

# Text Preprocessing

Natural Language Processing & AI Lab.,  
Korea University

강의자: 박찬준



**KOREA**  
UNIVERSITY



Natural Language  
Processing  
& Artificial Intelligence



# NLP Basics

## Overview



## 자연언어란?

# 자연언어 (Natural language)

- 자연언어란?
  - 인간 고유의 언어
  - 정보전달의 수단
  - 인공지능에 대응되는 개념
  - 특정 집단에서 사용되는 모국어의 집합
    - 한국어, 영어, 불어, 독일어, 스페인어, 일본어, 중국어 등
- 인공언어란?
  - 특정 목적을 위해 인위적으로 만든 언어
  - 자연언어에 비해 엄격한 구문을 가짐
  - 형식언어, 에스페란토어, 프로그래밍 언어



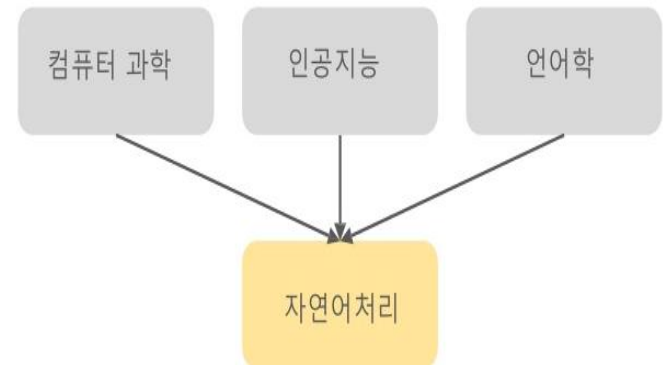
# 자연언어처리란?

## 자연언어처리

- 자연 언어 처리/ 자연어 처리
  - 컴퓨터를 통하여 인간의 언어를 처리하고 이용하려는 학문 분야
  - 인간의 언어를 이해하고, 이를 바탕으로 각종 정보처리에 적용함으로써 보다 빠르고 편리한 정보 획득
- 자연언어처리 응용 분야
  - 인간의 언어가 사용되는 실세계의 모든 영역
  - 정보검색, 질의응답 시스템
  - 기계번역, 자동통역
  - 문서작성, 문서요약, 문서분류, 철자오류 검색 및 수정, 문법 오류 검사

## 자연어처리 (NLP)

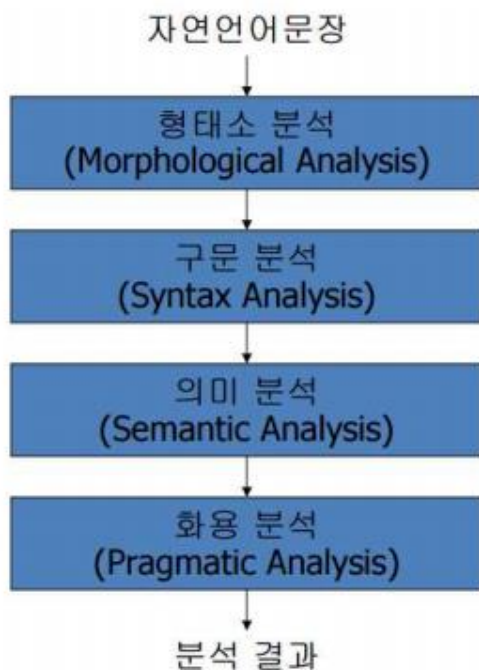
컴퓨터과학, 인공지능과 언어학이 합쳐진 분야





# 자연언어처리의 단계

## 자연언어처리의 단계



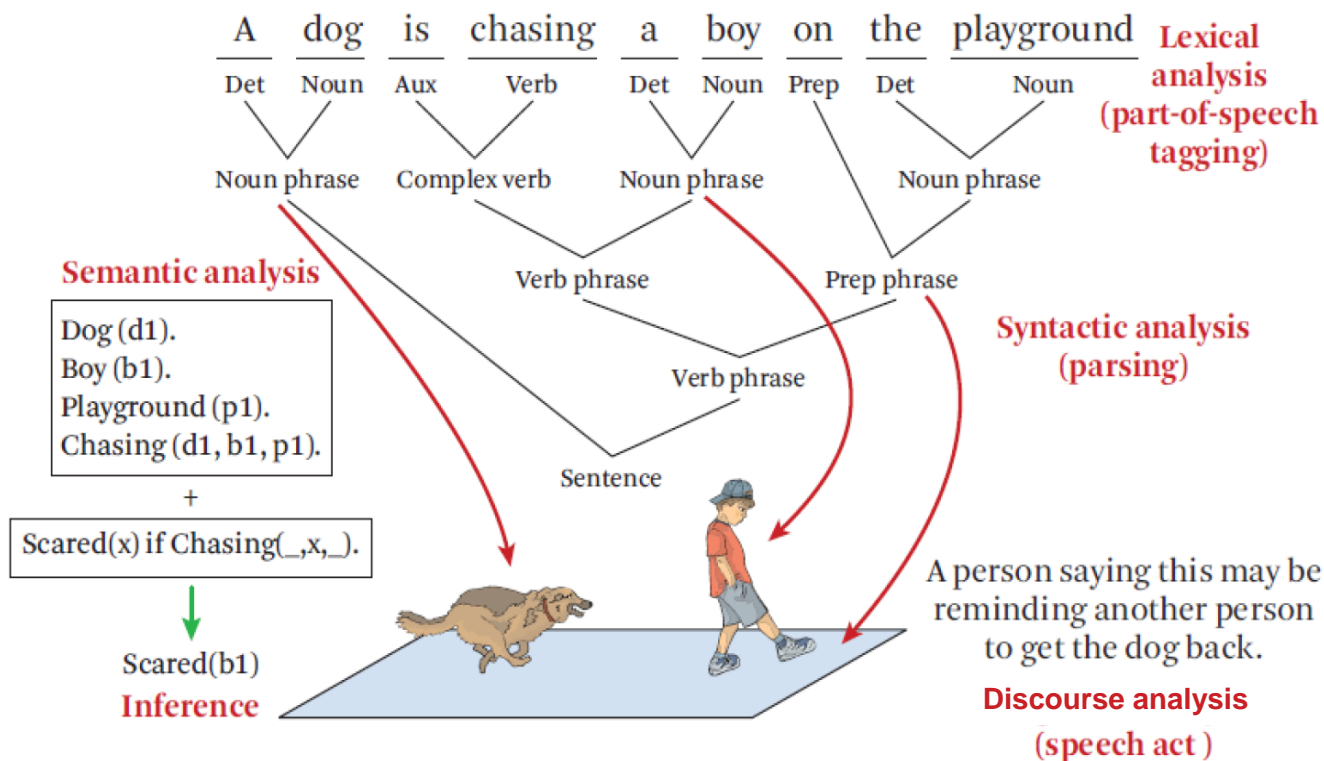
### General steps in NLP

- ▶ Lexical Analysis
    - Word(lexicon), Morphology, word segmentation
  - ▶ Syntax Analysis
    - Sentence structure, phrase, grammar,...
  - ▶ Semantic Analysis
    - Meaning, execute commands
  - ▶ Discourse Analysis
    - Meaning of a text
    - Relationship between sentences
- Ex) I disagree and so does John. (does-> disagree)



# 자연언어처리의 단계

## □ General steps in NLP





# Lexical Analysis

- ▶ Lexical analysis의 필요성
  - 입력된 문장을 잘 분할해서 효율성을 높이기 위함
- ▶ Lexical analysis에서의 주요 요인
  - Sentence splitting : 마침표(.), 느낌표(!), 물음표(?) 등을 기준으로 분리
  - Tokenizing : 문서나 문장을 분석하기 좋도록 나눔 (띄어쓰기 또는 형태소 단위로...)
  - Morphological : 토큰들을 좀 더 일반적인 형태로 분석해 단어 수를 줄여 분석의 효율성을 높임 (가장 작은 의미단위로 토큰화 함)
  - Stemming ('cars', 'car' => 'car'), lemmatization (단어를 원형으로), ...



# Syntax Analysis

## ▶ Syntax analysis의 필요성

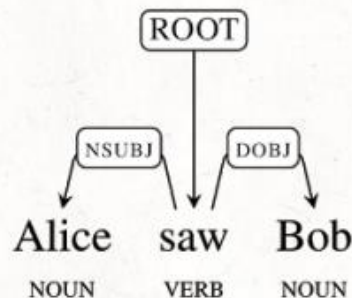
- 언어는 문장구성에 규칙이 필요함
- 문장 구성을 위한 규칙 / 문법을 구성

## ▶ Syntax analysis (구문분석)

- 각각의 어절단위로 구분, 해당 tag 부여 (parsing tree이용)
- 의존성 구문분석 트리로 표현 (dependency parsing)
- Saw(root)를 기준으로 Alice(subject), Bob(object) 관계를 나타냄
- 이 문법적 관계들은 아주 직접적인 연관이 있음

POS	UMLS	Penn Tree Bank Tag	Example
Noun	noun	NN, NNS, NNP, NNPS	table
Adjective	adj	JJ, JJR, JJS	blue
Adverb	adv	RB, RBR, RBS, WRB, RP	quickly
Pronoun	pron	PRP, PRP\$, WP, WP\$	she
Verb	verb	VB, VBD, VBG, VBN, VBP, VBZ	wrote
Determiner	det	DT, PDT, WDT	the
Preposition	prep	IN	with
Conjunction	conj	CC	and
Auxiliary	aux	VB, VBD, VBG, VBN, VBP, VBZ	does
Modal	modal	MD	could
Complement	compl	IN	that

Part-of-speech tagging



Dependency parsing





# Semantic Analysis

## ▶ Semantic analysis의 필요성

- 규칙에 따라 문장은 만들었는데 문장이 의미적으로 올바른 것인지 알아야 함
- 사람이 사과를 먹는다. (O)
- 사람이 비행기를 먹는다. (X)

## ▶ Semantic analysis란?

- Syntactic + meaning
- Two levels (lexical semantics)
- Representing meaning of words
- Word sense disambiguation (word bank)
  
- Compositional semantics
- How words combined to form a larger meaning

“6시에 KBS에서 뭐 하니?”



Semantic analysis

Question focus : 프로그램

Channel : KBS

Begin\_time : 18:00



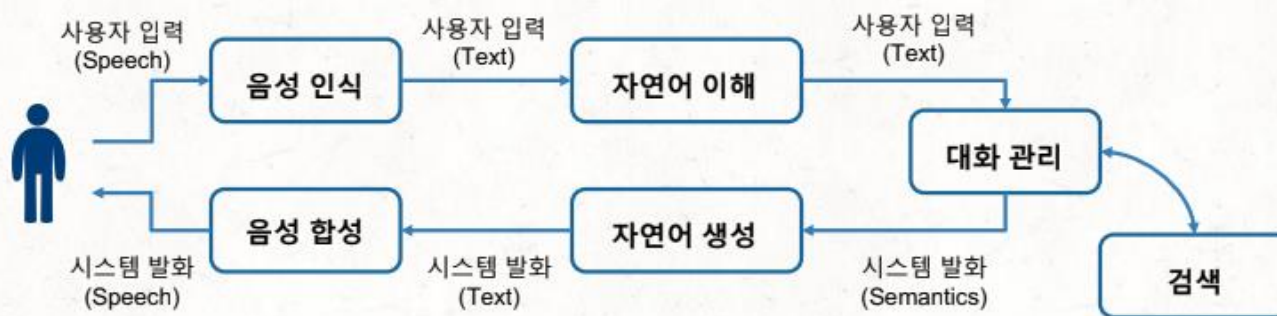
# Discourse Analysis

## ▶ Discourse analysis의 필요성

- 대화의 흐름을 파악하여 발화자의 의도에 맞도록 응답해야 함

## ▶ Discourse analysis란?

- 대화의 흐름상 어떤 의미를 가지는지를 찾음
- 문맥구조 분석 (문장들의 연관 관계)
- 의도분석 (전후관계를 통한 실제 의도)
- 대화 분석 (대표적인 담화분석)

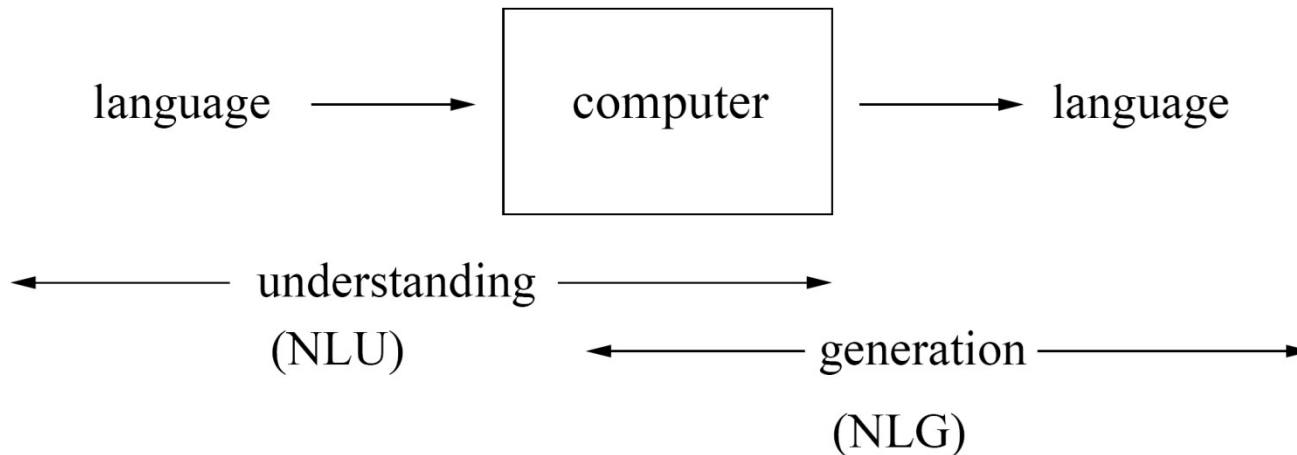


대화 시스템 프로세스



## 자연언어처리의 단계

- ❑ One simple (but practical) answer
  - ❑ Computer using natural language as input and/or output



- ❑ Or components enabling such a computer



# 자연언어처리를 위한 언어학

Overview



## Phonetics (음성학) & Phonology (음운론)

- The study of language sounds, how they are physically formed and systems of discrete sounds
  - disconnect => dis-k&-'nekt
  - “It is easy to recognize speech.”
  - “It is easy to wreck a nice beach.”
- 음성인식
  - Signal to symbol

음소

더 이상 작게 나눌 수 없는  
음운론상의 최소 단위



# 형태론

## 어절, 단어, 형태소

### - 어절

양쪽에 공백을 갖는 띄어쓰기 단위의 문자열

### - 단어 / 형태소

단일 품사를 갖는 단위 / 사전에 등록되어 있는 색인어의 집합

예: 나는 책을 읽었다.

파릇파릇한 싹이 나는 계절이다.

하늘을 나는 새를 보라.

I tried to go to school.

He tries to pass the exam.

나 + 는  
날다 + 는  
나다 + 는

### 형태소

의미를 가지는 언어 단위 중 가장  
작은 단위

의미 혹은 문법적 기능의  
최소단위



# 형태소 분석

## 형태소 분석

- ▶ 형태소 분석이란?
  - ▶ 형태소를 비롯하여, 어근, 접두사/접미사, 품사(POS, Part-of-Speech) 등 다양한 언어적 속성의 구조를 파악하는 것
- ▶ Mecab, Twitter, Komoran, Hannanum, KoNLPy 등

```
pprint(kkma.pos(a))
```

문장을 입력하세요: 세종대왕님은 글을 만드셨습니다.

```
[('세종', 'NNG'),  
 ('대왕', 'NNG'),  
 ('님', 'XSN'),  
 ('은', 'JX'),  
 ('글', 'NNG'),  
 ('을', 'JKO'),  
 ('만들', 'VV'),  
 ('시', 'EPH'),  
 ('었', 'EPT'),  
 ('습니다', 'EFN'),  
 ('.', 'SF')]
```



## 형태소 분석

### 형태소 분석 (Morphological Analysis)

- 입력된 문자열을 분석하여  
형태소(morpheme)라는 최소 의미 단위로 분리
- 사전 정보와 형태소 결합 정보 이용
- 정규 문법(Regular Grammar)으로 분석 가능
- 언어에 따라 난이도가 다름
  - 영어, 불어 : 쉬움
  - 한국어, 일본어, 아랍어, 터키어 : 어려움





# 형태소 분석의 어려움

## 형태론적 다양성

- 첨가어
  - 한국어, 일본어, 터키어 등
  - 다수의 형태소가 결합하여 어절 형성
  - 터키어는 평균 7개의 형태소가 결합
- 굴절어
  - 라틴어 (영어, 불어 등은 첨가어와 굴절어의 특징이 모두 있음)
  - 어간이 변함 (영어의 예 : run, ran, run)
- 스와힐리어
  - 수(number)를 위한 형태소가 문두에 붙음
  - (예) 사람 : m+tu (단수), wa+tu (복수)
  - 나무 : m+ti (단수), mi+ti (복수)
- 아랍어
  - 자음이 어간이고 모음이 시제, 수 등을 표현
  - (예) ktb(쓰다)            kAtAb(능동) KUtlb(수동)
  - kttb(쓰게하다)    kAttAb(능동)KUttlb(수동)

## 통사적 다양성

- Postfix 언어 (Head-Final Languages)
  - 동사가 문장의 뒤에 위치
  - 한국어, 일본어 등
- Infix 언어
  - 동사가 문장의 중간에 위치
  - 영어, 불어 등
- Prefix 언어
  - 동사가 문장의 처음에 위치
  - 아일랜드어



## 형태소 분석의 어려움

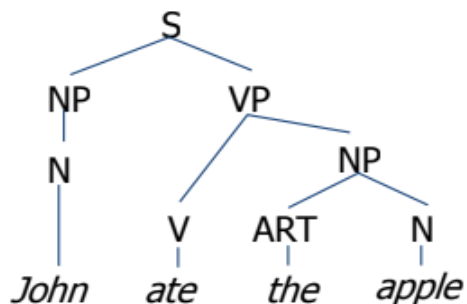
### 형태소 분석의 난점

- 중의성 (ambiguity)
  - "감기는"의 분석 결과
    - 감기(명사:cold) + 는(조사)
    - 감(동사 어간) + 기(명사화 어미) + 는 (조사)
    - 감(동사 어간) + 기는(어미)
- 접두사, 접미사 처리
- 고유명사, 사전에 등록되지 않은 단어 처리
  - 한국어, 독일어처럼 복합명사 내의 명사를 띄우지 않거나, 일본어처럼 띄어쓰기가 없으면 더욱 어려워짐
- 한국어 형태소 결합의 예 ("친구에게서였었다라고")
  - 친구(명사) + 에게(조사) + 서(조사) + 이(서술격조사) +
  - 였(과거시제어미) + 었(회상어미) + 다(어말어미) +
  - 라고(인용격조사)



## 문법, 구문 분석

- 문법 (Grammar) :
  - 문장의 구조적 성질을 규칙으로 표현한 것
- 구문 분석기 (Parser) :
  - 문법을 이용하여 문장의 구조를 찾아내는 process
  - 문장의 구문 구조는 Tree 형태로 표현할 수 있다. 즉, 몇 개의 형태소들이 모여서 구문 요소(구: phrase)를 이루고, 그 구문 요소들간의 결합구조를 Tree형태로써 구문 구조를 이루게 된다.





## 문법 (Grammars)

- Grammar : a set of rewrite rules

(ex)  $S \rightarrow NP VP$   
 $NP \rightarrow ART N$   
 $NP \rightarrow N$   
 $VP \rightarrow V NP$

- Context Free Grammar (CFG) :
  - 각 rule의 LHS(Left-Hand side)가 하나의 symbol로 이루어진 문법 규칙
- Grammar Rule 을 이용해서 문장(sentence)을 생성할 수도 있고(sentence generation), 분석할 수도 있다(sentence parsing).



## Sentence Generation

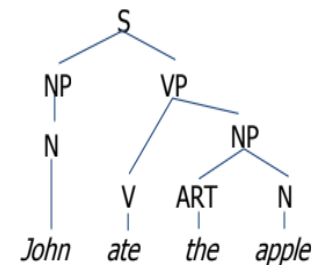
(ex) By rewrite rule

$S \rightarrow NP \ VP$   
 $\rightarrow N \ VP$   
 $\rightarrow John \ VP$   
 $\rightarrow John \ V \ NP$   
 $\rightarrow John \ ate \ ART \ N$   
 $\rightarrow John \ ate \ the \ N$   
 $\rightarrow John \ ate \ the \ apple.$

## Bottom-up Parsing

(ex) *John ate the apple.*

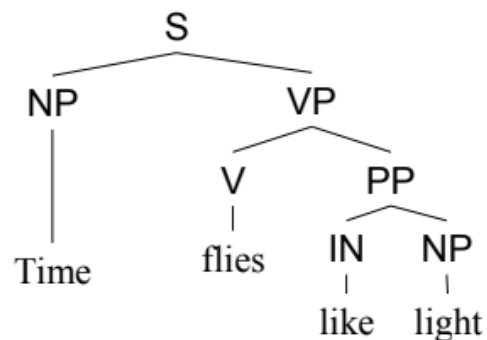
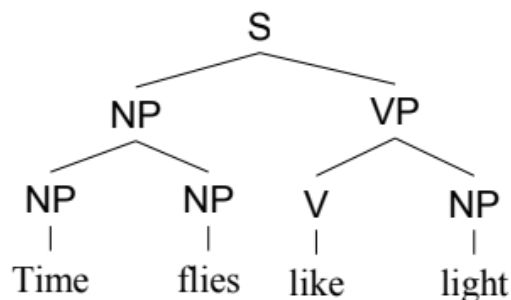
$\rightarrow N \ V \ ART \ N$   
 $\rightarrow NP \ V \ ART \ N$   
 $\rightarrow NP \ V \ NP$   
 $\rightarrow NP \ VP$   
 $\rightarrow S$





## 구문 분석의 어려움

### 구문 분석 - Structural Ambiguities



- Structural Ambiguities
  - Time flies like light. ⇒ 2가지 이상의 구조로 분석됨
    - flies (noun or verb), like(verb or preposition)
  - A man see a woman with a telescope on the hill. ⇒ 5가지 이상



## 의미 분석 (Semantic Analysis)

- 통사 분석 결과에 해석을 가하여 문장이 가진 의미를 분석
- 형태소가 가진 의미를 표현하는 지식 표현 기법이 요구됨
- 통사적으로 옳으나 의미적으로 틀린 문장이 있을 수 있음
  - 돌이 걸어간다 (cf. 사람이 걸어간다)
  - 바람이 달린다 (cf. 말이 달린다)
- Ambiguity
  - 말이 많다 (horse, speech)

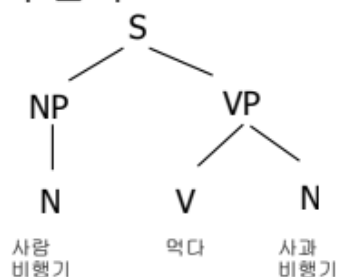


## 의미 분석

### 의미 분석 - cont'd

- 문법적으로는 맞지만 의미적으로 틀린 문장들
  - 사람이 사과를 먹는다. (o)
  - 사람이 비행기를 먹는다. (x)
  - 비행기가 사과를 먹는다. (x)

구문 구조



의미적 제약

[먹다

[ agent : 먹을수 있는 주체  
object : 먹을 수 있는 대상  
....]]





# 화용 분석 (Pragmatic Analysis)

- 문장이 실세계(real world)와 가지는 연관관계 분석
- 실세계 지식과 상식의 표현이 요구됨
- 지시(anaphora), 간접화법(indirect speech act) 등의 분석
  - Anaphora : 대명사의 지시 대상  
The city councilmen refused the women a permit because  
(1) *they* feared violence.  
(2) *they* advocated revolution.
  - Speech Act : 상대방에게 행동을 요구하는 언어 행위  
Can you give me a salt?  
Would you mind opening the window?



# Data Preprocessing

## Overview



## Why use preprocessing?

- 대표적인 예시
  - 아버지가 방에 들어가신다.

아버지가 방에 들어가신다. 아버지가 방에 들어가신다.



두 가지 뜻으로 읽힐 수 있는 것처럼 말이죠



## Why use preprocessing?

- 띄어쓰기나 맞춤법에 따라 다른 의미
    - 중의적 표현이나 반어법 등의 표현으로 의미를 찾기 어려움
- ⇒ 그래서, 텍스트 전처리 필요!
- ⇒ 이를 통해 의미를 찾음



## Data Preprocessing

- Crawling을 통해 얻은 corpus에서 필요에 맞게 전처리가 필요
- 용도에 맞게 텍스트를 사전에 처리하는 작업
  - 주어진 원래 데이터를 그대로 사용하기보다는 원하는 형태로 변형해서 분석하는 경우가 많음
  - 유사한 말
    - 데이터 가공 (Data Manipulation), 데이터 핸들링 (Data Handling), 데이터 클리닝(Data Cleaning) 등...



## Data Preprocessing

- 컴퓨터가 텍스트를 이해할 수 있도록 data preprocessing 방법
  - HTML 태그, 특수문자, 이모티콘
  - 정규표현식
  - 불용어 (Stopword)
  - 어간추출(Stemming)
  - 음소표기법(Lemmatizing)



## Library - KoNLPy

- KoNLPy
  - <https://konlpy-ko.readthedocs.io/ko/v0.4.3/#>
  - 한국어 **자연어처리**를 위한 대표적 python Library
    - Twitter, Komoran, Mecab 등 다양한 형태소 분석기 내장하고 있음



## Library - NLTK

- NLTK(Natural Language Toolkit)
  - <https://www.nltk.org>
  - 영어로 된 텍스트의 자연처리를 위한 대표적인 python Library
    - classification, tokenization, stemming tagging, parsing, and semantic reasoning 등 50개가 넘는 library를 제공하며 쉬운 interfaces 제공





## Library - Gensim

- Gensim
  - <https://radimrehurek.com/gensim/>
  - 주로 Topic modeling, Corpus 및 Word Embedding model을 지원해줌
  - 한국어 및 다양한 언어를 지원해줌



## Tokenization

- Tokenization(토큰화)
  - 주어진 코퍼스(corpus)에서 토큰(token)이라 불리는 단위로 나누는 작업을 토큰화(tokenization)라고 부름
    - 보통 의미있는 단위의 토큰을 정의함
  - 토큰의 기준을 단어(word)로 하는 경우, 단어 토큰화(word tokenization)라고 함
    - 단어(word)는 단어 단위 외에도 단어구, 의미를 갖는 문자열로도 간주되기도 함



## Tokenization

- Tokenization(토큰화)

Input text : 한국어를 처리하는 예시입니다 ㅋㅋㅋ



Output text : " 한국어 ", " 를 ", " 처리", " 하는", " 예시", " 입", "  
니다", "ㅋㅋㅋ"



## Tokenization

- Tokenization(토큰화) 고려할 사항
  - 구두점이나 특수문자를 단순 제외해서는 안됨
  - 줄임말과 단어 내에 띄어쓰기가 있는 경우
    - She's -> "she", "' ", "s"
    - Don't -> "Do", "n't"
    - Data-mining -> "Data-mining"



## Normalization

- Normalization(정규화)
  - 표현 방법이 다른 단어들을 통합시켜서 같은 단어로 만듦
    - HTML 문서로부터 가져온 corpus라면 문서 내에 있는 HTML 태그 제거
    - 뉴스 기사의 경우 게재 시간 제거, 기자 이름 제거



## Cleaning

- Cleaning(정제)
  - 갖고있는 corpus로부터 noisy를 제거
    - 대, 소문자 통합
    - 등장 빈도 적은 단어 제거
    - 길이가 짧은 단어 제거



## Stemming - 어간추출

- Stemming (어간 추출)
  - 대표적 포터 알고리즘의 어간 추출은 이러한 규칙들을 가짐
    - ALIZE → AL
    - ANCE → 제거
    - ICAL → IC

formalize → formal  
allowance → allow  
electricical → electric  
먹었다(ate), 먹을(will eat) → 먹다(eat)



## Lemmatization – 표제어 추출

- Lemmatization (표제어 추출)
  - 품사 정보가 보존된 형태의 기본형으로 변환
  - 표제어 추출에 가장 섬세한 방법은 => 형태학적 파싱
  - 형태소란?
    - 의미를 가진 가장 작은 단위
    - 어간(stem) : 단어의 의미를 담고 있는 단어의 핵심 부분
    - 접사(affix) : 단어에 추가적인 의미를 주는 부분

**Cats → cat(어간) + s(접사)**

**Dies → die**

**Watched → watch**

**Has → have**





## Stopword

- Stopword (불용어)
  - 갖고 있는 데이터에서 유의미한 단어 토큰만을 선별하기 위해서는 큰 의미가 없는 단어 토큰을 제거하는 작업이 필요
    - I, my, me, over, 조사, 접미사 같은 단어들은 문장에서는 자주 등장하지만 실제 의미 분석을 하는데는 거의 기여하는 바가 없음
    - [한국어]조사, 접미사 : 나, 너, 은, 는, 이, 가, 하다, 합니다 등

Family **is not an** important thing. It's everything.



['Family', 'important', 'thing', '.', 'It', "'s", 'everything', '.']



# Regular Expression

- Regular Expression (정규 표현식)
  - 특정 규칙을 정하여 텍스트 데이터를 정리함

모듈 함수	설명
<code>re.compile()</code>	정규표현식을 컴파일하는 함수입니다. 다시 말해, 파이썬에게 전해주는 역할을 합니다. 찾고자 하는 패턴이 빈번한 경우에는 미리 컴파일해놓고 사용하면 속도와 편의성면에서 유리합니다.
<code>re.search()</code>	문자열 전체에 대해서 정규표현식과 매치되는지를 검색합니다.
<code>re.match()</code>	문자열의 처음이 정규표현식과 매치되는지를 검색합니다.
<code>re.split()</code>	정규 표현식을 기준으로 문자열을 분리하여 리스트로 리턴합니다.
<code>re.findall()</code>	문자열에서 정규 표현식과 매치되는 모든 경우의 문자열을 찾아서 리스트로 리턴합니다. 만약, 매치되는 문자열이 없다면 빈 리스트가 리턴됩니다.
<code>re.finditer()</code>	문자열에서 정규 표현식과 매치되는 모든 경우의 문자열에 대한 이터레이터 객체를 리턴합니다.
<code>re.sub()</code>	문자열에서 정규 표현식과 일치하는 부분에 대해서 다른 문자열로 대체합니다.



# Colab

Jupyter notebook in the cloud





## Colab

- Jupyter notebook environment that requires no setup and runs entirely in the cloud
- Supports hardware acceleration (GPU/TPU)
- <https://colab.research.google.com>



# Enable Saving

This notebook is open with private outputs. Outputs will not be saved. You can disable this in [Notebook settings](#).

 basic\_classification.ipynb 

File Edit View Insert Runtime Tools Help

View on GitHub

New Python 2 notebook

New Python 3 notebook

Open notebook... Ctrl+O

Upload notebook...

Rename...

Move to trash

Save a copy in Drive...

Save a copy as a GitHub Gist...

Save a copy in GitHub...

Save Ctrl+S

Save and pin revision Ctrl+M S

Revision history

Download .ipynb

Download .py



Update Drive preview

Print Ctrl+P

CONNECTED EDITING

본인의 구글 드라이브로 복사한 후 해당 파일에 작업하세요. 아니면 브라우저 종료 시 모든 결과물이 사라집니다.

## Train your first neural network: basic classification

[View on TensorFlow.org](#)  [Run in Google Colab](#)  [View source on GitHub](#)

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details, this is a fast-paced overview of a complete TensorFlow program with the details explained as we go.

This guide uses [tf.keras](#), a high-level API to build and train models in TensorFlow.

```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

1.12.0



# Hardware Acceleration

This notebook is open with private outputs. Outputs will not be saved. You can disable this in [Notebook settings](#).

The screenshot shows the Google Colab interface for a notebook titled 'basic\_classification.ipynb'. The 'Runtime' menu is open, displaying various execution options. The 'Change runtime type' option is highlighted with a red dashed box. In the background, the 'Notebook settings' dialog is open, showing the 'Runtime type' set to 'Python 3' and the 'Hardware accelerator' set to 'None'. A dropdown menu for hardware accelerators is open, with 'GPU' selected and highlighted by a red dashed box. Other options visible are 'None' and 'TPU'. The 'Omit code cell outputs' checkbox is checked. The 'SAVE' button is visible in the bottom right of the settings dialog.

basic\_classification.ipynb

File Edit View Insert Runtime Tools Help

Run all Ctrl+F9

Run before Ctrl+F8

Run the focused cell Ctrl+Enter

Run selection Ctrl+Shift+Enter

Run after Ctrl+F10

Interrupt execution Ctrl+M I

Restart runtime... Ctrl+M .

Restart and run all...

Reset all runtimes...

Change runtime type

Manage sessions

Table of contents Code snippets

Copyright 2018 The TensorFlow Authors.

Licensed under the Apache License, Version 2.0 (the "License");

MIT License

Train your first neural network: basic

Import the Fashion MNIST data

Explore the data

Preprocess the data

Build the model

This guide trains a neural network model to classify in the details, this is a fast-paced overview of a complete

This guide uses `tf.keras`, a high-level API to build and t

Run in Google Colab

Notebook settings

Runtime type Python 3

Hardware accelerator None

☒ Omit code cell outputs

GPU

TPU

CANCEL SAVE



## Colab 실습

### ▼ Welcome to Colab!

Colab은 클라우드 환경에서 구동되는 Jupyter notebook (ipython notebook) 개발 환경입니다.

또한, Colab은 GPU 혹은 TPU 하드웨어 가속을 지원합니다. (일반적으로 GPU 가속이 더 좋은 성능을 보입니다)

한 번 시험해볼까요?

```
[ ] print("Hello Colab!")
```

☞ Hello Colab!

그래프를 그리는 matplotlib와 같은 라이브러리를 이용하여 시각화를 할 수도 있습니다. 이 때, 시각화된 그래프가 환경에 바로 표시됩니다.

```
[ ] import matplotlib.pyplot as plt
import numpy as np

x = np.arange(20)
y = [x_i + np.random.randn(1) for x_i in x]
a, b = np.polyfit(x, y, 1)
```



감사합니다.

박찬준

[bcj1210@naver.com](mailto:bcj1210@naver.com)