

개발환경설정

with  python™

OS별 머신러닝 & 딥러닝을 구현할 수 있도록 도와주는 패키지
(웹사이트 주소: <https://www.anaconda.com/products/individual>)

The screenshot shows the 'Anaconda Installers' section of the Anaconda website. It is organized into three columns for Windows, macOS, and Linux. Each column contains links to graphical and command-line installers for Python 3.8.

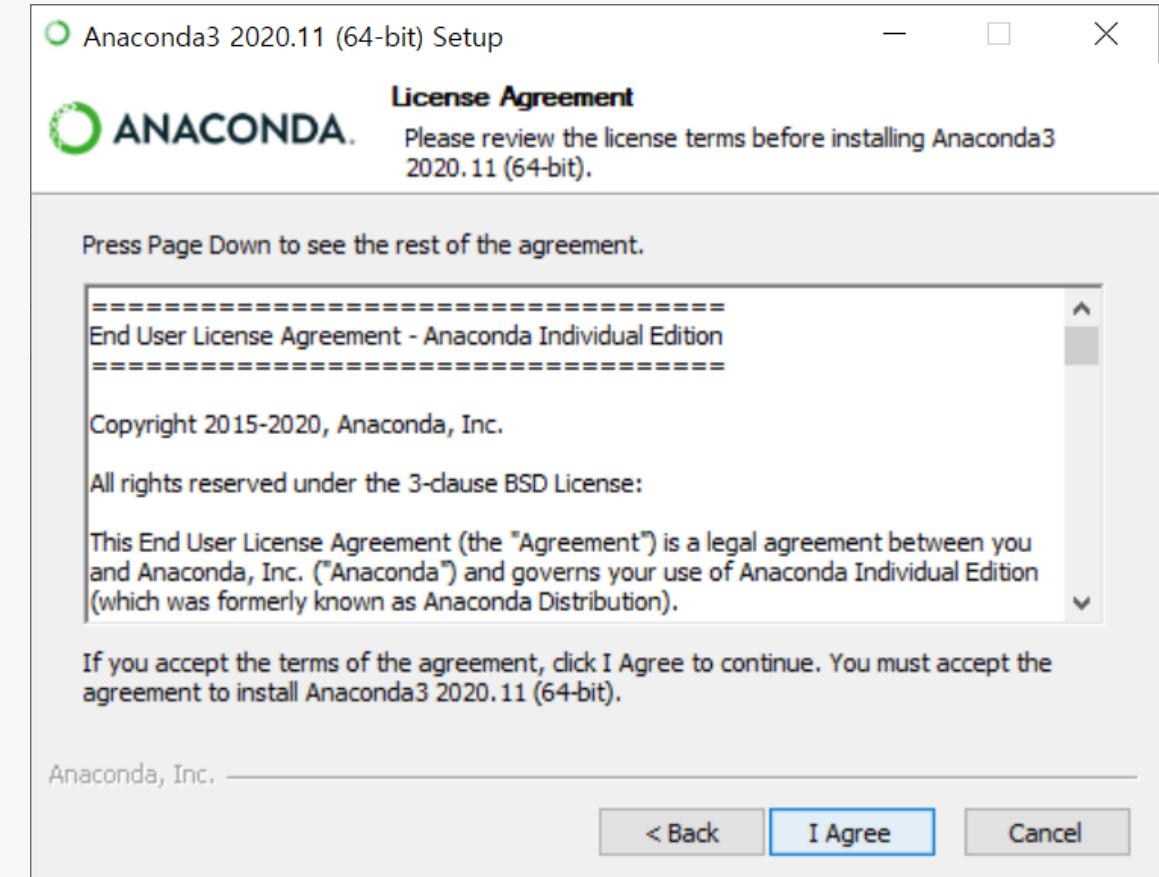
Platform	Python Version	Installer Type	File Name	Size
Windows	Python 3.8	Graphical Installer	64-Bit Graphical Installer	457 MB
		Graphical Installer	32-Bit Graphical Installer	403 MB
macOS	Python 3.8	Graphical Installer	64-Bit Graphical Installer	435 MB
		Command Line Installer	64-Bit Command Line Installer	428 MB
Linux	Python 3.8	(x86) Installer	64-Bit (x86) Installer	529 MB
		(Power8 and Power9) Installer	64-Bit (Power8 and Power9) Installer	279 MB

ADDITIONAL INSTALLERS

The archive has older versions of Anaconda Individual Edition installers. The Miniconda installer homepage can be found [here](#).

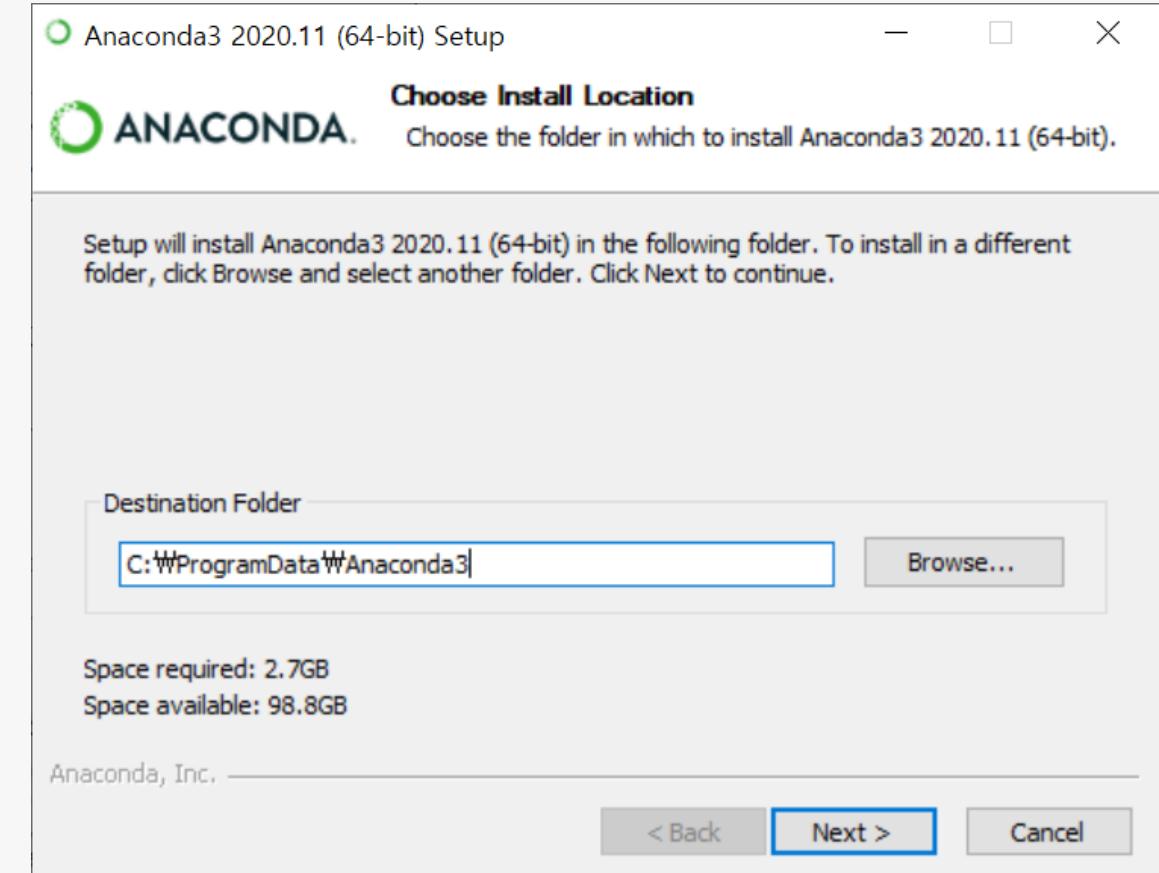
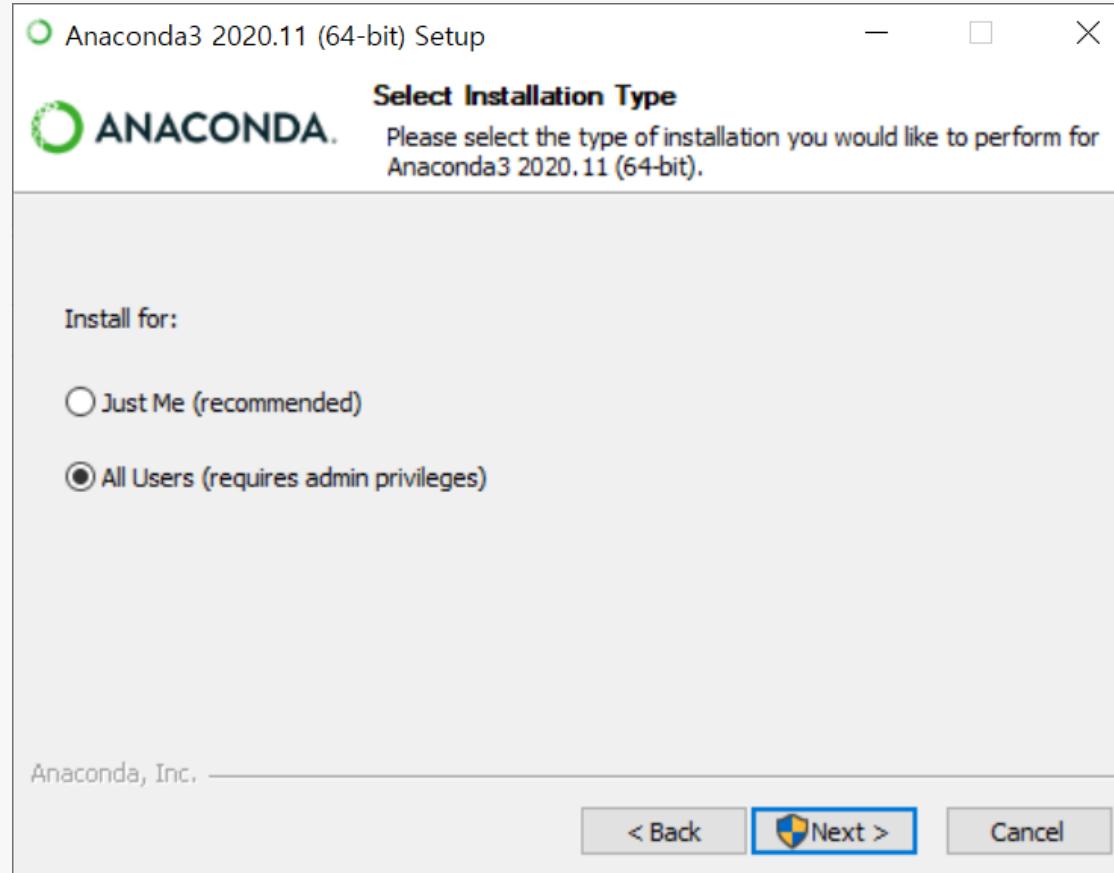
00. 아나콘다 – Windows 10

(2020년 12월 기준)



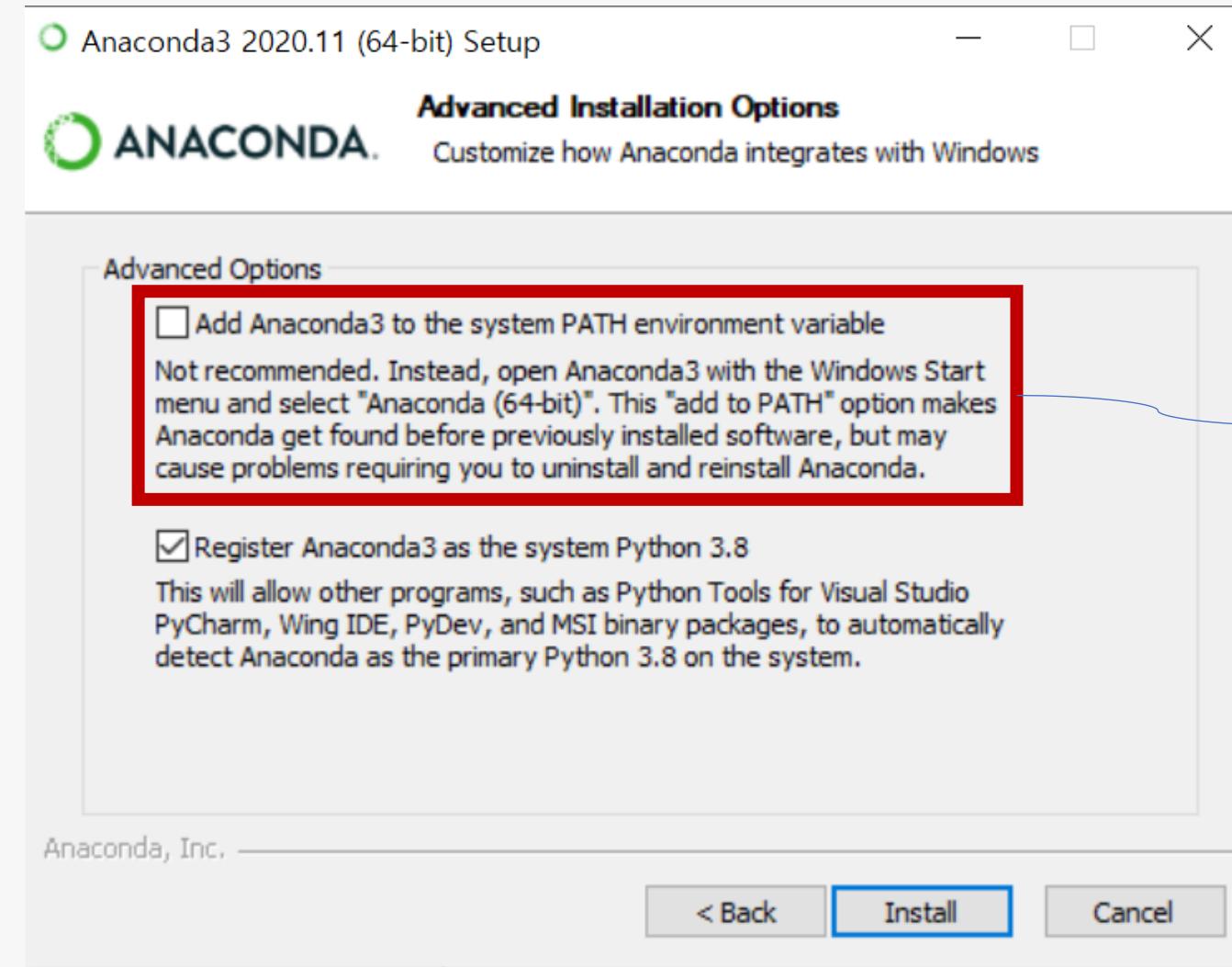
00. 아나콘다 – Windows 10

(2020년 12월 기준)



00. 아나콘다 – Windows 10

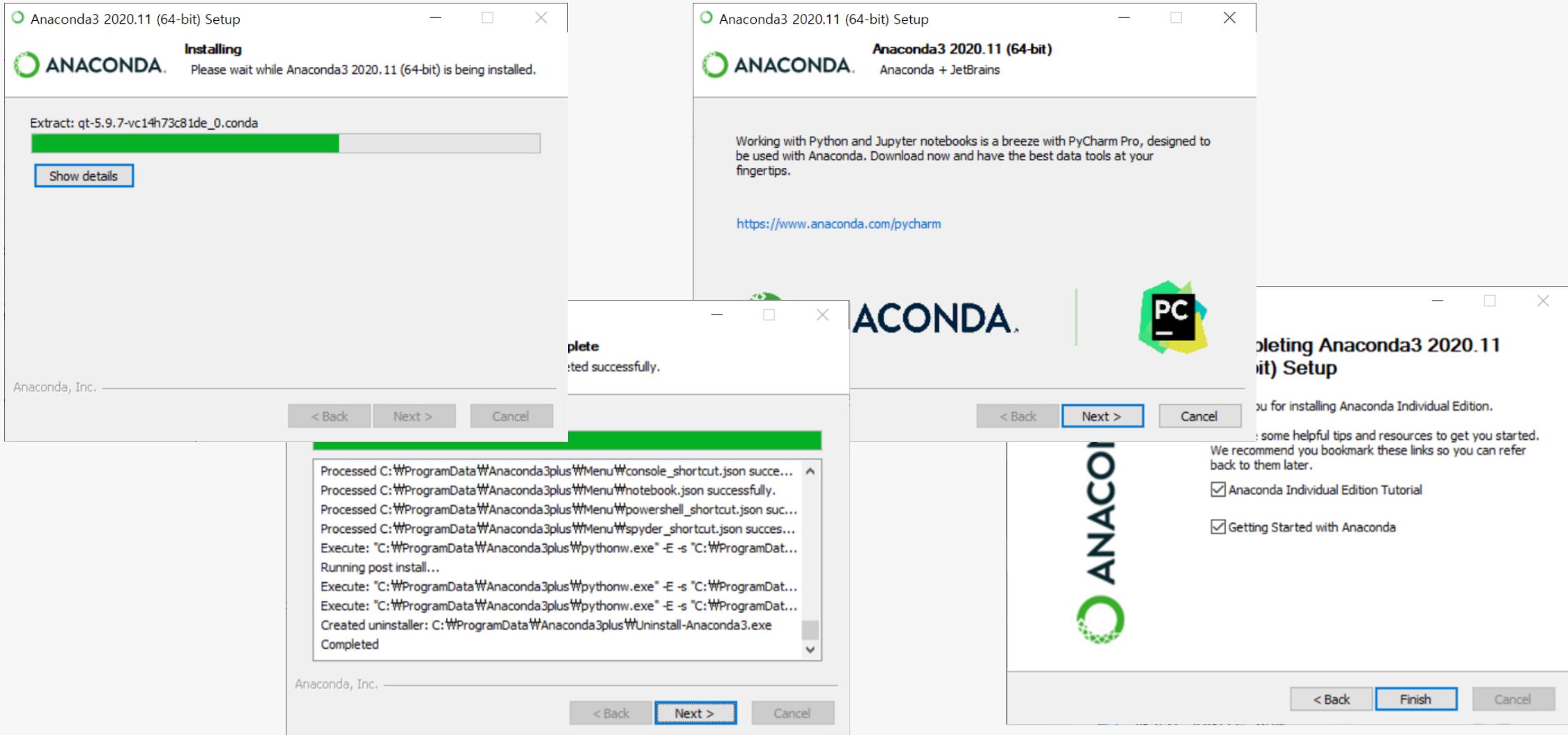
(2020년 12월 기준)



- 환경 변수의 개념을 잘 모른다면 추가하지 않는다.
- 환경 변수는 언제든지 추가가 가능하다.

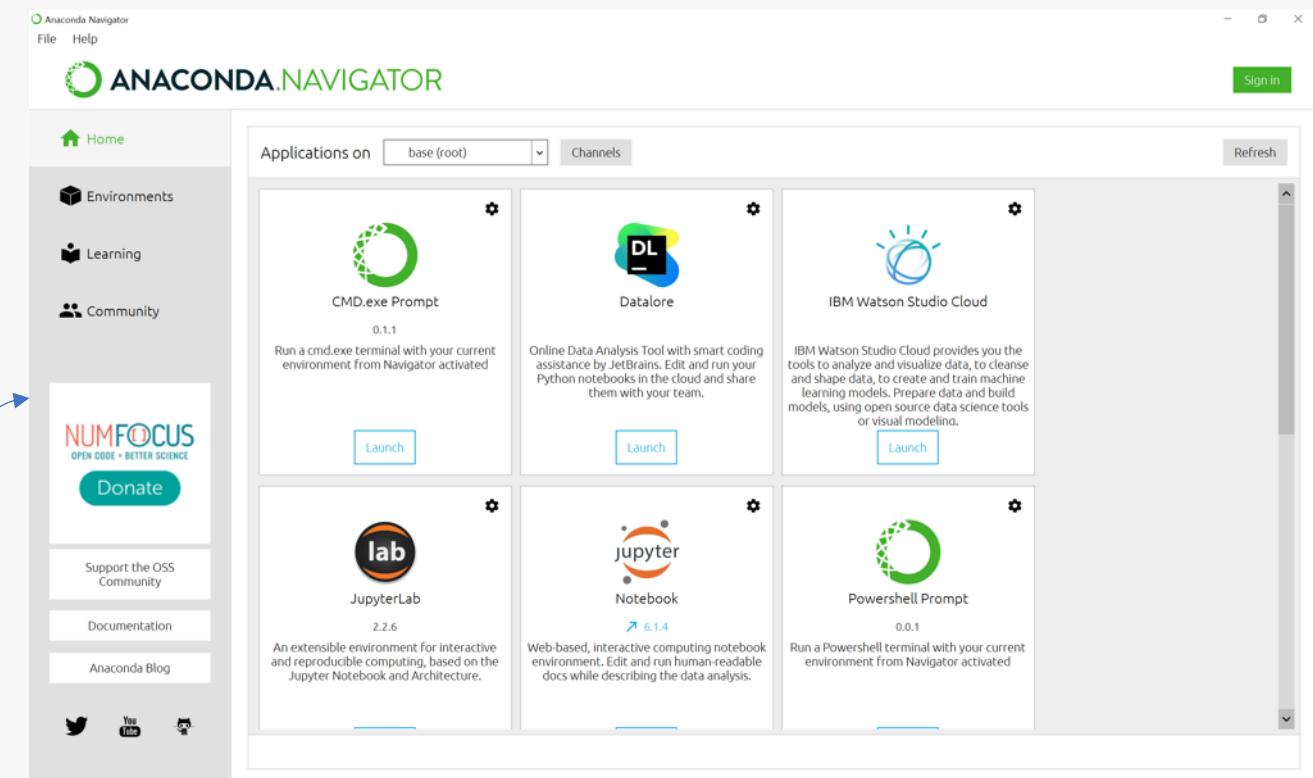
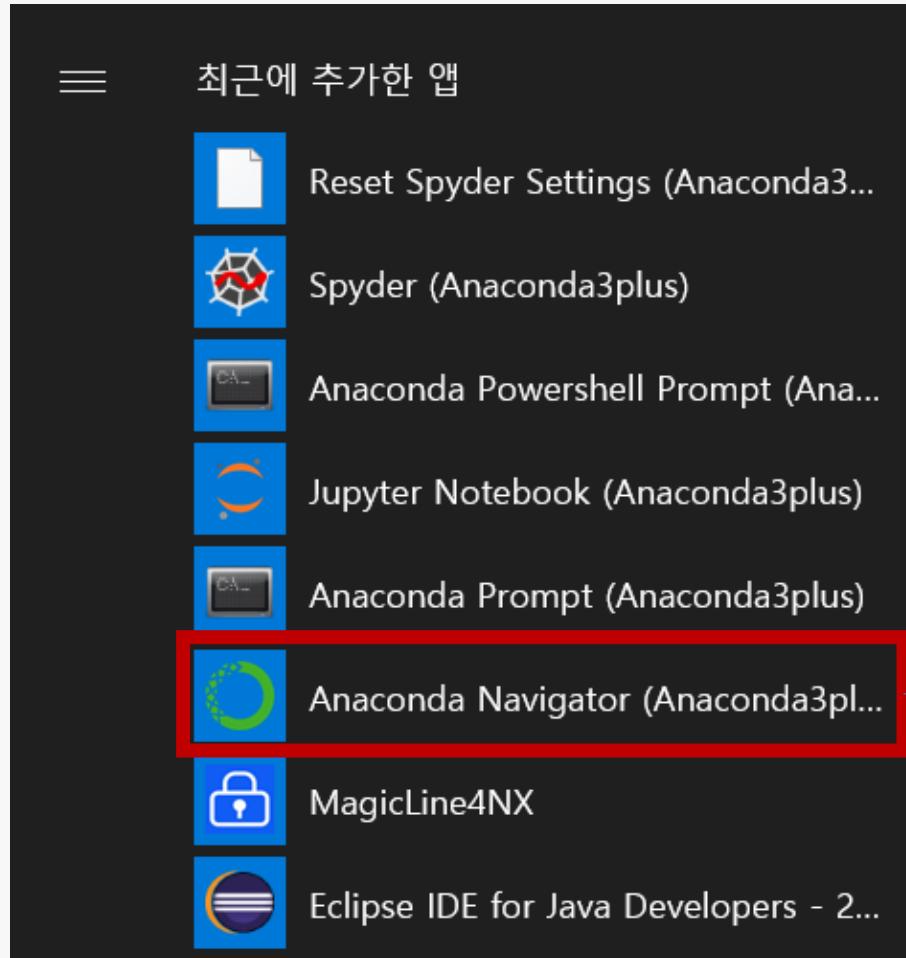
00. 아나콘다 – Windows 10

(2020년 12월 기준)

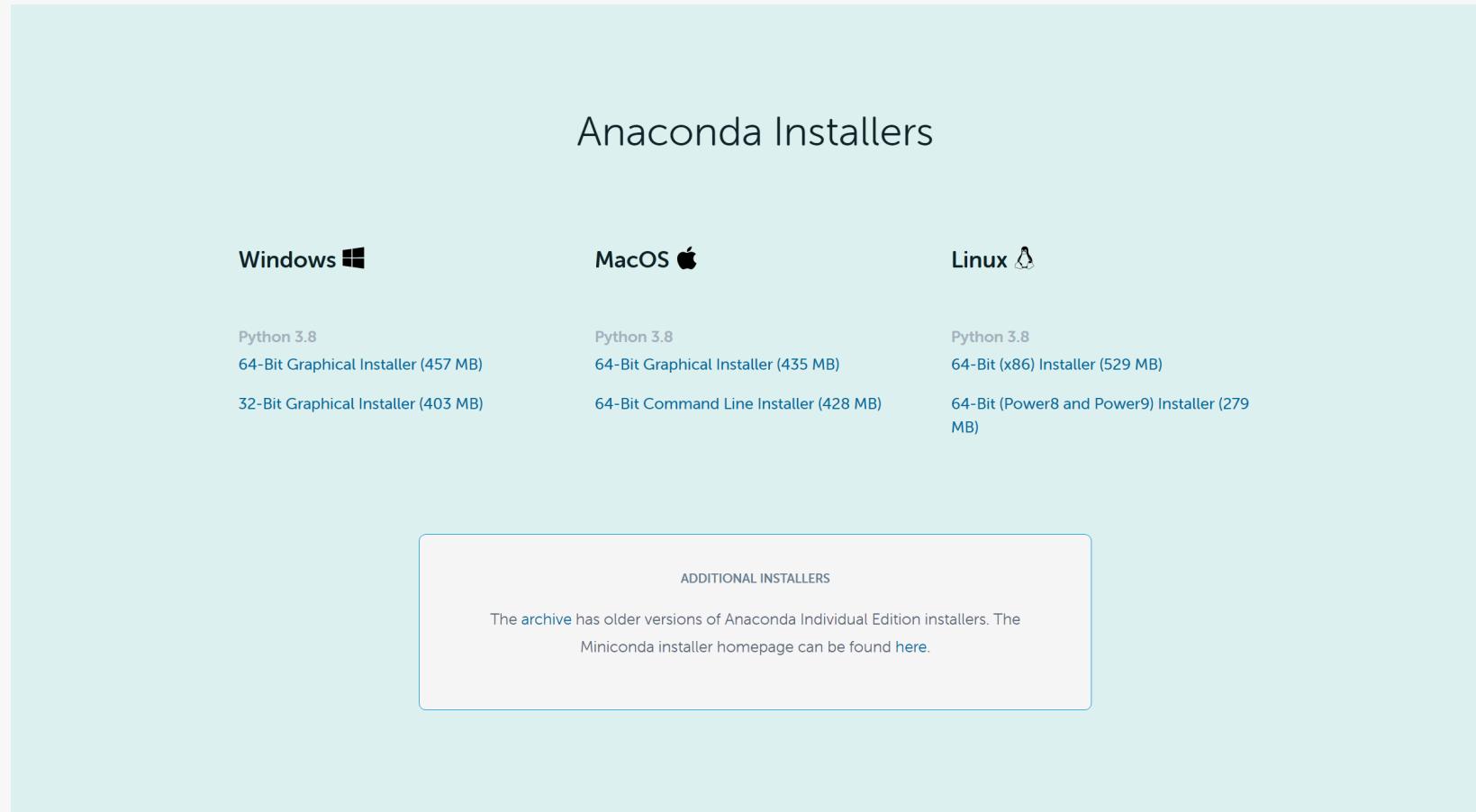


00. 아나콘다 – Windows 10

(2020년 12월 기준)

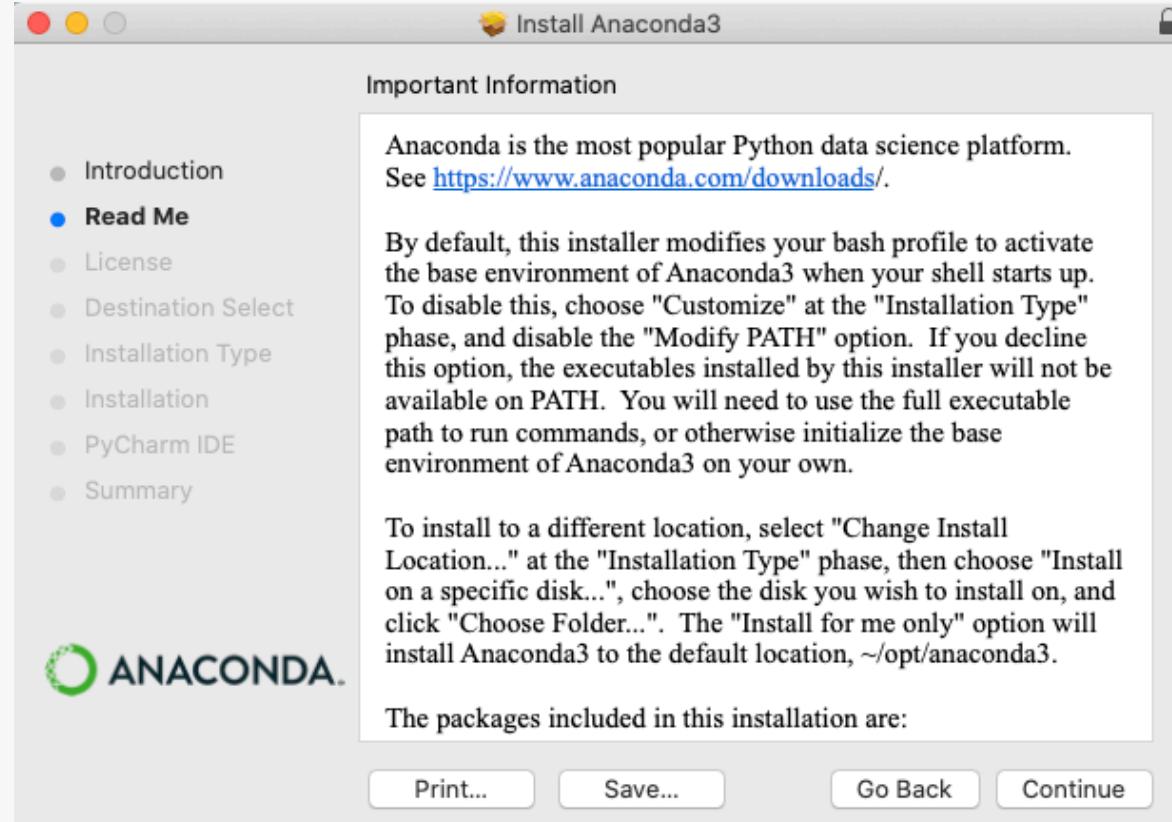
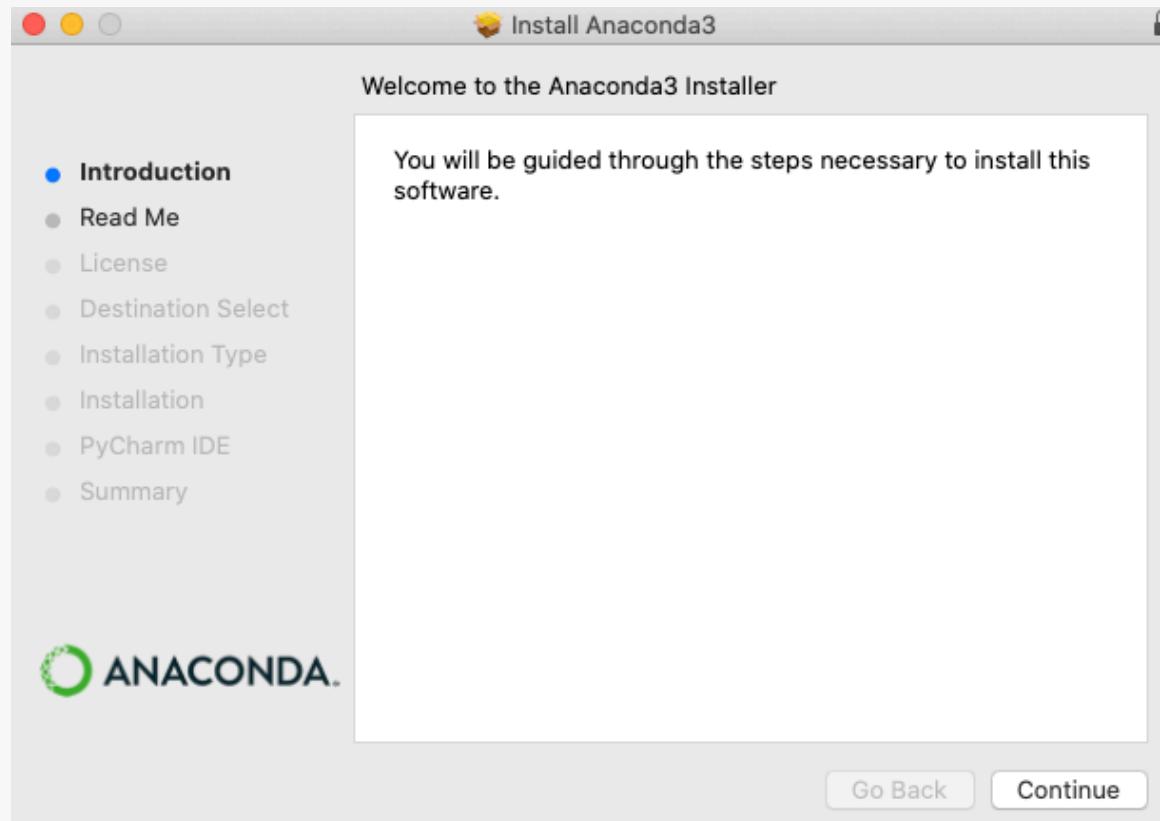


Shell Script 실행 방법 모른다면, 64-Bit Graphical Installer 선택
Shell Script 실행 방법 알고 있다면, 64-Bit Command Line Installer 선택



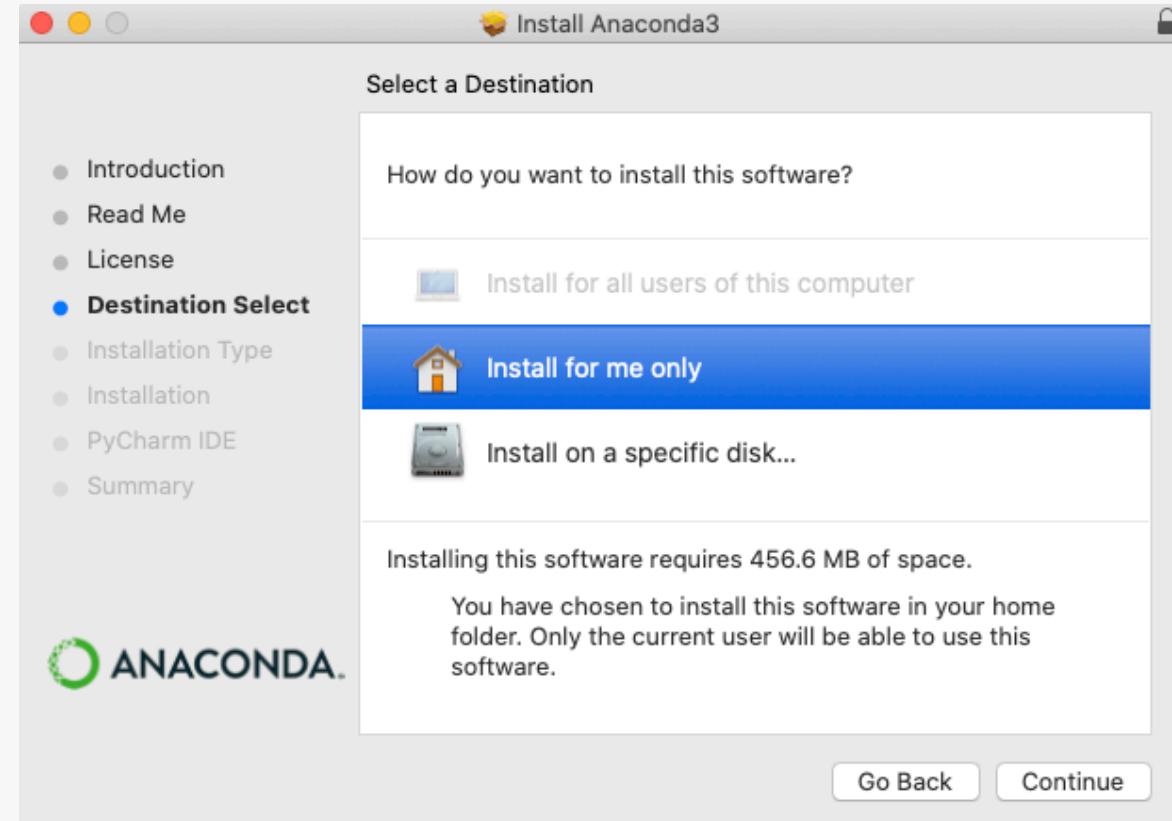
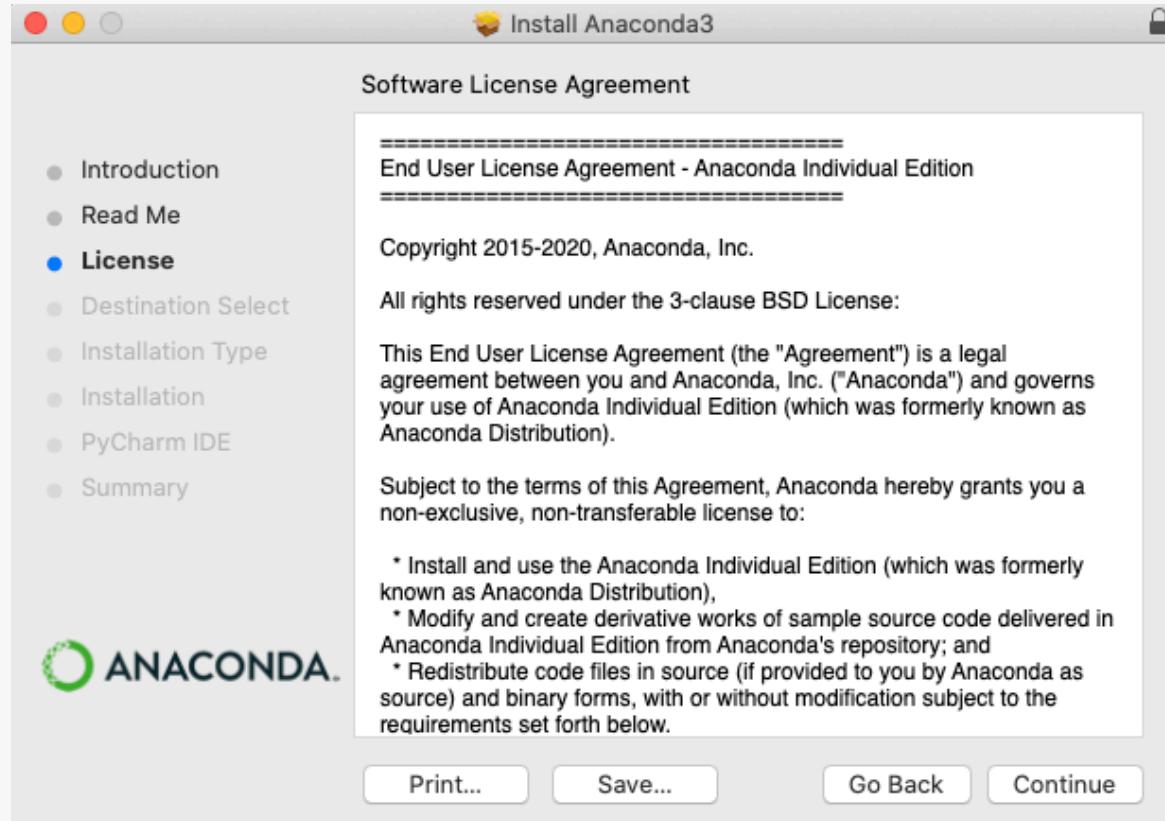
01. 아나콘다 – MacOS

(2020년 12월 기준)



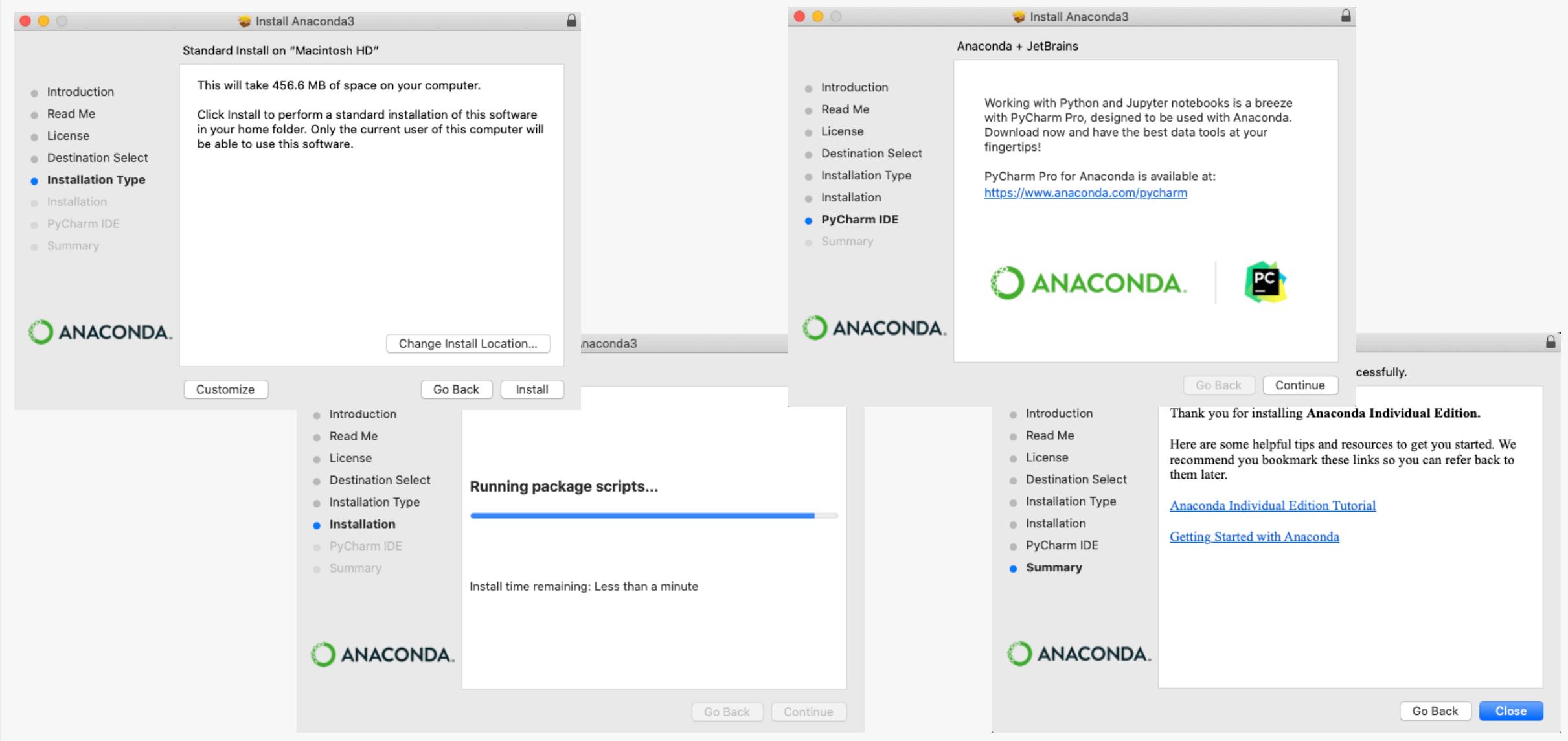
01. 아나콘다 – MacOS

(2020년 12월 기준)



01. 아나콘다 - MacOS

(2020년 12월 기준)



01. 아나콘다 - MacOS

(2020년 12월 기준)

The screenshot shows the Anaconda Navigator interface on a Mac OS X desktop. The window title is "Anaconda Navigator". On the left, there's a sidebar with links for Home, Environments, Learning, and Community, along with a "NUMFOCUS OPEN CODE + BETTER SCIENCE" section featuring a "Donate" button. The main area displays a grid of data science applications:

Application	Description	Version	Action
Datalore	Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.	2.2.6	Launch
IBM Watson Studio Cloud	IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.	2.2.6	Launch
JupyterLab	An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.	2.2.6	Launch
jupyter Notebook	Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.	6.1.4	Launch
PyCharm Community	An IDE by JetBrains for pure Python development. Supports code completion, listing, and debugging.	2020.2.1	Launch
Qt Console	PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.	4.7.7	Launch
Spyder	Scientific PYthon Development EnvIRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.	4.1.5	Launch
VS Code	Streamlined code editor with support for development operations like debugging, task running and version control.	1.52.1	Launch
Glueviz	Multidimensional data visualization across files. Explore relationships within and among related datasets.	1.0.0	Install
Orange 3	Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.	3.26.0	Install
PyCharm Professional	A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.	2020.2.1	Install
RStudio	A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.	1.1.456	Install

텍스트 마이닝: 텍스트 분석

with  python™

00. 텍스트 분석

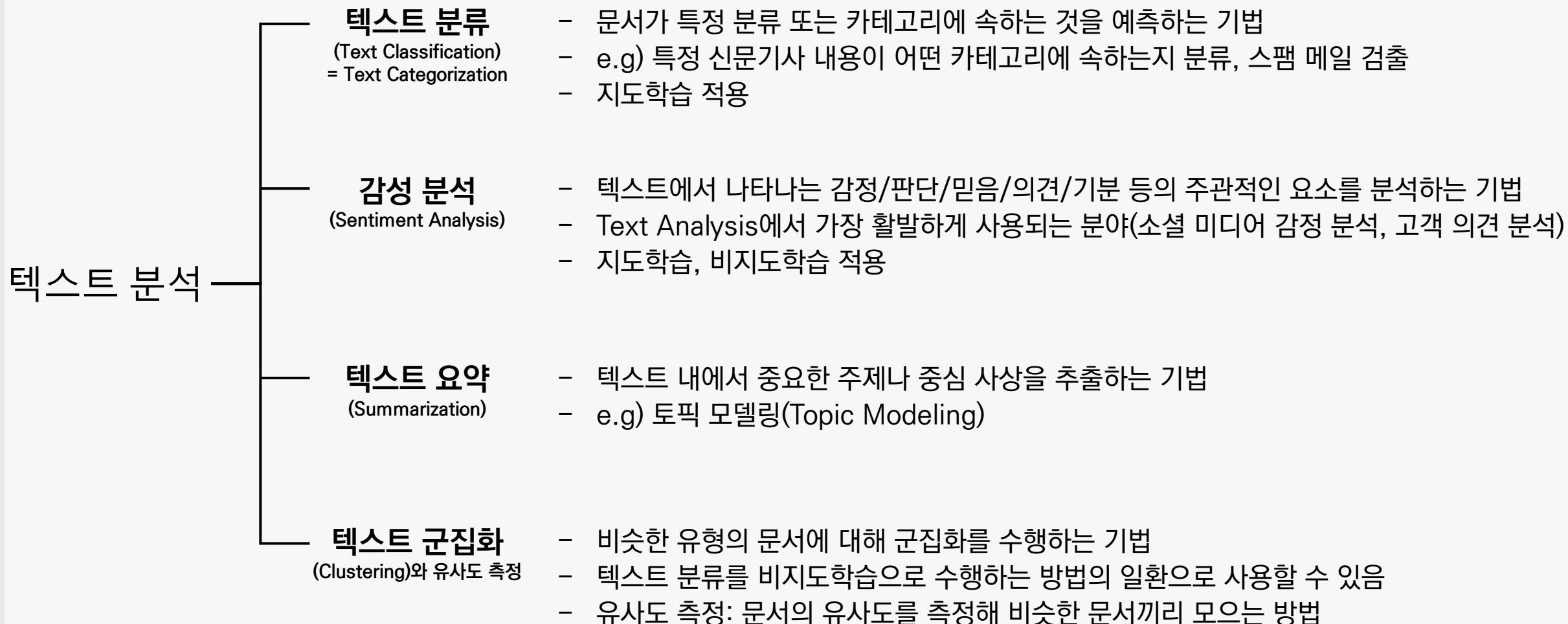
NLP(National Language Processing)

- 머신이 인간 언어를 이해하고 해석하는 데 중점
- 텍스트 분석을 향상하게 하는 기반 기술

텍스트 분석(Text Analytics, 텍스트 마이닝)

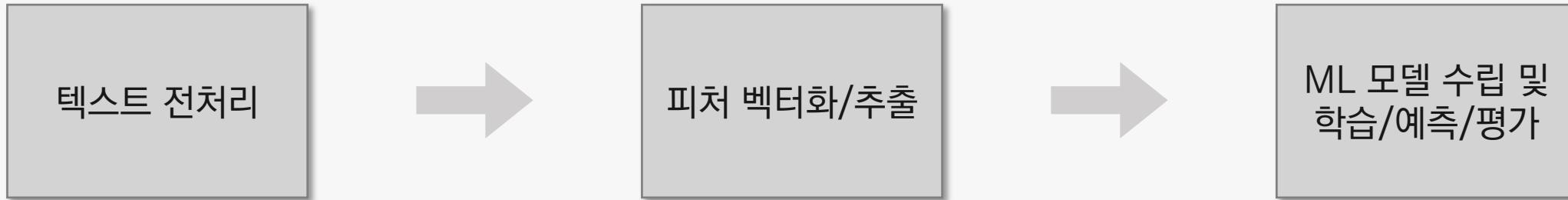
- 비정형 텍스트에서 의미 있는 정보를 추출하는 데 중점
- 비즈니스 인텔리전스나 예측 분석 등의 분석 작업 수행

00. 텍스트 분석



01. 텍스트 분석 이해

텍스트 분석 수행 프로세스

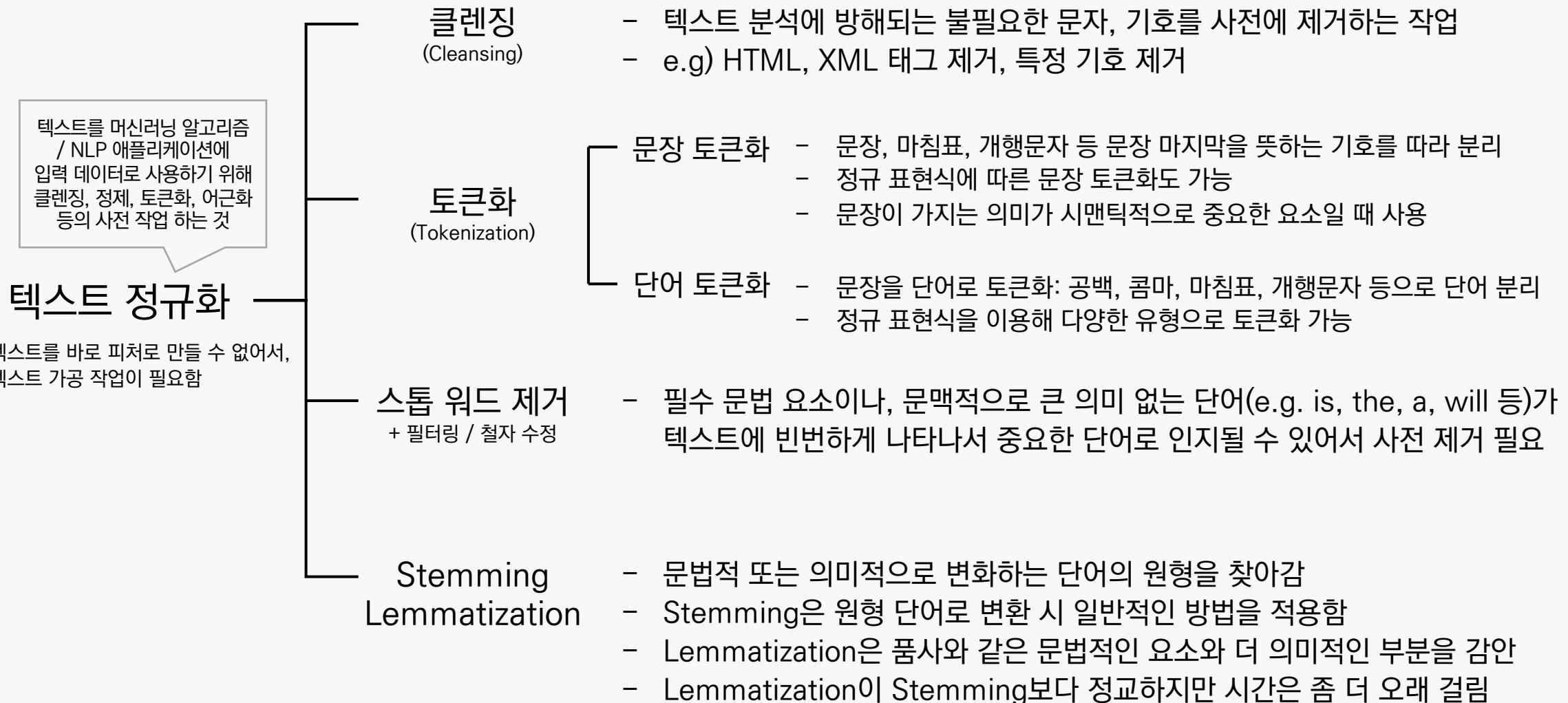


- 텍스트를 피처로 만들기 전 단계
 - 클렌징: 대/소문자 변경, 특수문자 삭제
 - 단어 등 토큰화 작업
 - 의미 없는 단어(Stop word) 제거
 - 어근 추출(Stemming/Lemmatization)
-
- 사전 준비 작업으로 가공된 텍스트에서 피처를 추출하고 벡터 값을 할당
 - 방법: BOW(Bag of Words), Word2Vec
-
- 피처 벡터화된 데이터 세트에 ML 모델 적용하여 학습/예측/평가

NLP, 텍스트 분석 패키지 (in Python)

: NLTK(Natural Language Toolkit for Python) / Gensim / SpaCy

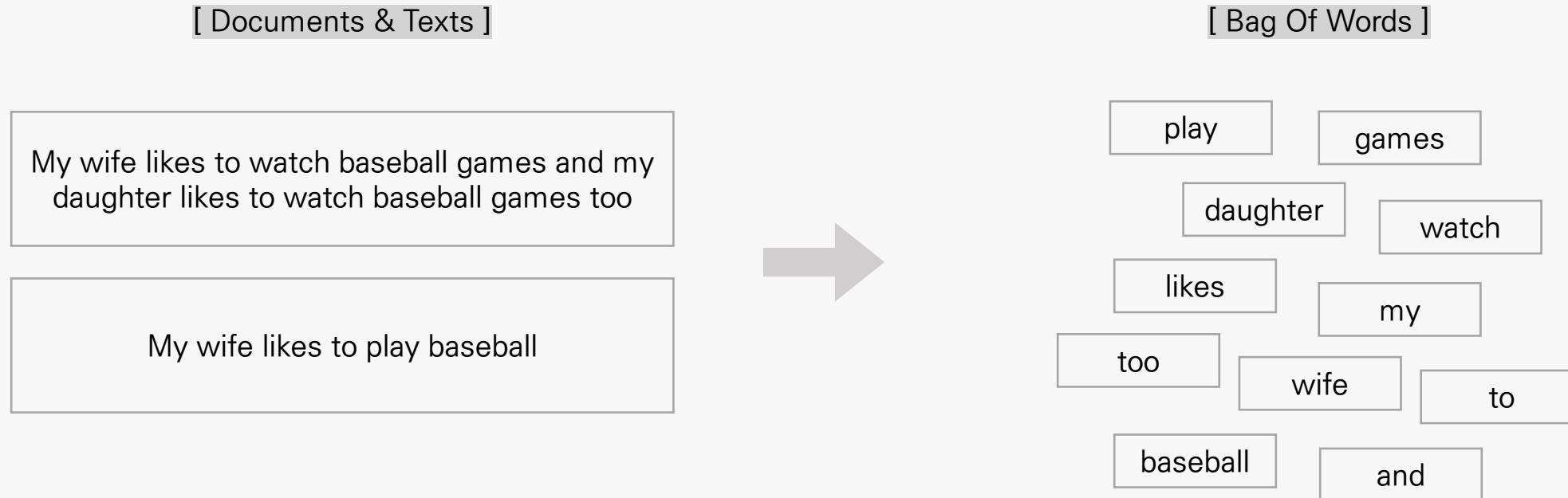
02. 텍스트 전처리: 텍스트 정규화



03. Bag of Words: BOW

BOW

: 문서 내 모든 단어를 문맥, 순서에 상관 없이 일괄적으로 단어에 빈도값을 부여해 피처값을 추출하는 모델



03. Bag of Words: BOW

BOW 처리 예시

- 문장 ①: My wife likes to watch baseball games and my daughter likes to watch baseball games too
- 문장 ②: My wife likes to play baseball

[Step 1]

- 모든 문장의 단어에서 중복을 제거하고 각 단어를 칼럼 형태로 나열
- 각 단어에 고유 인덱스를 부여



'and': 0, 'baseball': 1, 'daughter': 2, 'games': 3, 'likes': 4,
'my': 5, 'play': 6, 'to': 7, 'too': 8, 'watch': 9, 'wife': 10

[Step 2]

- 개별 문장에서 단어가 나타나는 횟수(Occurrence)를 기재

	index 0	index 1	index 2	index 3	index 4	index 5	index 6	index 7	index 8	index 9	index 10
문장 ①	1	2	1	2	2	2		2	1	2	1
문장 ②		1			1	1	1	1			1

03. Bag of Words: BOW

BOW 장단점

장점

- 쉽고 빠른 구축이 가능
- 단순한 단어 발생 횟수에 기반하지만, 문서 특징을 잘 나타낼 수 있는 모델

단점

- 문맥 의미(Semantic Context) 반영 부족
단어 순서를 고려하지 않기 때문에, 단어의 문맥적인 의미가 무시됨
이를 보완하기 위해 n_gram 기법을 쓸 수 있으나 제한적임
- 희소 행렬 문제 발생
BOW로 피처 벡터화 하면 희소 행렬 형태의 데이터 세트가 만들어지기 쉬움
매우 많은 단어가 칼럼으로 만들어지기 때문에 대부분 값이 0으로 채워지게 됨

* 희소 행렬(Sparse Matrix)

: 대규모 칼럼으로 구성된 행렬에서 대부분의 값이
0으로 채워지는 행렬

↔ 밀집행렬(Dense Matrix)

03. Bag of Words: BOW

BOW 피처 벡터화

: 텍스트를 특정 의미를 가지는 숫자형 값인 벡터 값으로 변환하는 것

- 머신러닝 알고리즘은 일반적으로 숫자형 피처를 데이터로 입력 받아 동작
 - 텍스트 데이터를 머신러닝 알고리즘에 바로 입력할 수 없음
 - 피처 벡터화가 필요

* 피처 벡터화

: 기존 텍스트 데이터를 또 다른 형태의 피처 조합으로
변경하는 것으로 넓은 범위의 피처 추출에 포함됨
Text Analysis에서는 피처 벡터화와 피처 추출을 같은 의미로 사용하고는 함

03. Bag of Words: BOW

BOW 모델에서의 피처 벡터화 수행

: 모든 문서의 모든 단어를 칼럼 형태로 나열하고,
각 문서에서 해당 단어의 횟수나 정규화된 빈도를 값으로 부여하는 데이터 세트 모델로 변경하는 것

* BOW 모델에서 피처 벡터화 수행하기

: M개의 텍스트 문서가 있고, 모든 단어를 추출했을 때 N개의 단어가 있을 경우,
피처 벡터화 하면 M개의 문서는 각각 N개 값이 할당된 피처의 벡터 세트가 됨
→ $M \times N$ 개의 단어 피처로 이뤄진 행렬을 구성하게 됨

BOW의 피처 벡터화 방식 2가지

카운트 기반의 벡터화

- 단어 피처에 값을 부여할 때, 각 문서에서 해당 단어가 나타나는 횟수(Count)를 부여하는 경우
- 단점: 문서의 특징을 나타내는 것보다 언어 특성상 문서에서 자주 쓰이는 단어에도 높은 값을 부여하게 됨

TF-IDF 기반의 벡터화

- 개별 문서에서 자주 쓰인 단어에 높은 가중치를 주되, 모든 문서에 전반적으로 나타나는 단어에는 패널티를 주는 방식으로 값을 부여
- 카운트 기반의 벡터화 단점을 보완하기 위한 방식

03. Bag of Words: BOW

사이킷런의 Count 및 TF-IDF 벡터화 구현: CountVectorizer, TfidfVectorizer

- 사이킷런의 CountVectorizer 클래스

-
- ```
graph TD; A[사전 데이터 가공] --> B[토큰화]; B --> C[텍스트 정규화]; C --> D[피처 벡터화];
```
- 사전 데이터 가공
    - 모든 문자를 소문자로 변환 (Default로 lowercase = True)
  - 토큰화
    - 디폴트는 단어 기준(analyzer = True)
    - n\_gram\_range를 반영하여 토큰화
  - 텍스트 정규화
    - Stop Words 필터링만 수행
  - 피처 벡터화
    - 토큰화된 단어를 피처로 추출
    - 단어 빈도수 벡터값 적용

| 파라미터 명        | 파라미터 설명                                                                                                                                                                                                                                                                                                                                          |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| max_df        | <ul style="list-style-type: none"><li>- 전체 문서에서 너무 높은 빈도수를 가지는 단어 피처를 제외하기 위한 파라미터</li><li>- 높은 빈도수를 가지는 단어는 문법적 특성으로 반복될 가능성이 높아 이를 제거하기 위해 사용</li><li>- max_df = 100(정수 값): 전체 문서에 걸쳐 100개 이하로 나타나는 단어만 피처로 추출</li><li>- max_df = 0.95(부동소수점 값): 전체 문서에 걸쳐 빈도수 0~95% 단어만 피처로 추출</li></ul>                                                      |
| min_df        | <ul style="list-style-type: none"><li>- 전체 문서에서 너무 낮은 빈도수를 가지는 단어 피처를 제외하기 위한 파라미터</li><li>- 수백~수천 개의 전체 문서에서 특정 단어가 min_df에 설정된 값보다 적은 빈도수를 가지면, 크게 중요하지 않거나 가비지(garbage)성 단어일 확률이 높음</li><li>- min_df = 2(정수 값): 전체 문서에 걸쳐 2번 이하로 나타나는 단어는 피처로 추출하지 않음</li><li>- min_df = 0.02(부동소수점 값): 전체 문서에 걸쳐 하위 2% 이하 빈도수를 가지는 단어는 피처로 추출하지 않음</li></ul> |
| max_feature   | <ul style="list-style-type: none"><li>- 추출하는 피처 개수를 제한하며 정수로 값을 지정</li><li>- max_features = 2000: 가장 높은 빈도를 가지는 단어 순으로 정렬해 2000개까지만 피처로 추출</li></ul>                                                                                                                                                                                             |
| stop_words    | <ul style="list-style-type: none"><li>'english'로 지정하면 영어의 스톱 워드로 지정된 단어를 추출에서 제외</li></ul>                                                                                                                                                                                                                                                       |
| n_gram_range  | <ul style="list-style-type: none"><li>- Bag of Words 모델의 단어 순서를 어느 정도 보강하기 위한 n_gram 범위를 설정</li><li>- 튜플 형태로 범위 최솟값, 범위 최댓값을 지정<ul style="list-style-type: none"><li>- (1,1): 토큰화된 단어를 1개씩 피처로 추출,</li><li>- (1,2): 토큰화된 단어를 1개씩(minimum 1), 그리고 순서대로 2개씩(maximum 2) 묶어 피처로 추출</li></ul></li></ul>                                               |
| analyzer      | <ul style="list-style-type: none"><li>- 피처 추출을 수행한 단위를 지정, 디폴트는 'word'</li><li>- Word가 아니라 character의 특정 범위를 피처로 만드는 특정한 경우를 적용할 때 사용</li></ul>                                                                                                                                                                                                  |
| token_pattern | <ul style="list-style-type: none"><li>- 토큰화를 수행하는 정규 표현식 태평을 지정</li><li>- 디폴트 값은 '/b/w/w+/b/'로 공백/개행 문자로 구분된 단어 분리자(/b) 사이 영숫자 2 이상 단어를 토큰으로 분리</li><li>- analyzer = 'word'로 설정했을 때만 변경 가능하나, 디폴트 값을 변경할 경우는 거의 발생하지 않음</li></ul>                                                                                                                |
| tokenizer     | <ul style="list-style-type: none"><li>- 토큰화를 별도의 커스텀 함수로 이용할 때 적용</li><li>- CountVectorizer 클래스에서 어근 변환 시, 이를 수행하는 함수를 tokenizer 파라미터에 적용하면 됨</li></ul>                                                                                                                                                                                          |

\* 참고: 사이킷런에서 TF-IDF 벡터화는 TfidfVectorizer 클래스를 이용, 파라미터와 변환 방법은 CountVectorizer와 동일

# 03. Bag of Words: BOW

## BOW 벡터화를 위한 희소 행렬

\* 희소 행렬(Sparse Matrix)

: 대규모 칼럼으로 구성된 행렬에서 대부분의 값이 0으로 채워지는 행렬

### - COO 형식

: 0이 아닌 데이터만 별도의 데이터 배열에 저장하고, 그 데이터가 가리키는 행과 열의 위치를 별도의 배열로 저장하는 방식

### - CSR 형식

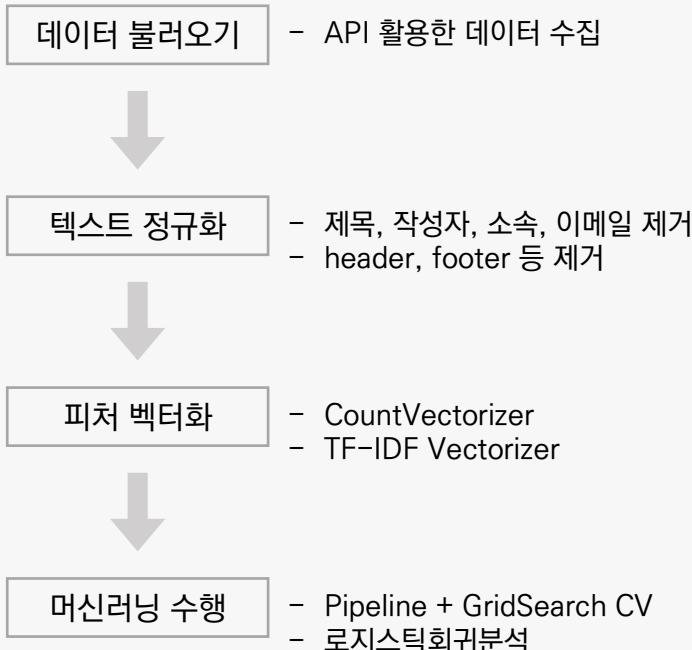
: 행 위치 배열 내에 있는 고유한 값의 시작 위치만 다시 별도의 위치 배열로 가지는 변환 방식

: COO 형식이 행과 열의 위치를 나타내기 위해 반복적인 위치 데이터를 사용해야 하는 문제점을 해결한 방식

: 고유 값의 시작 위치만 알면, 행 위치 배열을 다시 만들 수 있어서 COO 방식보다 메모리가 적게 들고 빠른 연산이 가능

# 04. 텍스트 분류 실습: 20 뉴스그룹 분류

## fetch\_20newsgroups() API 예제 데이터를 활용한 실습

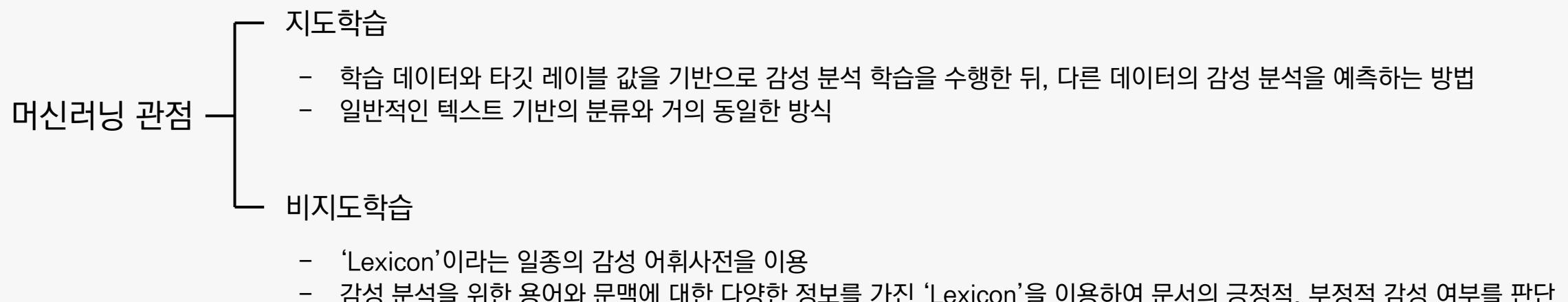


(Source: Gupta, 2018)

# 05. 감성 분석

## 감성 분석(Sentiment Analysis)

- : 문서의 주관적 감성/의견/감정/기분 등을 파악하기 위한 방법 (소셜 미디어, 여론조사, 온라인 리뷰, 피드백 ..)
- : 문서 내 텍스트가 나타내는 주관적인 단어와 문맥을 기반으로 감성 수치를 계산하는 방법



# 05. 감성 분석

## 비지도학습 기반 감성 분석: Lexicon

### \* NLP 패키지의 WordNet 모듈

- : 시맨틱(semantic, 문맥상 의미) 분석을 제공하는 어휘 사전
- : 각 품사로 구성된 개별 단어를 Synset(Sets of cognitive synonyms) 개념으로 표현

- : 긍정(positive) 감성 또는 부정(negative) 감성 정도를 의미하는 감성 지수(Polarity score) 수치를 가짐
- : 감성 지수는 단어 위치나 주변 단어, 문맥, POS(Part of Speech) 등을 참고해 결정됨

## 대표적인 감성 사전

### NLTK

- : 감성 사전인 Lexicon 모듈을 포함한 많은 서브 모듈을 보유
- : 예측 성능이 좋지 못해 실무에서는 일반적으로 다른 감성 사전을 적용함

### SentiWordNet

- : NLTK 패키지의 WordNet과 유사하게 감성 단어 전용의 WordNet을 구현(WordNet의 Synset 개념을 감성 분석에 적용)
- : 긍정 감정/부정 감성/객관성 지수를 할당 후, 문장별로 단어들의 긍정 감성 지수와 부정 감성 지수를 합산하여 최종 감성 지수를 계산하고 감성을 결정

### VADER

- : 주로 소셜 미디어의 텍스트에 관한 감성 분석을 제공하기 위한 패키지
- : 뛰어난 감성 분석 결과를 제공하며, 비교적 빠른 수행 시간을 보장해 대용량 텍스트 데이터에 잘 사용됨

### Pattern

- : 예측 성능 측면에서 가장 주목받는 패키지 (20.07 기준 파이썬 2.X 버전에서만 동작)

# 06. 토픽 모델링

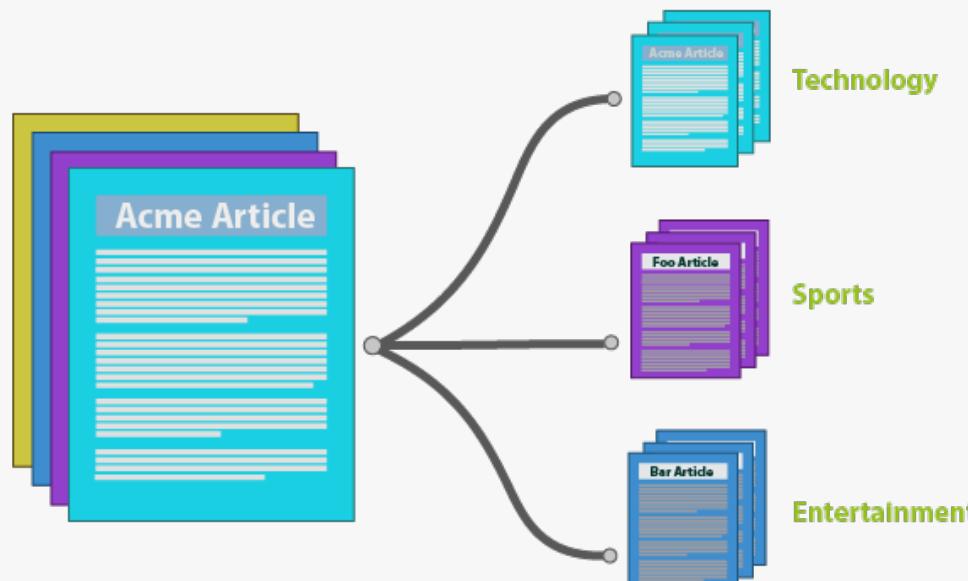
## 토픽 모델링(Topic Modeling)

- : 문서 집합에 숨어 있는 주제를 찾아내는 것
- : 머신러닝 기반의 토픽 모델은 숨겨진 주제를 효과적으로 표현할 수 있는 중심 단어를 함축적으로 추출
- : LSA(Latent Semantic Analysis)와 LDA(Latent Dirichlet Allocation) 기법을 주로 사용

# 07. 문서 군집화

## 문서 군집화(Document Clustering)

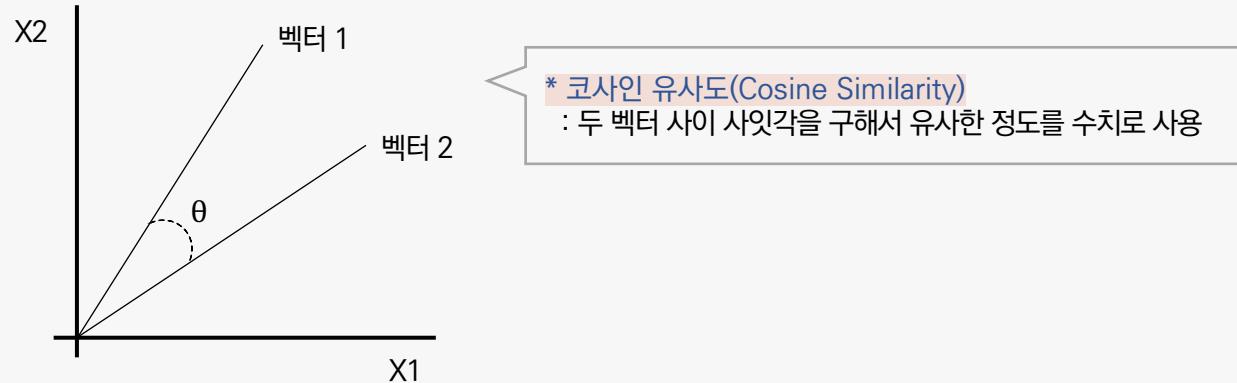
- : 비슷한 텍스트 구성의 문서를 군집화 하는 것
- : 동일한 군집에 속하는 문서를 같은 카테고리 소속으로 분류할 수 있으므로 텍스트 분류 기반의 문서 분류와 유사  
단, 학습 데이터 세트가 필요 없는 비지도학습 기반으로 동작



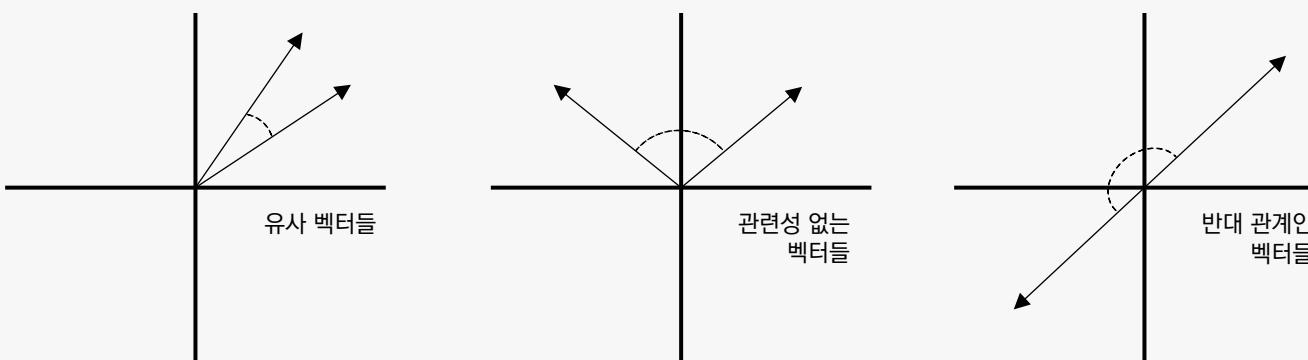
# 08. 문서 유사도

## 문서 유사도 측정 방법- 코사인 유사도(Cosine Similarity)

: 벡터와 벡터의 유사도를 비교할 때, 벡터 크기보다는 벡터의 상호 방향성이 얼마나 유사한지에 기반하여 비교



### 두 벡터 사잇각



# 08. 문서 유사도

## 두 벡터 A와 B의 코사인 값

: 두 벡터 A와 B의 내적 값은 두 벡터 크기를 곱한 값의 코사인 각도 값을 곱한 것

$$A * B = \|A\| \|B\| \cos \theta$$

## 유사도

: 유사도  $\cos\theta$ 는 두 벡터 내적을 총 벡터 크기의 합으로 나눈 것

즉, 내적 결과를 총 벡터 크기로 정규화(L2 Norm)한 것

\* 코사인 유사도가 문서 유사도 비교에 많이 사용되는 이유

: 문서를 피처 벡터화 변환하면 차원이 매우 많은 희소 행렬이 되기 쉬워, 문서와 문서 벡터 간 크기에 기반한 유사도 지표 (e.g. 유clidean 거리 기반 지표)는 정확도가 떨어지기 쉬움

$$\text{similarity} = \cos\theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

# 08. 문서 유사도

## 문서 유사도 측정 방법- 자카드 유사도(Jaccard Similarity)

: 두 문장을 각각 단어의 집합으로 만든 뒤 두 집합을 통해 유사도를 측정하는 방식 중 하나

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

\* 자카드 유사도(Cosine Similarity)

: 두 문장 사이의 합집합과 교집합의 원리를 이용해서 구한다

### 자카드 유사도 구하는 방법

$$\left. \begin{array}{l} \text{Set } A = ["A", "B", "C"] \\ \text{Set } B = ["D", "A", "E"] \end{array} \right\} J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{|1|}{|3| + |3| - |1|} = \frac{1}{5} = 0.2$$

# 09. 한글 텍스트 처리

## 한글 NLP 처리의 어려움

### 띄어쓰기

- 영어의 경우 띄어쓰기를 잘못하면 잘못되거나 없는 단어로 인식되나, 한국어는 의미가 왜곡됨

### 다양한 조사

- 경우의 수가 많기 때문에 어근 추출(Stemming/Lemmatization) 등의 전처리 시 제거하기가 까다로움

## KoNLPy

: 파이썬의 대표적인 한글 형태소 패키지

: 형태소 분석(Morphological analysis)

- 말뭉치를 형태소 어근 단위로 쪼개고 각 형태소에 품사 태깅(POS tagging)을 부착하는 작업

# 10. 텍스트 분석 실습

## 캐글 Mercari Price Suggestion Challenge

- train\_id: 데이터 id
- name: 제품명
- item\_condition\_id: 판매자가 제공하는 제품 상태
- category\_name: 카테고리 명
- brand\_name: 브랜드 명
- price: 제품 가격, 예측을 위한 타깃 속성
- shipping: 배송비 무료 여부, 1이면 무료, 0이면 유료
- item\_description: 제품 설명

The screenshot shows the Kaggle Mercari Price Suggestion Challenge page. At the top, it says 'Featured Code Competition' and 'Mercari Price Suggestion Challenge'. Below that, it asks 'Can you automatically suggest product prices to online sellers?'. It shows 'Mercari · 2,382 teams · 3 years ago'. The navigation bar includes 'Overview' (which is underlined), 'Data', 'Notebooks', 'Discussion', 'Leaderboard', 'Rules', 'Team', 'My Submissions', and 'Late Submission' (which is highlighted). The main content area has tabs for 'Description', 'Evaluation', 'Prizes', 'Timeline', and 'Kernels-FAQ'. Under 'Description', it says: 'It can be hard to know how much something's really worth. Small details can mean big differences in pricing. For example, one of these sweaters cost \$335 and the other cost \$9.99. Can you guess which one's which?'. Under 'Sweater A:', it says: 'Vince Long-Sleeve Turtleneck Pullover Sweater, Black, Women's, size L, great condition.' Under 'Sweater B:', it says: 'St. John's Bay Long-Sleeve Turtleneck Pullover Sweater, size L, great condition'.

# 11. 정리

## 머신러닝 기반의 텍스트 분석 프로세스

### ① 텍스트 정규화 작업: 텍스트 사전 정제 작업 등

: 피처 벡터화를 진행하기 이전에 수행하는 다양한 사전 작업

: 텍스트 클렌징 및 대소문자 변경, 단어 토큰화, 의미 없는 단어 필터링, 어근 추출 등

### ② 단어를 피처 벡터화로 변환

: BOW의 대표 방식인 Count 기반과 TF-IDF 기반 피처 벡터화

### ③ 생성된 피처 벡터 데이터 세트에 머신러닝 모델을 학습하고 예측/평가

: 피처 벡터 데이터 세트는 희소 행렬이며, 머신러닝 모델은 희소 행렬 기반에서 최적화되어야 함

# 자연어 처리 : 자연어 처리 개요

with  python™

# 01. 단어 표현

## 자연어 처리

: 컴퓨터가 인간의 언어를 이해하고 분석 가능한 모든 분야

“어떻게 자연어를 컴퓨터에 인식시킬 수 있을까?”

## 유니코드

- 컴퓨터가 텍스트를 인식하는 방법 (영어: 아스키코드)

언 1100010110111000

어 1100010110110100



컴퓨터가 문자를 인식하기 위해 만들어진 값으로,  
언어적인 특성이 전혀 없어 자연어 처리 모델에 부적합함

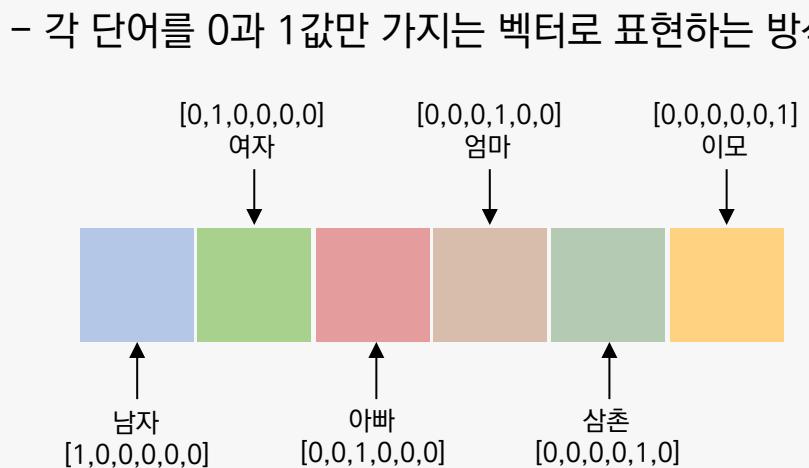
# 01. 단어 표현

## 단어 표현(Word Representation)

- : 텍스트를 자연어 처리를 위한 모델에 적용할 수 있도록, 언어적인 특성을 반영해 단어를 수치화하는 방법을 찾는 것
- : 주로 단어를 벡터로 표현해 수치화 → 단어 임베딩(word embedding) or 단어 벡터(word vector)로도 불림

### 원-핫 인코딩(one-hot encoding) 방식

\* 원-핫 인코딩  
: 단어의 인덱스를 정한 후, 각 단어 벡터에서 그 단어에 해당하는 인덱스 값을 1로 표현



- 각 단어를 표현하는 벡터 크기: 6
- 각 단어의 벡터는 총 6개의 값을 가지며, 하나만 1이 됨

### 원-핫 인코딩 방식의 문제점

1. 단어 벡터의 크기가 너무 크고 값이 희소(sparse)함
2. 벡터값 자체에는 단어의 의미나 특성이 전혀 표현되지 않음

# 01. 단어 표현

## 원-핫 인코딩 문제 해결

### 분포 가설(Distributed hypothesis)

- 같은 문맥에 나오는 단어는 비슷한 의미를 가진다는 개념
- 분포 가설을 기반으로 두 가지 인코딩 방식을 만들어냄

#### 1. 카운트 기반(count-base) 방법

- 문맥 안에 단어가 동시에 등장하는 횟수를 세는 방법
- 동시 등장 횟수를 하나의 행렬로 나타낸 뒤 그 행렬을 수치화하여 단어 벡터로 만드는 방법을 사용
  - ① 특이값 분해(Singular Value Decomposition, SVD)
  - ② 잠재의미분석(Latent Semantic analysis, LSA)
  - ③ Hyperspace Analogue to Language(HAL)
  - ④ Hellinger PCA(Principal Component Analysis)

\* 동시에 등장하는 횟수

: 동시 출현, 공기, Co-occurrence

# 01. 단어 표현

## 원-핫 인코딩 문제 해결

### 카운트 기반(count-base) 방법 예시

성진과 창욱은 야구장에 갔다.

성진과 태균은 도서관에 갔다.

성진과 창욱은 공부를 좋아한다.



|       | 성진과 | 창욱은 | 태균은 | 야구장에 | 도서관에 | 공부를 | 갔다. | 좋아한다. |
|-------|-----|-----|-----|------|------|-----|-----|-------|
| 성진과   | 0   | 2   | 1   | 0    | 0    | 0   | 0   | 0     |
| 창욱은   | 2   | 0   | 0   | 1    | 0    | 1   | 0   | 0     |
| 태균은   | 1   | 0   | 0   | 0    | 1    | 0   | 0   | 0     |
| 야구장에  | 0   | 1   | 0   | 0    | 0    | 0   | 1   | 0     |
| 도서관에  | 0   | 0   | 1   | 0    | 0    | 0   | 1   | 0     |
| 공부를   | 0   | 1   | 0   | 0    | 0    | 0   | 0   | 1     |
| 갔다.   | 0   | 0   | 0   | 1    | 1    | 0   | 0   | 0     |
| 좋아한다. | 0   | 0   | 0   | 0    | 0    | 1   | 0   | 0     |

### 카운트 기반 방식 장점

1. 적은 시간으로 단어 벡터를 만들 수 있음
2. 데이터가 많을 경우에 단어가 잘 표현되며 효율적임

# 01. 단어 표현

## 원-핫 인코딩 문제 해결

### 2. 예측 방법

- 신경망 구조 혹은 어떠한 모델을 사용해 특정 문맥에서 나올 단어를 예측하며 단어를 벡터로 만드는 방식
  - ① Word2vec
  - ② NNLM(Neural Network Language Model)
  - ③ RNNLM(Recurrent Neural Network Language Model)

예측 방법 예시: Word2vec

\* Word2vec의 두 가지 모델  
: CBOW, Skip-Gram

- CBOW(Continuous Bag of Words): 문맥 내 주변 단어를 통해 단어를 예측하는 방법

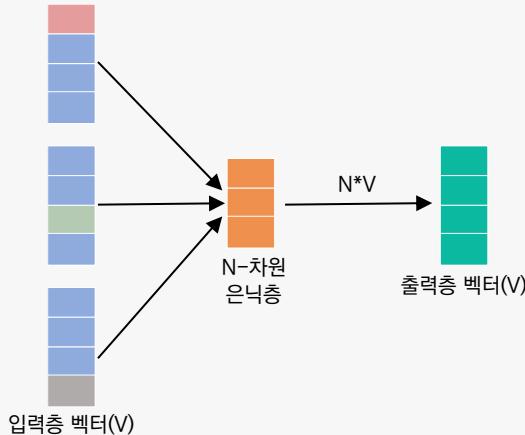
창욱은 냉장고에서 \_\_\_\_\_ 꺼내서 먹었다.

- Skip-Gram: 단어를 가지고 특정 문맥 내 주변 단어를 예측하는 방법

\_\_\_\_\_ 음식을 \_\_\_\_\_

# 01. 단어 표현

## CBOW 모델 학습 방법

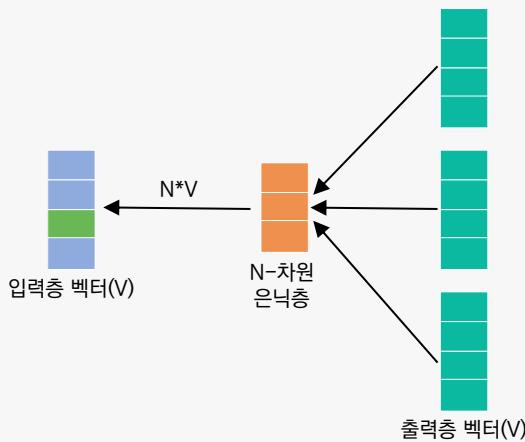


- ① 주변 단어를 원-핫 벡터로 만들어 입력값으로 사용 (입력층 벡터)
- ② 가중치 행렬(weight matrix)을 각각의 원-핫 벡터에 곱하여 n-차원 벡터를 생성 (N-차원 은닉층)
- ③ n-차원 벡터를 모두 더한 후, 개수로 나누어 평균 n-차원 벡터를 생성 (출력층 벡터)
- ④ n-차원 벡터에 다시 가중치 행렬을 곱하여 원-핫 벡터와 같은 차원의 벡터로 만듦
- ⑤ 만들어진 벡터를 실제 예측하려는 단어의 원-핫 벡터와 비교하여 학습  
+ 학습 종료 후, 가중치 행렬의 각 행을 단어 벡터로 사용

### \* Word2vec 두 모델의 차이점

CBOW: 입력값으로 여러 개 단어 사용, 학습을 위해 하나의 단어와 비교  
Skip-Gram: 입력값으로 하나의 단어를 사용, 학습을 위해 주변의 여러 단어와 비교

## Skip-Gram 모델 학습 방법



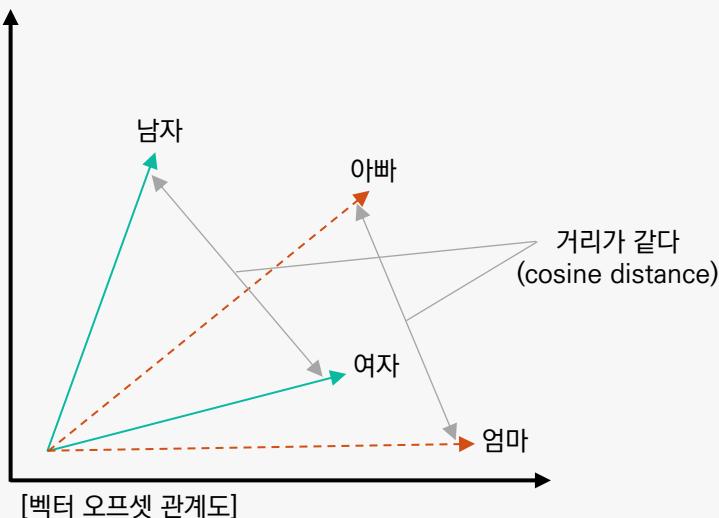
- ① 하나의 단어를 원-핫 벡터로 만들어 입력값으로 사용 (입력층 벡터)
- ② 가중치 행렬을 원-핫 벡터에 곱하여 n-차원 벡터를 생성 (N-차원 은닉층)
- ③ n-차원 벡터에 다시 가중치 행렬을 곱하여 원-핫 벡터와 같은 차원의 벡터로 만듦 (출력층 벡터)
- ④ 만들어진 벡터를 실제 예측하려는 주변 단어 각각의 원-핫 벡터와 비교하여 학습  
+ 학습 종료 후, 가중치 행렬의 각 행을 단어 벡터로 사용

# 01. 단어 표현

## Word2vec 정리

### 장점

- ① 카운트 기반 방법으로 만든 단어 벡터보다 단어 간의 유사도를 잘 측정함
- ② 단어들의 복잡한 특징까지도 잘 잡아냄
- ③ Word2vec 방식으로 만들어진 단어 벡터는 서로에게 유의미한 관계를 측정할 수 있음



### \* CBOW vs Skip-Gram

- : Skip-Gram 성능이 좋아 일반적인 경우 Skip-Gram을 사용함
- : 단, 절대적인 것은 아님

### \* 추가\_ 카운트 기반 방법 vs 예측 기반 방법

- : 보통의 경우 예측 기반 방법 성능이 좋아서 주로 예측 기반 방법을 사용함
- : 두 가지 방법을 모두 포함하는 “Glove”라는 단어 표현 방법도 자주 사용됨

## 02. 텍스트 분류

### 텍스트 분류(Text Classification)

: 자연어 처리 기술을 활용하여 특정 텍스트가 어느 범주(Class)에 속하는지 분류하는 문제

- ① 이진 분류(Binary classification) 문제 – 2가지 범주에 대해 구분하는 문제
- ② 다중 범주 분류(Multi class classification) 문제 – 3개 이상의 범주로 분류하는 문제

### 텍스트 분류 예시

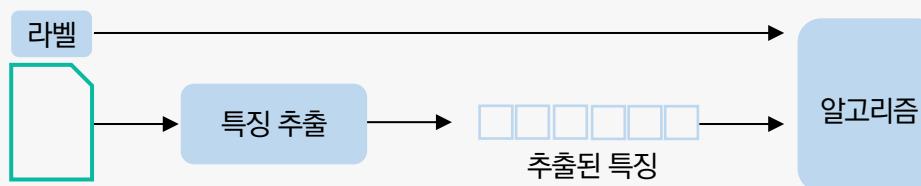
- 스팸 분류: 일반 메일과 스팸 메일을 분류하는 문제
- 감정 분류: 주어진 글이 긍정적인지 부정적인지 판단하는 문제, 필요에 따라 범주 추가 가능
- 뉴스 기사 분류: 스포츠, 경제, 사회, 연예 등 다양한 주제의 기사를 각 주제에 맞게 분류해야 함
- 품사 분류(POS tagging): 각 단어를 기준으로 어떤 품사를 가지는지 분류

## 02. 텍스트 분류

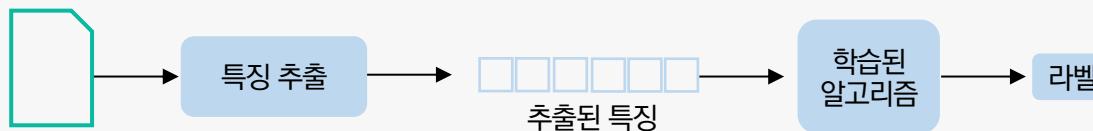
### 지도 학습을 통한 텍스트 분류

- : 지도 학습에는 글(데이터)가 속한 범주에 대한 값(라벨)이 주어져 있음
- : 주어진 범주로 글을 모두 학습하고, 학습한 결과를 이용하여 새로운 글의 범주를 예측하는 방법

[학습과정]



[예측과정]



#### 대표적인 지도학습 방식

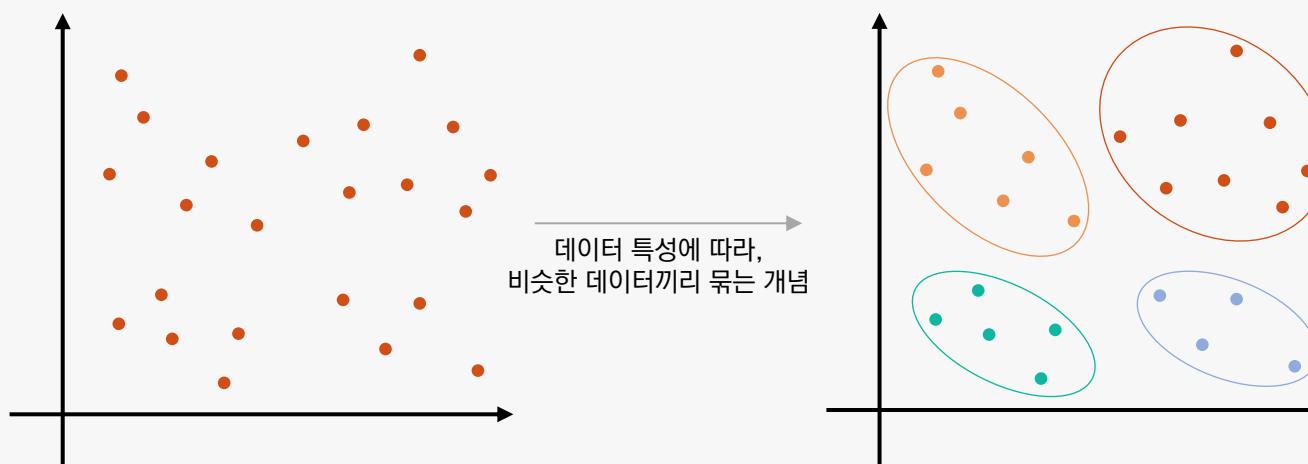
- 나이브 베이즈 분류(Naïve Bayes Classifier)
- 서포트 벡터 머신(Support Vector Machine)
- 신경망(Neural Network)
- 선형 분류(Linear Classifier)
- 로지스틱 분류(Logistic Classifier)
- 랜덤 포레스트(Random Forest)

## 02. 텍스트 분류

### 비지도 학습을 통한 텍스트 분류

- : 각 데이터가 범주로 나누어지지 않은 상태로 존재
- : 특성을 찾아내어 적당한 범주를 만들고, 각 데이터를 나누어야 함

k-평균 군집화(K-means clustering) 예시



#### 대표적인 비지도학습 방식

- K-평균 군집화(K-means Clustering)
- 계층적 군집화(Hierarchical Clustering)

# 03. 텍스트 유사도

## 텍스트 유사도(Text Similarity)

- : 텍스트가 얼마나 유사한지 표현하는 방식
- : 단어의 개수, 형태소, 자소 등 다양한 방식으로 텍스트 유사도 측정이 가능함
- : 본 수업에서는 딥러닝 기반 텍스트 유사도 측정 방식을 살펴볼 것

### 딥러닝 기반 텍스트 유사도 측정 방식

- : 단어, 형태소, 유사도 종류에 상관 없이 텍스트를 벡터화한 후 벡터화된 각 문장 간의 유사도를 측정하는 방식
- : 자카드 유사도, 유클리디언 유사도, 맨하탄 유사도, 코사인 유사도

#### \* 유사도 측정 방식

- : 자카드 유사도는 텍스트 벡터화 없이 유사도 측정 가능
- : 자카드 유사도를 제외한 방식은 텍스트 벡터화 필요

## 03. 텍스트 유사도

### 자카드 유사도(Jaccard Similarity, 자카드 지수)

- : 두 문장을 각각 단어의 집합으로 만든 뒤, 두 집합을 통해 유사도를 측정하는 방식
- : 두 집합의 교집합인 공통된 단어 개수를 두 집합의 합집합(전체 단어 수)으로 나누어 측정
- : 결괏값은 공통 원소 개수에 따라 0과 1 사이 값이 나오며, 1에 가까울수록 유사도가 높음을 의미함

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\text{token in } A \cap \text{token in } B}{\text{token in } A \cup \text{token in } B}$$

\* A, B: 각 문장 / token: 단어

# 03. 텍스트 유사도

## 코사인 유사도

- : 두 개의 벡터값에서 코사인 각도를 구하는 방법
- : -1과 1 사이 값을 가지고 1에 가까울수록 두 문장이 유사함을 의미
- : 방향성 개념이 더해지는 계산법으로 다른 유사도 접근법에 비해 성능이 좋은 편

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

# 03. 텍스트 유사도

## 유클리디언 유사도

- : 거리를 측정하는 가장 기본적인 유사도 공식
- : n차원 공간에서 두 점 사이의 최단 거리를 구하는 접근법
- : 유클리디언 유사도로 구하는 거리 – 유클리디언 거리(Euclidean Distance) or L2 거리

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

### \* 유클리디언 유사도

: 단순히 두 점 사이 거리를 구하는 것이기 때문에  
타 유사도와 다르게 값에 제한이 없음  
→ 정규화하여 0과 1사이 값을 갖도록 만들어줌

# 03. 텍스트 유사도

## 맨하탄 유사도(Manhattan Similarity)

- : 맨하탄 거리를 통해 유사도를 측정하는 방법
- : 맨하탄 거리- 사각형 격자로 이뤄진 지도에서 출발점부터 도착점까지의 최단 거리를 구하는 공식 (가로지르기 X)
- : L2 거리라고 불림

$$\text{MaDistance} = \sum_{i=1}^n |a_i - b_i|$$

\* 맨하탄 유사도  
: 거리를 통해 유사도를 측정하는 방법으로,  
값이 계속하여 커질 수 있음  
→ 정규화하여 0과 1사이 값을 갖도록 만들어줌

## 04. 자연어 생성

### 사람이 생성하는 언어

- : 어떠한 주제에 관해 목적 의식을 가지고 문법에 맞추고 적합한 단어를 사용하여 문장을 생성
- : 사람은 언어를 활용하여 소통(Communication)하며 살아감

### 컴퓨터가 생성하는 언어

- : 사실 기반의 기사를 제외하고 감정 및 논리력이 들어가는 글을 작성하는 데는 한계가 있음
- : 컴퓨터는 0과 1로 이루어져 있어서 숫자로 정량화하기 어려운 내용은 작성하기 어려움

### 컴퓨터 ‘언어 생성’의 목적

- : 사람의 대화를 수집하여 배우게 하고 지속적으로 평가하는 과정을 반복하여 특정 목적에 맞는 텍스트 생성

# 05. 기계 이해

## 기계 이해(Machine Comprehension)

: 기계가 텍스트를 통해 정보를 학습한 뒤, 사용자가 관련 질문을 던졌을 때 응답하는 문제  
→ 기계가 논리적 추론을 할 수 있는지 확인하는 작업

### 기계 이해 예시

#### - 텍스트

자연어 처리 또는 자연 언어 처리는 인간의 언어 현상을 컴퓨터와 같은 기계를 이용하여 묘사할 수 있도록 연구하고 이를 구현하는 인공지능 주요 분야 중 하나이다. 자연 언어 처리는 연구 대상이 먼저이기 때문에 언어 자체를 연구하는 언어학과 언어 현상의 내적 기재를 탐구하는 언어 인지 과학과 연관이 있다. 구현을 위해 수학적 통계적 도구를 많이 활용하며 특히 기계학습 도구를 많이 사용하는 대표적인 분야이다. 정보검색, QA 시스템, 문서 자동 분류, 신문기사 클러스터링, 대화형 Agent 등 다양한 응용이 이뤄지고 있다.

#### - 질문

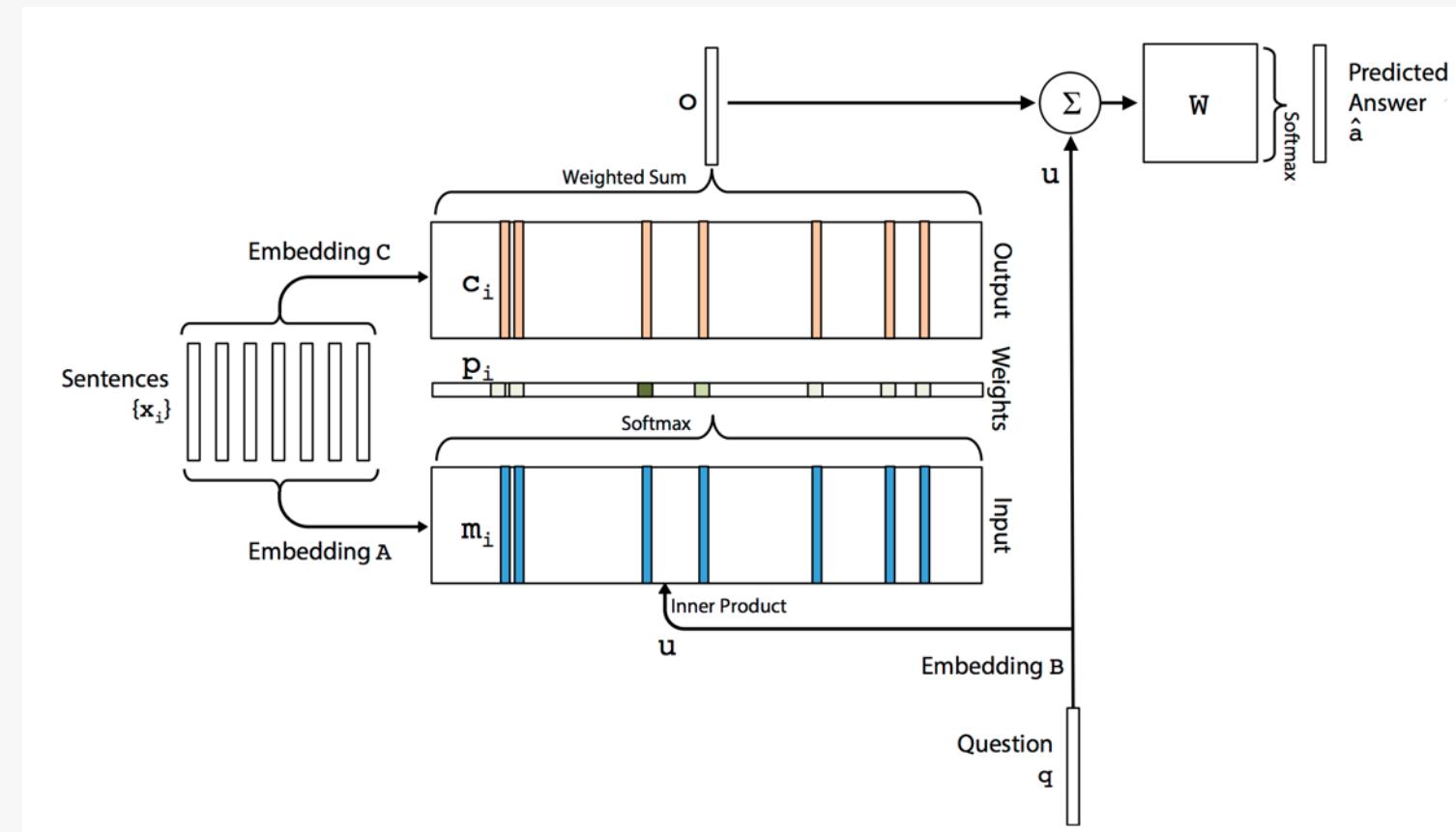
자연어 처리는 어느 분야 중 하나인가?

#### \* 기계 이해 수행 과정

1. 텍스트와 질의에 관한 정보를 알기 위해 각 문장 의미 벡터를 추출할 수 있어야 함  
예시의 경우, 단어 표현 벡터를 활용하여 문장의 의미를 표현하는 벡터를 추출할 수 있음
  2. 텍스트와 질의 문장 정보의 관계를 파악하기 위해 문장 유사도를 구하는 방식이 필요할 수 있음
- + 경우에 따라서는, 질의 내용에 관한 정보 기반으로 텍스트에 관한 언어 정보를 새롭게 생성할 수도 있음  
→ 언어 생성 모델과 밀접한 관련이 있음

# 05. 기계 이해

## 메모리 네트워크(Memory Network)



▲ 메모리 네트워크 도식화 / 출처: <https://arxiv.org/abs/1503.08895>

### \* 메모리 네트워크 개념

입력값으로 질의를 입력했을 때, 텍스트 정보를 통해 어떻게 답할 것인지를 판단

- ① 정보를 주기 위한 문장으로 문장 표현 벡터 생성 (Question  $q$ , Sentences( $x$ ))
- ② 문장 벡터를 모아 두 개의 행렬로 만들고, 만들어진 행렬로 질의와의 유사도를 측정
- ③ 유사도 높은 정보에 가중치를 주어 출력 (출력값: Embedding  $c$ )

# 05. 기계 이해

## 데이터셋

\* 데이터셋

: 위키피디아나 뉴스 기사 기반으로 데이터를 구성하며  
대부분 텍스트와 지문, 정답 형태

- : 기계 이해 테스크에서는 자연 언어를 이해하는 과제에서 기계가 텍스트 내용 추론을 잘하는지 파악하기 위해 학습 수행
- : QA(Question Answering) 형태의 데이터셋을 활용하여 기계에 학습하게 함

## bAbI

- 페이스북 AI 연구팀에서 생성한 데이터셋
- 기계가 데이터를 학습하여 텍스트를 이해하고 추론하는 목적
- 시간 순서로 나열된 텍스트 문장 정보와 이에 관한 질문으로 구성
- 문장 내용을 알 수 있고 논리적인 관계를 파악할 수 있도록 구성

### Task 1: Single Supporting Fact

Mary went to the bathroom.  
John moved to the hallway.  
Mary travelled to the office.  
Where is Mary? A:office

### Task 2: Two Supporting Facts

John is in the playground.  
John picked up the football.  
Bob went to the kitchen.  
Where is the football? A:playground

### Task 3: Three Supporting Facts

John picked up the apple.  
John went to the office.  
John went to the kitchen.  
John dropped the apple.  
Where was the apple before the kitchen? A:office

### Task 4: Two Argument Relations

The office is north of the bedroom.  
The bedroom is north of the bathroom.  
The kitchen is west of the garden.  
What is north of the bedroom? A: office  
What is the bedroom north of? A: bathroom

### Task 5: Three Argument Relations

Mary gave the cake to Fred.  
Fred gave the cake to Bill.  
Jeff was given the milk by Bill.  
Who gave the cake to Fred? A: Mary  
Who did Fred give the cake to? A: Bill

### Task 6: Yes/No Questions

John moved to the playground.  
Daniel went to the bathroom.  
John went back to the hallway.  
Is John in the playground? A:no  
Is Daniel in the bathroom? A:yes

### Task 7: Counting

Daniel picked up the football.  
Daniel dropped the football.  
Daniel got the milk.  
Daniel took the apple.  
How many objects is Daniel holding? A: two

### Task 8: Lists/Sets

Daniel picks up the football.  
Daniel drops the newspaper.  
Daniel picks up the milk.  
John took the apple.  
What is Daniel holding? milk, football

### Task 9: Simple Negation

Sandra travelled to the office.  
Fred is no longer in the office.  
Is Fred in the office? A:no  
Is Sandra in the office? A:yes

### Task 10: Indefinite Knowledge

John is either in the classroom or the playground.  
Sandra is in the garden.  
Is John in the classroom? A:maybe  
Is John in the office? A:no

▲ bAbI 데이터셋 예시 / 출처: <https://arxiv.org/abs/1502.05698>

# 05. 기계 이해

## 데이터셋

### SQuAD(Stanford Question Answering Dataset)

- 스텐퍼트 자연어 처리 연구실에서 만든 데이터셋
- 위키피디아의 내용을 크라우드 소싱하여 QA 데이터셋으로 생성

- SQuAD
  - : 46개의 주제에 관해 약 10만 개 질문 데이터셋으로 구성
  - : 시간, 장소, 이유 등 다양한 형태의 질문이 있음
  - : 10만 개의 어휘와 다양한 길이의 지문을 포함

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?  
**gravity**

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?  
**graupel**

Where do water droplets collide with ice crystals to form precipitation?  
**within a cloud**

- 그림에서 “precipitation”에 관한 설명을 볼 수 있음
- “What causes precipitation to fall”과 같은 질문이 제시되며, 질문의 답으로 텍스트 내의 단어를 선택하게 하는 데이터 구성

데이터 구성 특성에 따라 모델링에서는,

① 정답 선택 시

텍스트 토큰 위치의 시작점과 끝점을 지정하도록 학습

② 모델 학습 평가 시

정답 위치와 완벽하게 일치하는지를 보는 EM(Exact Matching) 점수와

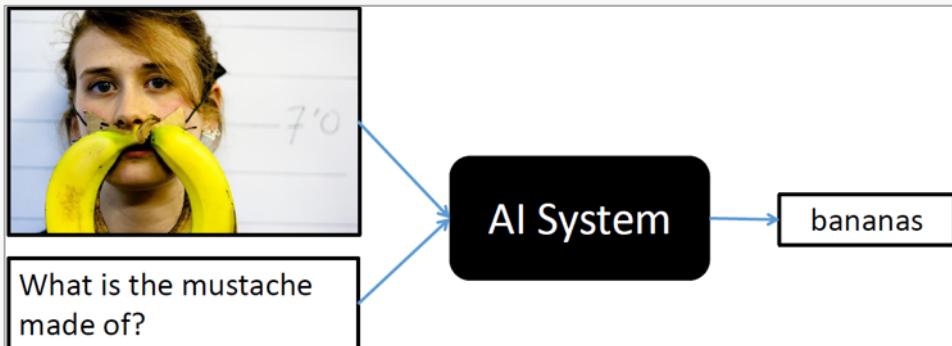
정답 위치와 겹치는지를 보는 F1 점수 사용

▲ SQuAD 데이터 / 출처: <https://arxiv.org/abs/1606.05250>

# 05. 기계 이해

## Visual Question Answering(VQA)

: 이미지 정보와 텍스트 질의를 통해 이미지 컨텍스트에 해당하는 응답을 알려주는 태스크



▲ VAQ 진행 구조 / 출처: <https://visualqa.org/challenge.html>

- 바나나를 입 주변에 둔 사진과 “What is the mustache made of?” 질의 제시
- VQA에서는 이 두 정보를 AI System을 통해 bananas라는 답을 얻도록 학습해야 함

실제 VQA 모델은 두 가지 모델이 합쳐진 형태

① Image Embedding (VGGNet) 모델

이미지 정보를 추출하는 모델

② Question Embedding (LSTM) 모델

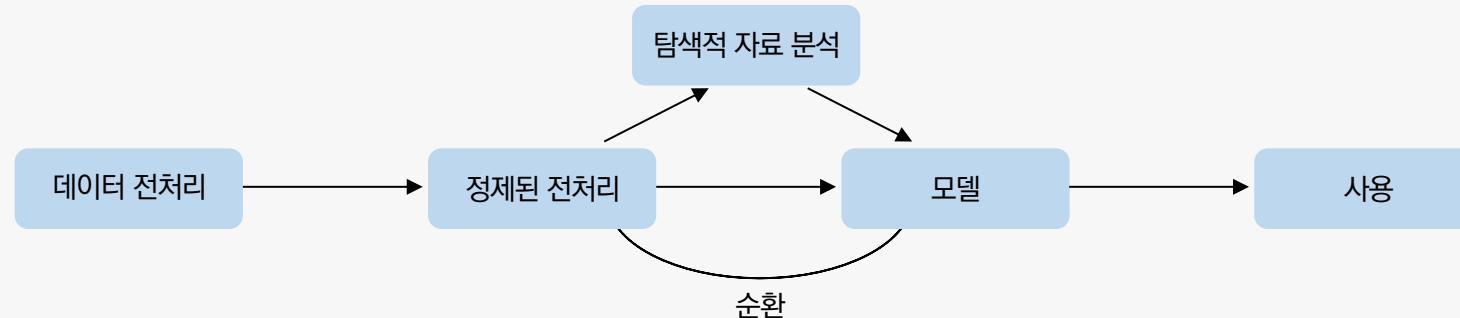
질문 텍스트에 관한 정보를 추출하는 모델

# 06. 데이터 이해하기

## 탐색적 데이터 분석(EDA; Exploratory Data Analysis)

- : 데이터를 이해하는 과정, 데이터의 패턴이나 잠재적인 문제점 등을 발견할 수 있음
- : 정해진 틀 없이 데이터에 관한 최대한 많은 정보를 추출: 평균값, 중앙값, 최솟값, 최댓값, 범위, 분포, 이상치 등

## 탐색적 데이터 분석 과정



# 자연어 처리 텍스트 분류

with  python™

# 00. 텍스트 분류

## 텍스트 분류

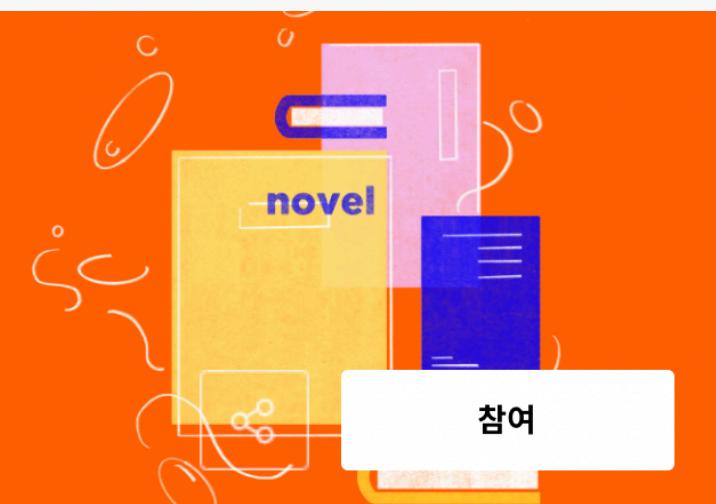
: 자연어 처리 기술을 활용해 글의 정보를 추출하고 문제에 맞게 사람이 정한 범주(Class)로 분류하는 문제

- 영어 텍스트 분류
- 한글 텍스트 분류

**소설 작가 분류 AI 경진대회**  
월간 데이콘 9 | 소설 문체 | NLP | Logloss

⭐ 상금 : 100만원+애플워치  
⌚ 2020.10.29 ~ 2020.12.04 17:59 [+ Google Calendar](#)

👤 779팀 📄 마감

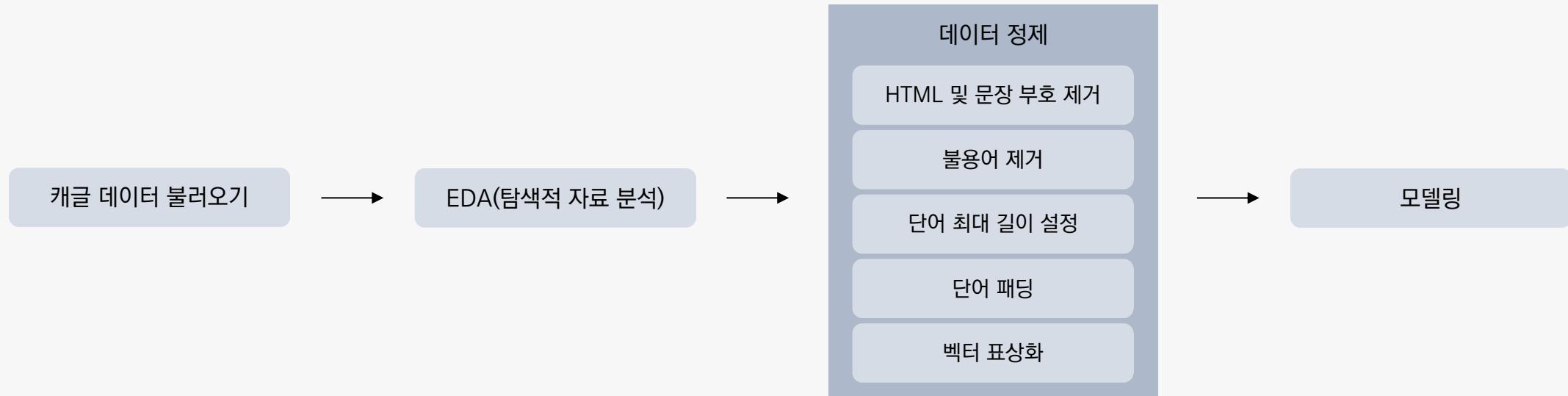


**참여**

출처: <https://dacon.io/competitions/official/235670/data/>

# 01. 데이터 분석 및 전처리

## 데이터 처리 과정

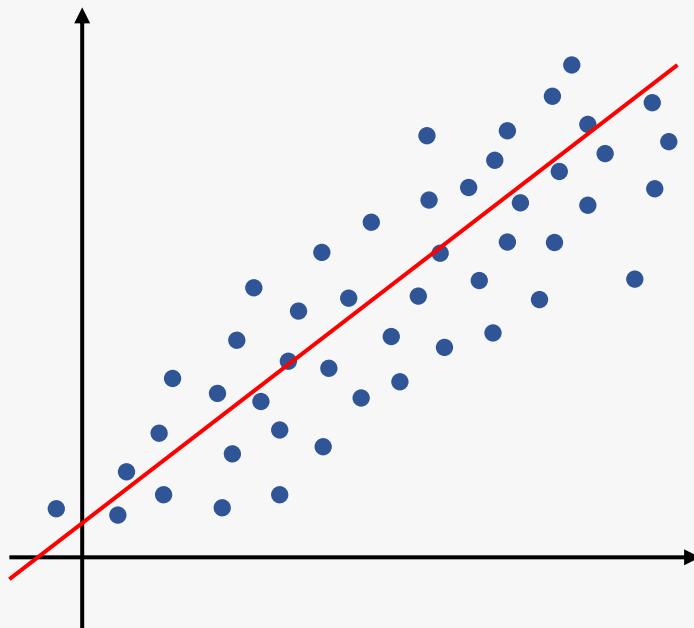


# 02. 모델링

## 회귀 모델

### ① 선형 회귀 모델

- 종속변수와 독립변수 간의 상관관계를 모델링하는 방법
- 하나의 선형 방정식으로 표현해 예측할 데이터를 분류하는 모델



$$y = w_1x_1 + w_2x_2 + \dots + b$$

- $w_1, w_2, b$ : 학습하고자 하는 파라미터
- $x_1, x_2$ : 입력값, 벡터  $x$ 로 다루며 주로 단어 또는 문장 표현 벡터를 입력

\* 모델에 입력하는 값  
:  $x_1, x_2$  변수에 넣음

# 02. 모델링

## 회귀 모델

### ② 로지스틱 회귀 모델

- 선형 모델의 결괏값에 로지스틱 함수를 적용해 0 – 1 사이의 값을 갖게 하여 확률로 표현
- 결과가 1에 가까우면 정답이 1이라고 예측하고 결과가 0에 가까우면 정답이 0이라고 예측

#### TF-IDF

: 사이킷런의 TfidfVectorizer 사용

→ 입력값이 텍스트로 이루어진 데이터 형태여야 함

#### word2vec

: 각 단어를 word2vec으로 벡터화

: 단어로 표현된 리스트를 입력값으로 사용

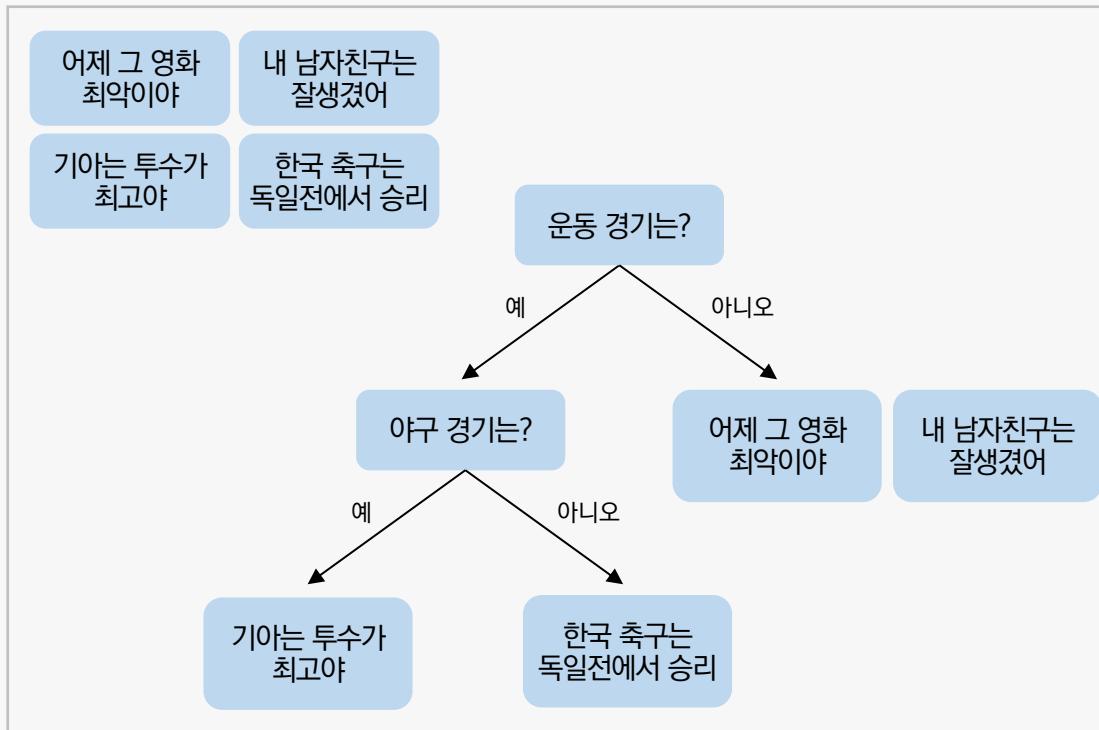
→ 텍스트 데이터를 불러온 후 각 단어의 리스트로 나누어야 함

# 02. 모델링

## 회귀 모델 ② 랜덤 포레스트 분류 모델

### 의사결정 트리

- 자료구조 중 하나인 트리 구조와 같은 형태로 이루어진 알고리즘
- 각 노드는 하나의 질문이 되며, 질문에 따라 다음 노드가 달라지며 결과를 얻게 되는 형태

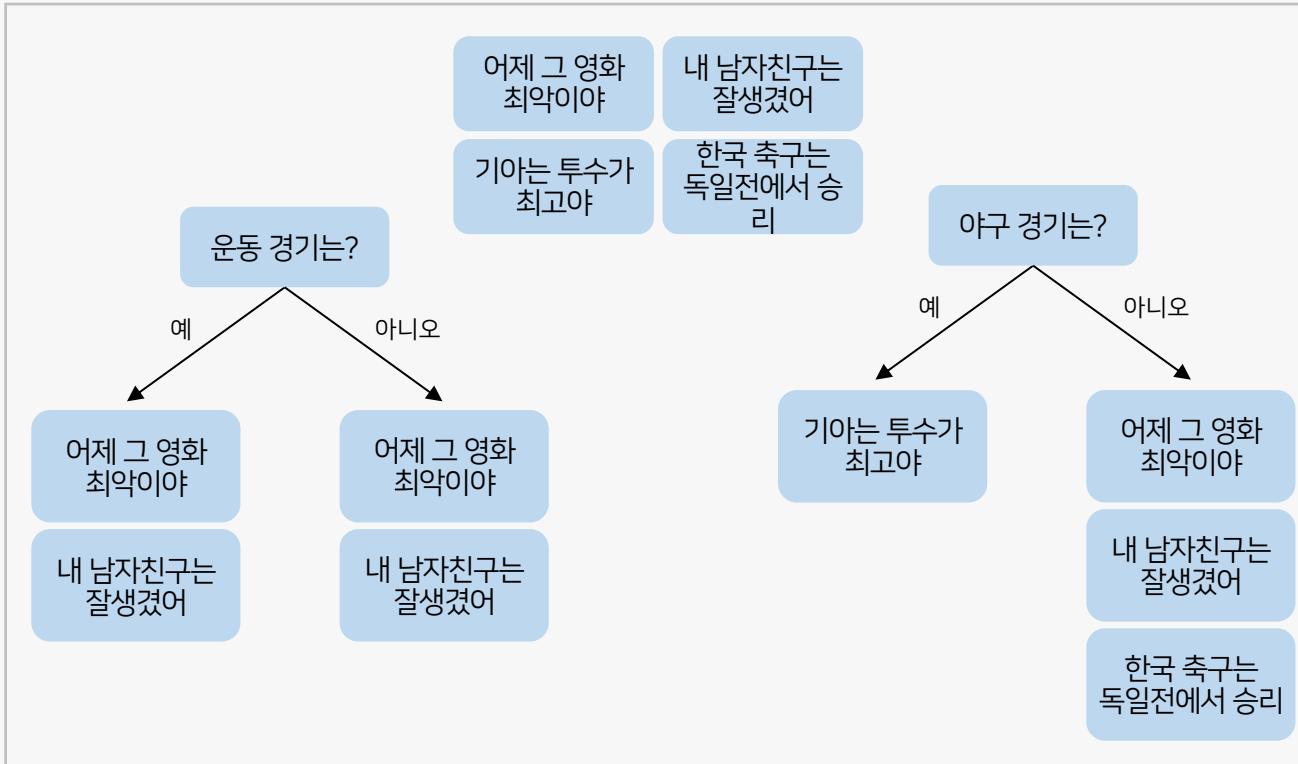


# 02. 모델링

## 회귀 모델 ② 랜덤 포레스트 분류 모델

### 랜덤 포레스트

- 트리 구조가 모인 형태로, 각 트리에서 구한 결과를 종합하여 결과를 도출함



- 두 개의 결괏값을 합쳐 야구와 관련된 문장이 하나임을 확인
- 실제 모델에서는 많은 의사결정 트리가 만들어지고 각 결과 평균으로 최종 결과를 구함

# 02. 모델링

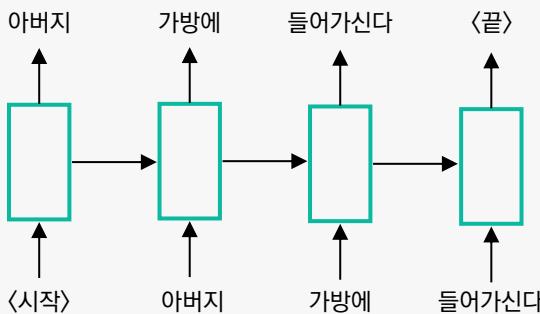
## 회귀 모델

### ③ 순환 신경망(RNN) 분류 모델

\* 순환 신경망 분류 모델

: 언어 모델에서 많이 쓰이는 딥러닝 모델

- 문장 데이터를 입력하고 문장 흐름에서 패턴을 찾아 분류하는 모델
- 주어진 단어 벡터로 모델을 학습하지 않고, 텍스트 정보를 입력하여 문장에 대한 특징 정보를 추출



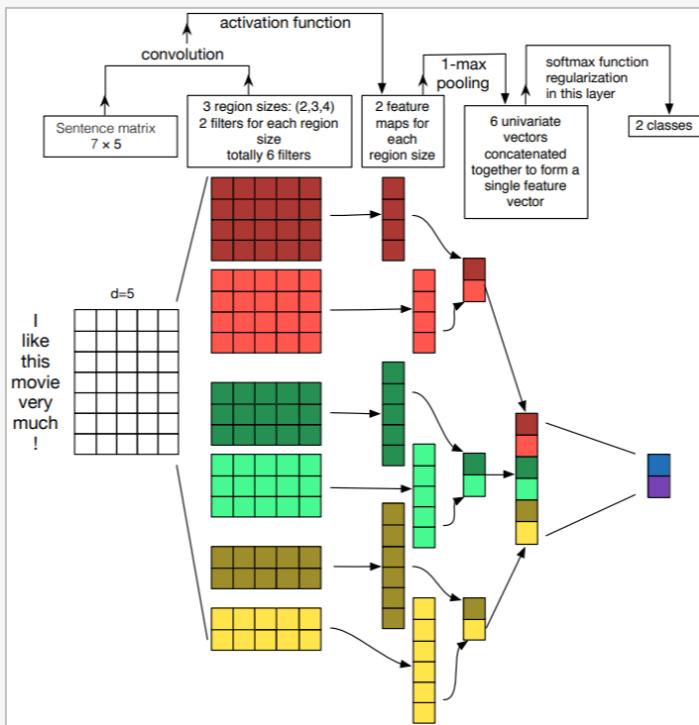
- 한 단어에 대한 정보를 입력하면 다음에 나올 단어를 맞추는 모델
- 앞에 입력한 모든 정보를 활용해 다음 단어를 예측
- 현재 정보: 입력 상태(input state)
- 이전 정보: 은닉 상태(hidden state)

# 02. 모델링

## 회귀 모델

### ④ 합성곱 신경망(CNN) 분류 모델

- 딥러닝의 부흥을 이끈 핵심 알고리즘 모델로, 전통적 신경망 앞에 여러 계층의 합성곱(convolution) 계층을 쌓은 모델
- 입력받은 이미지에 대한 가장 좋은 특징을 만들도록 학습하고, 추출된 특징을 활용해 이미지를 분류하는 방식



▲ CNN을 활용한 텍스트 분류 과정 / 출처: <https://arxiv.org/abs/1510.03820>

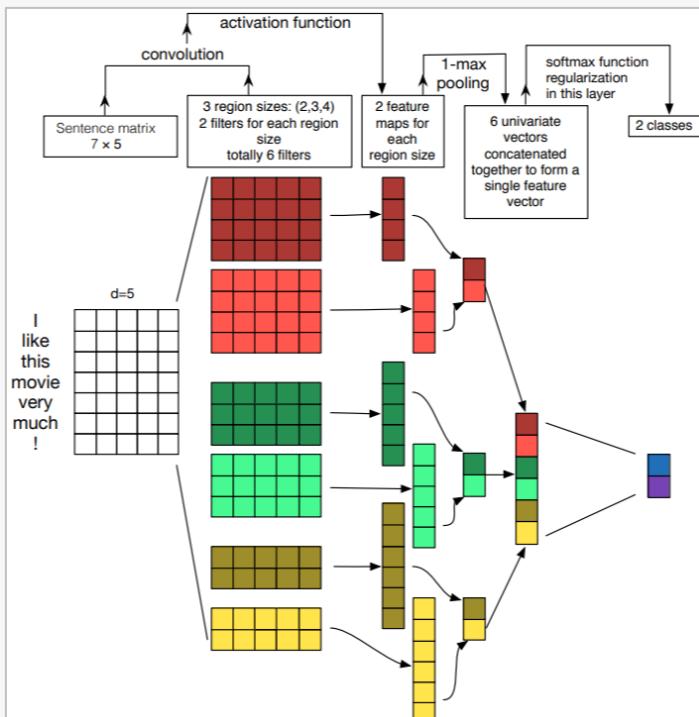
- 문장의 지역 정보를 보존하면서 각 문장 선분의 등장 정보를 학습에 반영하는 구조로 풀어감
- 학습 시, 각 필터 크기를 조절하면서 언어의 특징 값을 추출하는데 기존의 n-gram 방식과 유사함
- “I like this movie very much!”
  - !를 포함한 총 7개의 단어, 각 5차원( $d=5$ )을 보유
  - 2, 3, 4개 단어 필터를 추출해 피쳐 맵을 생성하고 맥스 풀링을 수행한 후, 각 값의 무언을 의미하는지 추출

# 02. 모델링

## 회귀 모델

### ④ 합성곱 신경망(CNN) 분류 모델

- 딥러닝의 부흥을 이끈 핵심 알고리즘 모델로, 전통적 신경망 앞에 여러 계층의 합성곱(convolution) 계층을 쌓은 모델
- 입력받은 이미지에 대한 가장 좋은 특징을 만들도록 학습하고, 추출된 특징을 활용해 이미지를 분류하는 방식



▲ CNN을 활용한 텍스트 분류 과정 / 출처: <https://arxiv.org/abs/1510.03820>

### 모델 구현을 위한 컨볼루션층 구조 도식화

|                 |                            |
|-----------------|----------------------------|
| Input           | [배치크기, 문장길이]               |
| Word Embedding  | [배치크기, 문장길이, 임베딩크기]        |
| Dropout         |                            |
| Convolution     | [배치크기, 컨볼루션값, 필터크기, 커널사이즈] |
| Max-pooling     | [배치크기, 컨볼루션값, 필터크기]        |
| Fully-connected | [배치크기, 컨볼루션값, 필터크기]        |
| Dropout         |                            |
| Dense Layer     | [배치크기, 로짓값(예측 확률값)]        |

# 자연어 처리 : 이론심화학습

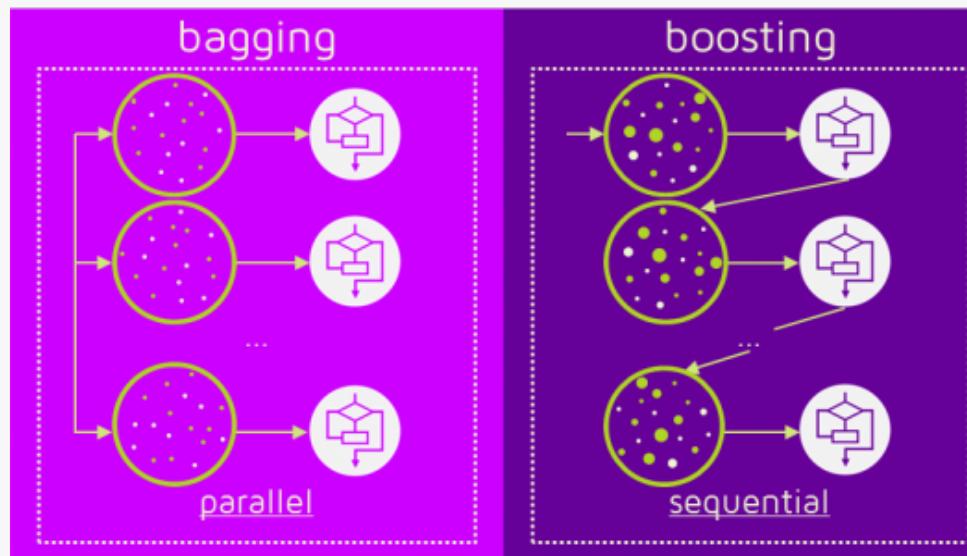
with  python™

# 01. 모델링

## 머신러닝 기반 모델

### 배깅 vs 부스팅

- 배깅(Bagging): 여러 개의 학습 알고리즘/모델을 통해 각각의 결과를 예측한 뒤, 모든 결과를 동등하게 보고 취합하여 결과를 얻는 방식
- 부스팅(Boosting): 각각의 결과를 순차적으로 취합하면서, 잘못 예측한 부분에 가중치를 주어 다시 학습시키는 방식



▲ 양상별 기법 / 출처: <https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/>

# 02. 머신러닝 모델링

## 머신러닝 기반 모델

### ① XGBoost 모델

- eXtreme Gradient Boosting의 약자
- 부스팅(Boosting)기법 중, 트리 부스팅(Tree Boosting) 기법을 활용한 모델
- Tree Boost 기법
  - : 여러 개의 의사결정 트리를 사용하지만, 단순히 결과를 평균 내는 것이 아니라 결과를 보고 오답에 가중치를 부여하는 방식
  - : 가중치가 적용된 오답에는 관심을 가지고 정답이 될 수 있도록 결과를 만든 후, 해당 결과에 대한 다른 오답을 찾아 작업을 진행하는 방식
- XGBoost
  - : 트리 부스팅 방식에 경사 경사하강법을 통해 최적화하는 방법
  - : 연산량을 줄이기 위해 의사결정 트리 구성 시, 병렬 처리를 사용하여 빠른 학습이 가능함

# 02. 딥러닝 모델링

## 딥러닝 기반 모델

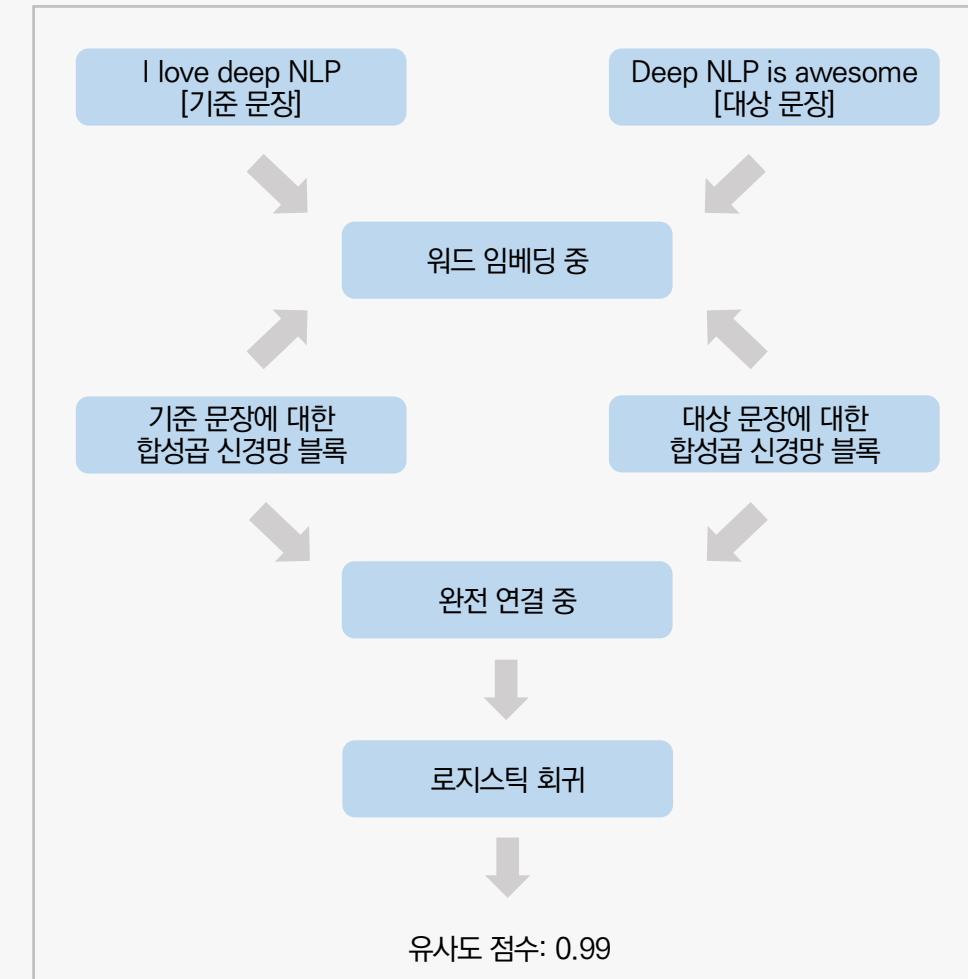
### ① CNN 텍스트 유사도 분석 모델

- 합성곱 신경망 구조를 활용한 딥러닝 모델
- 문장에 대한 의미 벡터를 추출해서 그 벡터에 대한 유사도를 측정

#### \* CNN 텍스트 유사도 분석

기준 문장: 기준이 되는 문장  
대상 문장: 기준 문장에 대해 비교해야 하는 문장

- 문장이 의미적으로 가까울수록 유사도 점수 높아짐

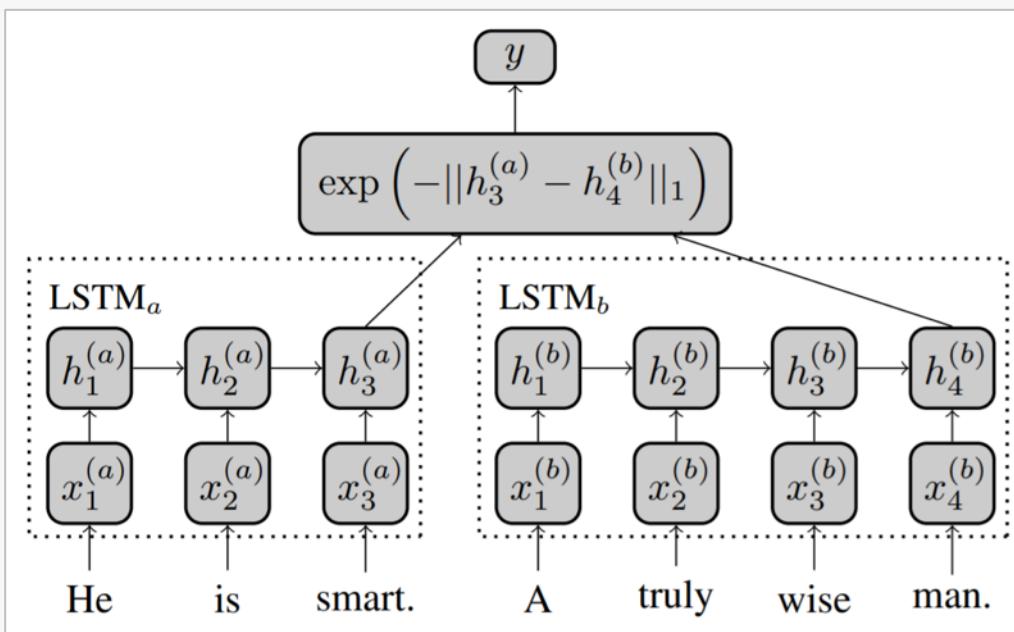


# 02. 딥러닝 모델링

## 딥러닝 기반 모델

### ② MaLSTM

- 순환 신경망 기반의 모델: 문장의 시퀀스 형태로 모델을 학습
- 2016년 MIT 조나스 뮐러가 쓴 논문(Siamese Recurrent Architectures for Learning Sentence Similarity)에서 처음 소개



▲ MaLSTM 모델 구조 / 출처:  
[https://people.csail.mit.edu/jonasmueller/info/MuellerThyagarajan\\_AAAI16.pdf](https://people.csail.mit.edu/jonasmueller/info/MuellerThyagarajan_AAAI16.pdf)

#### \* MaLSTM 모델

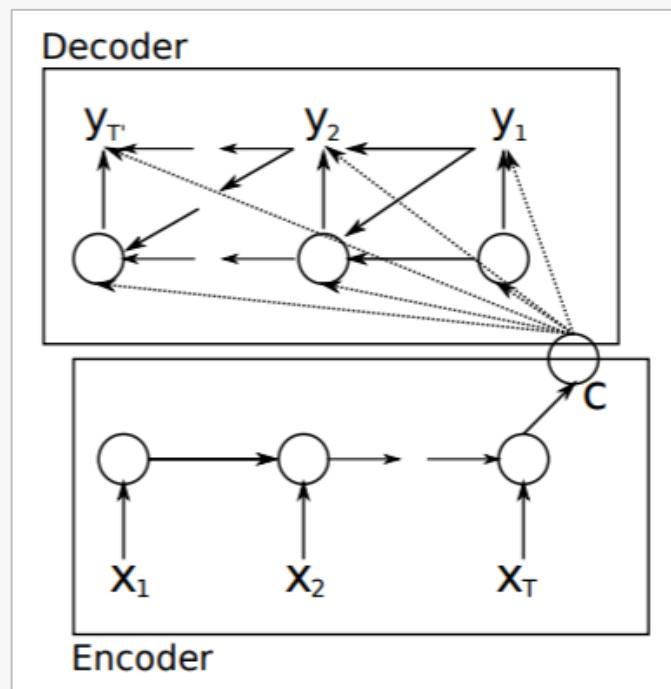
- : 순환 신경망 기반의 LSTM 층을 적용해 각 문장의 의미 벡터를 뽑음
- : 의미 벡터- 각 LSTM 마지막 스텝인  $LSTM_a h_3^{(a)}$ ,  $LSTM_b h_3^{(b)}$  값이 은닉 상태 벡터로 사용됨
  - 문장의 모든 단어 정보가 반영된 값으로 전체 문장을 대표하는 벡터가 되고, 두 벡터의 맨하탄 거리를 계산하여 두 문장 사이의 유사도를 측정
  - 계산된 유사도를 실제 라벨과 비교하여 학습

# 03. 시퀀스 투 시퀀스 모델

## 시퀀스 투 시퀀스(Sequence to Sequence) 모델

: 시퀀스 형태의 입력값을 시퀀스 형태의 출력으로 만들 수 있게 하는 모델

- 기계 번역 분야를 비롯한, 텍스트 요약, 이미지 설명, 대화 모델 등 다양한 분야에서 활용



▲ 시퀀스 투 시퀀스 모델 구조 / 출처:  
<https://arxiv.org/pdf/1406.1078.pdf>

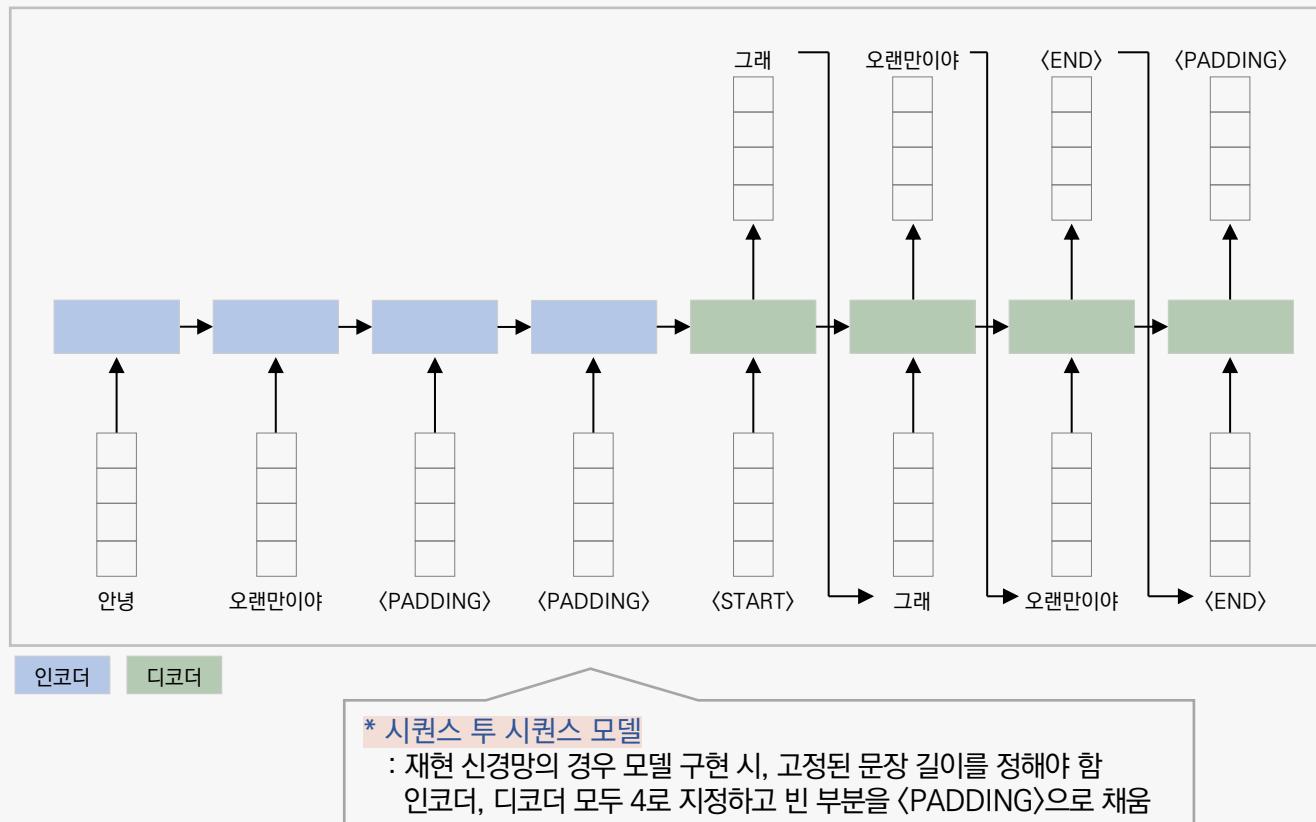
### \* 시퀀스 투 시퀀스 모델

: 하나의 텍스트 문장이 입력으로 들어오면,  
하나의 텍스트 문장을 출력하는 구조

- 재귀 순환 신경망(RNN) 모델을 기반으로 함
- 인코더(Encoder) 부분과 디코더(Decoder) 부분으로 나뉨
  - ① 인코더에서 입력값을 받아 입력값의 정보를 담은 벡터를 만듦
  - ② 디코더에서 만들어진 벡터를 활용해 재귀적으로 출력값을 만듦
- 그림 설명
  - : 인코더에서 재귀 순환 신경망의 스텝마다 입력값이 들어가며, 하나의 단어가 됨
  - : 인코더 부분 전체 재귀 순환 신경망의 마지막 부분에서 'c'로 표현된 하나의 벡터가 나옴
    - \* 벡터- 인코더 부분의 정보를 요약해 담고 있으며, 재귀 순환 신경망의 마지막 은닉 상태 벡터값
  - : 디코더에서는 벡터를 사용하여 새로운 재귀 순환 신경망을 시작
  - : 신경망의 각 스텝마다 하나의 출력값이 나오며, 하나의 단어가 됨
  - : 각 스텝의 출력값이 다음 스텝의 입력값으로 사용됨

# 03. 시퀀스 투 시퀀스 모델

## 시퀀스 투 시퀀스 모델을 이용한 한글 시각화

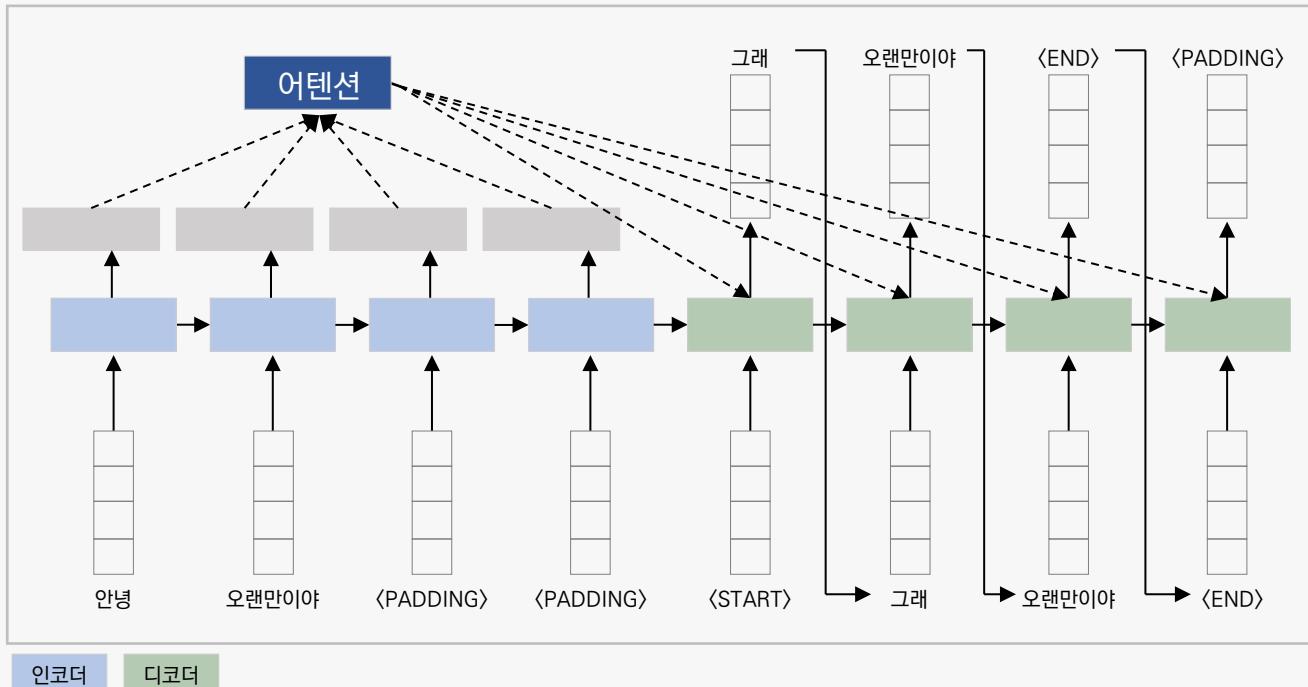


- 인코더 부분
  - : 신경망의 각 스텝마다 단어가 하나씩 들어가며, 단어는 임베딩된 벡터로 바뀐 뒤 입력값으로 사용됨
- 디코더 부분
  - : 최초 입력값으로 문장 시작을 나타내는 <START> 토큰 사용
  - : 단어가 임베딩된 벡터 형태로 입력값에 들어가고 각 스텝마다 출력
  - : 출력된 단어가 다음 스텝의 입력값으로 사용되며, <END> 토큰이 나오면 문장의 끝으로 보는 형태로 학습 진행
- 참고

| 스페셜 토큰 | 의미           |
|--------|--------------|
| <PAD>  | 의미 없는 패딩 토큰  |
| <SOS>  | 시작 토큰        |
| <END>  | 종료 토큰        |
| <UNK>  | 사전에 없는 단어 의미 |

# 03. 시퀀스 투 시퀀스 모델

## 시퀀스 투 시퀀스 모델 보완: 어텐션 방법

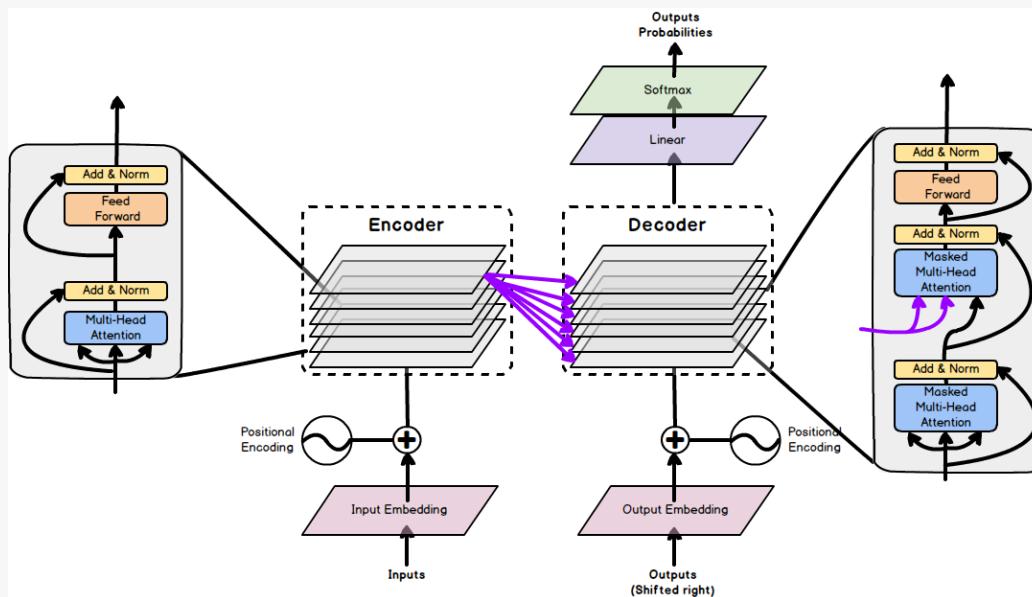


- 시퀀스 투 시퀀스의 문제점
  - : 정보를 고정된 길이에 담아야 하여 문장이 길수록 정보 손실이 있음
  - : 재귀 순환망 특유 문제인 장기 의존성 문제가 발생할 수 있음
- 어텐션 방법
  - : 은닉 상태의 값을 통해 어텐션을 계산하고
  - : 디코더의 각 시퀀스 스텝마다 계산된 어텐션을 입력으로 넣음
  - : 어텐션도 학습을 진행하게 되며, 학습을 통해
  - : 디코더의 각 시퀀스 스텝마다 어텐션의 가중치는 다르게 적용됨

# 04. 트랜스포머 모델

## 트랜스포머(Transformer) 모델

- : 2017년 구글이 소개한 논문(Attention is all you need)에 나온 모델
- : 시퀀스 투 시퀀스의 인코더와 디코더 구조를 가지고 있지만, 어텐션 구조만으로 전체 모델을 만듦
  - 인코더에서 입력한 문장에 대한 정보와 디코더에 입력한 문장 정보를 조합해서 디코더 문장 다음에 나올 단어를 생성하는 방법
  - 시퀀스 투 시퀀스와는 다르게 순환 신경망을 활용하지 않고, 셀프 어텐션 기법을 사용해 문장에 대한 정보를 추출



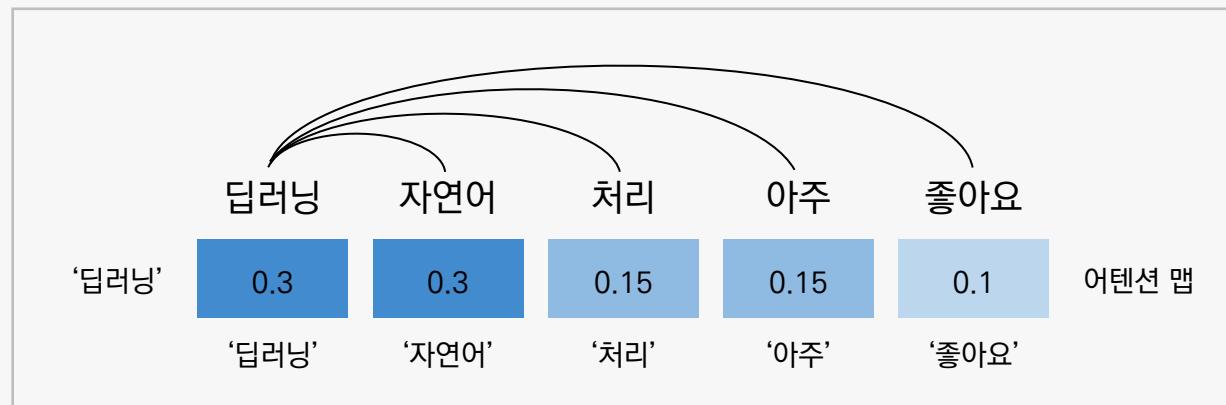
Thesis URL: <https://arxiv.org/abs/1706.03762>

# 04. 트랜스포머 모델

## 셀프 어텐션(Self-Attention)

- 문장에서 각 단어끼리 얼마나 관계가 있는지를 계산해서 반영하는 방법
- 문장 안에서 단어들 간의 관계를 측정할 수 있으며, 각 단어를 기준으로 다른 단어들과의 관계 값을 계산

\* 어텐션 스코어(attention score)



- '딥러닝'이라는 단어를 기반으로 나머지 단어와의 관계를 측정
- 순서대로 모든 단어의 스코어를 각각 구해야 함
- 어텐션 스코어: 각 단어 간의 관계를 측정한 값
- 어텐션 맵: 어텐션 스코어 값을 하나의 테이블로 만든 것

# 04. 트랜스포머 모델

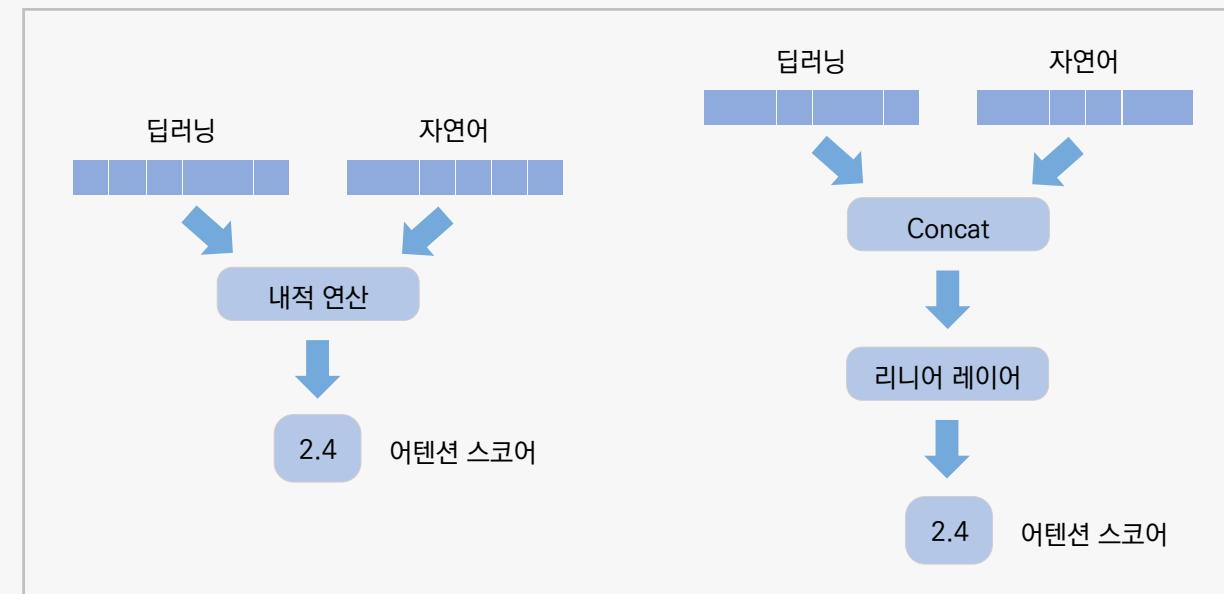
## 셀프 어텐션(Self-Attention)

① 단어 벡터로 구성된 문장



- 문장에 대한 정보가 벡터로 구성됨

② 어텐션 스코어 구하기

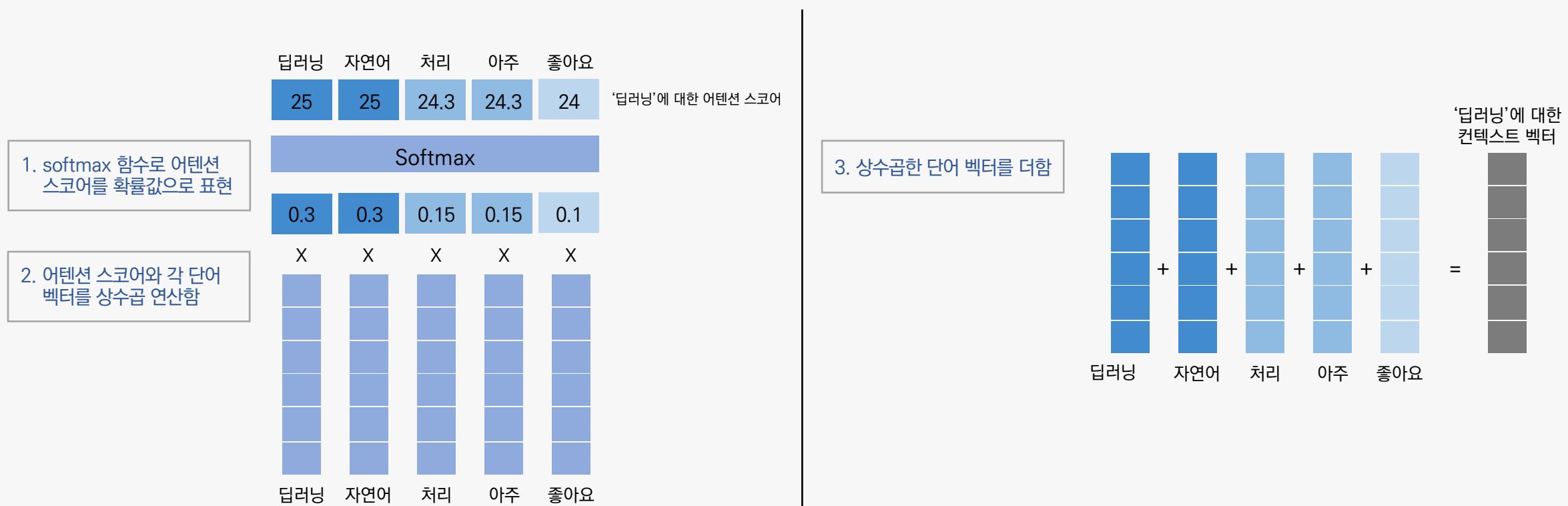


- 트랜스포머 모델에서는 단어 벡터끼리 내적 연산을 하여 어텐션 스코어를 구함

## 04. 트랜스포머 모델

## 셀프 어텐션(Self-Attention)

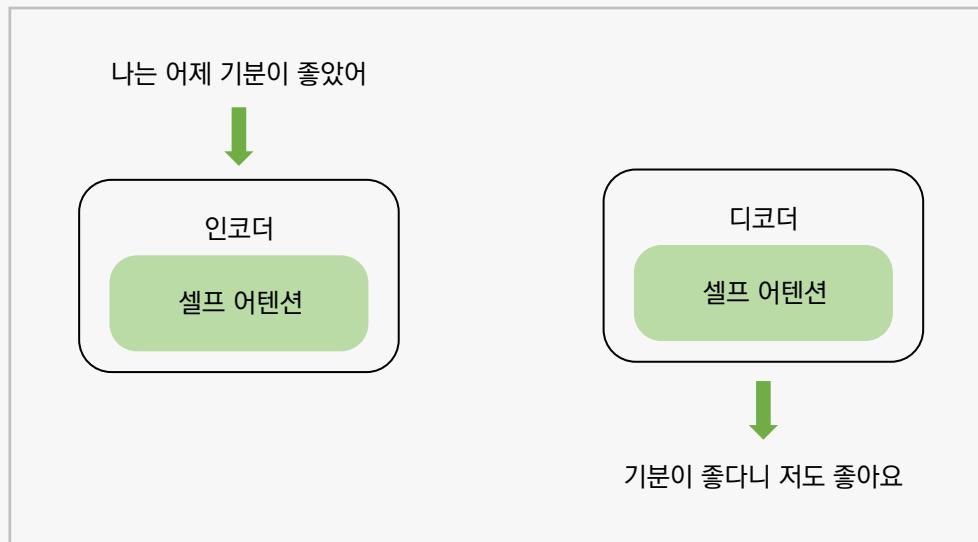
### ③ 어텐션 맵에 소프트맥스 함수 적용하기



# 04. 트랜스포머 모델

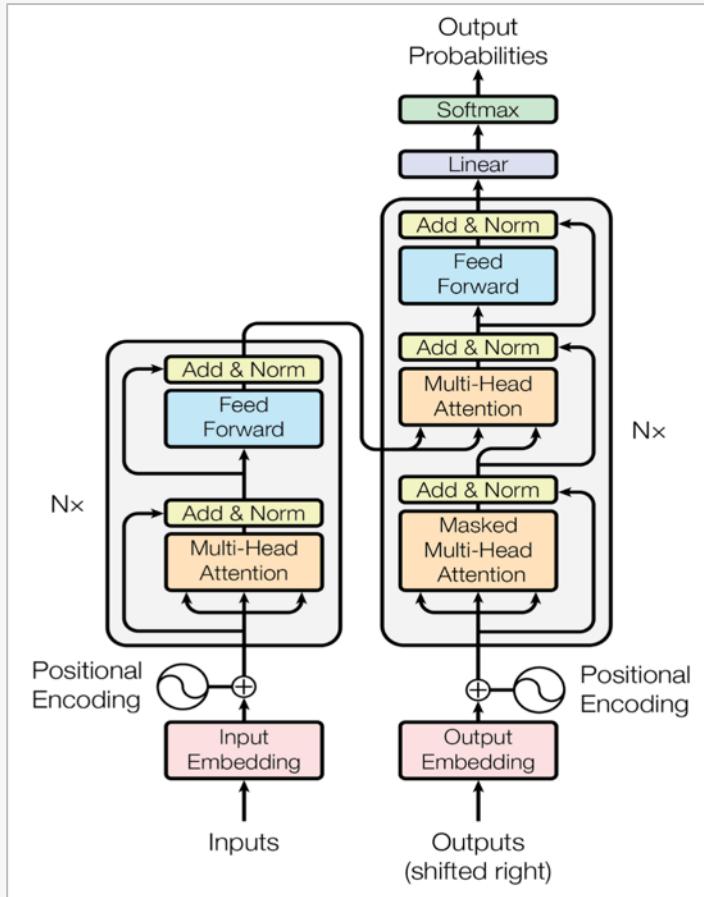
## 트랜스포머 모델 구현

- : 모델은 크게 인코더와 디코더로 구성되어 있음
- : 인코더에 입력이 들어가면 셀프 어텐션 기법으로 문장 정보를 추출하고, 디코더에서 출력 문장을 만듦



# 04. 트랜스포머 모델

## 트랜스포머 네트워크의 전체 구조



▲ 트랜스포머 네트워크 구조 / 출처: <https://jay.tech.blog/2019/01/19자연어처리nlp-bert>

### - 구현해야 하는 모듈 목록

1. 멀티 헤드 어텐션(Multi-head attention)
2. 서브시퀀트 마스크 어텐션(Subsequent masked attention)
3. 포지션-와이즈 피드 포워드 네트워크(Position-wise feed forward network)
4. 리지듀얼 커넥션(Residual connection)

# 04. 트랜스포머 모델

## 1. 멀티 헤드 어텐션(Multi-head attention)

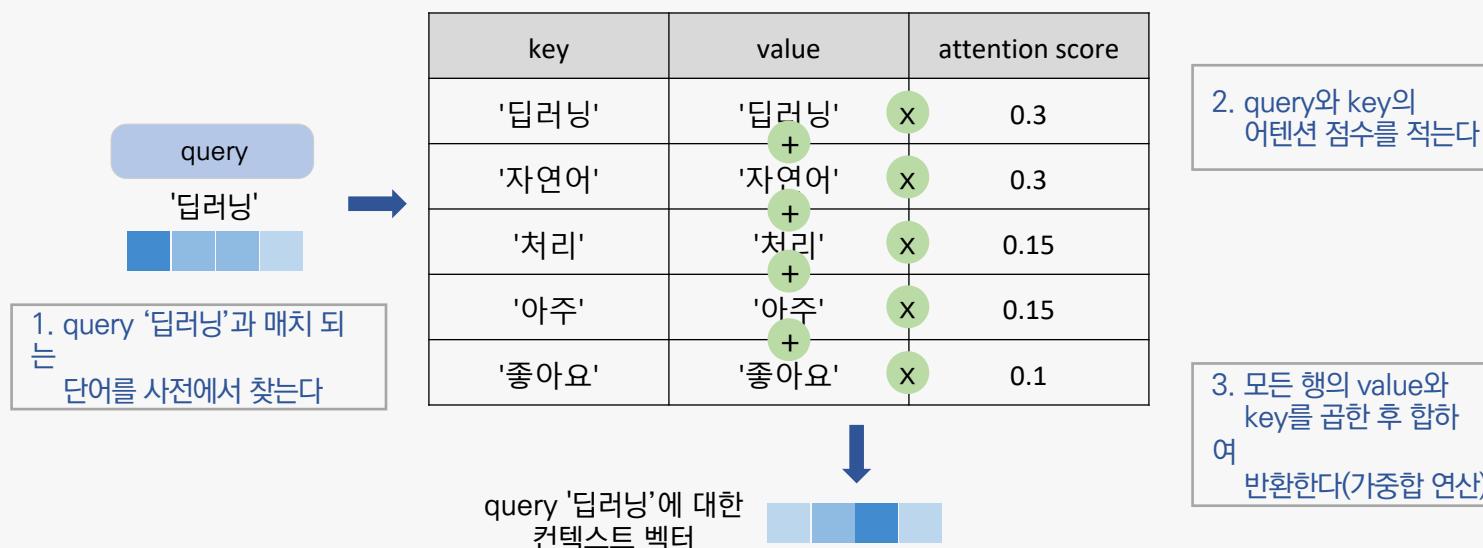
### \* 멀티 헤드 어텐션

: 내적 어텐션, 순방향 어텐션 마스크,  
멀티 헤드 어텐션 3부분으로 나누어 구현

: 내적 어텐션 구조가 중첩된 형태

### ① 스케일 내적 어텐션

- query(질의), key(키), value(값)으로 구성



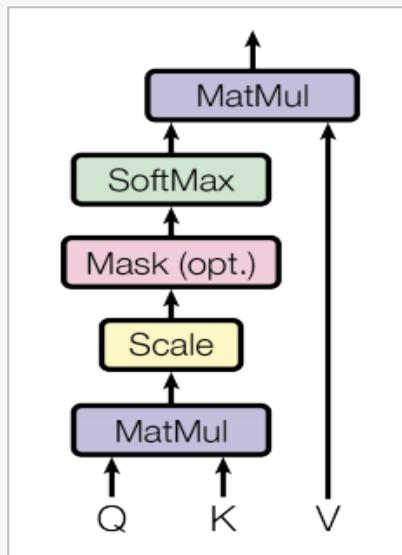
# 04. 트랜스포머 모델

## 1. 멀티 헤드 어텐션(Multi-head attention)

: 내적 어텐션 구조가 중첩된 형태

### ① 스케일 내적 어텐션

- 중간에 크기를 조정하는 과정(scaling)이 추가된 것
- query와 key를 내적한 값에 key 벡터의 차원 수를 제곱근한 값으로 나눈 뒤, 소프트맥스 함수를 적용



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

▲ 내적 연산 어텐션 그래프 / 출처: <https://jay.tech.blog/2019/01/19자연어처리nlp-bert>

### \* 멀티 헤드 어텐션

: 내적 어텐션, 순방향 어텐션 마스크,  
멀티 헤드 어텐션 3부분으로 나누어 구현

# 04. 트랜스포머 모델

## 1. 멀티 헤드 어텐션(Multi-head attention)

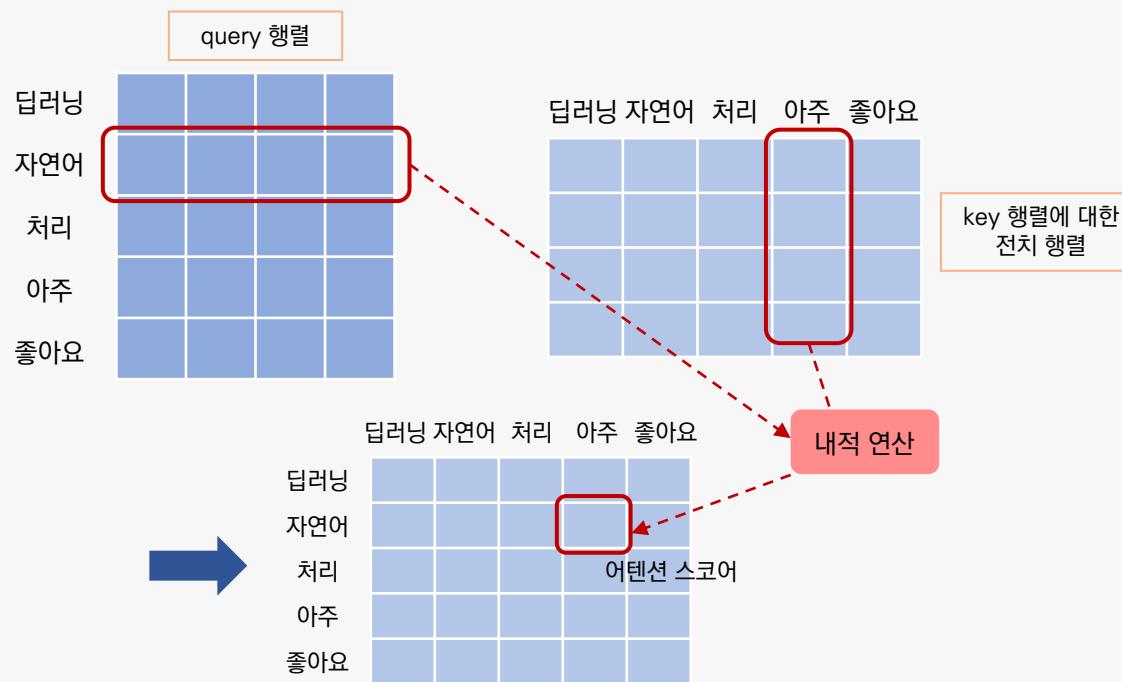
\* 멀티 헤드 어텐션

: 내적 어텐션, 순방향 어텐션 마스크,  
멀티 헤드 어텐션 3부분으로 나누어 구현

: 내적 어텐션 구조가 중첩된 형태

### ① 스케일 내적 어텐션

- query, key, value 값이 행렬로 들어와서 계산되는 과정



- 선형대수학의 외적 연산(outer product) 방법

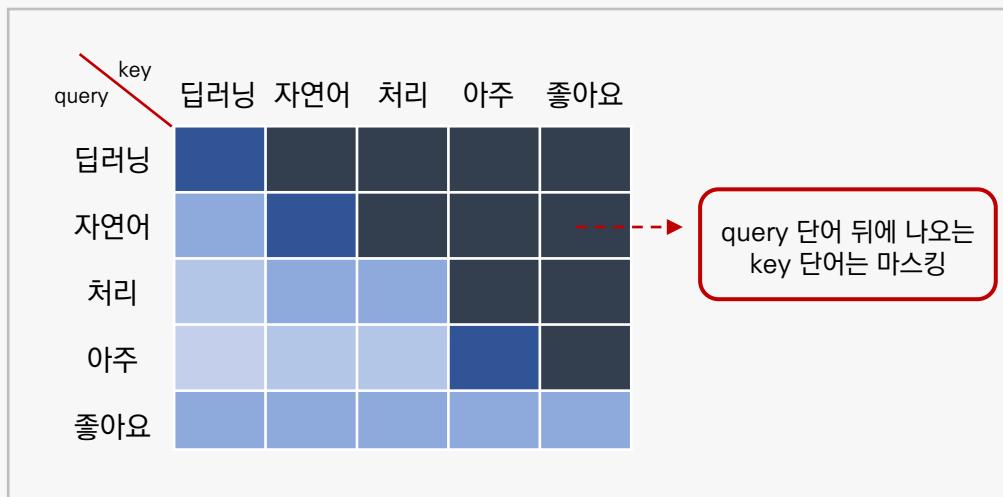
$$u \otimes v = uv^T = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} [v_1 \quad v_2 \quad v_3] = \begin{bmatrix} u_1v_1 & u_1v_2 & u_1v_3 \\ u_2v_1 & u_2v_2 & u_2v_3 \\ u_3v_1 & u_3v_2 & u_3v_3 \\ u_4v_1 & u_4v_2 & u_4v_3 \end{bmatrix}$$

# 04. 트랜스포머 모델

## 2. 서브시퀀트 마스크 어텐션(Subsequent masked attention)

### ② 순방향 마스크 어텐션

- 트랜스포머 모델의 경우, 입력에 전체 문장이 들어가기 때문에 위치에 상관없이 모든 단어를 참고하여 예측
- 자신도 예측하지 않은 상태에서 뒤의 단어를 예측한다는 오류를 방지하기 위해 마스크 기법을 사용함



# 04. 트랜스포머 모델

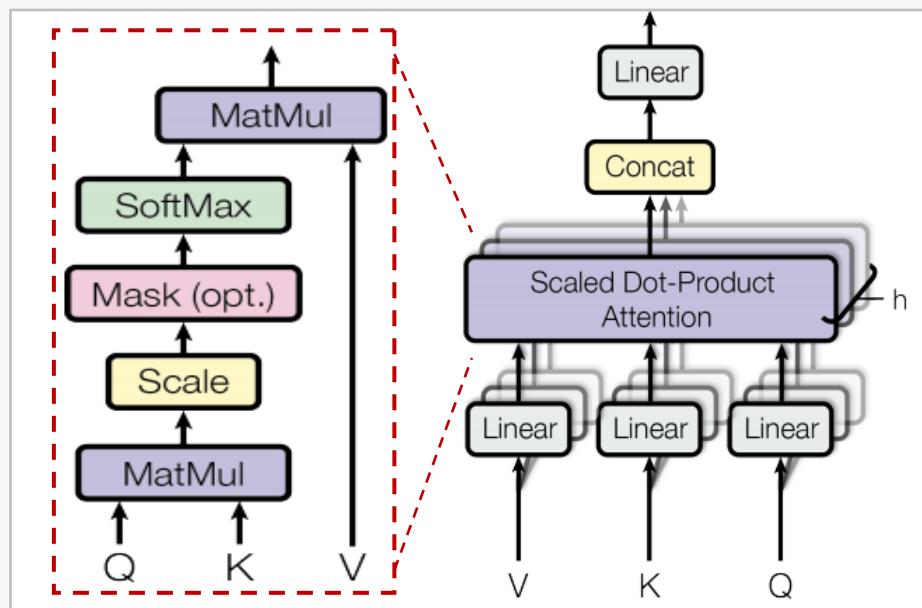
## 1. 멀티 헤드 어텐션(Multi-head attention)

\* 멀티 헤드 어텐션  
: 내적 어텐션, 순방향 어텐션 마스크,  
멀티 헤드 어텐션 3부분으로 나누어 구현

: 내적 어텐션 구조가 중첩된 형태

### ③ 멀티 헤드 어텐션

- 어텐션 맵을 여럿 만들어 다양한 특징에 대한 어텐션을 볼 수 있게 하는 방식



▲ 멀티 헤드 어텐션 / 출처: <https://jay.tech.blog/2019/01/19자연어처리nlp-bert>

## 04. 트랜스포머 모델

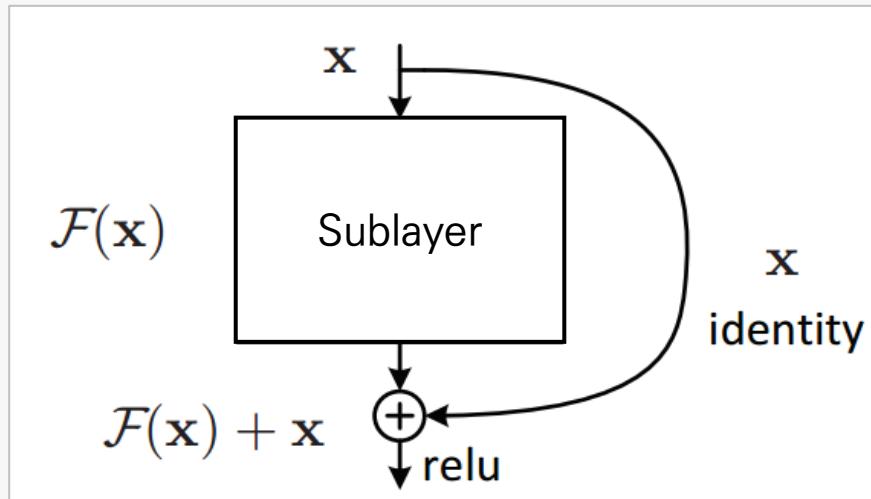
### 3. 포지션-와이즈 피드 포워드 네트워크(Position-wise feed forward network)

: 한 문장에 있는 단어 토큰 벡터 각각에 대해 연산하는 네트워크

$$\text{FFN}(x) = \max(0, xW_1 + b_1) W_2 + b_2$$

# 04. 트랜스포머 모델

## 4. 리지듀얼 커넥션(Residual connection)



▲ 리지듀얼 커넥션 / 출처: <https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec>

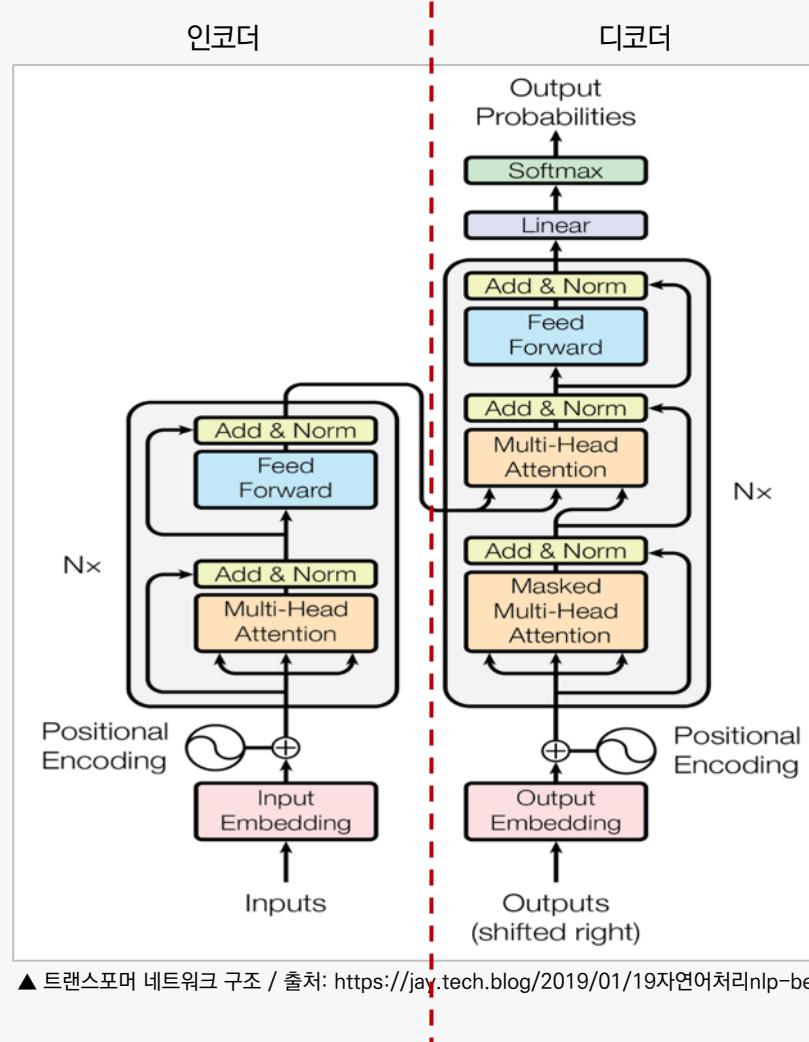
- 입력 정보  $x$ 와 네트워크 레이어를 거쳐 나온 정보  $\mathcal{F}(x)$ 가 있다면, 두 정보를 더해 앞에 있던 정보  $x$ 를 보존하게 됨
- 트랜스포머 네트워크에서는 리지듀얼 커넥션을 한 뒤, 레이어에 대한 값을 노멀라이즈하는 레이어 노멀라이제이션을 수행

# 04. 트랜스포머 모델

## 인코더/디코더

### 인코더

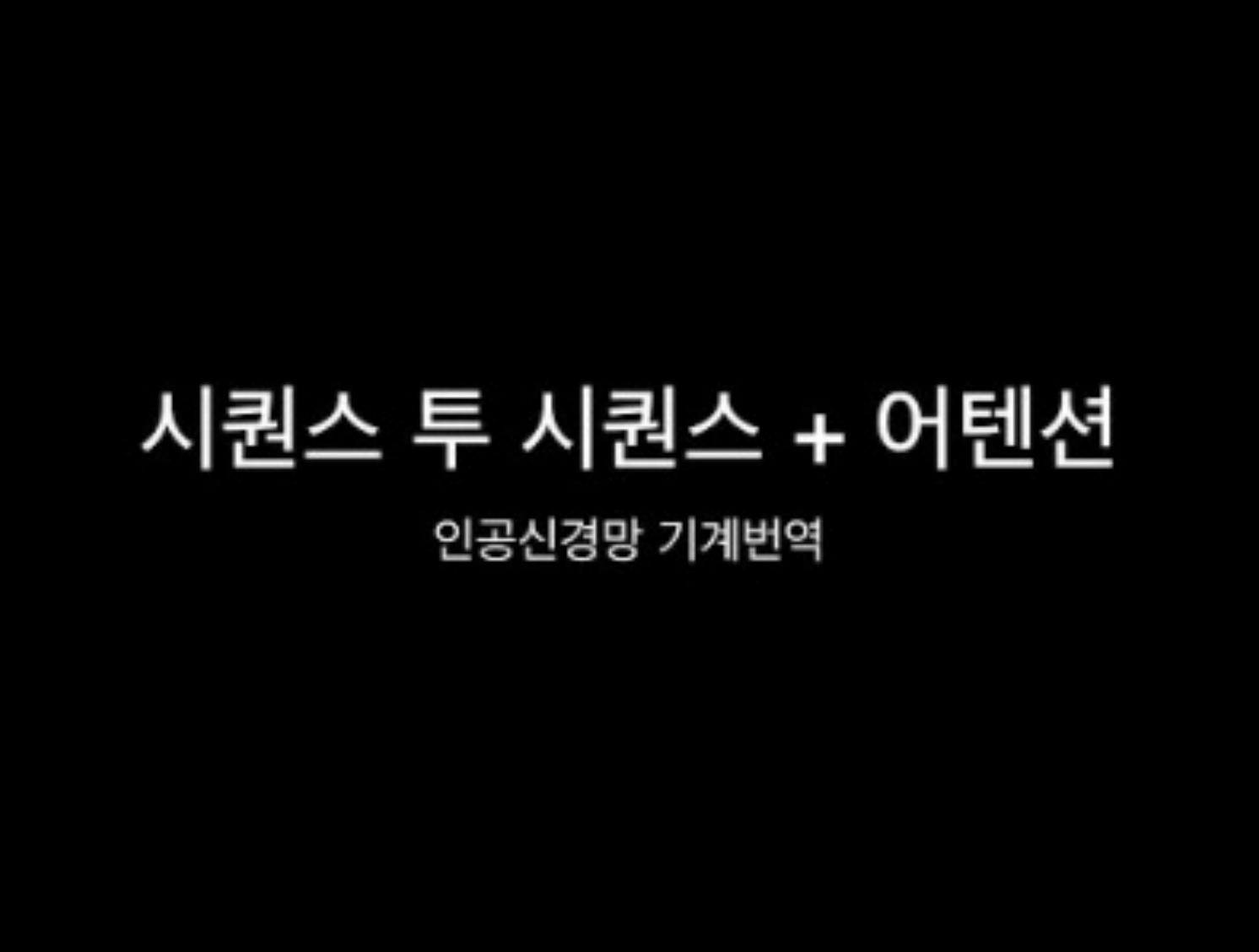
- 모델 입력값에 대한 정보를 추출하는 모듈
- 구성
  - : 멀티 헤드 어텐션
  - : 포지션 와이즈 피드 포워드 네트워크
  - : 레이어 노멀레이제이션
  - : 드롭아웃



### 디코더

- 인코더에 하나의 레이어가 더 추가된 형태
- 2개의 어텐션과 1개의 피드 포워드 레이어 구성

# 영상 (시퀀스 투 시퀀스 + 어텐션)



시퀀스 투 시퀀스 + 어텐션  
인공신경망 기계번역

# 영상 (트랜스포머 어텐션 이즈 올 유 니드)

트랜스포머

어텐션 이즈 올 유 니드

<https://arxiv.org/pdf/1706.03762.pdf>

# 자연어 처리 : 사전학습모델

with  python™

# 00. 사전 학습 모델

## 사전학습모델

\* 용어 정리: 하위문제(downstream task)  
사전 학습한 가중치를 활용하여 학습하고자 하는 본문제

: 기존 Xavier 등 임의 값으로 초기화하던 모델의 가중치들을 다른 문제(task)에 학습한 가중치들로 초기화하는 방법  
예시) 감정 분석 문제를 학습하면서 얻은 언어에 대한 이해를 학습한 후, 유사도 문제 학습에 활용하는 것



# 00. 사전 학습 모델

## 언어 모델 preview

: 특정 단어가 주어졌을 때, 다음(or 특정 위치) 단어가 어떤 단어인지 예측하는 것을 해결해주는 모델

### - 장점

: 해당 학습을 통해 모델은 언어에 대한 전반적인 이해(Natural Language Understanding; NLU)를 함

: 라벨이 필요 없는 대표적인 비지도 학습 문제로 데이터 제약이 없고 하위 문제 모델 성능도 대부분 향상함

⇒ 대부분의 자연어 처리 연구에서는 언어 모델을 사전 학습의 핵심 문제로 활용

# 00. 사전 학습 모델

## 사전 학습한 가중치를 활용하는 방법

### 특징 기반(feature-based) 방법

- 사전 학습된 특징을 하위 문제 모델에 부가적인 특징으로 활용하는 방법
- e.g. 단어에 대한 임베딩 벡터가 단어에 대한 특징값(즉, 특징)이 됨
- word2vec: 학습한 임베딩 특징을 학습하고자 하는 모델의 임베딩 특징으로 활용

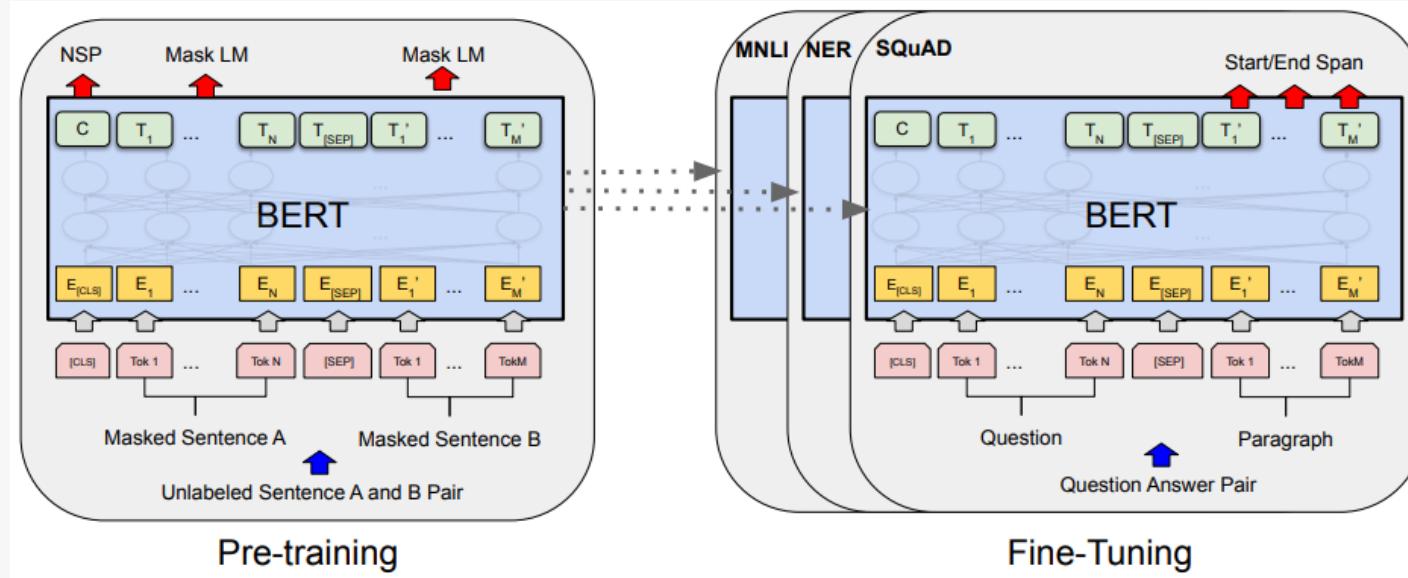
### 미세 조정(fine-tuning)

- 사전 학습한 모든 가중치에 하위 문제를 위한 최소한의 가중치를 추가하여 모델을 미세 조정하는 방법
- e.g. 감정 분석 문제에 사전 학습시킨 가중치와 더불어 텍스트 유사도를 위한 부가적인 가중치를 추가해 텍스트 유사도 문제를 학습하는 것

# 01. 버트

## 버트(BERT)

- : 2018년 구글에서 공개한 논문《BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding》에서 제안된 모델
- : 비지도 사전 학습을 한 모델에 추가로 하나의 완전 연결 계층만 추가한 후 미세 조정하는 방식
- : 11개의 자연어 처리 문제에서 최고(state-of-the-art) 성능을 보임
- : 사전 학습하는 모델이 양방향성(bidirectional)을 띨



# 01. 베트

## 언어 모델(Language modeling)

- : 단어들의 시퀀스에 관한 확률 분포
- : 단어 모음이 있을 때, 해당 단어 모음이 어떤 확률로 등장할지를 나타내는 값

## 언어 모델의 목적 함수 예시

### - Word2vec 모델 중 CBOW 모델의 목적 함수

$$\log p(w_t | w_{t-c}, w_{t-c+1}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+c-1}, w_{t+c})$$

#### \* CBOW 모델

: 앞뒤로 총  $2c$ 개의 단어 모음이 있을 때,  
t번째 위치에 올 단어에 대한 확률 분포를 찾는 언어 모  
델

### - 고전적인 언어 모델

$$\log p(w_t | w_{t-1}, w_{t-2}, \dots, w_2, w_1)$$

: t번째 위치에 올 단어들의 확률분포를 앞선  $t-1$ 개의 단어들을 통해 찾는다.  
앞서 나온 단어들을 통해 바로 직후의 단어가 어떤 단어가 될지를 예측하는 모델

# 01. 버트

## 버트의 사전 학습 문제

\* 버트의 사전 학습 2가지

- : ① 마스크 언어 모델
- ② 다음 문장 예측

### 1. 마스크 언어 모델

- 양방향성을 가지고 언어 모델을 학습하기 위한 방법
- 일부 단어를 마스킹하여 모델이 해당 단어를 알지 못하게 한 후 모델을 통해 마스킹된 단어를 예측하게 함
- 입력값으로 들어간 문장 안의 다른 단어를 통해 마스킹된 단어를 예측하도록 학습하는 것

|           | 입력값                            |
|-----------|--------------------------------|
| 입력값       | 딥러닝 자연어 처리 공부는 매우 재밌고 유익하다     |
| 마스킹 후보    | 딥러닝 자연어 처리 공부는 매우 재밌고 유익하다     |
| 마스킹 후 입력값 | 딥러닝 자연어 [MASK] 공부는 내일 재밌고 유익하다 |

- 입력값의 단어 중 15% 단어를 마스킹함
- 마스킹 단어 구성 비율
  - 80%: 스페셜 토큰 [MASK]
  - 10%: 임의의 단어로 대체
  - 10%: 마스킹하지 않음

# 01. 버트

## 버트의 사전 학습 문제

\* 버트의 사전 학습 2가지

- : ① 마스크 언어 모델
- ② 다음 문장 예측

### 2. 다음 문장 예측(next sentence prediction)

- 주어진 두 문장이 이어진 문장인지 아닌지를 예측하는 것을 학습
- 데이터셋을 구성할 때, 한 문장에 대해 50% 확률로 다음 문장을 이어서 전체 텍스트를 모델의 입력값으로 넣고 나머지 50% 확률은 임의의 다른 문서를 기준 문장과 함께 모델의 입력값으로 넣음
- 다음 문제 예측을 사전 학습함으로써 문장 간 관계를 모델이 학습하게 됨

#### + 스페셜 토큰

| 스페셜 토큰 | 역할                                | 첨가 위치        |
|--------|-----------------------------------|--------------|
| [CLS]  | 다음 문장인지 아닌지 여부에 관한 이진 분류 예측을 위함   | 모든 입력값 앞     |
| [SEP]  | 두 개의 문장을 넣기 때문에 모델이 문장을 구분할 수 있도록 | 각 문장이 끝나는 지점 |

# 01. 버트

## 버트의 사전 학습 문제

: 버트는 마스크 언어 모델, 다음 문장 예측을 동시에 사전 학습함

- 최종 모델 입력값



# 01. 버트

## 버트의 모델

### \* 트랜스포머와의 차이

: ReLU 함수 대신 GELU 함수 사용

- : 트랜스포머 모델의 인코더 부분만을 사용해 모델 학습
- : 정규 분포의 누적분포함수(cumulative distribution functions)인 GELU는 0 주위에서 부드럽게 변화해 학습 성능을 높임

### + 버트의 하이퍼파라미터 설정

|        | L  | H    | A  | 전체 파라미터 수 |
|--------|----|------|----|-----------|
| 버트 베이스 | 12 | 768  | 12 | 110M      |
| 버트 라지  | 24 | 1024 | 16 | 340M      |

- L: 트랜스포머 층(블록) 개수  
어텐션 층, 피드 포워드 층을 포함하는 블록이 총 몇 번 반복됐는지를 나타냄
- H: 모델의 전체적인 차원 수를 의미  
트랜스포머 모델에서는 전체 층 출력값의 차원 수가 동일해야 함
- A: 멀티 헤드 어텐션에서의 헤드의 개수

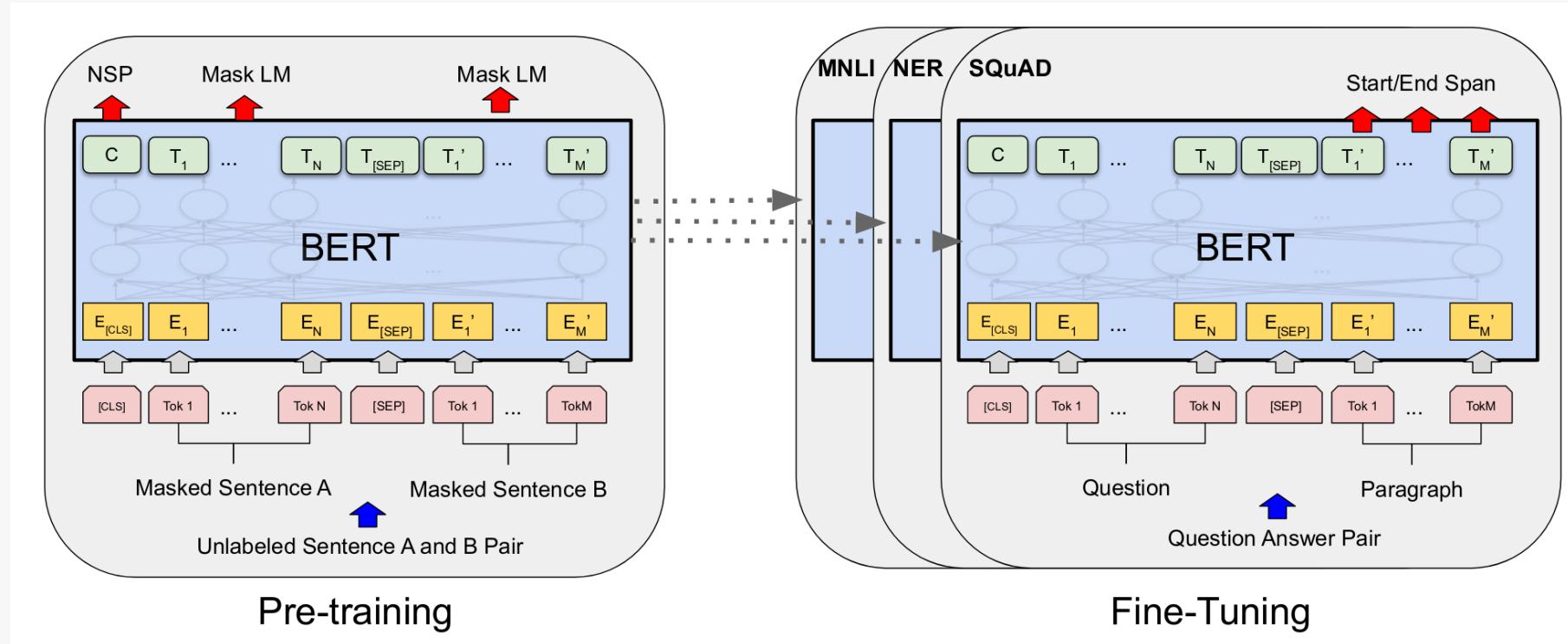
# 01. 버트

## 버트의 미세 조정

### \* 버트 미세 조정

: 사전 학습했던 가중치를 그대로 사용하여,  
미세 조정 후의 모델도 동일한 모델 크기를 사용해야 함

: 사전 학습된 가중치를 가진 버트는 여러 하위 문제에 미세 조정(fine-tuning)될 수 있음



# 01. 버트

## 버트의 미세 조정

### \* 버트 미세 조정

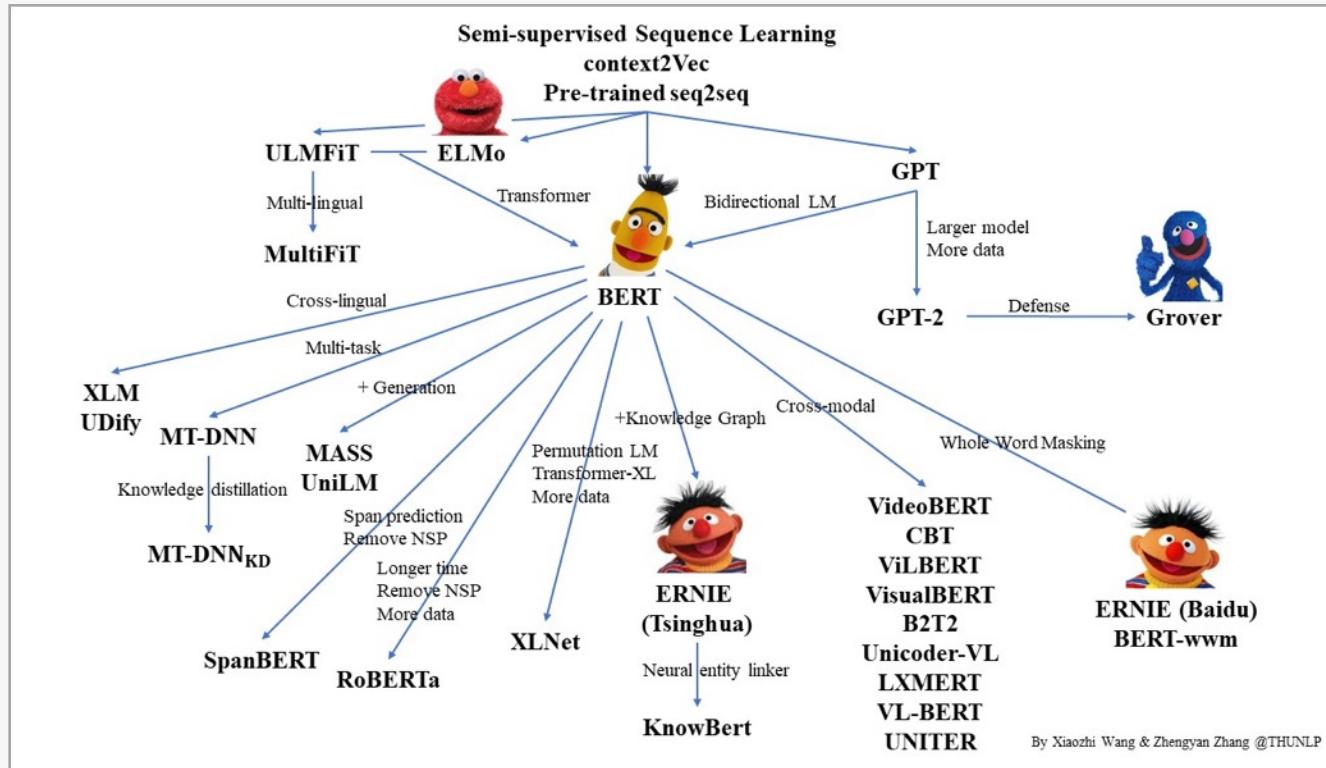
: 사전 학습했던 가중치를 그대로 사용하여,  
미세 조정 후의 모델도 동일한 모델 크기를 사용해야 함

: 사전 학습된 가중치를 가진 버트는 여러 하위 문제에 미세 조정(fine-tuning)될 수 있음

| 분야                                     | 예시                                                                                                                   |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| 언어적 용인 가능성<br>Linguistic Acceptability | 입력: 그는 한 번도 고향에 갔다.<br>답: 가능 / 불가능                                                                                   |
| 자연어 추론<br>Natural Language Inference   | 전제: 버트를 이용한 자연어 처리를 수행하는 연구자가 사무실에 있다.<br>가설: 연구자는 딥러닝을 알지 못한다.<br>답: 수반, 모순, 중립                                     |
| 유사도 예측<br>Similarity Prediction        | 입력: 문장① 버트와 트랜스포머 중에 무엇을 먼저 알아야 할까요?<br>문장② BERT와 Transformer 중에서 무엇부터 시작해야 하나요?<br>답: 유사 / 비유사                      |
| 감정 분석<br>Sentiment Analysis            | 입력: 나는 이 영화를 용서한다.<br>답: 긍정 / 부정                                                                                     |
| 개체명 인식<br>Named Entity Recognition     | 입력: 단백질은 단 한 가지 아미노산이라도 부족하면 합성되지 않는다.<br>답: "BIO"        "BIO"                                                      |
| 기계독해<br>Reading Comprehension          | 지문: 시절의 대부분을 건천동에서 보냈고 외가인 아산에서 소년기를 보냈다.<br>이정은 자신의 네 아들에게 (... 종략...) 순신(舜臣)이라는 이름이 붙었다.<br>질문: 이순신의 외가는?<br>답: 아산 |

# 01. 버트

## 버트를 중심으로 한 사전 학습 모델 관계도



### - 모델 분류 기준

- ① **버트 개선(성능, 속도, 메모리)을 노력한 모델**  
: SpanBERT, RoBERTa, ERNIE 등
- ② **버트 알고리즘 문제를 실험적으로 증명하며 개선하려는 모델**  
: XLNet
- ③ **버트를 자연어 처리가 아닌 다른 영역에서 활용하는 모델**  
: VideoBert, VisualBERT

▲ 버트를 중심으로 한 사전 학습 모델 관계도 / 출처: <https://github.com/thunlp/PLMpapers>

## 02. 버트를 활용한 미세 조정 학습

### 버트 파일 불러오기

: 버트 활용을 위해서는 ① 데이터를 전처리하는 토크나이저, ② 모델 가중치를 가진 모델을 불러와야 함

### 버트 문장 전처리 진행하기

#### - 버트에 필요한 입력값

| 입력값            | 역할                                                                        |
|----------------|---------------------------------------------------------------------------|
| input_ids      | 문장을 토크나이즈해서 인덱스 값으로 변환<br>일반적으로 단어를 서브워드 단위로 변환하는 워드 피스 토크나이저 활용          |
| attention_mask | 패딩된 부분이 학습의 영향을 받지 않도록 처리하는 입력값<br>1: 어텐션에 영향을 받는 토큰, 0: 어텐션 영향을 받지 않는 토큰 |
| token_type_ids | 두 개의 시퀀스를 입력으로 활용할 때, 0과 1로 문장 토큰 값을 분리                                   |

#### - 버트 스페셜 토큰의 역할

| 스페셜 토큰 | 역할                 |
|--------|--------------------|
| [UNK]  | 모르는 단어에 대한 토큰      |
| [MASK] | 마스크 토큰, 사전 학습에서 활용 |
| [PAD]  | 최대 길이를 맞추는 용도      |
| [SEP]  | 문장 종결을 알림          |
| [CLS]  | 문장 시작을 알림          |

## 02. 버트를 활용한 미세 조정 학습

### 추가: 혼동 행렬 표와 F1 Score

#### - 혼동 행렬(Confusion Matrix)

| 혼동 행렬(Confusion Matrix) 표 |              | 예측값(Predicted)     |                    |
|---------------------------|--------------|--------------------|--------------------|
|                           |              | 양성(Positive)       | 음성(Negative)       |
| 정답값(Observed)             | 양성(Positive) | True Positive(TP)  | False Negative(FN) |
|                           | 음성(Negative) | False Positive(FP) | True Negative(TN)  |

$$1. \text{Accuracy}(\text{정확도}) = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$2. \text{Precision}(\text{정밀도}) = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$3. \text{Recall}(\text{재현율}) = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$4. \text{F1 Score} = \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

#### \* Accuracy

: 전체 경우의 수 중에서 올바르게 예측한 비율

#### \* Precision

: 1이라고 예측한 것 중에 실제 값도 1인 것의 비율

#### \* Recall (= sensitivity; 민감도)

: 실제 값이 1인 것 중에서 내가 1이라고 맞춘 것의 비율

#### \* F1 Score

: 정밀도와 재현율을 조화 평균하여 하나의 수치로 나타낸 지표

# 03. GPT

## GPT(Generative Pre-training) 1

: 큰 자연어 처리 데이터를 비지도 학습으로 사전 학습한 후, 학습된 가중치로 풀고자 하는 문제를 미세 조정하는 방법론의 모델

### - 모델 구조

- 버트와 마찬가지로 트랜스포머 모델을 사용
- 단, GPT1에서는 트랜스포머의 디코더 구조만 사용 (버트는 인코더 구조만 사용)

\* GPT1에서 디코더를 사용한 것  
: 순방향 마스크 어텐션을 사용한 것으로 볼 수 있음

### - 사전 학습

- 버트와 달리 하나의 사전 학습 방식(전통적인 언어 모델 방식) 사용
- 앞 단어를 활용해 다음 단어를 예측하는 방식으로 사전 학습 진행

| input                  | label |
|------------------------|-------|
| "<START>"              | "나는"  |
| "<START>", "나는"        | "학교에" |
| "<START>", "나는", "학교에" | "간다"  |

\* GPT1 사전 학습  
: 별도 라벨이 존재하지 않는 데이터도 학습이 가능  
→ 비지도 학습으로 분류  
: 많은 데이터로 모델 가중치를 사전 학습할 수 있음  
: 버트와 달리, 실제 문제 대상으로 학습을 진행할 때도 언어 모델을 함께 학습

# 03. GPT

## GPT(Generative Pre-training) 2

: OpenAI 제안, 2018년 발표된 GPT1 모델의 성능을 향상한 모델로서 텍스트 생성에서 특히 좋은 성능을 보임

### - 모델 구조

- 대부분 GPT1과 동일, 트랜스포머의 디코더를 기반으로 하는 모델
- 차이점: 레이어 노멀라이제이션이 각 부분 블록의 입력 쪽으로 이동 (기존에는 각 레이어 직후 레지듀얼 커넥션과 함께 적용)

### - 학습 데이터 및 모델 크기

|     | GPT1 | GPT2     |
|-----|------|----------|
| 레이어 | 12개  | 117만 개   |
| 가중치 | 48개  | 1,542만 개 |

### - 모델 입력

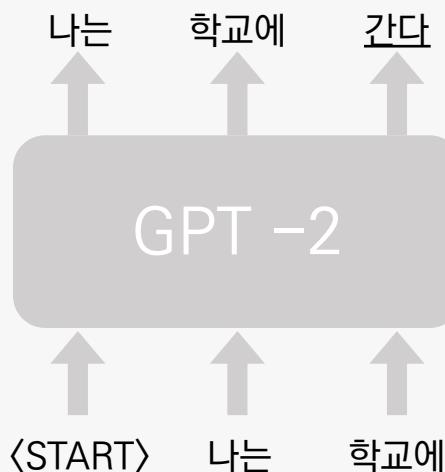
- BPE(Byte Pair Encoding) 방식을 사용해 텍스트를 나누어 입력값으로 사용
- 글자와 문자 사이 적절한 단위를 찾아 나누는 방식으로 높은 성능을 보임

# 04. GPT를 활용한 미세 조정 학습

## GPT2 미세 조정 학습

: GPT2 모델은 언어를 생성하는 문제를 가지고 학습 → 언어 모델을 사전 학습 문제로 사용

### - 모델 언어 생성 예시



### - 언어 생성 모델



#### - past

: 모델에서 연산한 현재 스텝까지의 결괏값

#### - attention

: 모든 레이어에서 연산한 어텐션 맵 값

#### - hidden\_states

: 모든 레이어에서 출력한 hidden state 벡터 값

# 04. GPT를 활용한 미세 조정 학습

## GPT2 사전 학습 모델 문장 생성

### - GPT2 스페셜 토큰의 역할

| 스페셜 토큰 | 역할               |
|--------|------------------|
| <unk>  | 모르는 단어에 대한 토큰    |
| <pad>  | 배치 데이터 길이 맞추는 용도 |
| <s>    | 문장 시작을 알림        |
| </s>   | 문장 종결을 알림        |

### - GPT2 분류 모델을 위한 스페셜 토큰

| 스페셜 토큰    | 역할                                                        |
|-----------|-----------------------------------------------------------|
| <unused0> | SEP 토큰으로, 두 개 문장 입력 시 문장을 구분<br>(자연어 추론과 텍스트 유사도 모델에서 활용) |

# 04. GPT3

## GPT(Generative Pre-training)3

- : OpenAI 제안, 2020년 발표된 모델로 GPT-2에 비해 모델 및 데이터 사이즈 확대, 학습 길이가 증가된 모델
- : 언어 문제 풀이, 글짓기, 번역, 간단한 사칙연산, 주어진 문장 내에서 간단한 웹 코딩이 가능
- : 인간이 작성한 글과 구별하기 어렵다는 장점이자 단점을 지님

### - 모델 구조

- 대부분 GPT2와 동일, 트랜스포머의 디코더를 기반으로 하는 모델
- 차이점: 트랜스포머 레이어에서 alternating dense, locally banded sparse attention 사용
- 약 1,750억 개의 매개변수를 가짐

### - 주요 하이퍼파라미터

| 하이퍼파라미터     | 설명                 |
|-------------|--------------------|
| $d_{model}$ | 인코더/디코더의 입출력 크기    |
| num_layers  | 인코더/디코더의 구성 층      |
| num_heads   | 어텐션 수행 시, 병렬의 개수   |
| $d_{ff}$    | 피드 포워드 신경망의 은닉층 크기 |

# 05. Switch Transformer

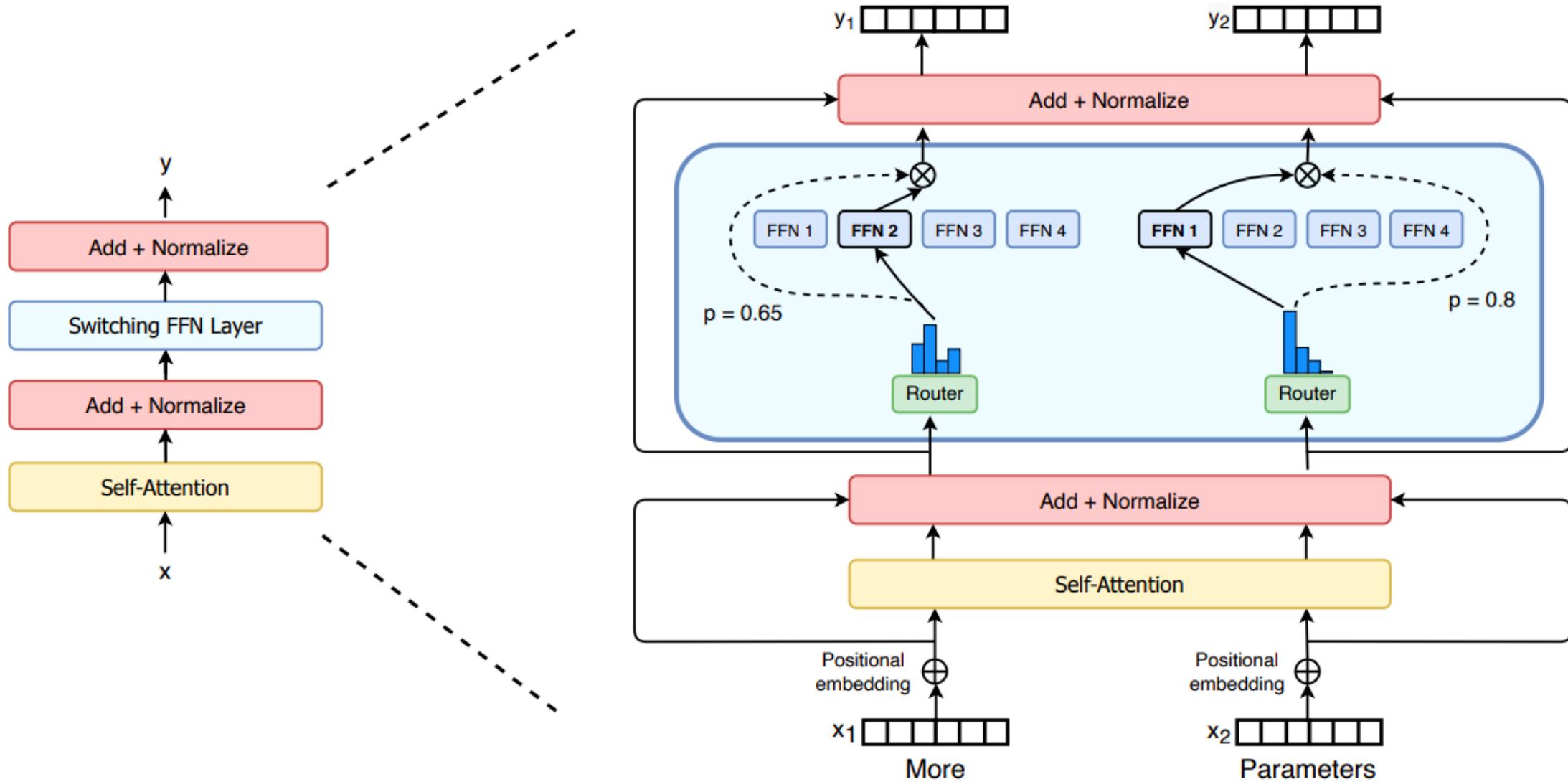
## Switch Transformer

- : Google Brain AI team 제안, 2021년 발표된 모델로 1조 매개변수 변압기를 가진 최초의 조 단위 파라미터 모델
- : 기존 모델 대비 시간과 자원을 더 효율적으로 쓸 수 있는 기법을 적용한 모델로 데이터 처리 규모가 크게 늘어남
- : 현재(21년 2월) 기준 논문과 소스코드 형태로만 공개

### - 모델 구조

- 최대 110억 개 파라미터를 사용한 언어 모델 T5 시리즈와 동일 연산을 활용하지만, 4-7배 빠른 속도로 모델을 학습시킴
- 1990년대 초 제작된 AI 모델 패러다임을 혼합된 모델로, 게이팅 네트워크를 통해 주어진 데이터에 적합한 패러다임을 적용

# 05. Switch Transformer



# 참고 문헌

권철민, 『파이썬 머신러닝 완벽 가이드』, 위키북스(2020), p.466 – p.561

전창욱 외, 『텐서플로2와 머신러닝으로 시작하는 자연어 처리』, 위키북스(2020), p.361 – p.541

Gupta, S., (2018, Jan). **Automated Text Classification Using Machine Learning.**

KDnuggets News. [shorturl.at/jnGX5](http://shorturl.at/jnGX5)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). **Attention is All you Need.** ArXiv, abs/1706.03762.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krüger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). **Language Models are Few-Shot Learners.** ArXiv, abs/2005.14165.

Fedus, W., Zoph, B., & Shazeer, N. (2021). **Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity.** ArXiv, abs/2101.03961.