

# Python 기반 자연어 처리

## 언어란?

★

# 언어<sup>1</sup> 言語

단어장 저장

발음

파생어 언어-적

표준국어대사전 고려대한국어대사전 우리말샘 < >

예문 열기

명사

1. 생각, 느낌 따위를 나타내거나 전달하는 데에 쓰는 음성, 문자 따위의 수단. 또는 그 음성이나 문자 따위의 사회 관습적인 체계.

언어 감각.

## 언어란?

생각, 느낌을 '전달'하는 과정



## 자연어란?

**자연-어** 自然語

한국어, 영어, 중국어

+ 단어장 저장

표준국어대사전

고려대한국어대사전

우리말샘



명사

- 언어 일반 사회에서 자연히 발생하여 쓰이는 언어. 에스페란토 따위의 인공 언어에 상대하여 이르는 말이다.

**인공-어** 人工語

프로그래밍 언어

+ 단어장 저장

표준국어대사전

고려대한국어대사전

우리말샘



명사

- 정보·통신 컴퓨터가 직접 판독하고 실행할 수 있는 언어. 0과 1의 조합으로 구성된다. 컴퓨터의 내부 구조에 대한 전문 지식이 없으면 프로그램 작성이나 오류 수정이 매우 어렵다.
- 언어 세계 여러 나라에서 공통으로 사용하기 위하여 만든 언어. '에스페란토', '이도' 따위가 있다.

## 자연어란?

### 자연언어

- 인간 고유의 언어, 정보 전달의 수단, 인공언어에 대응되는 개념
- 특정 집단에서 사용되는 모국어의 집합
- 예) 한국어, 영어, 불어, 독일어, 스페인어, 일본어, 중국어 등

### 인공언어

- 특정 목적을 위해 인위적으로 만든 언어
- 자연언어에 비해 엄격한 구문을 가짐
- 예) 형식언어, 프로그래밍 언어

## 자연어처리

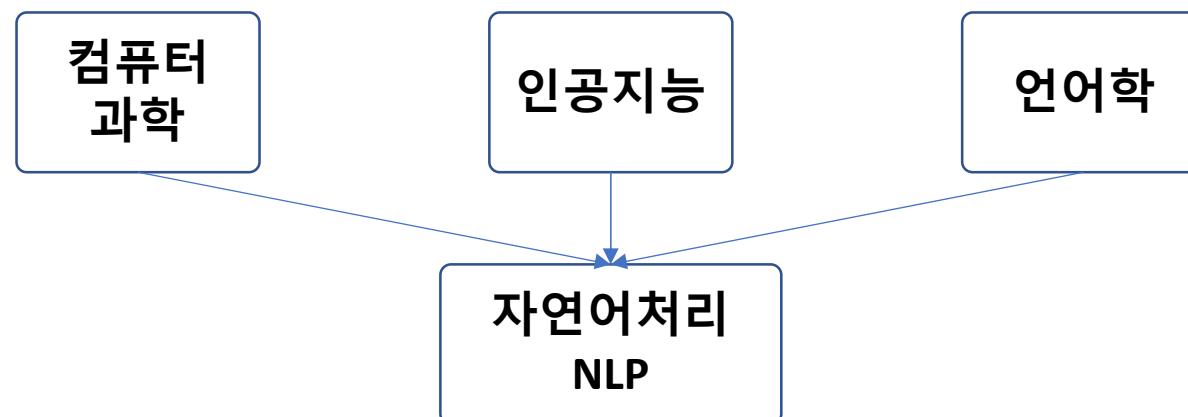
컴퓨터를 이용하여 자연어(인간 언어)의 이해, 생성 및 분석을 다루는 기술



## 자연어 처리의 목적

무엇을 위해 자연어 처리를 할까?

- 이해하기 위해서
- 대답하기 위해서

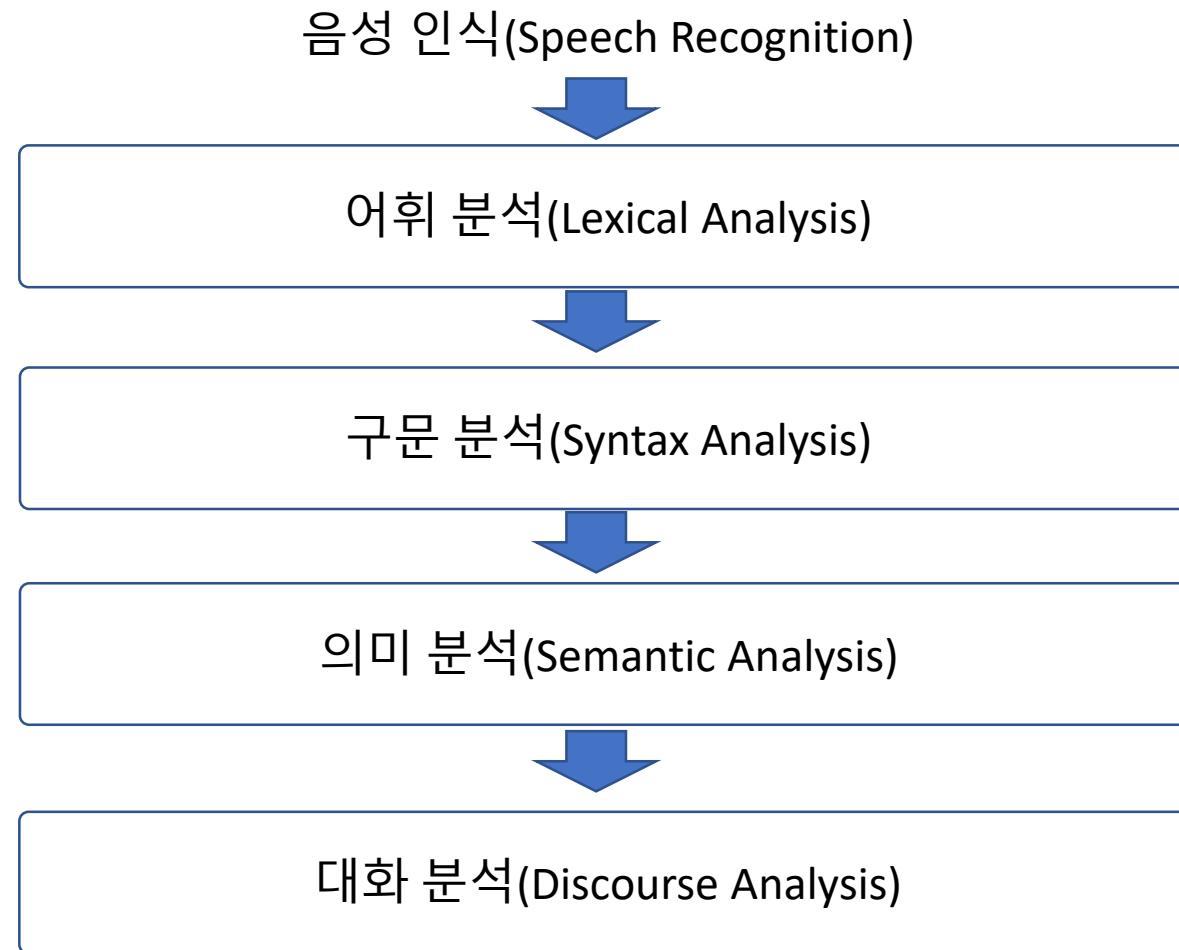


## 자연어 처리가 어려운 이유

### 자연어 처리의 어려움

- 동일한 표현이 다양한 방법으로 해석될 수 있는 **애매성(ambiguity)**을 내포
- 형태소 수준의 애매성 ex) 감기는
- 구문 수준의 애매성 ex) 두 개의 침실이 있는 스위트룸을 예약했다.

## 자연어 처리의 단계



## 자연어 처리의 단계

음성 인식(Speech Recognition)



어휘 분석(Lexical Analysis)



구문 분석(Syntax Analysis)



문장

의미 분석(Semantic Analysis)



문맥

대화 분석(Discourse Analysis)

## 자연어 처리의 단계

### 어휘 분석(Lexical Analysis)

- 입력된 문장을 잘 분할해서 효율성을 높이기 위함
- **Sentence splitting:** 마침표(.), 느낌표(!), 물음표(?) 등을 기준으로 분리
- **Tokenizing:** 문서나 문장을 분석하기 좋도록 나눔(띄어쓰기 또는 형태소 단위)
- **Morphological**
  - 토큰들을 좀 더 일반적인 형태로 분석해 단어 수를 줄여 분석의 효율성을 높임(가장 작은 의미단위로 토큰화)
  - Stemming, Lemmatization

## 자연어 처리의 단계

### 구문 분석(Syntax Analysis)

- 문장을 이루고 있는 구성 성분으로 분해하고 그들 사이의 위계 관계를 분석하여 문장의 구조를 결정
- 문장 구성을 위한 규칙, 문법을 구성

- 각각의 어절 단위로 구분, 해당 tag 부여
- 의존구문분석(Dependency parsing) : 개별 단어 사이의 관계를 고려해 연결
- 구성성분분석(Constituency parsing): 텍스트를 반복적으로 하위 구문으로 분리
- ROOT를 기준으로 SUBJECT와 OBJECT 관계를 나타냄
- SyntaxNet

## 자연어 처리의 단계

### 의미 분석(Semantic Analysis)

- 규칙에 따라 문장은 만들었는데 문장이 의미적으로 올바른 것인지 알아야함

“사람이 사과를 먹는다”  
“사람이 비행기를 먹는다”

- Syntactic + Meaning
- CharCNN

“스타벅스 언제 문닫니?”

Question focus: Closing time  
Where: 스타벅스

## 자연어 처리의 단계

### 대화 분석(Discourse Analysis)

- 대화의 흐름을 파악하여 발화자의 의도에 맞도록 응답해야 함
- 대화의 흐름상 어떤 의미를 가지는지 찾음
- 문맥 구조 분석: 문장들의 연관관계
- 의도분석: 전후 관계를 통한 실제 의도
- 대화분석: 대표적인 담화분석

## 자연어 처리를 위한 언어학

### 음소

- 더 이상 작게 나눌 수 없는 음운론상의 최소 단위

### 어절

- 양쪽에 공백을 갖는 띄어쓰기 단위의 문자열

### 단어

- 단일 품사를 갖는 단위

### 형태소

- 사전에 등록되어 있는 색인어의 집합
- 의미를 가지는 언어 단위 중 가장 작은 단위

## 최근 20년간 자연어 처리 문제

이야기

문장내에서 '이해하기'

문장외에서 정보 '찾기'

Q/A

번역

감성분석

주어복원

주제분류

검색

자연어이해

...

...

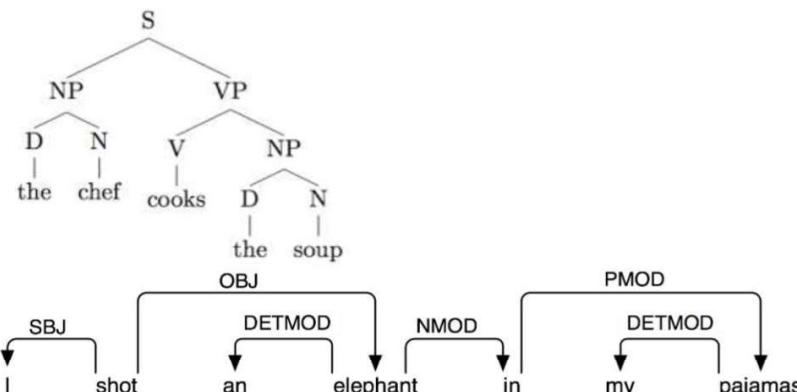
대화

개체명분석

Date Time Location Person  
 At the W party Thursday night at Chateau Marmont, Cate Blanchett barely made it up in the elevator.

NLP Stack

구문분석



의존구문분석

형태소분석

Ex) 밥/NNG+먹/VV+기/ETN 짖/VA+대/EF

정규화

Ex) 채널 삼백번 틀어줘 → 채널 300번 틀어줘

띄어쓰기

Ex) 무한도전좀 보여 줘 → 무한도전 좀 보여줘

## 자연어 처리 분야 : 기계번역(NMT, Neural Machine Translation)



The screenshot shows the Naver Translation interface with two parallel translation boxes side-by-side, illustrating the differences between Google Translate (left) and Papago (right) for the same Korean input sentence.

**Input Sentence:** 한국의 수도와 일본의 수도는 얼마나 떨어져있어?

**Google Translate Output:** How far aways is the capital of the capital of Korea and Japan?

**Papago Output:** How far (...)?

**Annotations:**

- Left Column (Google Translate):** Includes a red box around the 'How far aways' part of the output, and a note below stating "수도 (...) capital; waterworks, water supply[service]; ascetic practice, practice ascetism".
- Right Column (Papago):** Includes a red box around the question mark in the output, and a note below stating "[거리] 얼마나 멀어져있는가? [정도] 어느 정도, 얼마나큼".

## 자연어 처리 분야 : 가상비서(Virtual Assistants)



구글 홈



아마존 에코



KT 기가지니



카카오 미니



SK 퓨전 누구



네이버 클로바



네이버  
웨이브



애플  
홈파드



MS  
코타나

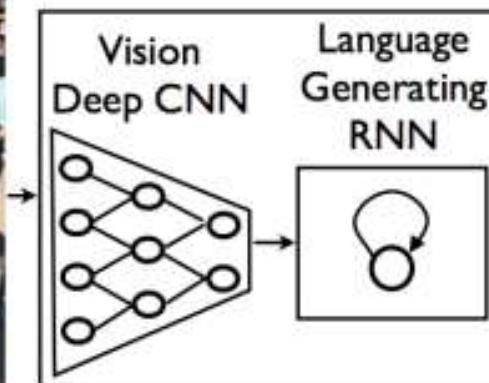


LG  
씽큐



샤오미

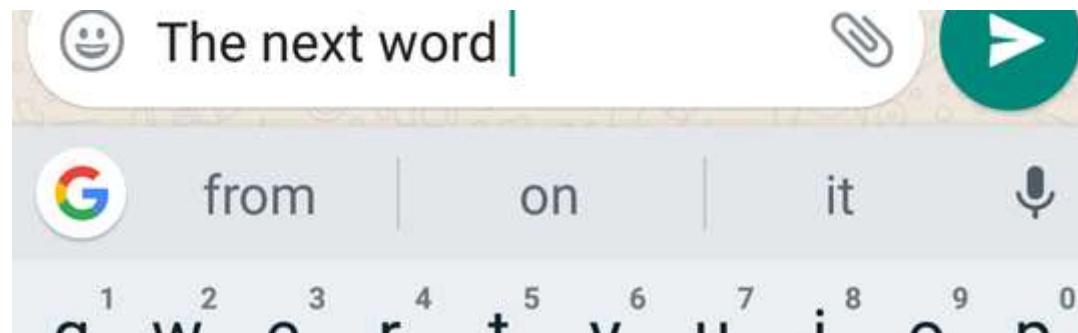
## 자연어 처리 분야 : Image Captioning



**A group of people  
shopping at an  
outdoor market.**

**There are many  
vegetables at the  
fruit stand.**

## 자연어 처리 분야 : Language Modeling



## 자연어 처리 분야 : Named Entity Recognition

"There was nothing about this storm that was as expected," said **Jeff Masters**, a meteorologist and founder of **Weather Underground**. "**Irma** could have been so much worse. If it had traveled 20 miles north of the coast of **Cuba**, you'd have been looking at a (Category) 5 instead of a (Category) 3."

Person

Organization

Location

## 자연어 처리 분야 : 기계독해(Machine Reading Comprehension)

대한민국의 가수 겸 배우이다. 본명은 이지은이며, 아이유(IU)는 예명이다.<sup>[30]</sup> 아이유(IU)는 “음악으로 나와 내가 하나가 된다”라는 뜻을 가지고 있다. 2008년에 데뷔하였다. 국민 여동생이라 고도 불렸다.

2010년대 이후부터 현재까지 악성 슬로 가수 중에서 독보적인 위치를 점하고 있으며, 다양한 장르를 소화해내는 만능형 가수라는 평가를 받는다.<sup>[31]</sup> 2010년 전소리와 좋은 날로 부종한 이후 10년째 가요계 정상의 위치를 굳건히 지키며 활약하고 있다.

가수 활동 이외에 각종 예능 프로그램 및 광고 모델로도 활발하게 참여하고 있고, 다양한 드라

### Korean Machine Reading Comprehension using BERT

Question : 아이유는 언제 데뷔했어?

Top 1 : 2008년 (99.724%)

Top 2 : 2008년이 (0.265%)

Top 3 : 2008년이 데뷔 (0.005%)

응답시간 : 1.23 (sec)

back →

## 자연어 처리 분야 : Text Summarization



before

## Text Summarization using NLP Summary

- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]

after

## 자연어 처리 분야 : Sentiment Analysis

# Sentiment Analysis



Positive

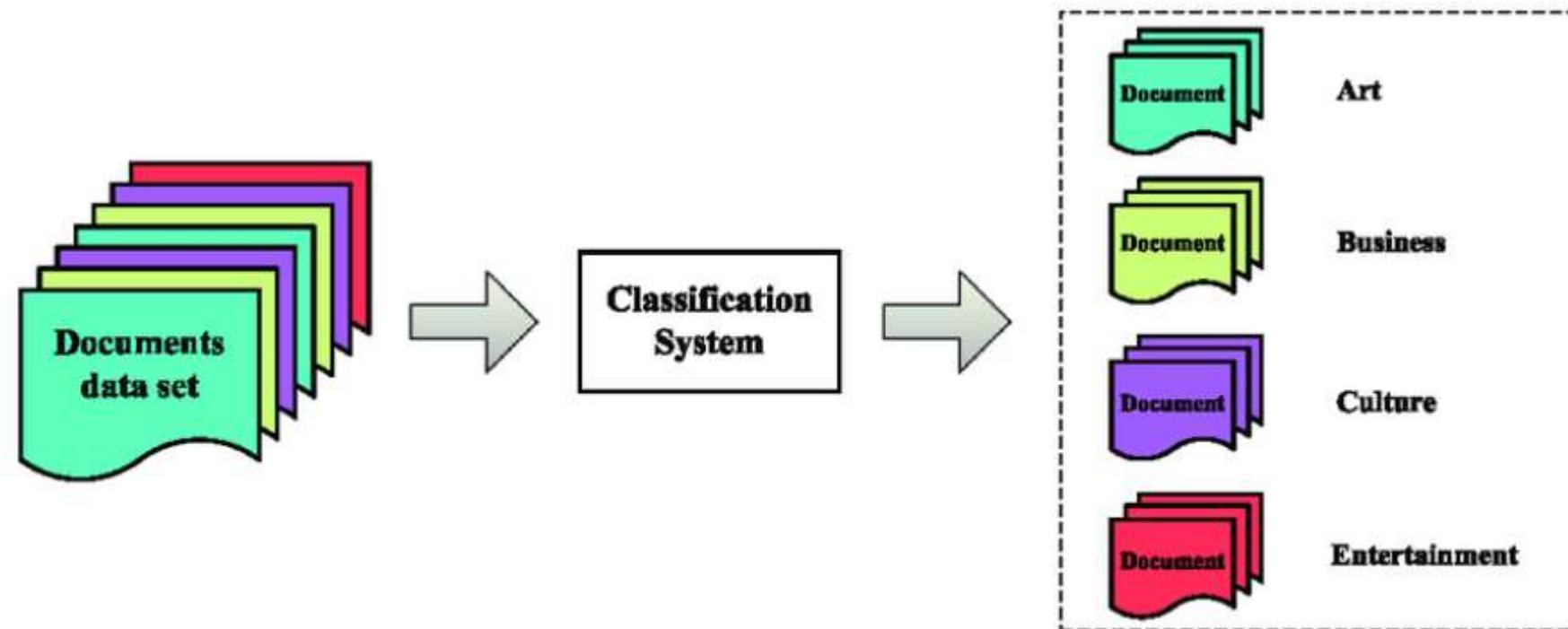


Negative



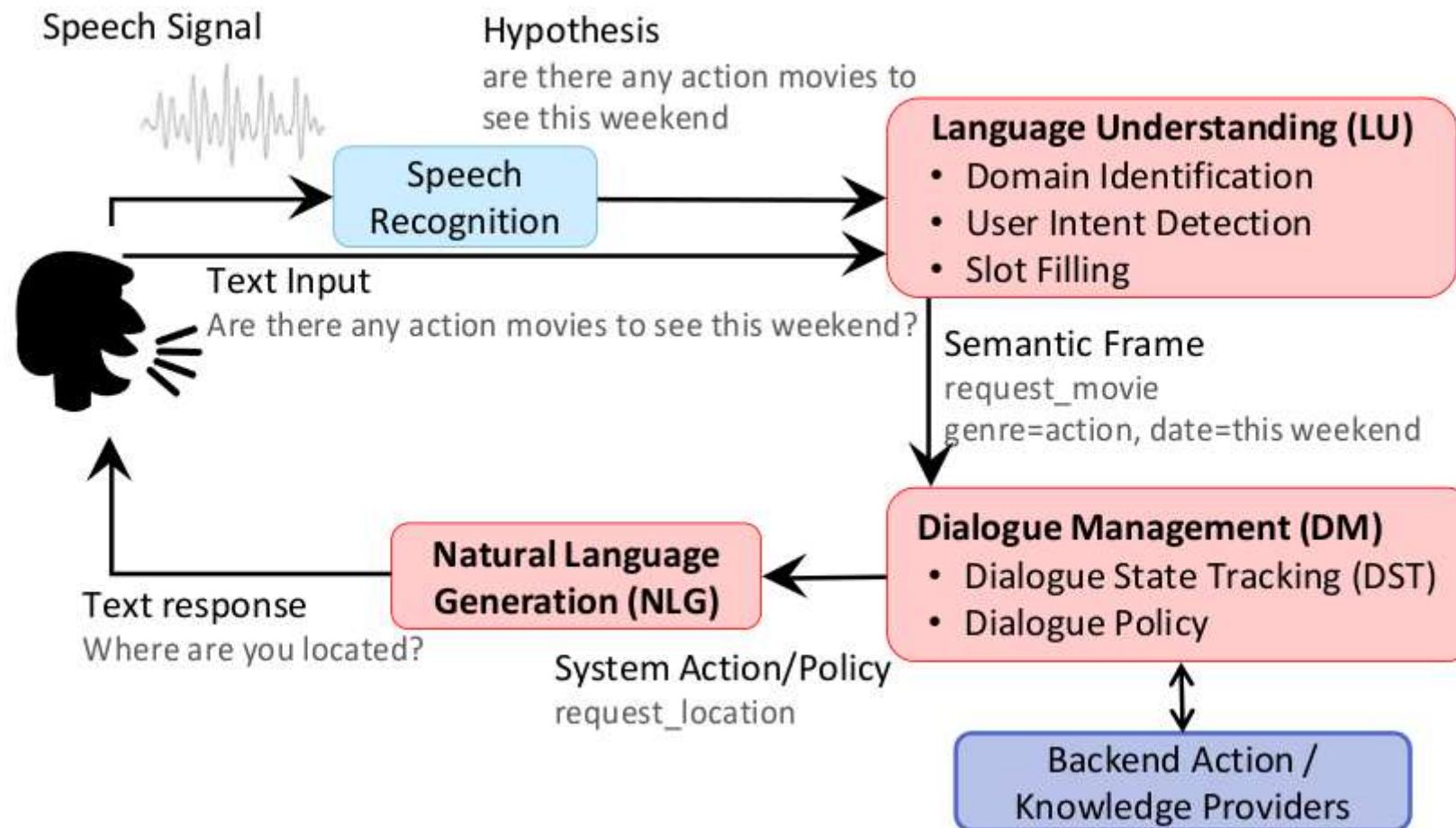
Neutral

## 자연어 처리 분야 : Text Categorization



A simple example algorithm framework for text categorization.

## 자연어 처리 분야 : Dialogue System



텍스트정제

**텍스트정제**

## 자연어 처리의 기본개념

### 말뭉치(corpus, 코퍼스)

- 자연어 처리로 정제하는 텍스트 데이터의 집합
- 대용량의 정형화된 텍스트의 집합
- 코포라(corpora): 둘 이상의 코퍼스 모음
- Google Books Ngram corpus, Brown corpus, American National corpus

### 코퍼스가 필요한 이유

- 통계분석 수행: 빈도 분포, 단어의 동시 발생 등
- 언어규칙 정의: 문법 교정 시스템
- 문제 진술을 해결하기 위해 필요한 데이터 타입을 결정

## 자연어 처리의 기본개념

### nltk.corpus

<https://www.nltk.org/api/nltk.corpus.html>  
[http://www.nltk.org/nltk\\_data/](http://www.nltk.org/nltk_data/)

```
import nltk  
from nltk.corpus import brown as cb  
from nltk.corpus import gutenberg as cg
```

```
print cb.categories()
```

```
[u'adventure', u'belles_lettres', u'editorial', u'fiction', u'government', u'hobbies', u'humor', u'learned', u'lore', u'mystery', u'news', u'religion', u'reviews', u'romance', u'science_fiction']
```

## 자연어 처리의 기본개념

### 텍스트 마이닝(Text Mining)

- 자연 언어 처리 기술을 활용하여 반정형/비정형 텍스트 데이터를 정형화하고, 특징을 추출하기 위한 기술
- 추출된 특징으로부터 의미 있는 정보를 발견할 수 있도록 하는 기술

## 자연어 처리용 파이썬 패키지

### NLTK(Natural Language Toolkit)

- 자연어 처리 및 문서분석용 파이썬 패키지  
<https://www.nltk.org/index.html>

### KoNLPy

- 한국어 자연어 처리  
<https://konlpy-ko.readthedocs.io/ko/v0.4.3/>
- 5가지 형태소 분석기를 제공
  - 한나눔(Hannanum), 꼬꼬마(Khma), 코모란(Komoran), 은전한잎(Mecan), OKT(이전의 Twitter)

## 텍스트 정제과정

### 다양한 이론

- **7단계**

- ①공란처리, ②대소문자 통일, ③숫자제거, ④문장부호/특수문자 제거,  
⑤불용어(stopword) 제거, ⑥어근 동일화(stemming), ⑦엔그램(n-gram)

- **3단계**

- ①정규표현식: 대소문자 통일, 숫자/불용어 제거, 어근 동일화, n-gram 적용
- ②사전처리
- ③품사분석

## 텍스트 정제과정

### 6단계

1. 사전정제
2. 토큰화
3. 정규화(normalization)
4. 불용어 제거
5. 결합제약 적용
6. 품사표식(part of speech tagging)

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

- 텍스트에서 단어 사이의 불필요한 빈칸을 제거
- 영어인 경우, 대문자와 소문자를 통일
- 특별한 의미를 가지지 않는 숫자 제거
- 느낌표나 물음표 같은 문장부호 및 특수문자를 제거
- 전체 텍스트를 일단 문장 단위로 정리

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

의심한그득 +. + 앞으로는 사람들은 많을 퇴근시간 말구 미리가서 사와야겠드 ㅓ('ù') 와  
맥주로 터진 입, 청포도와 고구마스틱으로 달래봄니당  
막상 먹다보니 양이 부족하진 않았는데 괜히 먹을 때 많이 먹자며 배 터지기 직전까지 냅나리 :D  
식후땡 아스크림 ㅓ(>\_\_\_< ' )  
브라우니쿠키 먹고 출근하세요 ♥  
gs편의점에서 파는 진~~~한 브라우니 쿠키인데 JMTgr  
ㅋㅋㅋㅋㅋㅋㅋㅋㅋ  
ㅠㅠㅠㅠㅠㅠㅠ

- 개행문자 제거
- 특수문자 제거
- 공백 제거
- 중복 표현 제거
- 이메일, 링크 제거
- 제목 제거
- 불용어 제거 조사 제거
- 띄어쓰기, 문장분리 보정
- 사전 구축

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

- 자연어를 어떤 단위로 살펴볼 것인가
  - 어절: 문장 성분의 최소 단위로서 띄어쓰기의 단위가 됨.
  - 형태소: 의미를 가지는 요소로서는 더 이상 분석할 수 없는 가장 작은 말의 단위
  - n-gram

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

His mother went to school with him

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

His / mother / went / to / school / with / him

텍스트정제

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

그의 어머니는 그와 함께 학교에 갔다

텍스트정제

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

그/의/ 어머니/는/ 그/와/ 함께/ 학교/에 가/았/다

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

- **토큰(Token)**

- 하나의 유용한 의미적 단위로 함께 모여지는 일련의 문자열
- 한국어의 경우, 명사, 동사, 조사 모두 토큰
- 토큰은 **형태소**일 수도 있고, 하나의 **단어**일 수도 있고, 두 단어 이상 결합된 복합어일 수도 있음

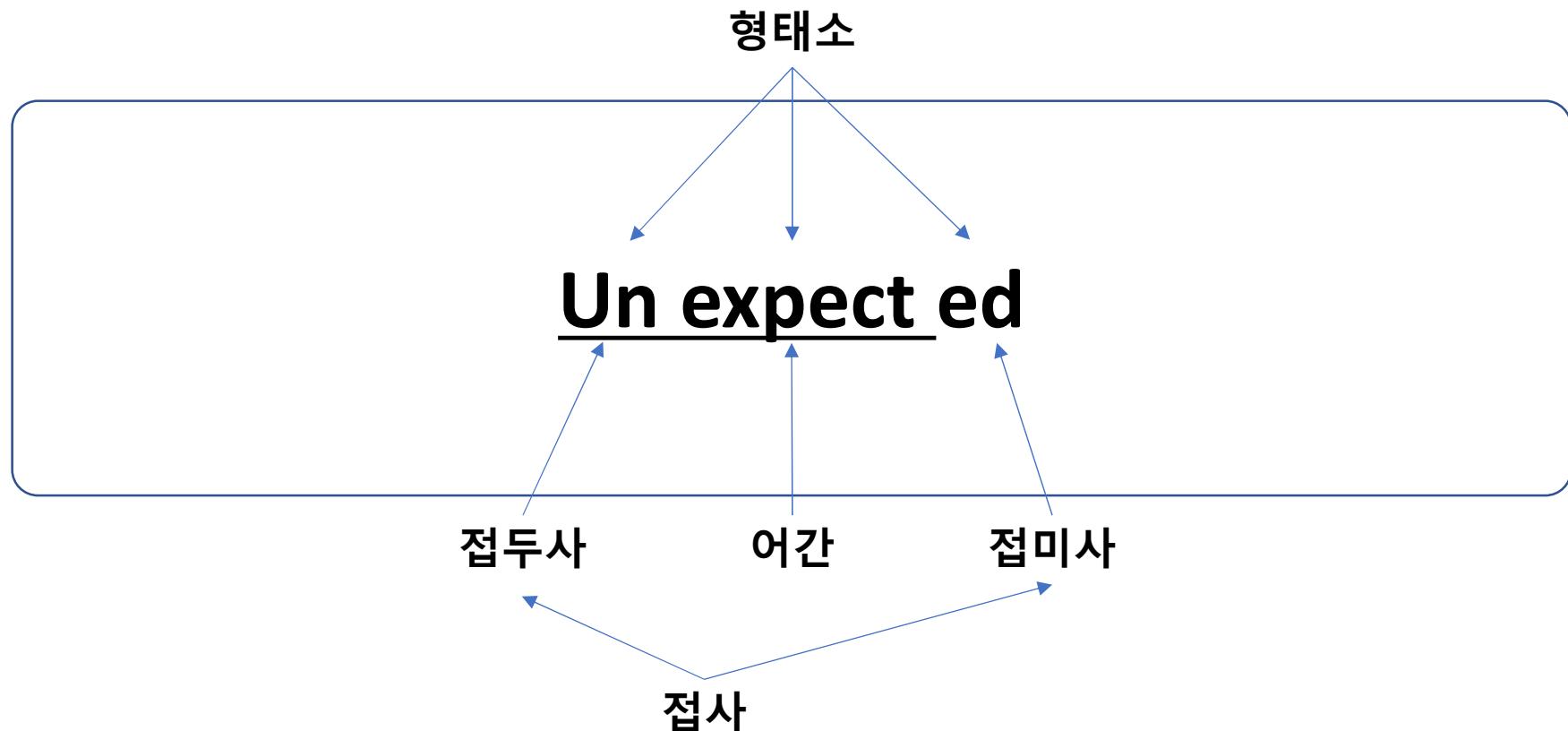
### 형태소(Morpheme)

- 더 이상 분석이 불가능한, 의미의 최소 단위
- 실질형태소: 체언(명사, 대명사), 용언(동사, 형용사)
- 형식형태소: 문법적 의미만 가진 말(조사, 접미사, 접두사, 어미나 어근 등)

### 단어(Word)

- 스스로 일정한 뜻을 담고있으면서 자립하여 쓸 수 있는 의미의 최소 단위

## 텍스트 정제과정



## 텍스트 정제과정

### [참고]

- **굴절어**
  - 단어 자체는 독립적으로 구분되고, 문법적 역할에 따라 단어 형태가 변화
  - 띄어쓰기 단위가 토큰이기 때문에 쉽게 구분 가능
  - 예: 영어
- **교착어**
  - 하나의 낱말에 첨가되는 조사나 접사에 의해 그 문법적 기능이 달라짐
  - 띄어쓰기만으로 토큰을 구분하기 어려움
  - 예: 한국어
- **고립어**
  - 단어의 어순에 의해 문법적 의미를 나타냄
  - 띄어쓰기를 거의 하지 않음
  - 예: 중국어

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

- 정규화(Normalization)
  - 형태소 분석을 하고 불규칙한 변화를 통일된 형태로 추출하는 과정
- 형태소 분석
  - 단어를 구성하는 각각의 형태소를 인식
  - 용언의 활용, 불규칙 활용이나 축약 탈락현상이 일어난 형태소는 원형으로 복원하는 과정

**John's** = John + 's, **books** = book + s

**flies** = fly + es, **taking** = take + ing, **prettier** = pretty + er

**깨뜨리셨더군요** = 깨(어근) + 뜨리(힘줌접사) + 시(높임어미) + 었(과거어미) + 더(어미) + 군(감탄어미) + 요(종결어미)

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > **3.정규화** > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

- 형태소분석은 중의성 때문에 어려움

텍스트정제

## 텍스트 정제과정

감기는

텍스트정제

## 텍스트 정제과정

# 감기는



감기(명사) + 는(조사)

텍스트정제

## 텍스트 정제과정

# 감기는



(줄을)감다(수동형)

텍스트정제

## 텍스트 정제과정

# 감기는



(머리를)감기다(사동형)

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

### 형태소 분석

#### 1. 어간 추출(stemming)

- 단어의 어미나 접두사, 접미사 등으로 해서 형태가 달라진 단어들을 형태소 분석을 통해 그 어간을 추출하여 동일한 단어로 간주하는 작업
- 어근 동일화를 통해 텍스트 정리
- 포터 스테머(PorterStemmer) 어간 추출 알고리즘: 영어

#### 2. 표제어 추출(lemmatization)

- 어간 추출의 단점 보완: 단어의 의미적 단위를 고려하지 않고 단어를 축약형으로 정리
- 형태소 분석을 통해서 더 정확한 단어수준 분석을 수행
- 원형 복원 : 품사정보가 보존된 형태의 기본형으로 변환하는 방법

텍스트정제

## 텍스트 정제과정

love, loves, loving, loved

텍스트정제

## 텍스트 정제과정

love, loves, loving, loved

어간추출

lov

표제어 추출

love

텍스트정제

## 텍스트 정제과정

innovation, innovations, innovate,  
innovates, innovative

텍스트정제

## 텍스트 정제과정

innovation, innovations, innovate,  
innovates, innovative

어간추출

innovat

표제어 추출

**innovation**  
(innovation, innovations)  
**innovate**  
(innovate, innovates)  
**innovative**  
(innovative)

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

- **형태소 분석**

어간추출	표제어 추출
문맥에 대한 지식없이 한 단어로 작동	문장에서 단어와 문맥을 고려
POS 태그를 고려하지 않음	POS 태그를 고려
비슷한 기본 의미를 가진 단어를 그룹화하는데 사용	사전, 또는 WordNet 종류의 사전을 만드는데 사용

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

- **불용어(Stop word) 제거**

- 기능어는 문장에서 실질적인 의미를 별로 가지고 있지 않기 때문에 불용어로 간주하여 제거
- 영어: 관사, 전치사
- 한국어: 조사

문장 = 지시어+ 기능어

(구체적인 대상이나 행동 상태를 가리킴)

(문법적인 기능)

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > **5.결합제약 적용** > 6.품사표식

- 복합어, 축약어, 신조어 등을 처리
- n-gram 적용 : 2번 이상의 단어들이 연이어 등장하는 빈도가 많을 때, 이들을 하나의 단어로 묶어서 처리하는 방법
- 텍스트의 복잡성이 감소되기도 하고, 동시에 증가하기도 함

텍스트정제

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

# Public Relations

## PR (마케팅)

위키백과, 우리 모두의 백과사전.

PR(피알) 또는 퍼블릭 릴레이션스(Public Relations → 대중 관계, 공중 관계)는 정부, 정당, 기업, 개인 등의 마케팅 주체가 대중(공중)과의 호의적인 관계를 위해 하는 모든 활동을 지칭한다. 마케팅 주체가 대중 매체를 이용하여 마케팅 활동을 하는 홍보(弘報)는 PR의 수단이다. 마케팅 주체는 현대 사회에서 가장 영향력 있는 정치 집단이자 소비 집단인 대중에게 호의를 얻기 위해 여러 가지 활동을 펼치는데, 기업이 직접적인 이익이 없어 보이는 자선 행사를 주최하거나 비영리 단체에 기부금을 납부하거나 또는 단체 봉사 활동을 실시하는 것도 모두 대중의 호의를 얻기 위한 행동이다.

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

# Public Relations

- 홍보
  - 공중 관계 ← 하나의 복합어로 간주
- 
- 하나의 단어로 묶어서 정리하기 때문에 단어수가 줄어든다  
→ 텍스트의 복잡성이 감소
  - 공중, 관계, 공중관계라는 3종류의 단어가 분석 대상이 된다  
→ 텍스트의 복잡성이 증가

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > 6.품사표식

### 품사표식(Part of Speech Tagging, POS 태깅)

- 문장에서 하나의 단어가 여러가지 품사(POS: part of speech)의 역할을 할 수 있기 때문에 정확하게 어떤 품사인가를 표시(tagging)하는 작업이 필요
- 말의 중의성 및 모호성

Tag	Description	설명	Example
CC	coordinating conjunction		
CD	cardinal digit		
DT	determiner		
EX	existential there		'there' is..., 'there' exists..
FW	foreign word		
IN	preposition/subordinating conjunction		
JJ	adjective	형용사	'big'
JJR	adjective, comparative	형용사, 비교급	'bigger'
JJS	adjective, superlative	형용사, 최상급	'biggest'
LS	list marker		'1)'
MD	modal		'could', 'will'
NN	noun, singular	명사, 단수형	'desk'
NNS	noun, plural	명사, 복수형	'desks'
NNP	proper noun, singular	고유명사, 단수형	'Harrison'
NNPS	proper noun, plural	고유명사, 복수형	'Americans'
PDT	predeterminer		'all the kids'
POS	possessive ending		'parent's'
PRP	personal pronoun	인칭 대명사	'I', 'he', 'she'
PRP\$	possessive pronoun	소유 대명사	'my', 'his', 'hers'
RB	adverb	형용사	'very', 'silently'
RBR	adverb, comparative	형용사, 비교급	'better'
RBS	adverb, superlative	형용사, 최상급	'best'
RP	particle		'give up'
TO	to		go 'to' the store
UH	interjection		'errrrrm'
VB	verb, base form	동사, 원형	'take'
VBD	verb, past tense	동사, 과거형	'took'
VBG	verb, gerund/present participle	동사, 현재분사	'taking'
VBN	verb, past participle	동사, 과거분사	'taken'
VBP	verb, sing. Present, non-3d		'take'
VBZ	verb, 3rd person sing. Present		'takes'
WDT	wh-determiner		'which'
WP	wh-pronoun		'who', 'what'
WP\$	possessive		'wh-pronoun', 'whose'
WRB	wh-adverb		'where', 'when'

## 텍스트 정제과정

1.사전정제 > 2.토큰화 > 3.정규화 > 4.불용어 제거 > 5.결합제약 적용 > **6.품사표식**

- **규칙(Rule)기반**
  - 먼저 수십만 단어와 그에 대한 표식으로 구성된 대용량 사전을 사용하여 문장 내에서 단어가 수행할 수 있는 품사의 목록을 부여
  - 미리 마련한 규칙에 근거해서 해당되지 않는 품사표식을 제거하는 순서로 이루어짐
- **확률(Stochastic)기반**
  - 하나의 단어가 어떤 품사의 역할을 수행하는지 확률적으로 계산하기 위해 만들어진 학습 말뭉치를 사용

텍스트정제

## 자연어 처리용 파이썬 패키지

### JDK 1.7+ 설치

<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>

### Java SE Development Kit 8u261

This software is licensed under the [Oracle Technology Network License Agreement for Oracle Java SE](#)

Windows x64

166.28 MB



jdk-8u261-windows-x64.exe



You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

I reviewed and accept the Oracle Binary Code License Agreement for Java SE

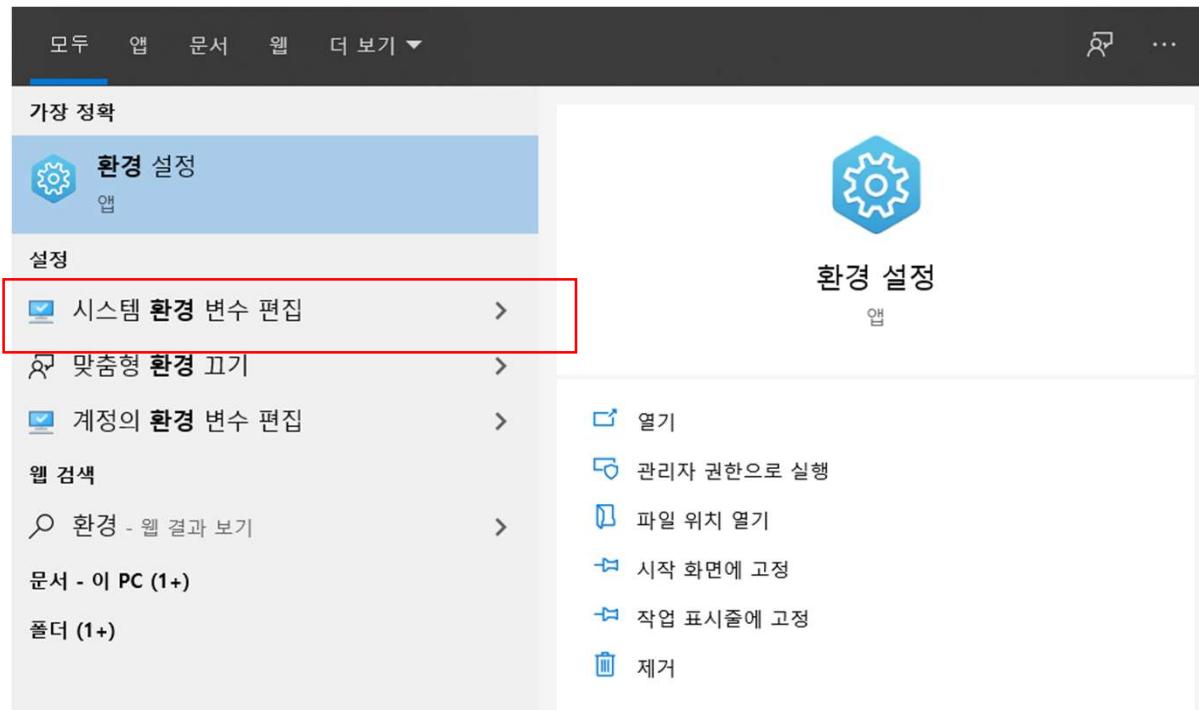
*You will be redirected to the login screen in order to download the file.*

Download jdk-8u261-windows-x64.exe

## 자연어 처리용 파이썬 패키지

**JAVA\_HOME 설정** [https://docs.oracle.com/cd/E19182-01/820-7851/inst\\_cli\\_jdk\\_javahome\\_t/index.html](https://docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/index.html)

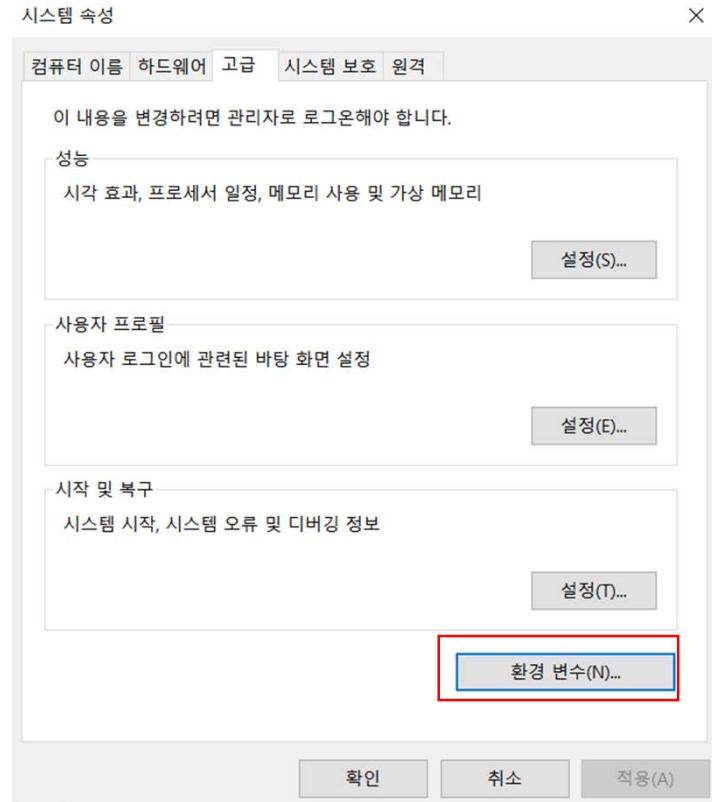
### 1. 시스템 환경 변수 편집



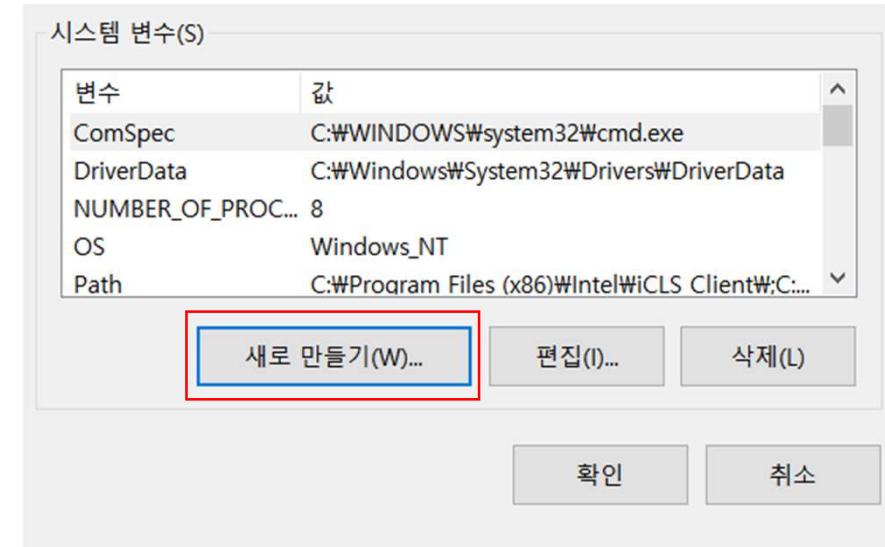
## 자연어 처리용 파이썬 패키지

### JAVA\_HOME 설정

2. 시스템 속성 창 > 환경 변수



3. 시스템 변수 > 새로 만들기

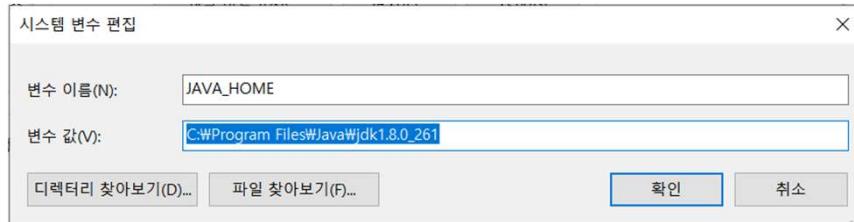


[https://docs.oracle.com/cd/E19182-01/820-7851/inst\\_cli\\_jdk\\_javahome\\_t/index.html](https://docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/index.html)

## 자연어 처리용 파이썬 패키지

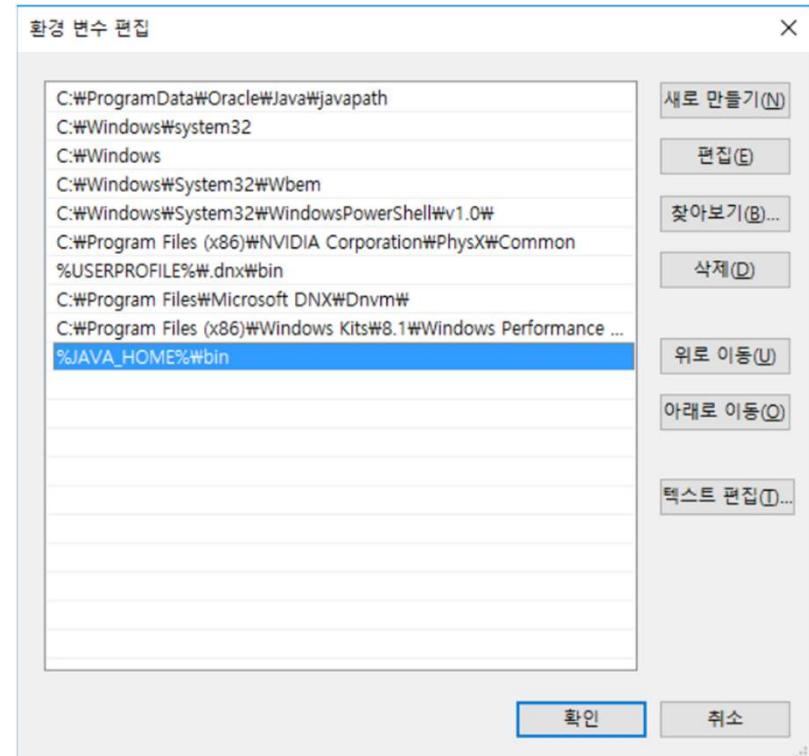
### JAVA\_HOME 설정

4. JAVA\_HOME 추가(Java 설치 경로  
복사: C:\Program Files\Java\jdk1.8.0\_261)



### 5. Path 추가

시스템 변수(S) 목록중 'Path' 선택 후 편집  
클릭, 새로 만들기(N) 클릭 후 아래의 값 추가

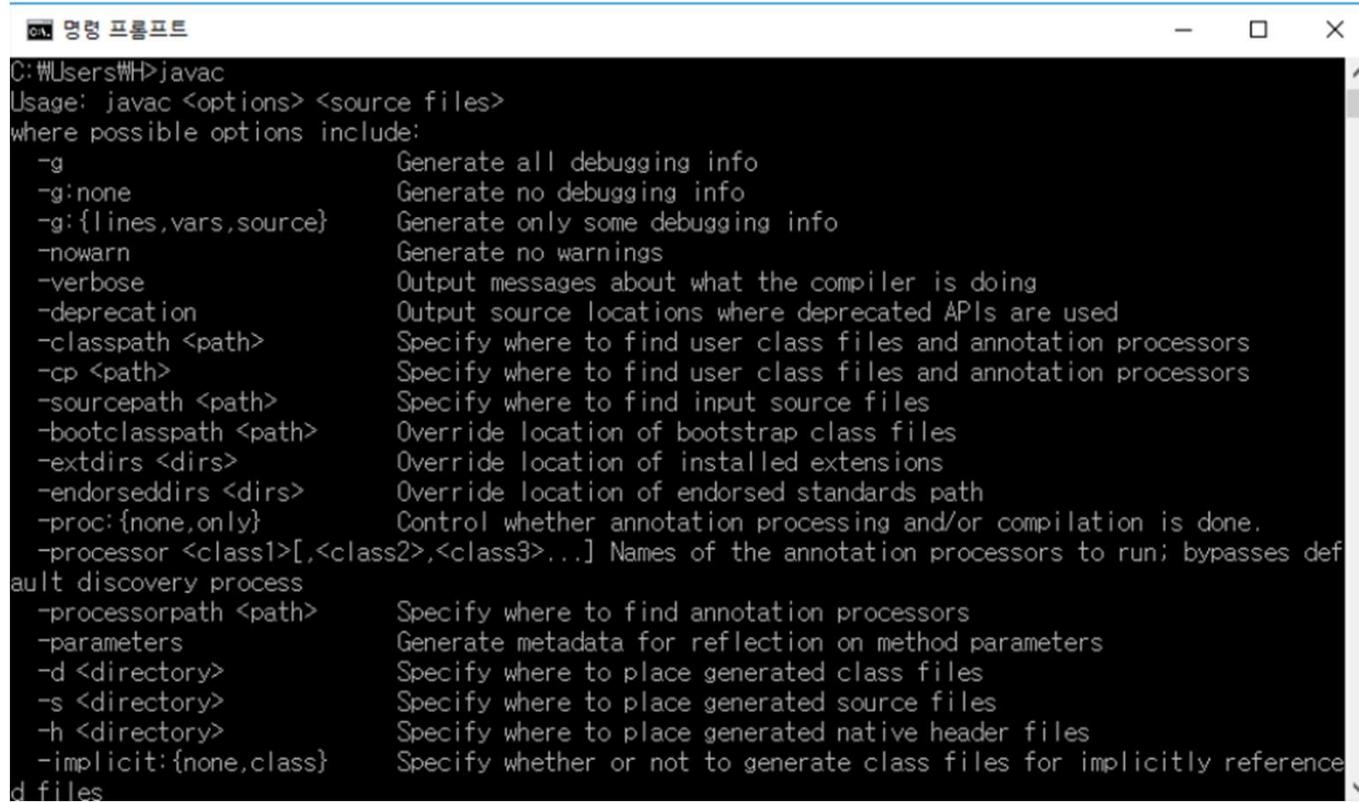


[https://docs.oracle.com/cd/E19182-01/820-7851/inst\\_cli\\_jdk\\_javahome\\_t/index.html](https://docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/index.html)

## 자연어 처리용 파일 패키지

**JAVA\_HOME 설정** [https://docs.oracle.com/cd/E19182-01/820-7851/inst\\_cli\\_jdk\\_javahome\\_t/index.html](https://docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/index.html)

### 6. 명령 프롬프트 > javac 실행



```
C:\Users\H>javac
Usage: javac <options> <source files>
where possible options include:
  -g                      Generate all debugging info
  -g:none                 Generate no debugging info
  -g:{lines,vars,source}   Generate only some debugging info
  -nowarn                 Generate no warnings
  -verbose                Output messages about what the compiler is doing
  -deprecation            Output source locations where deprecated APIs are used
  -classpath <path>       Specify where to find user class files and annotation processors
  -cp <path>               Specify where to find user class files and annotation processors
  -sourcepath <path>      Specify where to find input source files
  -bootclasspath <path>   Override location of bootstrap class files
  -extdirs <dirs>          Override location of installed extensions
  -endorseddirs <dirs>    Override location of endorsed standards path
  -proc:{none,only}        Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path>    Specify where to find annotation processors
  -parameters              Generate metadata for reflection on method parameters
  -d <directory>           Specify where to place generated class files
  -s <directory>           Specify where to place generated source files
  -h <directory>           Specify where to place generated native header files
  -implicit:{none,class}  Specify whether or not to generate class files for implicitly referenced files
```

## 자연어 처리용 파이썬 패키지

### JPyte1 (>=0.5.7)을 다운로드 받고 설치

```
> pip install jpyte1
```

### KoNLPy 설치하기

```
> pip install konlpy
```

### 주요 클래스 테스트

<https://konlpy.org/ko/latest/api/konlpy.tag/>

## 자연어 처리용 파이썬 패키지

**mecab 설치** <https://github.com/Pusnow/mecab-python-msvc>

1. **mecab-ko-msvc 설치**: C 기반으로 만들어진 mecab이 윈도우에서 실행될 수 있도록 하는 역할
2. **mecab-ko-dic-msvc**: 기본 사전 설치
3. **python wheel 설치**
4. **실행**

## 자연어 처리용 파이썬 패키지

### 1. mecab-ko-msvc

<https://github.com/Pusnow/mecab-ko-msvc/releases/tag/release-0.9.2-msvc-3>

64bit 다운로드

C:/mecab/ 폴더 아래 압축풀기

### 2. mecab-ko-dic-msvc

<https://github.com/Pusnow/mecab-ko-dic-msvc/releases/tag/mecab-ko-dic-2.1.1-20180720-msvc>

zip 파일 다운로드

C:/mecab/ 폴더 아래 압축풀기

### 3. python wheel 설치(계속)

[https://github.com/Pusnow/mecab-python-msvc/releases/tag/mecab\\_python-0.996\\_ko\\_0.9.2\\_msvc-2](https://github.com/Pusnow/mecab-python-msvc/releases/tag/mecab_python-0.996_ko_0.9.2_msvc-2)

[mecab\\_python-0.996\\_ko\\_0.9.2\\_msvc-cp37-cp37m-win\\_amd64.whl](#) 다운로드

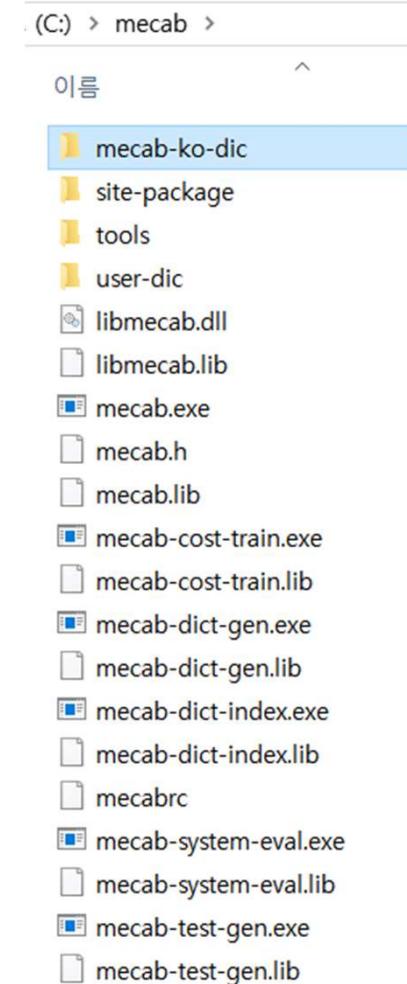
C:/mecab/site-package 폴더 아래로 이동

## 자연어 처리용 파이썬 패키지

### 3. python wheel 설치(계속)

```
C:\mecab\site-package>pip install mecab_python-0.996_ko_0.9.2_msvc-cp37-cp37m-win_amd64.whl
```

```
관리자: Anaconda Prompt (Anaconda3)
(base) C:\mecab\site-package>pip install mecab_python-0.996_ko_0.9.2_msvc-cp37-cp37m-win_amd64.whl
Processing c:\mecab\site-package\mecab_python-0.996_ko_0.9.2_msvc-cp37-cp37m-win_amd64.whl
Installing collected packages: mecab-python
Successfully installed mecab-python-0.996-ko-0.9.2-msvc
```



## 자연어 처리용 파이썬 패키지

### 4. 실행

```
from konlpy.tag import Mecab
mecab = Mecab(dicpath="C:\\mecab\\mecab-ko-dic")
print(mecab.morphs(u'영등포구청역에 있는 맛집 좀 알려주세요.'))
```

## 한국어 처리

### 한국어 POS 예제

그의 어머니는 그와 함께 학교에 갔다

- 세종 품사표식 목록: 국립국어원의 세종 말뭉치 사전에 기반
- KoNLPy의 형태소 분석마다 품사 표식 목록은 조금씩 다르다

[https://docs.google.com/spreadsheets/d/1OGAjUvalBuX-oZvZ\\_-9tEfYD2gQe7hTGsgUpiIBSXl8/edit#gid=0](https://docs.google.com/spreadsheets/d/1OGAjUvalBuX-oZvZ_-9tEfYD2gQe7hTGsgUpiIBSXl8/edit#gid=0)

## 한국어 처리

### 한국어 POS 예제

그의 어머니는 그와 함께 학교에 갔다

그/의/ 어머니/는/ 그/와/ 함께/ 학교/에/가/있/다

## 한국어 처리

### 한국어 POS 예제

그의 어머니는 그와 함께 학교에 갔다

(NP 대명사) (JKG 관형격 조사)(NNG 일반명사)(JKS 주격조사)(NP 대명사) (JKB 부사격조사)

그/의/ 어머니/는/ 그/와/  
함께/ 학교/에/가/있/다

(MAG 일반부사) (NNG 일반명사) (JKB 부사격조사) (vw동사) (EC 연결어미) (EF 종결어미)

## 영어 처리

### 영어 POS 예제

His mother went to school with him.

- 브라운 말뭉치(Brown corpus) : 미국의 브라운 대학에서 구축, 87개의 표식 목록
- 펜트리뱅크(Penn Treebank) : 45개 표식 목록으로 구성
- C4 표식목록: 61개로 구분
- NLTK 패키지에서 사용하는 품사 표식 목록

## 영어 처리

### 영어 POS 예제

His mother went to school with him.

His / mother / went / to / school / with / him

## 영어 처리

### 영어 POS 예제

His / mother / went / to / school / with / him

His(PPR\$ 소유대명사) mother(NN 단수명사) went(VRD 동사과거형) to(TO, to)  
school(NN 단수명사) with(IN 전치사) him(PRP, 인칭대명사)

## 핵심어

### 핵심어(Keyword)

- 텍스트 자료의 중요한 내용을 압축적으로 제시하는 단어 또는 문구

### 핵심어분석

- 불용어 제거와 형태소 분석, 어간 추출 등의 자연어 처리를 시행한 후 텍스트에서 많이 등장하는 단어의 등장 빈도를 분석함으로써 핵심어를 추출하는 것

## 핵심어 분석

### 핵심어 분석 효과

- 텍스트의 주제가 무엇인지 짐작 가능
- 텍스트가 서로 어느 정도 비슷한지 파악
- 인터넷 등에서 문서를 검색할 때 기초
- 검색 엔진에서 검색결과의 우선 순위를 결정

### 방법

- 단순 빈도를 제시하는 방법
  - 워드 클라우드(Word Cloud, 단어 구름)로 빈도 분석 결과를 제시하는 방법
- TF-IDF(Term Frequency-Inverse Document Frequency, 어휘 빈도-문서 역빈도)를 계산

## 단순 빈도 분석

### 빈도 카운트

- 불용어 제거
- 출현 빈도 카운트
  - from collections import Counter

## 단순 빈도 분석

### 워드 클라우드(Word Cloud)

- 텍스트에 등장하는 단어를 그 등장 빈도에 따라 서로 크기가 다르게 구름형태로 표현
- 어떤 단어가 많이 등장하고 어떤 단어가 적게 등장하는가를 한 눈에 알 수 있게 하는 방식
- 단어들 사이의 연관성이나 의미 구조 등을 분석하는데 한계

- pip install wordcloud
- pip install -U pytagcloud
- pip install pygame
- pip install simplejson
- from wordcloud import WordCloud
- import pytagcloud



## 단순 빈도 분석

### 1. pytagcloud 관련 패키지 설치

- pip install -U pytagcloud
- pip install pygame
- pip install simplejson

### 2. font.json 설정

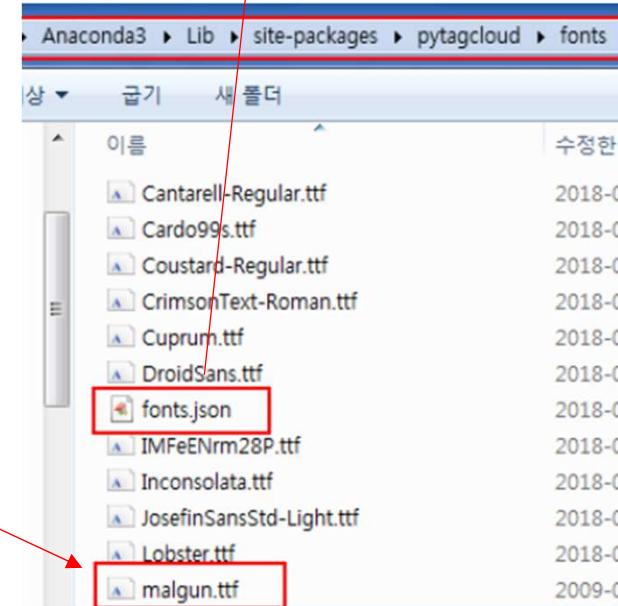
```
{  
    "name": "korean",  
    "ttf": "malgun.ttf",  
    "web": "http://fonts.googleapis.com/css?family=Nobile"  
}
```

### 3. 한글 폰트 설정

#### 한글 폰트를 pytagcloud/fonts 폴더에 복사



C:\Windows\Fonts



C:\ProgramData\Anaconda3\Lib\site-packages\pytagcloud\fonts

## TF-IDF(Term Frequency-Inverse Document Frequency)

### TF-IDF

- 어휘가 다른 문서에는 별로 등장하지 않고, **특정 문서에만 집중적으로 등장할 때 그 어휘야 말로 실질적으로 그 문서의 주제를 잘 담고 있는 핵심어라 할 수 있다.**

### 단어-문서 행렬TDM(Term-Document Matrix)

- 빈도 분석뿐 아니라 비슷한 단어들끼리 묶느 요인 분석이나 비슷한 문서끼리 묶는 군집분석, 단어들끼리의 동시 출현에 기반한 의미 연결망 분석, 토픽 모델링도 가능

## 사회 연결망 분석(SNA, Social Network Analysis)

### 사회 연결망 분석

- 분석 대상(node)이 서로 어떻게 관련을 맺고 연결망(network)을 구성하는가
- 대상들 간의 관계를 연결망 구조로 표현하고 이를 계량적으로 제시하는 분석기법
- 사회학, 경영, 인문, 공학 등 다양한 분야에서 활용
- SNS에서 서로 어떻게 연결되는가 분석

## 의미 연결망 분석(Semantic Network Analysis)

### 의미 연결망 분석

- 연결망 분석 기법을 텍스트 내 단어의 관계에 적용하여 텍스트의 의미 구조를 파악하려는 분석 기법
- 일정한 범위 내에서 어휘가 동시에 등장하면 이를 서로 연결된 것으로 간주

## 의미 연결망 분석(Semantic Network Analysis) - 사례

### 문서 X

스마트폰은 디자인!

### 문서 Y

카메라 성능이 좋아요

### 문서 Z

디자인도 좋네요

## 의미 연결망 분석(Semantic Network Analysis)

### 의미 연결망 분석(Semantic Network Analysis) - 사례

문서 X

스마트폰은 디자인!

문서 Y

카메라 성능이 좋아요

문서 Z

디자인도 좋네요

## 인접행렬(Adjacent Matrix)

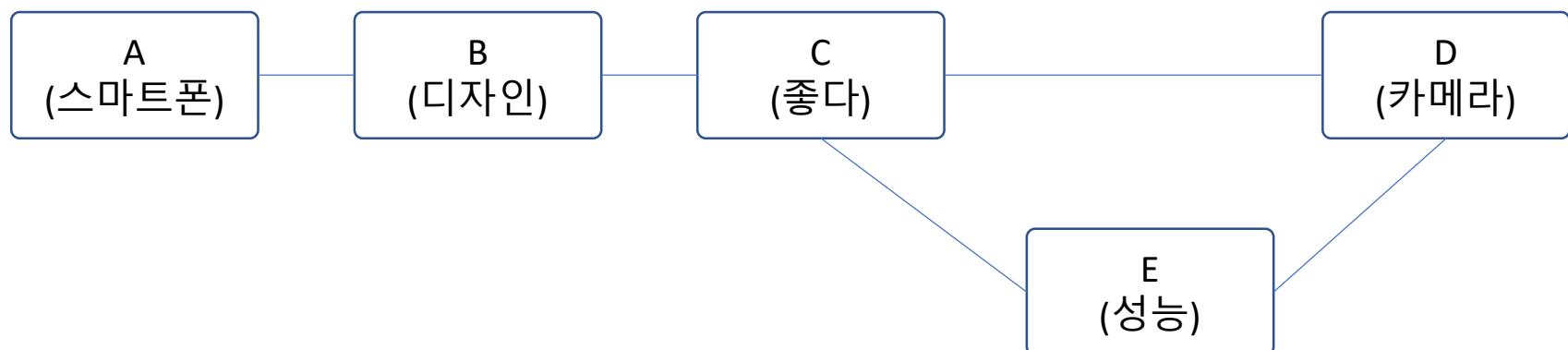
	A(스마트폰)	B(디자인)	C(좋다)	D(카메라)	E(성능)
A(스마트폰)	0	1	0	0	0
B(디자인)	1	0	1	0	0
C(좋다)	0	1	0	1	1
D(카메라)	0	0	1	0	1
E(성능)	0	0	1	1	0

## 의미 연결망 분석(Semantic Network Analysis) - 사례

### 인접행렬(Adjacent Matrix)

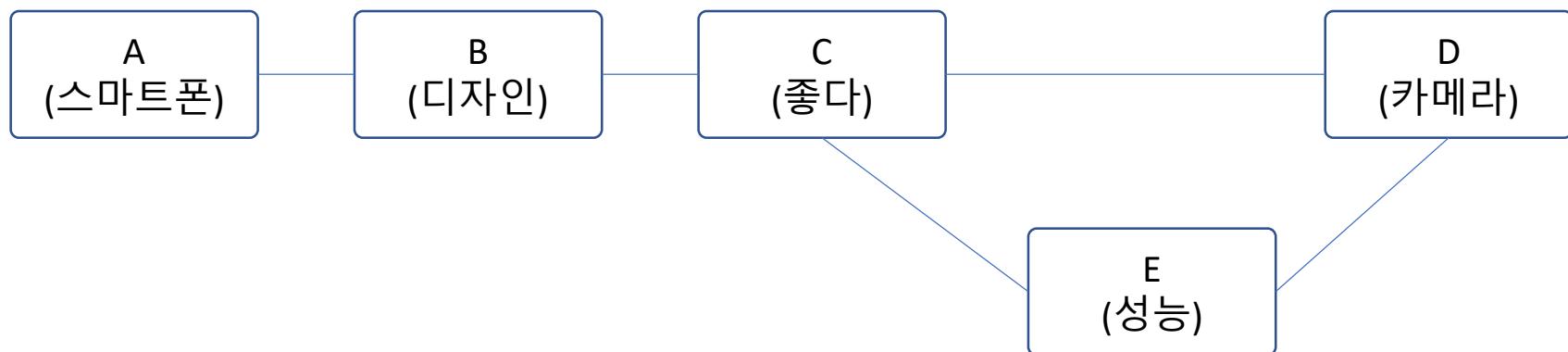
	A(스마트폰)	B(디자인)	C(좋다)	D(카메라)	E(성능)
A(스마트폰)	0	1	0	0	0
B(디자인)	1	0	1	0	0
C(좋다)	0	1	0	1	1
D(카메라)	0	0	1	0	1
E(성능)	0	0	1	1	0

### 연결망 그래프



## 의미 연결망(Semantic Network) 속성

### 연결망 그래프



	A(스마트폰)	B(디자인)	C(좋다)	D(카메라)	E(성능)
연결정도	1	2	3	2	2
연결중심성	$\frac{1}{4}=0.25$	$\frac{2}{4}=0.5$	$\frac{3}{4}=0.75$	$\frac{2}{4}=0.5$	$\frac{2}{4}=0.5$
매개중심성	$0/6=0$	$3/6=0.5$	$4/6=0.66$	$3/6=0.5$	$3/6=0.5$
근접중심성	$4/9=0.44$	$4/7=0.57$	$4/5=0.8$	$4/7=0.57$	$4/7=0.57$

## 의미 연결망(Semantic Network) 속성

### 연결 중심성(Degree centrality)

- 텍스트에서 한 단어에 직접 연결된 다른 단어의 수가 얼마나 많은지 측정
- 연결망의 크기에 따라 값을 비교하기 어렵기 때문에, 표준화 필요
  - ✓ 
$$\frac{\text{특정 노드 } i\text{와 직접 연결된 노드 수}}{\text{노드 } i\text{와 직간접적으로 연결된 모든 노드 수}}$$

## 의미 연결망(Semantic Network) 속성

### 매개 중심성(Betweenness centrality)

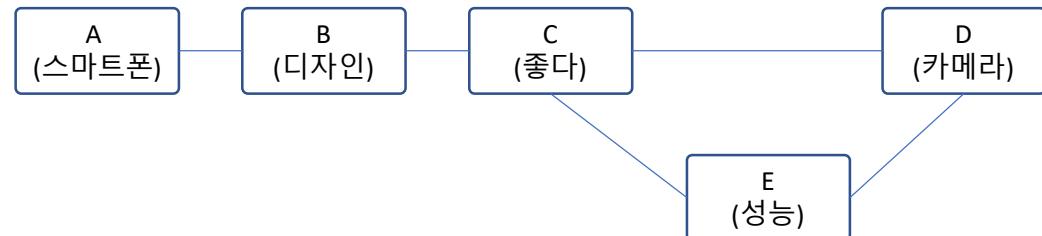
- 한 단어가 다른 단어들과의 연결망을 구축하는 데 매개자 역할을 얼마나 수행하는지 측정
- 단어들의 등장 빈도가 낮더라도 매개 중심성이 높으면 단어들 간 의미 부여 역할이 커짐

## 의미 연결망(Semantic Network) 속성

### 근접 중심성(Closeness centrality)

- 한 단어가 다른 단어에 얼마나 가깝게 있는지 측정
- 직접 연결 뿐 아니라 간접적으로 연결된 모든 단어들 사이의 거리를 측정

	A(스마트폰)
연결정도	1
근접중심성	4/9=0.44



전체 노드 수 - 1

$\frac{(A, B) \text{의 최단 경로} + (A, C) \text{의 최단 경로} + (A, D) \text{의 최단 경로} + (A, E) \text{의 최단 경로}}{\text{전체 노드 수} - 1}$

$$= \frac{5 - 1}{1 + 2 + 3 + 3} = \frac{4}{9}$$

## 의미 연결망(Semantic Network) 속성

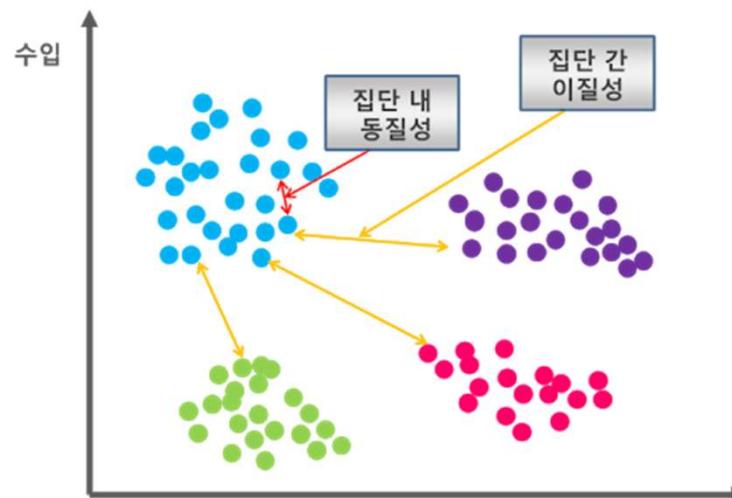
### 위세 중심성(Eigenvector centrality)

- 연결된 상대 단어의 중요성
- 중요한 단어와 많이 연결될수록 높아짐

## 군집분석(Clustering Analysis)

### 군집분석이란

- 주어진 대상물을 비슷한 것끼리 분류하여 묶는 분석방법
- 집단 내 동질성과 집단 간 이질성을 최대화
- 군집 내 거리(intra-cluster distance)는 최소화, 군집 간 거리(inter-cluster distance)는 최대화



## 군집분석(Clustering Analysis)

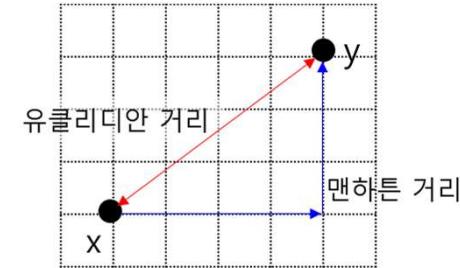
### 군집분석 응용 사례

- 광고 마케팅 분야에서 소비자 조사 데이터를 바탕으로 소비자를 비슷한 집단끼리 묶어내는 표적시장 세분화 전략
- 심리학의 심리검사 데이터를 바탕으로 비슷한 유형의 성격 집단으로 묶는 작업
- 유사 중복문서 검출(Near duplicates detection)
- 검색엔진 최적화(Search Engine Optimization: SEO)
- 추천 시스템(Recommendation System)
- 문서 요약(Document Summarization)

## 군집분석(Clustering Analysis)의 유사성 계산

### 유클리디안 거리

$$d_{ij} = \sqrt{(X_i - X_j)'(X_i - X_j)}$$



**마할라노비스 거리:** 변수의 상관성을 보정하여 두 벡터간의 거리를 측정

$$d_{ij} = (X_i - X_j)' S^{-1} (X_i - X_j)$$

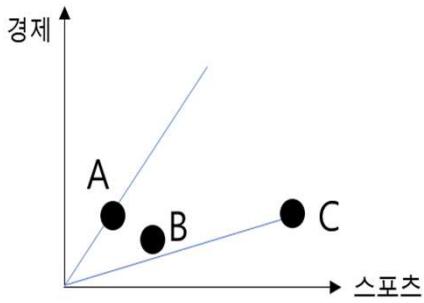
**민코프스키 거리 :** 맨하튼 거리와 유클리디안 거리를 한번에 표현

$$d_{ij} = \left[ \sum_{k=1}^p |X_{ik} - X_{jk}|^m \right]^{1/m}$$

## 군집분석(Clustering Analysis)의 유사성 계산

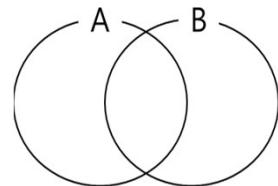
### 코사인 유사도

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$



- . B를 거리 기반으로 판단했을 경우: A와 근접하므로 경제기사
- . B를 방향(각도) 기반으로 판단했을 경우: C와 근접하므로 스포츠기사  
→ A와 B는 기사 내용이 부족, C는 기사 내용이 풍부
- 이 경우에는 크기는 고려하지 않고, 방향만 이용해서 유사도를 구하는 코사인 유사도를 사용

**자카드 유사도:** 두 집합의 교집합과 합집합을 통해 유사도를 측정하는 방법



$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

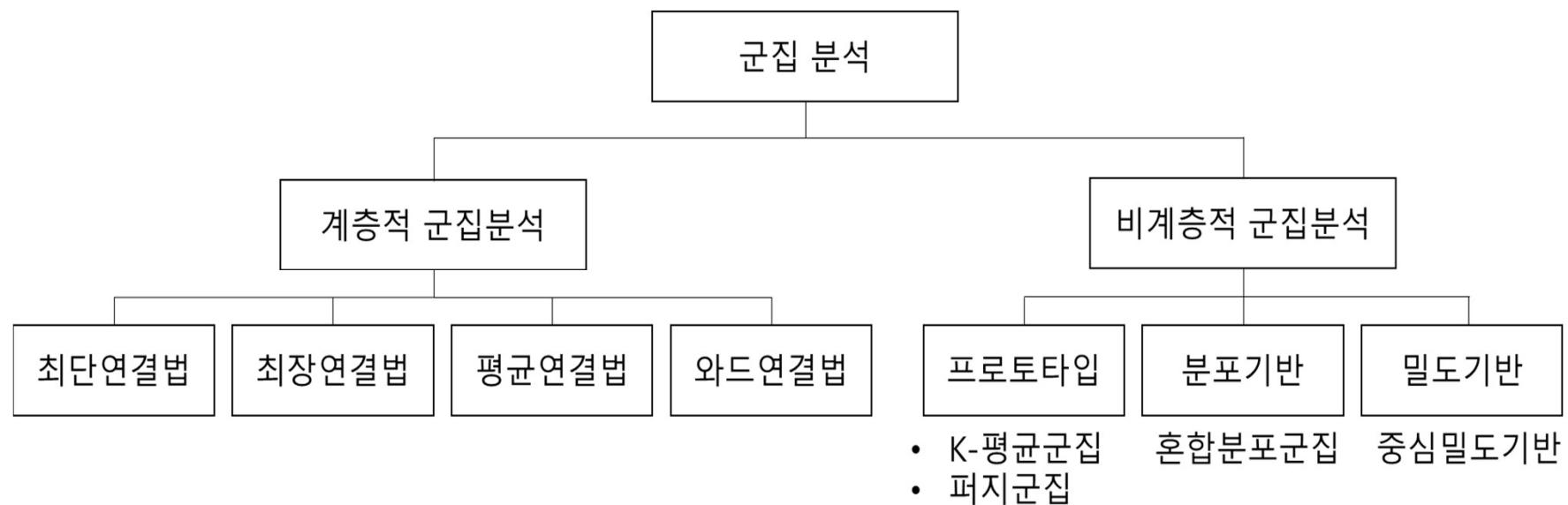
A = {상품 A를 구매한 소비자}, B={상품 B를 구매한 소비자}

→ 상품 A를 구매한 소비자와 상품 B를 구매한 소비자의 교집합이 클수록, 두 상품의 소비자 구매 관련 유사도가 높다고 판단 가능

## 군집분석(Clustering Analysis)

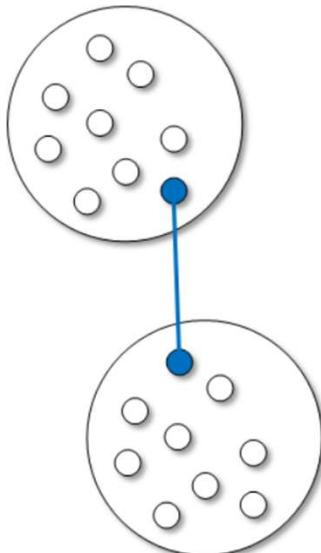
### 군집분석 구분

- 군집분석에는 n개의 군집으로 시작해 점차 군집의 개수를 줄여 나가는 **계층적 군집분석 방법**과, 모든 가능한 방법을 점검해 최적화한 군집을 형성하는 **비계층적 군집분석 방법**이 있다

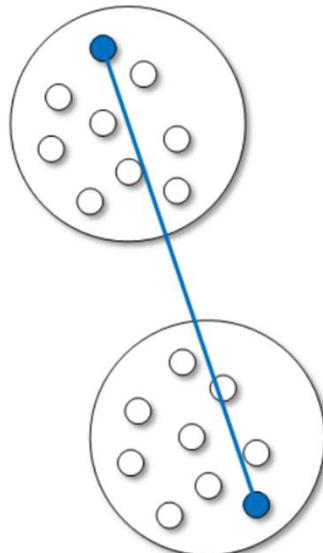


## 군집분석(Clustering Analysis)

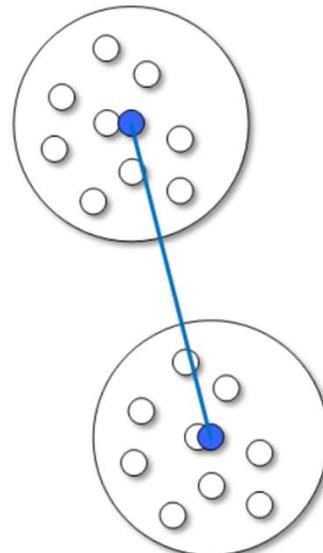
### 계층적 군집분석 구분



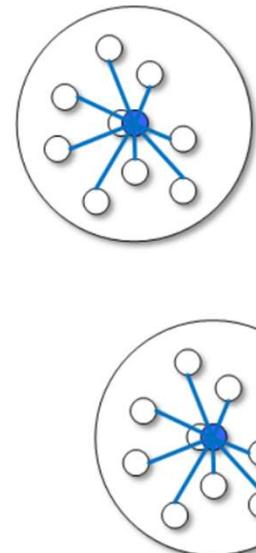
Single linkage  
최단연결법



Complete linkage  
최장연결법



Centroid linkage  
중심연결법



Ward linkage  
와드연결법

두 군집 사이에서  
가장 가깝게  
연결된 거리를  
유사도로 여김

두 군집 사이에서  
가장 멀게 연결된  
거리를 유사도로  
여김

두 군집의 평균  
거리를 유사도로  
여김

군집간의 거리에  
따라 연결하기  
보다는 군집내  
편자들의 제곱합에  
근거를 두고 군집

## 토픽모델링

- 문서를 이루는 핵심어를 바탕으로 문서에서 토픽, 즉 주제를 추출해 주는 확률모델 알고리즘
- 여러 문서로 이루어진 말뭉치가 있을 때, 각 문서들의 핵심어 분석을 통해서 토픽을 추출
- 잠재 디리클레 할당(LDA, Latent Dirichlet Allocation) 모형

## 토픽모델링

### 토픽모델링 vs 군집분석

- 군집분석: 각 문서가 하나의 토픽을 담고 있다고 가정하여 문서를 완전히 분리된 집단으로 묶음
- 토픽모델링: 각 문서에 여러 토픽이 담겨 있을 수 있다고 가정하기 때문에 문서를 몇 개의 군집으로 묶지 않음

### 군집분석과 비슷하지만 더 세밀하게 분석

각 문서별로 어떤 토픽들이 분포되어 있는가, 토픽이 어떤 문서들에 어떻게 분포되어 있고, 단어들이 토픽을 얼마나 드러내는가 살펴봄

## 토픽모델링

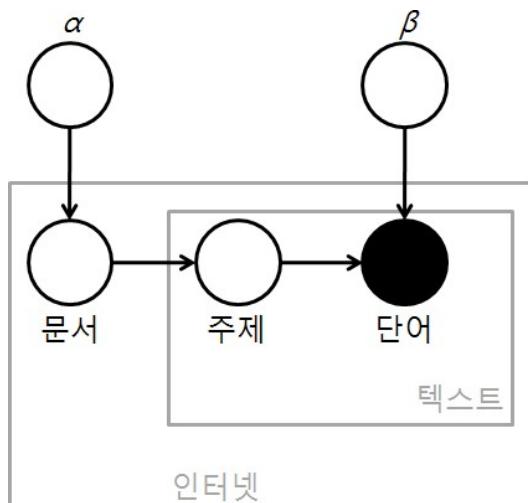
### 역동적 토픽 모델링

- 시간에 따라 토픽이 어떻게 변화했는지도 살펴 볼 수 있음

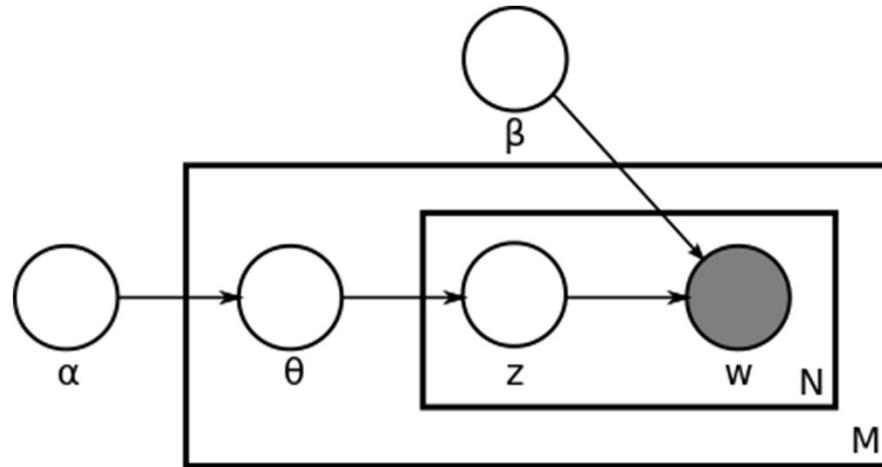
## 잠재 디리클레 할당(LDA, Latent Dirichlet Allocation)

### 잠재 디리클레 할당

- 말뭉치에서 문서를 통해 겉으로 드러난 단어들을 관찰하여 문서에 잠재된 토픽(주제)을 추론
- 말뭉치의 위계: 문서-토픽(주제)- 단어의 3가지 수준



## 잠재 디리클레 할당(LDA, Latent Dirichlet Allocation)



- $\alpha$ 는  $k$  차원 디리클레 분포의 매개변수이다.
- $\theta$ 는  $k$  차원 벡터이며,  $\theta^i$ 는 문서가  $i$ 번째 주제에 속할 확률 분포를 나타낸다.
- $z_n$ 는  $k$  차원 벡터이며,  $z_n^i$ 는 단어  $w_n$ 이  $i$ 번째 주제에 속할 확률 분포를 나타낸다.
- $\beta$ 는  $k \times V$  크기의 행렬 매개변수로,  $\beta_{ij}$ 는  $i$ 번째 주제가 단어집의  $j$ 번째 단어를 생성할 확률을 나타낸다.

여기에서  $w_n$ 은 실제 문서를 통해 주어지며, 다른 변수는 관측할 수 없는 잠재 변수이다.

이 모형은 다음과 같이 해석될 수 있다. 각 문서에 대해  $k$ 개의 주제에 대한 가중치  $\theta$ 가 존재한다. 문서 내의 각 단어  $w_n$ 은  $k$ 개의 주제에 대한 가중치  $z_n$ 을 가지는데,  $z_n$ 은  $\theta$ 에 의한 다항 분포로 선택된다. 마지막으로 실제 단어  $w_n$ 이  $z_n$ 에 기반하여 선택된다.

## 단어임베딩(Word Embedding)

### 단어 임베딩이란?

- 단어가 가진 의미를 그대로 보존하면서 의미와 맥락을 고려하여 단어를 벡터로 표현하는 것
- 개별 단어가 가진 속성, 즉 단어의 의미를 그대로 보존하면서 실수값으로 채워진 벡터로 표현하는 것

### 임베딩

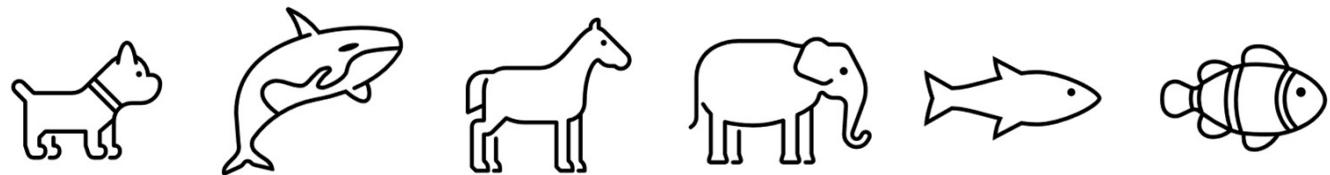
어떤 수학적 구조 혹은 대상을 본래 객체가 가진 연결성이나 대수적 성질을 그대로 유지하면서 특정 공간에 사상하는 것

## 단어 표현(Word Representation)

### 특징추출과 분류

- 분류를 하기위해서 데이터를 수학적으로 표현
- 특징추출(Feature Extraction) : 분류 대상의 특징을 파악

분류 대상



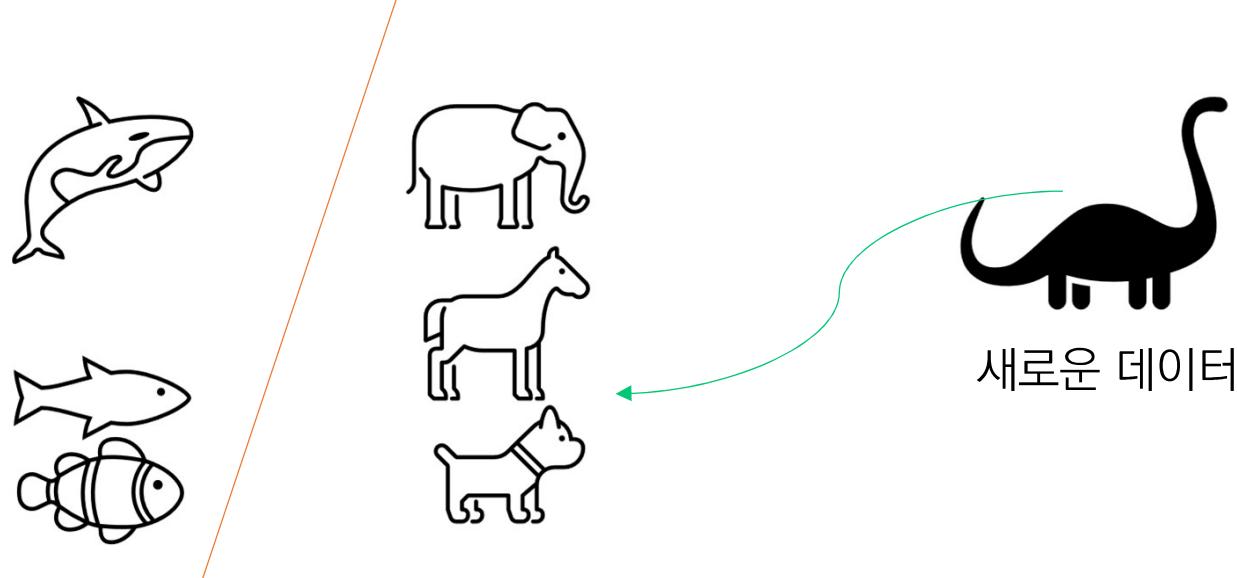
특징

크기가 다양  
다리의 개수가 다양

## 단어 표현(Word Representation)

### 특징추출과 분류

- 분류 대상의 특징(Feature)을 기준으로 분류대상을 그래프 위에 표현 가능
- 분류 대상들의 경계를 수학적으로 나눌수 있음
- 새로운 데이터 역시 특징을 기준으로 그래프에 표현하면, 어떤 그룹과 유사한지 파악 가능



## 단어 표현(Word Representation)

### 자연어에서의 특징 추출과 분류

- 과거에는 사람이 직접 특징(Feature)을 파악해서 분류
- 실제 복잡한 문제에서 분류 대상의 특징을 사람이 파악하기 어려울 수 있음
- 이러한 특징을 컴퓨터가 스스로 찾고(Feature extraction), 스스로 분류(Classification)하는 것이 기계학습의 핵심 기술

분류 대상

이순신은 조선 중기의 무신이다

지금 몇시야?

나 지금 너무 행복해

내일 날씨 알려줘

최초의 컴퓨터는 누가 만들었어?

아이유 노래 틀어줘

분류 대상의 특징

?

## 단어 표현(Word Representation)

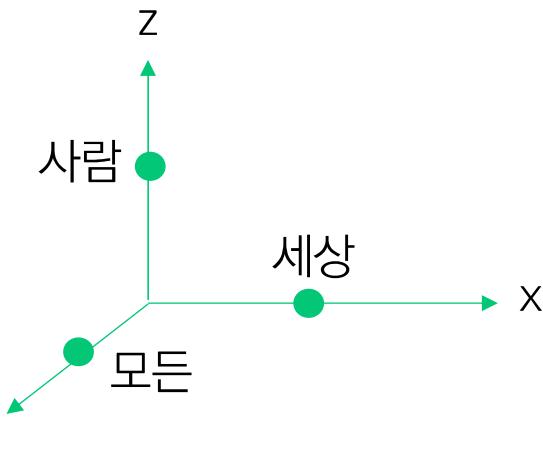
자연어를 어떻게 좌표평면 위에 표현할 수 있을까?

- 가장 단순한 표현 방법은 one-hot encoding 방식 → sparse representation

세상 모든 사람

단어	Vector
세상	[1,0,0]
모든	[0,1,0]
사람	[0,0,1]

n개의 단어는 n차원 벡터로 표현

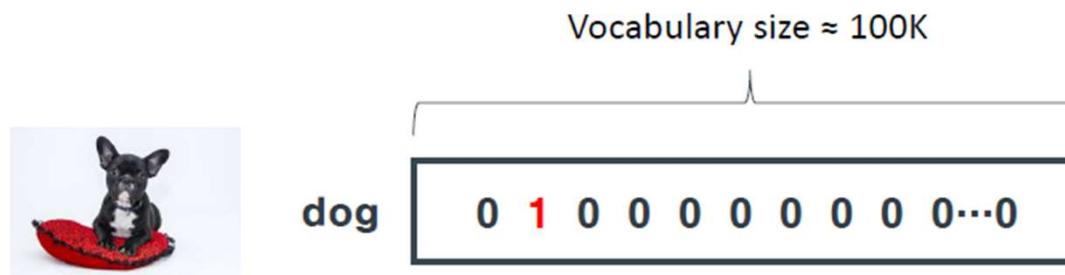


단어 벡터가 sparse해서 단어가 가지는 ‘의미’를 벡터 공간에 표현 불가능

## 단어 표현(Word Representation)

## One-hot representation

- 단어들간의 관계성을 고려하여 표현하지 않음
  - 매우 높은 dimension을 가지는 문제점: memory expensive, sparse



## 단어임베딩(Word Embedding)

### Word2Vec

- 자연어(특히 단어)의 의미를 벡터 공간에 임베딩
- 한 단어의 주변 단어들을 통해 그 단어의 의미를 파악
- 단어와 단어 사이의 관계를 벡터 사이의 연산관계로 환원하여 측정할 수 있도록 함
- 단어 사이의 의미 관계에 대해서 쉽게 일반화 할 수 있음
- 평가 방법: 유사한 벡터를 가진 단어가 과연 유사한 의미를 갖는지 여부, 단어를 재현하는 공간에서 거리가 얼마나 의미가 있는지 여부
- CBOW, skip-gram

## 단어임베딩(Word Embedding)

Word2Vec

الكلب

ورج



컴퓨터가 볼 때, 자연어는 우리가 보는 아랍어처럼 **기호**로만 보일 뿐!

## 단어임베딩(Word Embedding)

### Word2Vec

الكلب  
(개)

가 멍멍 짖었다

ورج

(강아지)

가 멍멍 짖었다



الكلب와 ورج가 무슨 뜻인지 모르겠지만, 주변 단어 형태가 비슷하니 의미도 비슷할 것이다!

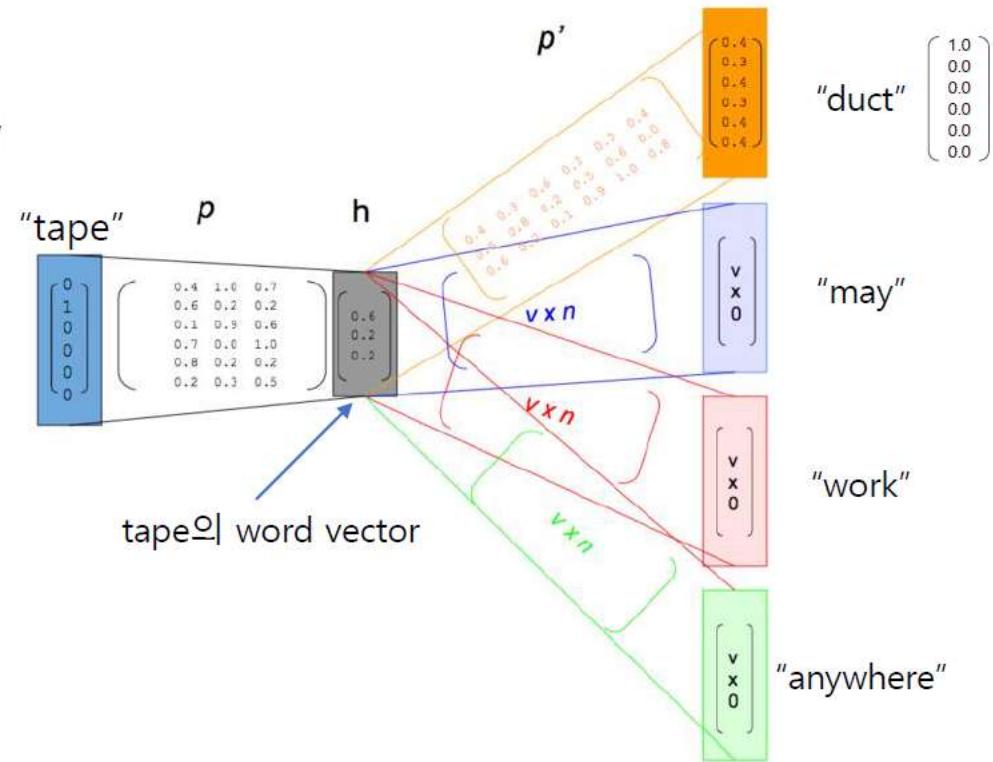
## Word2Vec

### Skip-gram

- 주변부의 단어를 예측하는 방식으로 학습
- 단어에 대한 dense vector를 얻을 수 있음

"Duct tape may works anywhere"

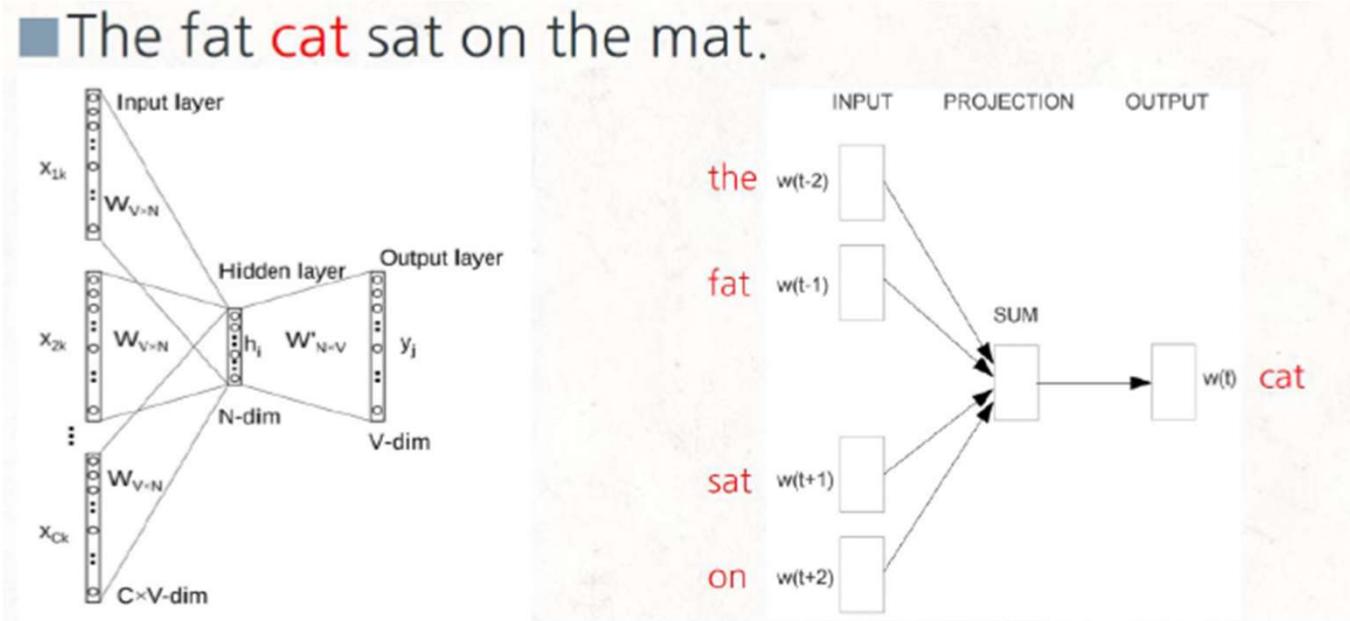
Word	One-hot-vector
"duct"	[1, 0, 0, 0, 0]
"tape"	[0, 1, 0, 0, 0]
"may"	[0, 0, 1, 0, 0]
"work"	[0, 0, 0, 1, 0]
"anywhere"	[0, 0, 0, 0, 1]



## Word2Vec

### CBOW

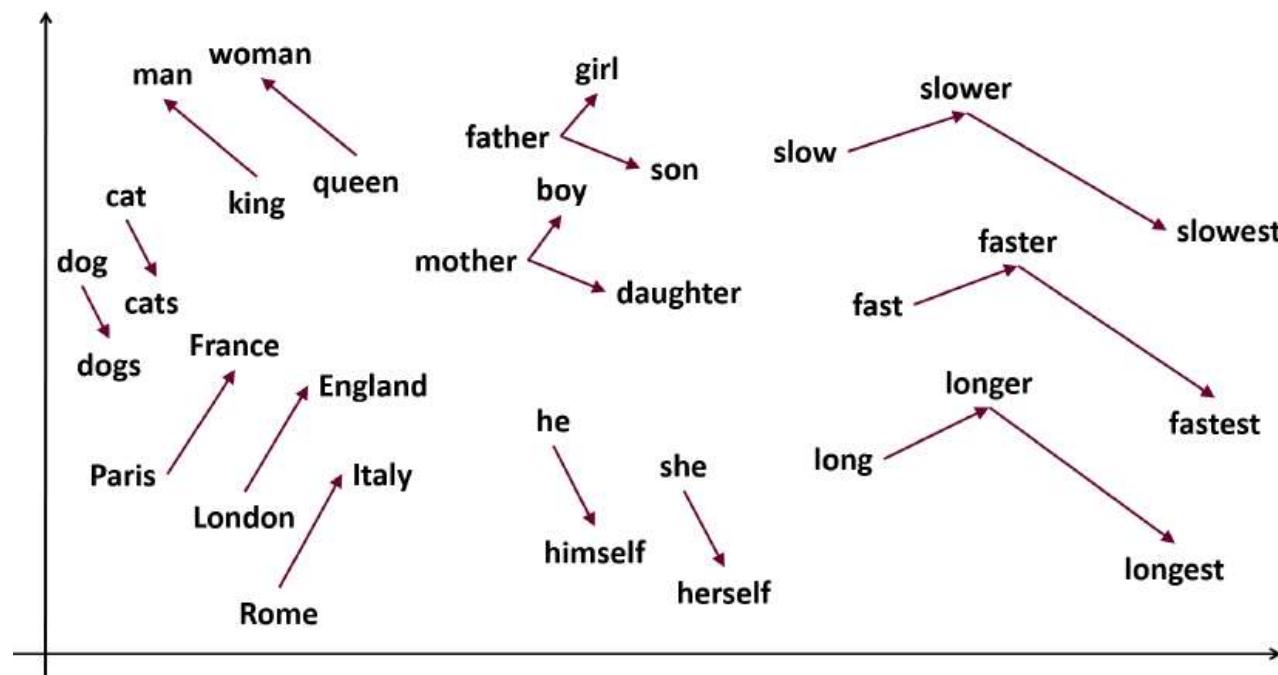
- Continuous Bag-Of-Words
- 문맥으로부터 단어를 예측하는 모델



## Word2Vec

단어의 의미가 벡터로 표현됨으로써 벡터 연산이 가능

$$\vec{\omega}_{king} - \vec{\omega}_{king} + \vec{\omega}_{woman} \approx \vec{\omega}_{queen}$$



## Word2Vec

<https://blog.naver.com/saltluxmarketing/221607368769>

### 기계가 이해하는 한국어의 의미

- [할아버지 - 할머니 + 농구] = [배구]
- [할아버지 - 할머니 + 노트북] = [태블릿]
- [할아버지 - 할머니 + 선풍기] = [청소기]
- [할아버지 - 할머니 + 바다] = [숲]
- [할아버지 - 할머니 + 연필] = [볼펜]
- [할아버지 - 할머니 + 파도] = [안개]
- [할아버지 - 할머니 + 물] = [기름]
- [할아버지 - 할머니 + 버스] = [택시]
- [할아버지 - 할머니 + 겨울] = [여름]
- [할아버지 - 할머니 + 신] = [환상]
- [할아버지 - 할머니 + 커피] = [와인]
- [할아버지 - 할머니 + 밥] = [키스]
- [할아버지 - 할머니 + 사탕] = [과자]
- [할아버지 - 할머니 + 소고기] = [닭고기]
- [할아버지 - 할머니 + 치킨] = [피자]
- [할아버지 - 할머니 + 손] = [주먹]
- [할아버지 - 할머니 + 초록색] = [노란색]
- [할아버지 - 할머니 + 기업] = [투자]
- [할아버지 - 할머니 + 사랑] = [행복]
- [할아버지 - 할머니 + 컴퓨터] = [개발자]
- [할아버지 - 할머니 + 시간] = [매일]
- [할아버지 - 할머니 + 공부] = [수업]
- [할아버지 - 할머니 + 인생] = [추억]

- [건물 - 콘크리트 + 사람] = [냄새]
- [건물 - 콘크리트 + 컴퓨터] = [인터넷페이스]
- [건물 - 콘크리트 + 사랑] = [언제나]
- [건물 - 콘크리트 + 바다] = [모래]
- [건물 - 콘크리트 + 물] = [녹]
- [건물 - 콘크리트 + 시간] = [속도]
- [건물 - 콘크리트 + 공부] = [적응]
- [건물 - 콘크리트 + 인생] = [미소]
- [건물 - 콘크리트 + 손] = [감]
- [건물 - 콘크리트 + 지식] = [구현]
- [건물 - 콘크리트 + 우정] = [목소리]
- [건물 - 콘크리트 + 세계] = [사이클]
- [건물 - 콘크리트 + 분노] = [상처]
- [건물 - 콘크리트 + 힙합] = [리듬]
- [건물 - 콘크리트 + 배고픔] = [방전]
- [건물 - 콘크리트 + 태양] = [빛]
- [건물 - 콘크리트 + 쾌락] = [공명]
- [건물 - 콘크리트 + 데이터] = [프로토콜]
- [건물 - 콘크리트 + 구름] = [열음]
- [건물 - 콘크리트 + 초록색] = [황색]
- [건물 - 콘크리트 + 자유] = [도파민]
- [여름 - 더위 + 겨울] = [마름]
- [여름 - 더위 + 인간] = [욕구]
- [여름 - 더위 + 바다] = [플랑크톤]
- [여름 - 더위 + 재미] = [자질]
- [선풍기 - 바람 + 눈] = [눈물]
- [사람 - 지능 + 컴퓨터] = [소프트웨어]
- [인생 - 사람 + 컴퓨터] = [관리자]
- [그림 - 연필 + 영화] = [스타]
- [오케스트라 - 바이올린 + 인간] = [육체]
- [손 - 박수 + 발] = [달리기]
- [삼겹살 - 소주 + 맥주] = [햄]

## Word embedding의 방법론에 따른 특징

### Sparse representation

- one-hot encoding
- n 개의 단어에 대한 n 차원의 벡터
- 단어가 커질 수록 무한대 차원의 벡터가 생성
- 주로 신경망의 입력단에 사용 신경망이 임베딩 과정을 대체 . e.g. tf.layers.
- 의미 유추 불가능
- 차원의 저주 (curse of dimensionality): 차원
- 이 무한대로 커지면 정보 추출이 어려워짐
- One hot vector 의 차원 축소를 통해 특징을 분류하고자 하는 접근도 있음

### Dense representation

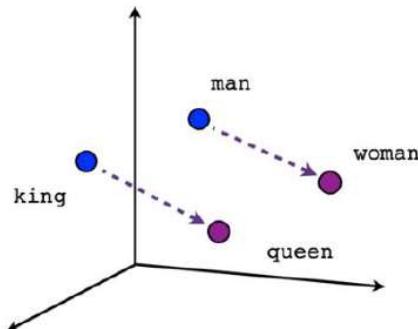
- Word embedding
- 한정된 차원으로 표현 가능
- 의미 관계 유추 가능
- 비지도 학습으로 단어의 의미 학습 가능

## Word2Vec

**gensim**

## Word2Vec

### Word2Vec의 한계



- 단어가 가지는 의미 자체를 다차원 공간에 벡터화 하는 것
- 중심 단어의 주변 단어들을 이용해 중심단어를 추론하는 방식으로 학습



#### 장점

- 단어 간의 유사도 측정에 용이
- 단어 간의 관계 파악에 용이
- 벡터 연산을 통한 추론이 가능 (e.g. 한국 - 서울 + 도쿄 = ?)

#### 단점

- 단어의 subword information 무시 (e.g. 서울 vs 서울시 vs 고양시)
- Out of vocabulary (OOV) 에서 적용 불가능

## FastText

### FastText

- 한국어는 다양한 용언 형태를 가짐
- Word2Vec의 경우 다양한 용언 표현들이 서로 독립된 vocab으로 관리

#### 동사원형: 모르다

모르네	모르기까지	모르겠으나	몰라야	몰랐다면	몰랐겠으나
모르데	모르기를	모르겠으면	몰라요	몰랐다면	몰랐겠으면
모르지	모르기는	모르겠으면서	몰라라	몰랐을	몰랐겠으면서
모르더라	모르기도	모르겠거나	몰랐다	몰랐을까	몰랐겠거나
모르리라	모르기만	모르겠거든	몰랐네	몰랐을지	몰랐겠거든
모르는구나	모르는	모르겠는데	몰랐지	몰랐을지도	몰랐겠는데
모르잖아	모르던	모르겠지만	몰랐더라	몰랐어	몰랐겠지만
모르려나	모른	모르겠더라도	몰랐으리라	몰랐어도	몰랐겠더라도
모르니	모른다	모르겠다가도	몰랐구나	몰랐어야	몰랐겠다가도
모르고	모른다면	모르겠던	몰랐잖아	몰랐어요	몰랐겠던
모르나	모른다면	모르겠다면	몰랐으려나	몰랐더라면	몰랐겠다면
모르면	모른답시고	모르겠다만	몰랐으니	몰랐더라도	몰랐겠다만
모르면서	모르겠다	모를까	몰랐거나	몰랐겠다	몰랐겠어
모르거든	모르겠네	모를지	몰랐거든	몰랐겠네	몰랐겠어도
모르는데	모르겠지	모를지도	몰랐는데	몰랐겠지	몰랐겠어서
모르지만	모르겠더라	모를수록	몰랐지만	몰랐겠더라	몰랐겠어야
모르더라도	모르겠구나	몰라	몰랐더라도	몰랐겠구나	몰랐겠어요
모르다가도	모르겠니	몰라도	몰랐다가도	몰랐겠니	몰랐겠더라면
모르기조차	모르겠고	몰라서	몰랐던	몰랐겠고	몰랐겠더라도

## FastText

**FastText** <https://research.fb.com/fasttext/>

- Facebook research 에서 공개한 open source library
- C++11

### 학습(Training)

- 기존의 word2vec과 유사하나, 단어를 n-gram으로 나누어 학습을 수행
- n-gram 의 범위가 2 ~ 5 일 때, 단어를 다음과 같이 분리하여 학습함
- “assumption” = {as, ss, su, ......., ass, ssu, sum, ump, mpt, ..., ption, assumption}
- 이 때, n-gram 으로 나눠진 단어는 사전에 들어가지 않으며, 별도의 n-gram vector 를 형성함

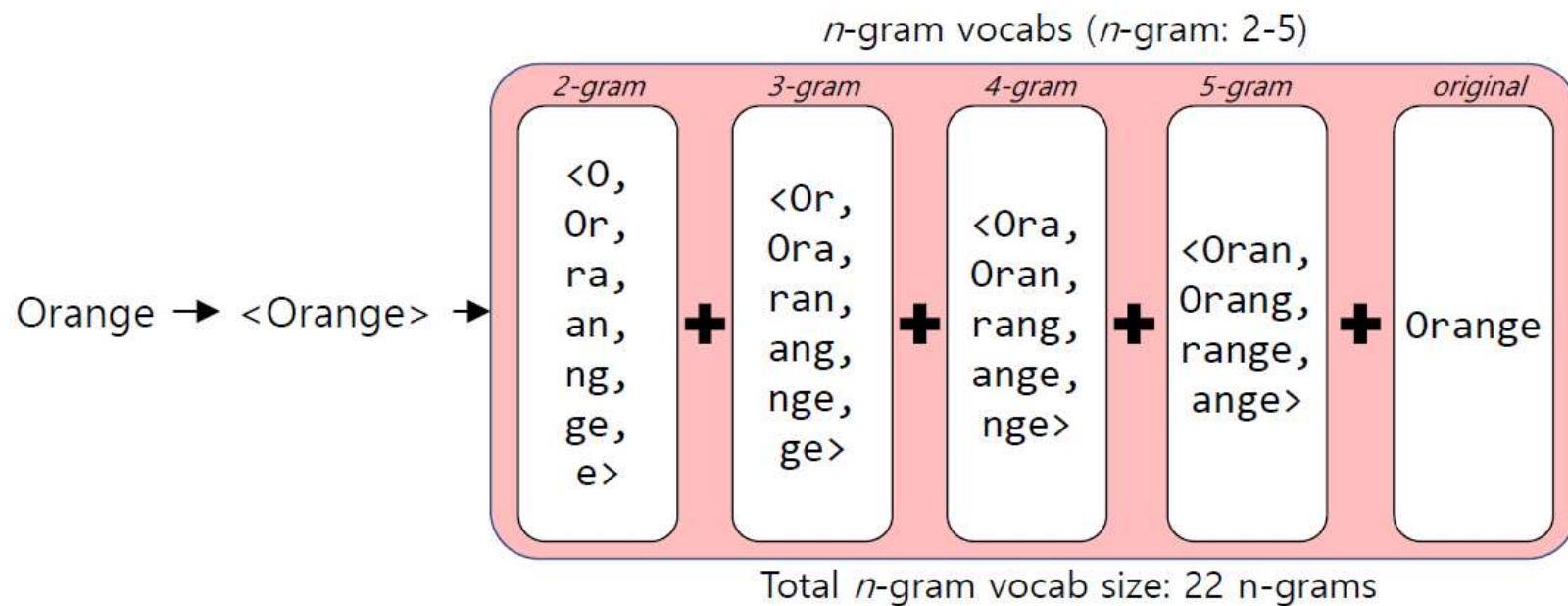
### 테스트(Testing)

- 입력 단어가 vocabulary 에 있을 경우, word2vec 과 마찬가지로 해당 단어의 word vector 를 return 함
- 만약 OOV 일 경우, 입력 단어의 n gram vector 들의 합산을 return 함

## FastText

FastText <https://research.fb.com/fasttext/>

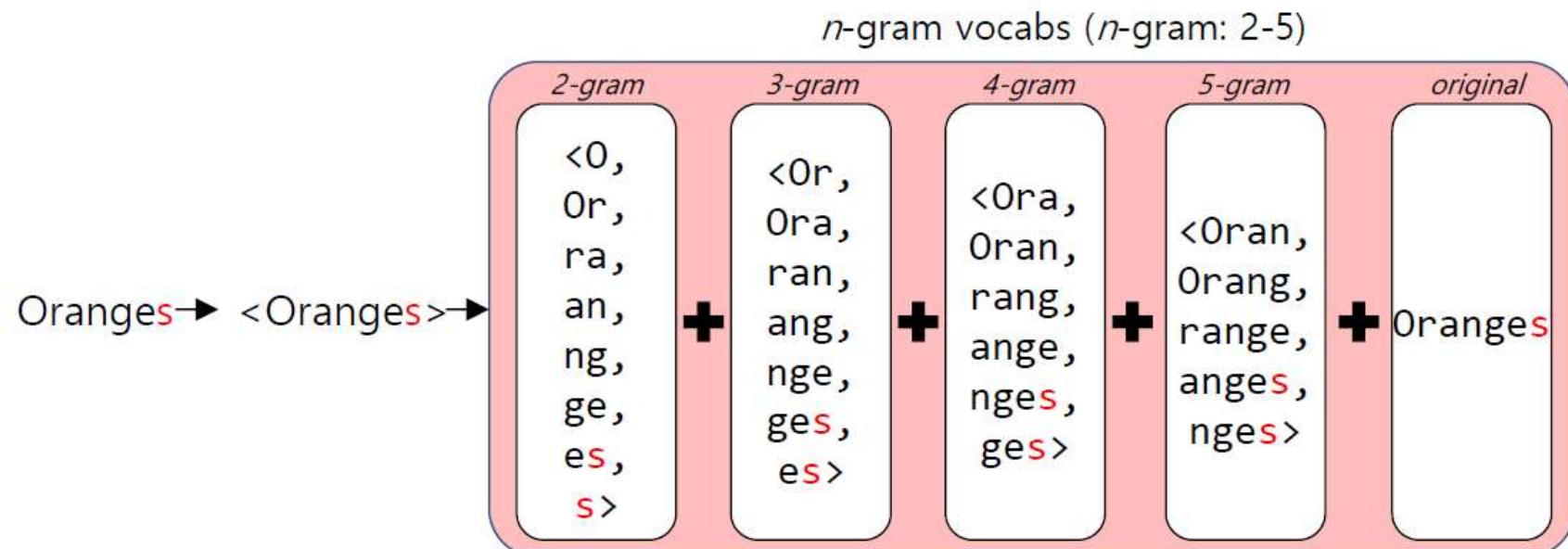
- 단어를 n-gram으로 분리한 후, 모든 n-gram vector를 합산한 후, 평균을 통해 단어 벡터를 획득



## FastText

FastText <https://research.fb.com/fasttext/>

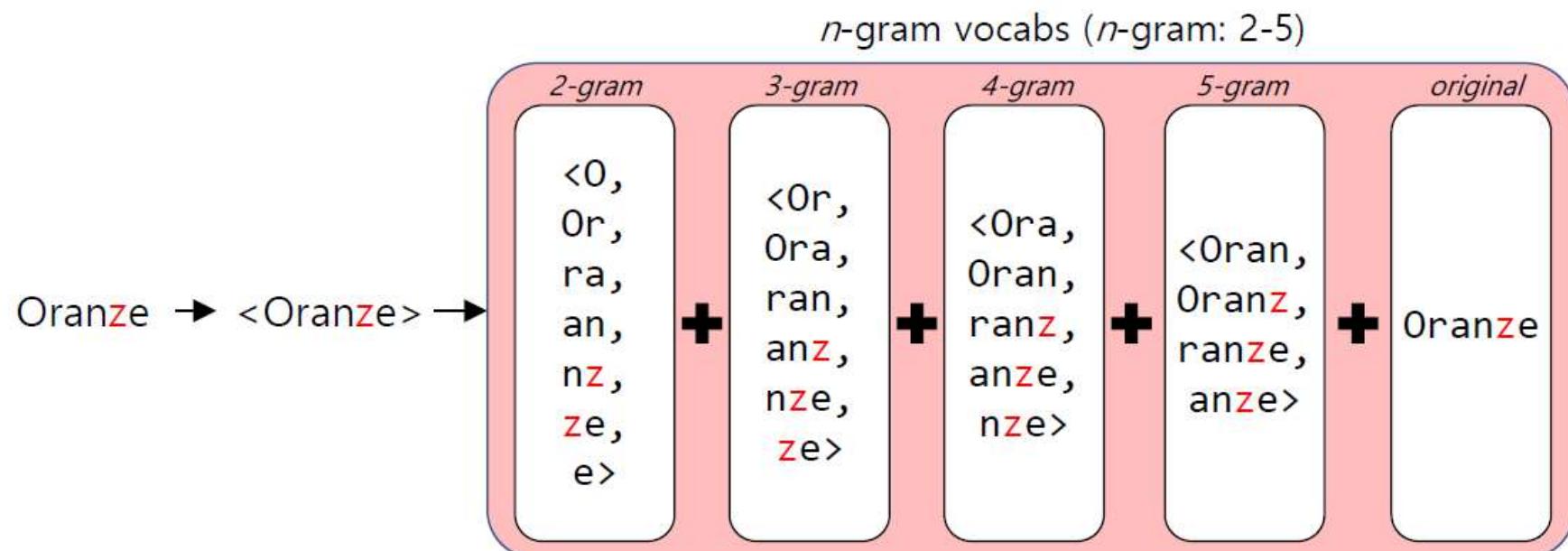
- 단어를 n-gram으로 분리한 후, 모든 n-gram vector를 합산한 후, 평균을 통해 단어 벡터를 획득



## FastText

FastText <https://research.fb.com/fasttext/>

- 단어를 n-gram으로 분리한 후, 모든 n-gram vector를 합산한 후, 평균을 통해 단어 벡터를 획득



## FastText

### FastText와 Word2Vec의 비교

- 오탈자, OOV 등장 회수가 적은 학습 단어에 대해서  
강세([https://link.springer.com/chapter/10.1007/978-3-030-12385-7\\_3](https://link.springer.com/chapter/10.1007/978-3-030-12385-7_3))
- Document frequency (DF) 가 낮을 경우 , word2vec 은 유의미한 단어를 찾을 수 없음
- Fasttext 의 경우 , subword information 이 유사한 단어를 찾았음

Input word	DF	Model	Most similar (1)	Most similar (2)	Most similar (3)	Most similar (4)	Most similar (5)
파일	10146	Word2vec	프로토콜	애플리케이션	url	디렉터리	포맷
		Fasttext	파일	확장자	음악파일	포멧	디렉터리
원인	11589	Word2vec	부작용	현상	요인	증상	질병
		Fasttext	주원인	초래	요인	직접	주요인
태생지	8	Word2vec	부타	구티에레즈	보아뱀	올림피코	집사장
		Fasttext	생지	탄생지	출생지	발생지	무생지
미스코리아	11	Word2vec	치평요람	神檀實記	컬투	방학기	김현승
		Fasttext	믹스코리아	라이코스코리아	악스코리아	보이스코리아	포브스코리아

## FastText

### FastText와 Word2Vec의 비교

- 오탈자, OOV 등장 회수가 적은 학습 단어에 대해서 강세

Input word	FastText Model	Most similar words in range of top 3		
페널티 (Penalty) OOV word	Baseline	리날디 (Rinaldi)	페레티 (Ferretti)	마세티 (Machete)
	Jamo-advanced	페널티골 (Penalty goal)	페널티 (Penalty)	드록바 (Drogba)
나프탈렌 (Naphthalene) OOV word	Baseline	야렌 (Yaren)	콜루바라 (Kolubara)	몽클로아아라바카 (Moncloa-Aravaca)
	Jamo-advanced	나프탈렌 (Naphthalene)	테레프탈산 (Terephthalic acid)	아디프산 (Adipic acid)
스테이크 (Steak) OOV word	Baseline	스탠히프 (Stanhope)	스탠너드 (Stannard)	화이트스네이크 (White Snake)
	Jamo-advanced	롱테이크 (Long take)	비프스테이크 (Beefsteak)	스테이크 (Steak)

## FastText

### Word embedding 방식의 한계점

- Word2Vec이나 FastText와 같은 Word Embedding 방식은 동형어, 다의어 등에 대해선 embedding 성능이 좋지 못하는 단점이 있음
- 주변 단어를 통해 학습이 이루어지기 때문에, '문맥'을 고려할 수 없음

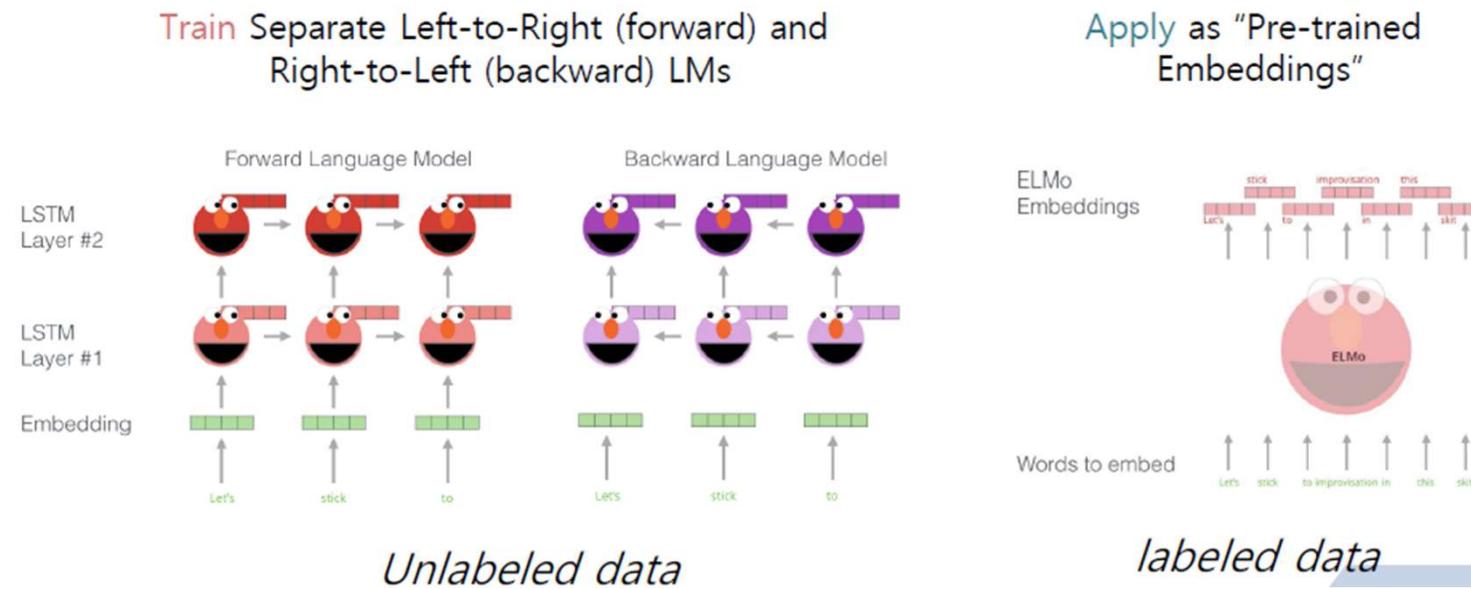
#### account

1. I don't have a bank **account**. (계좌)
2. Put it on my **account** please. (신용 거래, 외상 장부)
3. She gave the police a full **account** of the incident. (설명)
4. In English law a person is **accounted** innocent until they are proved guilty.(간주하다)
5. In 1992 Smith set up in business on his own **account**. (자신의)

## ELMo

### ELMo

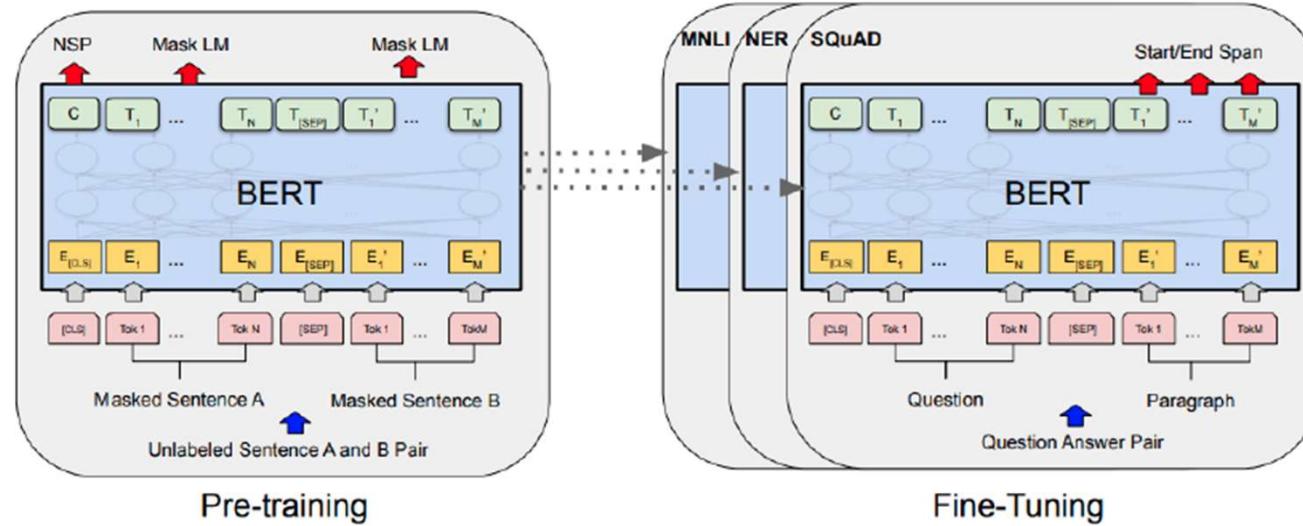
"Instead of using fixed embedding for each word, ELMo looks at the entire sentence before assigning each word in it an embedding."



## BERT

### BERT

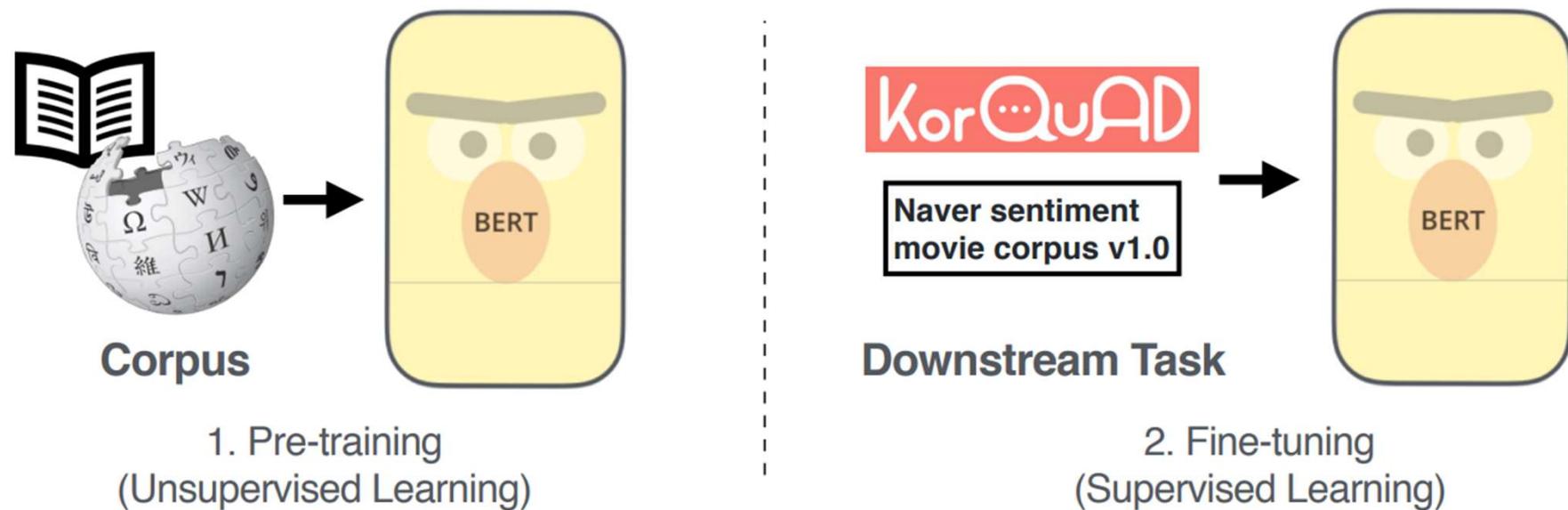
BERT pretrains both sentence and contextual word representations, using masked LM and next sentence prediction. BERT-large has 340M parameters, 24 layers!



## Language model

### PLM

- Pre-training then fine-tuning
- 대량의 코퍼스를 Pre-training한 다음, 각 Task에 따라서 fine-tuning을 진행하는 모델

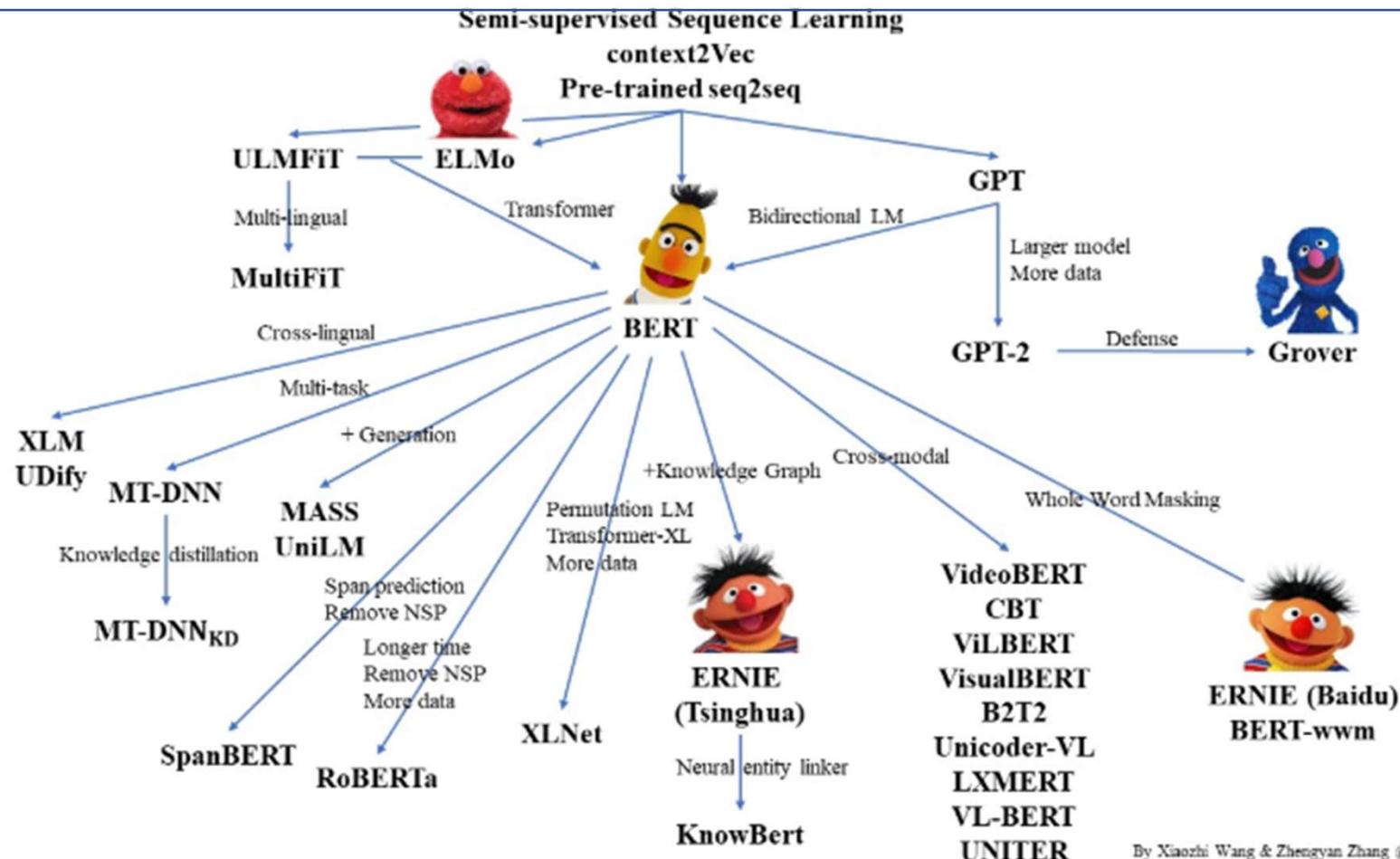


## Language model

### PLM

<https://github.com/thunlp/PLMpapers>

- BERT를 시작으로 계속해서 다양해짐



## Language model

### PLM

- BERT를 시작으로 계속해서 다양해짐

