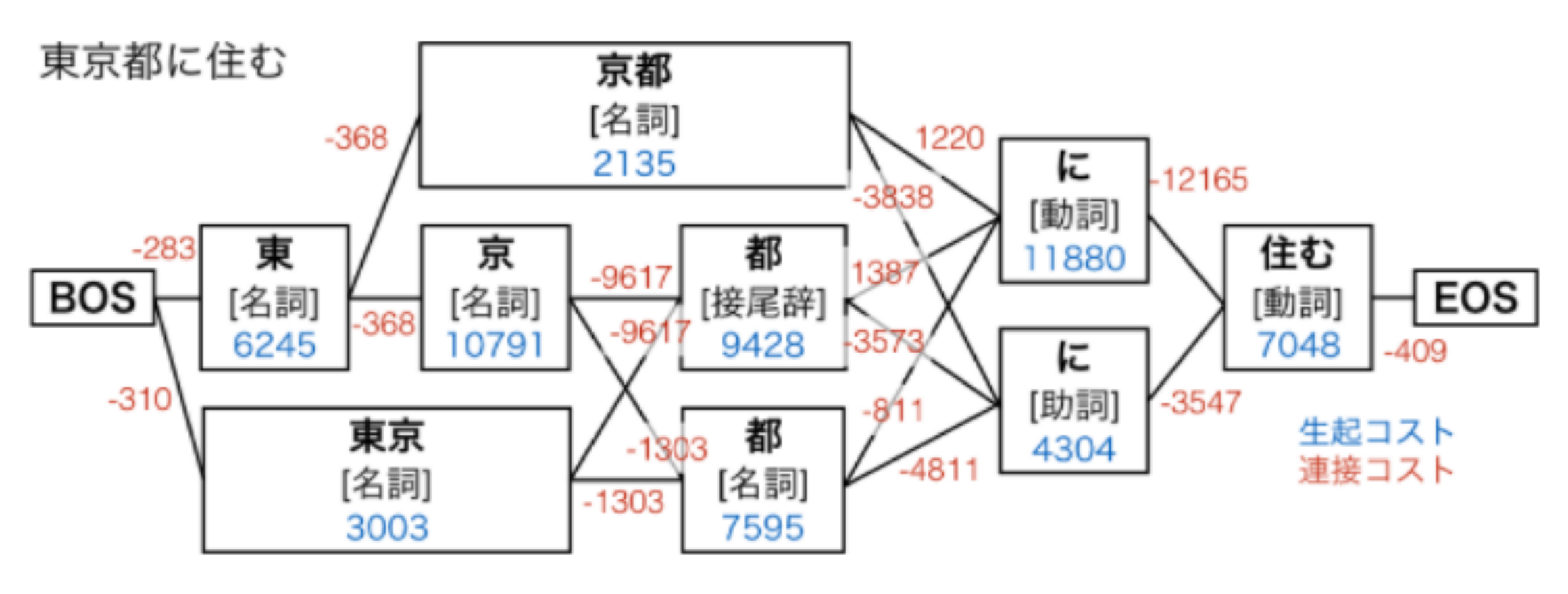
예)

해리포터,1785,3543,3500,NNP,인명,F,해리포터,Preanalysis,NNP,NNP,해리+포터,해리/NNP/인명/1/1+포터/NNP/인명/1/1
1/1 과 같이 루씬에서의 Position 증가 값을 미리 정의 할수 있음

****Side Effect를 충분히 고려하여 작성**

Mecab에서의 형태소 분석

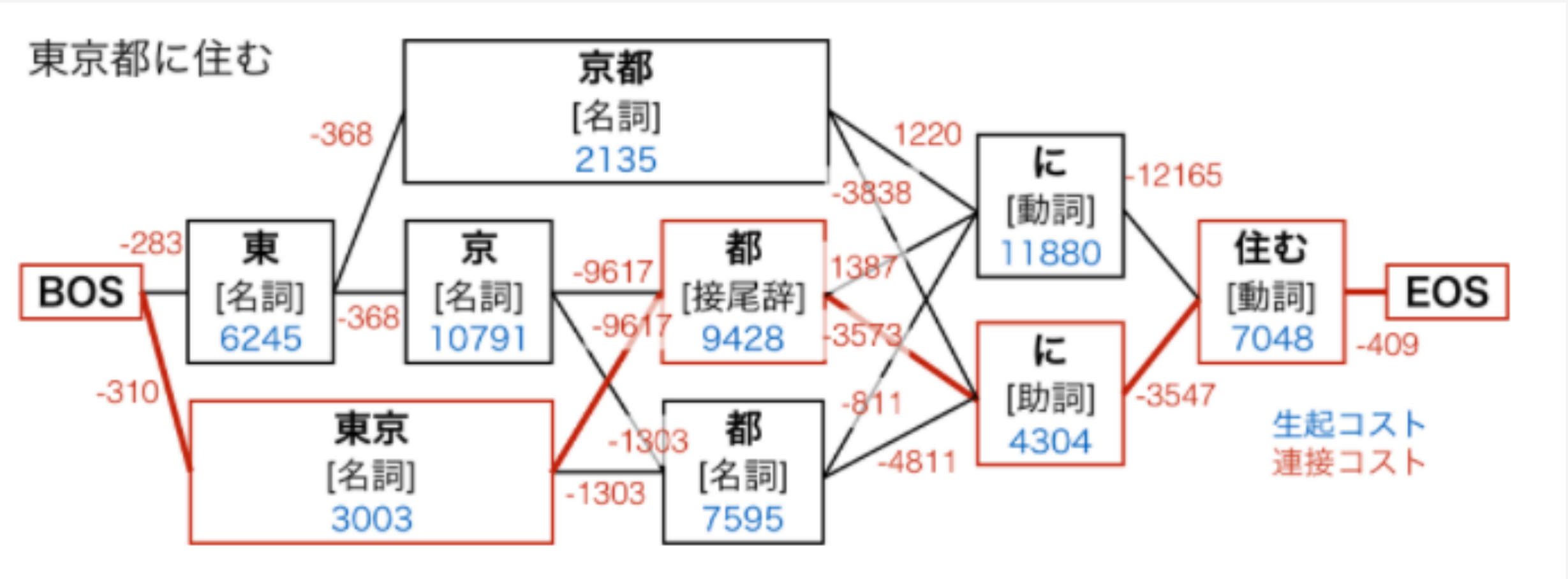
전체 경로 구축



각 노드 값은 단어의 생성 비용
연결은 품사의 연접비용 (앞뒤 단어와의 연접) -말뭉치 학습으로 계산
left-id.def, right-id.def, matrix.def에 저장

최적의 경로 선택

누적 비용을 최소화하는 경로를 선택 (비터비 알고리즘)



Mecab에서 비용계산

분석 문장 : “좋은 간호사가 있다. “

분석 : 좋은 / 간호사 / 가 / 있다.

간호사,1783,3538,2712,NNG,*,F,간호사,Compound,*,*,간호+사,간호/NNG/*/1/1+간호사/Compound/*/0/2+사/NNG/*/1/1

좌문맥ID 우문맥ID 연접비용

metrix.def 에서

간호사의 우문맥 3538

간호사의 좌문맥 1783 의 비용을 추출

문장의 전체 비용은 각 형태소의 비용과 형태소 간의 연접 비용의 합으로 계산

Mecab에서 비용계산

```
$ echo "서울시장애인협회" | mecab -d . -N2 -F"%m\t%f[0],%f[1],%phi,%phr,%pw,%pC,%pc\n"
```

```
서울시      NNP,,1094,1684,2497,-2135,362
```

```
장애인협회 NNG,,1093,1682,2619,697,3678
```

```
EOS
```

```
서울      NNP,지명,1096,1691,2219,-1898,321
```

```
시장      NNG,,1093,1683,-223,300,398
```

```
애인      NNG,,1093,1683,1152,820,2370
```

```
협회      NNG,,1093,1682,2461,820,5651
```

```
EOS
```

*.csv

비용: 2370 + 820 + 2461 = 5651

matric.def(1683 1093 820)

%phi : 좌문맥ID

%phr: 우문맥ID

%pw: 낱말 비용

%pC: 이전 형태소와의 연접비용

%pc: 연접비용 + 낱말 비용(누적)

단어간의 직접적인 연접 비용이 아니고 형태소간의 연접 비용
연접비용과 생성 비용 계산은 세종말뭉치 학습
단어의 빈도가 높으면 값이 낮게
품사간의 연접이 많이 일어나면 낮게
(클수록 손해)

seunjeon vs mecab-ko

seunjeon : mecab-ko-dic 을 이용하여 scala로 만든 형태소 분석기
mecab-ko : mecab-ko-dic 을 로딩해서 돌아가는 형태소분석기 (C++)

처리 속도 : mecab-ko > seunjeon
사용성 : mecab-ko < seunjeon
분석품질 : mecab-ko = seunjeon

elasticsearch-analysis-seunjeon 의 주요 옵션
인덱스 생성 시점에서 옵션을 첨가 가능

옵션인자	설명	기본값
user_words	사용자사전	[]
user_dict_path	사용자사전 경로	\$es_home/config/user_dict.csv
decompound	복합명사 분해여부	true
deinflect	활용어의 원형추출	true
index_eojeol	어절추출	true
index_poses	추출할 품사 지정	["N", "SL", "SH", "SN", "XR", "V", "M", "UNK"]
pos_tagging	품사태깅여부	true
max_uk_length	unknown 키워드로 뽑을 수 있는 최대 길이(한글)	8

seunjeon 사용자 사전

용법 :

1) \$es_home/config/user_dict.csv 파일로 만들거나

2) "seunjeon_tokenizer": { "type": "seunjeon_tokenizer", "user_words": ["낄끼빠빠,-100", "버카충","네네+치킨"] }, inline으로 넣을수 있음

문법 :

표현층 : 단어명, 단어명 + 단어명 (복합명사)

+ 자체를 단어로 사용하려면 '+' 예(c++ => c\+\+)

표현층 + 출현빈도 : "버카충,-100" (낮을수록 출현빈도 높다.)

트위터 형태소 분석기

작성 언어 : 스칼라

완전한 형태의 형태소분석이 아닌 색인어 추출에 목적

OpenKoreanText 로 명칭이 변경됨

Apache License 2.0

4가지 기능에 중점

1) 정규화 normalization

입니답ㅋㅋ -> 입니다 ㅋㅋ, 샤릉해 -> 사랑해

2) 토큰화 tokenization

한국어를 처리하는 예시입니다 ㅋㅋ -> 한국어Noun, 를Josa, 처리Noun, 하는Verb, 예시Noun, 입Adjective, 니다Eomi

3) 어근화 stemming (입니다 -> 이다)

한국어를 처리하는 예시입니다 ㅋㅋ -> 한국어Noun, 를Josa, 처리Noun, 하다Verb, 예시Noun, 이다Adjective, ㅋㅋ

4) 어구 추출 phrase extraction

한국어를 처리하는 예시입니다 ㅋㅋ -> 한국어, 처리, 예시, 처리하는 예시

트위터 형태소 분석기

JAVA에서 사용법 예시

```
String text = "한국어를 처리하는 예시입니달ㅋㅋㅋㅋ #한국어";
CharSequence normalized = TwitterKoreanProcessorJava.normalize(text);
System.out.println(normalized); // 한국어를 처리하는 예시입니다ㅋㅋ #한국어
```

```
Seq<KoreanTokenizer.KoreanToken> tokens = TwitterKoreanProcessorJava.tokenize(normalized);
System.out.println(TwitterKoreanProcessorJava.tokensToJavaStringList(tokens)); // [한국어, 를, 처리, 하는, 예시, 입니, 다, ㅋㅋ, #한국어]
System.out.println(TwitterKoreanProcessorJava.tokensToJavaKoreanTokenList(tokens));
// [한국어(Noun: 0, 3), 를(Josa: 3, 1), (Space: 4, 1), 처리(Noun: 5, 2), 하는(Verb: 7, 2), (Space: 9, 1), 예시(Noun: 10, 2), 입니(Adjective: 12, 2), 다(Eomi: 14, 1), ㅋㅋ(KoreanParticle: 15, 2), (Space: 17, 1), #한국어(Hashtag: 18, 4)]
```

```
Seq<KoreanTokenizer.KoreanToken> stemmed = TwitterKoreanProcessorJava.stem(tokens);
System.out.println(TwitterKoreanProcessorJava.tokensToJavaStringList(stemmed)); // [한국어, 를, 처리, 하다, 예시, 이다, ㅋㅋ, #한국어]
System.out.println(TwitterKoreanProcessorJava.tokensToJavaKoreanTokenList(stemmed));
// [한국어(Noun: 0, 3), 를(Josa: 3, 1), (Space: 4, 1), 처리(Noun: 5, 2), 하다(Verb: 7, 2), (Space: 9, 1), 예시(Noun: 10, 2), 이다(Adjective: 12, 3), ㅋㅋ(KoreanParticle: 15, 2), (Space: 17, 1), #한국어(Hashtag: 18, 4)]
```

```
List<KoreanPhraseExtractor.KoreanPhrase> phrases = TwitterKoreanProcessorJava.extractPhrases(tokens, true, true);
System.out.println(phrases); // [한국어(Noun: 0, 3), 처리(Noun: 5, 2), 처리하는 예시(Noun: 5, 7), 예시(Noun: 10, 2), #한국어(Hashtag: 18, 4)]
```

메이븐

```
<dependency>
  <groupId>com.twitter.penguin</groupId>
  <artifactId>korean-text</artifactId>
  <version>4.4</version>
</dependency>
```

트위터 형태소 분석기

E/S 에서 사용법 예시

GET /test/_analyze?analyzer=custom-analyzer&text=한국어를 처리하는 예시입니답ㅋㅋㅋㅋㅋ

1) 설치

```
cd ${ES_HOME}/plugins
mkdir tkt-elasticsearch
cd tkt-elasticsearch/
wget https://github.com/socurites/tkt-elasticsearch/blob/master/dist/tkt-elasticsearch-0.1.0-jar-with-dependencies.jar?raw=true -O tkt-elasticsearch-0.1.0.jar
```

PUT /test/

```
{
  "index" : {
    "analysis" : {
      "analyzer" : {
        "custom-analyzer" : {
          "tokenizer" : "custom-tokenizer"
        }
      },
      "tokenizer": {
        "custom-tokenizer" : {
          "type": "tkt-korean-tokenizer",
          "enableNormalize": false,
          "enableStemmer": false
        }
      }
    }
  }
}
```

트위터 형태소 분석기

사용자 사전 변경

사전 파일 경로

/src/main/resources/com/twitter/penguin/korean/util/

<https://github.com/twitter/twitter-korean-text/tree/master/src/main/resources/com/twitter/penguin/korean/util>

Nori 형태소 분석기

Mecab의 Contributor 인 Jim Ferenczi 개발

특징

- Elasticsearch 6.4 부터 사용가능
- 은전한닢의 mecab-ko-dic 사전을 사용
- Jim Ferenczi가 개발한 일본어 형태소 분석기인 Kuromoji 를 fork하여 mecab-ko-dic 를 적용시킴

구성

nori_tokenizer

- 사전 경로 정의
- 복합명사 처리 정의

nori_part_of_speech token filter

- 품사 제약

nori_readingform token filter

- 한자 독음

<https://www.youtube.com/watch?v=80WiaOdnjXU> (참조)

엘라스틱서치의 텍스트 분석

분석기의 정의

분석기란?

char filter (전처리)+tokenizer (토큰 분해)+token filter (토큰 변환)= Analyzer

내장 분석기

Standard Analyzer

Simple Analyzer

Whitespace Analyzer

Stop Analyzer

Keyword Analyzer

Pattern Analyzer

Language Analyzers

Fingerprint Analyzer

사용자 정의 분석기

특성

인덱스 단위로 저장

settings 옵션 명령으로 생성

0개 혹은 1개 이상 char_filter

1개 이상 tokenizer

0개 혹은 1개 이상 token filter

사용자 정의 분석기 추가,수정

1. 색인닫기 (POST /index_name/_close)
2. 추가,수정
3. 색인열기 (POST /index_name/_close)

```
curl -XPUT 'http://localhost:9200/books' -d '{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_custom_analyzer": {
          "type": "custom",
          "tokenizer": "standard",
          "char_filter": [
            "html_strip"
          ],
          "filter": [
            "lowercase",
            "asciifolding"
          ]
        }
      }
    }
  }
}
```

확인

http://localhost:9200/books/_settings?pretty

http://localhost:9200/books/_analyze?&text={text}

Built in - Character Filter, Tokenizer, Token Filter

캐릭터필터

- HTML Strip Character Filter
- Mapping Character Filter
- Pattern Replace Character Filter

토크나이저

- Standard Tokenizer
- Letter Tokenizer
- Lowercase Tokenizer
- Whitespace Tokenizer
- UAX URL Email Tokenizer
- Classic Tokenizer
- Thai Tokenizer
- Keyword Tokenizer
- Pattern Tokenizer
- Path Tokenizer

토큰필터

- Reverse Token Filter
- Elision Token Filter
- Truncate Token Filter
- Unique Token Filter**
- Pattern Capture Token Filter
- Pattern Replace Token Filter**
- Trim Token Filter**
- Limit Token Count Token Filter
- Hunspell Token Filter
- Common Grams Token Filter
- Normalization Token Filter
- CJK Width Token Filter
- CJK Bigram Token Filter
- Delimited Payload Token Filter
- Keep Words Token Filter
- Keep Types Token Filter
- Classic Token Filter
- Apostrophe Token Filter
- Decimal Digit Token Filter
- Fingerprint Token Filter
- Minhash Token Filter
- Standard Token Filter
- ASCII Folding Token Filter
- Flatten Graph Token Filter
- Length Token Filter**
- Lowercase Token Filter**
- Uppercase Token Filter
- NGram Token Filter**
- Edge NGram Token Filter**
- Porter Stem Token Filter
- Shingle Token Filter
- Stop Token Filter**
- Word Delimiter Token Filter
- Word Delimiter Graph Token Filter
- Stemmer Token Filter
- Stemmer Override Token Filter
- Keyword Marker Token Filter
- Keyword Repeat Token Filter
- KStem Token Filter
- Snowball Token Filter
- Phonetic Token Filter
- Synonym Token Filter**
- Synonym Graph Token Filter
- Compound Word Token Filters

사용자 정의 토크나이저

```
curl -XPUT 'http://localhost:9200/books' -d '{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_custom_analyzer": {
          "tokenizer": "standard"
        }
      }
    }
  }
}
```

```
curl -XPUT 'http://localhost:9200/books' -d '{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_custom_analyzer": {
          "tokenizer": "my_custom_tokenizer"
        },
        "tokenizer" : {
          "my_custom_tokenizer" : {
            "type" : "standard",
            "max_token_length" : 100
          }
        }
      }
    }
  }
}
```

실습!!
ngram 분석기 만들기

사용자 정의 필터 (동의어 필터)

```
curl -XPUT 'http://localhost:9200/books' -d '{
```

```
{
  "settings": {
    "index": {
      "analysis": {
        "analyzer": {
          "synonym_analyzer": {
            "tokenizer": "whitespace",
            "filter": ["synonym_filter"]
          }
        },
        "filter": {
          "synonym_filter": {
            "type": "synonym",
            "synonyms_path": "analysis/synonym.txt"
          }
        }
      }
    }
  }
}
```

동의어 필터 옵션

synonyms : 동의어를 인라인으로 정의

synonyms_path : 동의어 사전 위치를 정의 (추천)

format : 기본은 solr 포맷, wordnet 포맷 제공 (<http://wordnet.kaist.ac.kr/>)

solr 포맷 형식

1) i-pod,i pod => ipod (대체)

2) ipod,i-pod,i pod (확장)

3) foo => foo bar

foo => baz foo => foo bar, baz (병합)

기타 참고

독음처리 (<https://github.com/muik/transliteration>)

형태소 분석기 플러그인 개발

E/S에서의 플러그인

1. Java 플러그인

이 플러그인은 오직 jar파일만 포함
반드시 클러스터 내의 모든 노드에 설치해야 함
노드를 재구동 해야만 추가한 플러그인을 사용할 수 있음. (주로 형태소 분석기 추가시)

4. Rest 플러그인

Rest API를 직접 구성하고 특정 행위를 할수 있는 기능을 만드는 플러그인

2. Site 플러그인

이 플러그인은 자바스크립트, HTML, CSS 파일 같은 정적인 웹 콘텐츠를 내포
별도의 작업 없이 ES의 URL 접근을 통해 추가한 기능을 제공 받음.
Java 플러그인과는 달리 이 플러그인은 클러스터 내의 한 노드에만 설치하면
다른 노드에 자동으로 전파가 되고 별다른 재구동 절차 없이 바로 사용할 수 있음
Site 플러그인은 다음 URL로 접근할 수 있음.
대표적인 것으로 head bigdesk등
`http://installedNodeUrl:9200/_plugin/[플러그인 명]`

3. Mixed 플러그인

이 플러그인은 플러그인 이름 그대로 Java / Site 플러그인이 혼재
두 플러그인의 특성을 모두 가졌으며 사용하기 위해서 재구동 절차가 필요하며 Site 플러그인처럼 ES의 URL로 바로 접근 가능

E/S 분석기 플러그인 개발 과정

1. 프로젝트에 elasticsearch, lucene dependency 추가 (메이븐 pom.xml 에 추가)

```
<dependency>  
  <groupId>org.elasticsearch</groupId>  
  <artifactId>elasticsearch</artifactId>  
  <version>${elasticsearch.version}</version>  
  <scope>provided</scope>  
</dependency>
```

```
<dependency>  
  <groupId>org.apache.lucene</groupId>  
  <artifactId>lucene-test-framework</artifactId>  
  <version>${lucene.version}</version>  
  <scope>test</scope>  
</dependency>
```

형태소 분석기 플러그인 개발 과정

2. 기능을 수행하는 Filter class를 org.apache.lucene.analysis.TokenFilter class를 상속받아 구현

```
@Override
public final boolean incrementToken() throws IOException {
    clearAttributes();

    String resultToken = null;
    startOffset += this.tokenBuilder.length();

    while (resultToken == null) {
        if (!this.input.incrementToken())
            return false;

        resultToken = extractLetterOrDigit(
            this.input.getAttribute(CharTermAttribute.class).toString());
    }

    this.charTermAttribute.setEmpty().append(resultToken);
    this.positionIncrementAttribute.setPositionIncrement(1);
    this.offsetAttribute.setOffset(startOffset, startOffset + endOffset);

    return true;
}
```

형태소 분석기 플러그인 개발 과정

3. E/S 의 필터 생성을 담당하는 FilterFactory 클래스를

`org.elasticsearch.index.analysis.AbstractTokenFilterFactory` 클래스를 상속받아 구현

```
public class CustomTokenFilterFactory extends AbstractTokenFilterFactory {
    @Inject
    public CustomTokenFilterFactory(
        Index index, @IndexSettings Settings indexSettings,
        @Assisted String name, @Assisted Settings settings) {
        super(index, indexSettings, name, settings);
    }

    @Override
    public TokenStream create(TokenStream tokenStream) {
        return new CustomTokenFilter(tokenStream);    //실제 작동하려는 우리가 작성한 프로그램 호출
    }
}
```


형태소 분석기 플러그인 개발 과정

색인 분석이 필요한 경우 org.elasticsearch.plugins.AnalysisPlugin 인터페이스를 구현
AnalysisPlugin::getTokenFilters 를 필수로 오버라이드함

```
public class AnalysisArirangPlugin extends Plugin implements AnalysisPlugin {  
    @Override  
    public Map<String, AnalysisProvider<TokenFilterFactory>> getTokenFilters() {  
        return singletonMap("custom_filter", CustomTokenFilterFactory::new);  
    }  
}
```

main/resource/es-plugin.properties 파일을 생성 후
description : 플러그인에 대한 설명
version : 플러그인 버전
name : 플러그인 이름
classname : 로드해야할 플러그인의 전체 클래스명 (fully-qualified)
java.version : 자바 버전
elasticsearch.version : 엘라스틱서치 버전

필수 기재후 저장

빌드후

```
bin\elasticsearch-plugin.bat install  
    {플러그인경로}/elastic-basic-plugin-5.1.2.zip
```

플러그인 만들기 실습 (자동완성)

감사합니다.

다음 주제 :

Elasticsearch 질의 및 랭킹

1. Elasticsearch의 검색 질의 방식 및 각각의 질의 방식 별 특징
2. 검색 결과를 가져오는 순서의 변경 방법
3. Elasticsearch 랭킹 구조의 설계