

인공지능과 자연어 처리 기술의 동향

빅데이터 융합학 / 교수: 신병주



개요

A

워드 임베딩

B

언어 모델

C

실습

D

STEP 01

STEP 02

STEP 03

STEP 04

— 갑작스런 등장, ChatGPT

챗GPT에 대응 나선 美 대학...적응하거나 금지하거나

챗GPT의 명암...기술의 질주와 폭주 사이

챗GPT 교육혁명...피하기보단 활용하자

챗GPT 열풍에 올라타려는 AI 신약개발사...전문가들
은 “글쎄”

챗GPT 적용한 배경음악 AI검색 서비스 나왔다

네이버, '챗GPT'에 대항할 '서치GPT' 상반기 출시한다

— 기존 AI 들과의 차이

- 기존 대화형 AI (챗봇, AI 스피커): 선언적, **설계자가 정해 놓은 데이터베이스** 안에서 답변
- ChatGPT: 무수히 많은 질문에 **스스로의 학습**을 기반으로 사용자의 질문에 답변

챗GPT · 기존 검색 엔진 차이점		
 챗GPT		 검색 엔진
• 인공지능 기술인 언어 모델링을 사용해 사용자 질문에 답변 생성	인공지능 기술	• 키워드 검색을 통한 정보 제공
• 사용자 질문에 새로운 정보를 생성하는 기능이 있어 기존 검색 엔진보다 더 생성적 답변 제공	생성성	• 새로운 정보 생성할 수 없음
• 사용자 친화적 상호작용을 통해 질문을 이해하고 대답하는 방식	상호작용	• 키워드 검색을 통한 정보 제공으로 사용자와 상호작용은 없음
• 자연어처리 기술을 통해 사용자 질문을 이해해 의도에 맞는 결과를 제공 • 사용자 이전 질문을 기억하고 연관성을 고려해 유연성 있게 답변	질의·문맥 이해도	• 사용자 질문을 이해하고 답변하는 방식이 아닌 키워드 검색을 통한 정보 제공 방식 • 질문에 대한 답변을 제공하지 않으며 각 검색마다 독립적으로 정보 제공

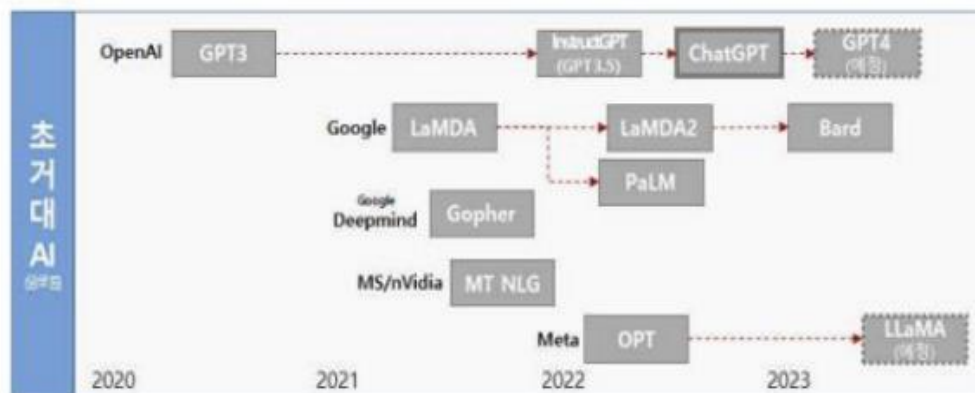
자료: 한국지능정보사회진흥원

인공지능의 지속적인 발전

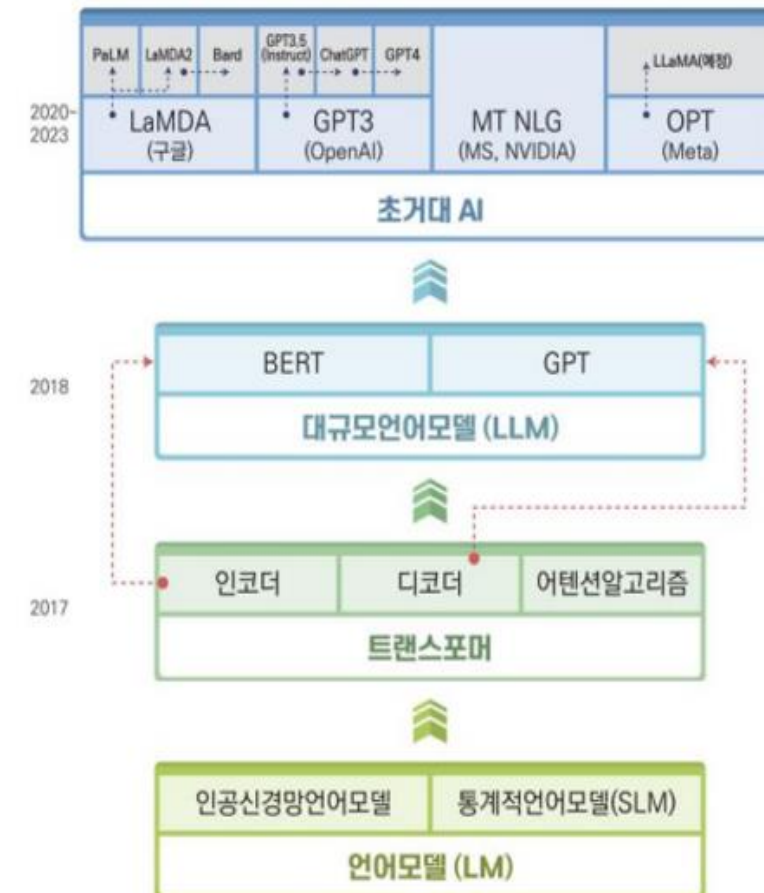


— 자연어 처리 인공지능의 발전

- 자연어처리는 전통적인 **통계적 언어모델(SLM)**에서 **인공신경망 언어 모델**로 발전
- **트랜스포머(Transformer)**의 등장 기점으로 **대규모 언어모델(LLM)**, **초거대 AI**로 발전 하면서 **ChatGPT** 등장



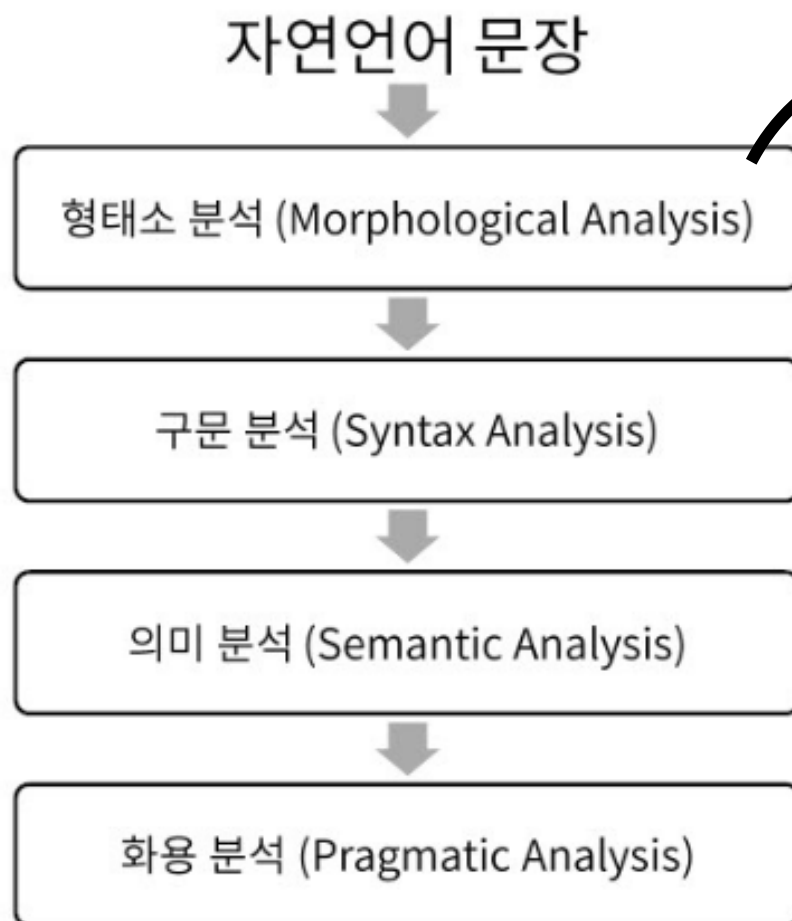
출처: SPRI 소프트웨어정책연구소



[출처 : 초거대 언어모델의 부상과 주요 이슈, SPRI 이슈 리포트 2023 재편집]

워드 임베딩

— 자연어 분석 단계



1. 형태소 분석 (Morphological Analysis)

입력된 문자열을 분석해 **형태소** 라는 **최소 의미 단위로 분리**하는 단계

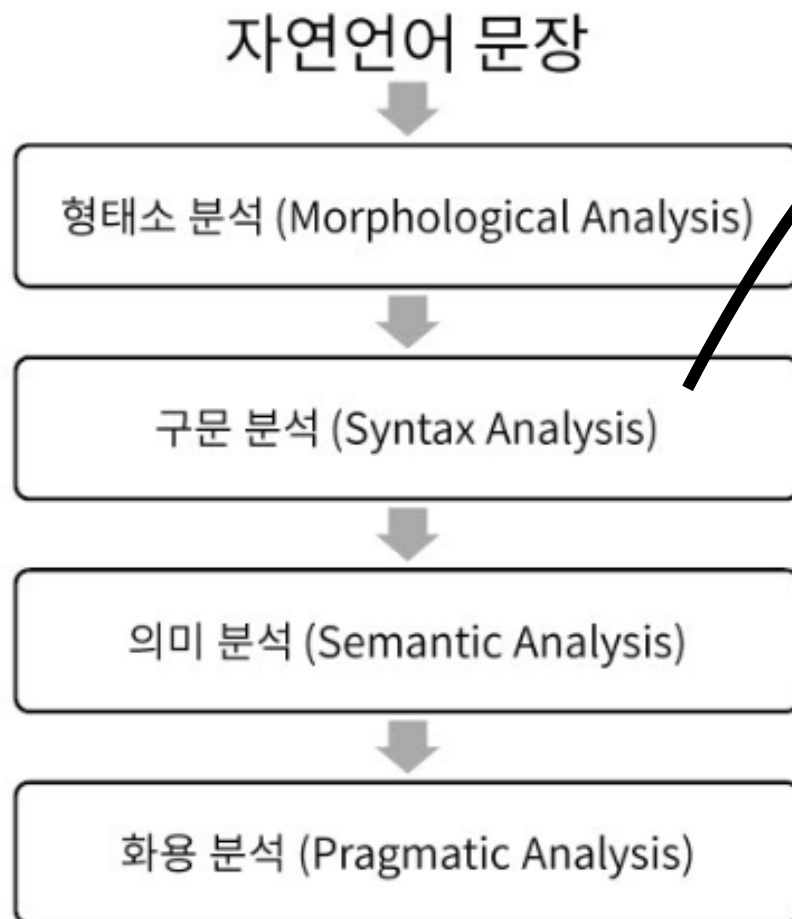
중의성

형태소 분석의 난점에는 하나의 어절에 대해 형태소 분석을 할 때, 분석 결과가 다양하게 나올 수 있다는 ‘**중의성**’이 있다.

예를 들어 “감기는”이라는 어절의 경우 아래와 같이 세 가지 의미로 분석이 가능하다.

1. 감기(명사:cold) + 는(조사)
2. 감(동사 어간) + 기(명사화 어미) + 는(조사)
3. 감(동사 어간) + 기는(어미)

— 자연어 분석 단계

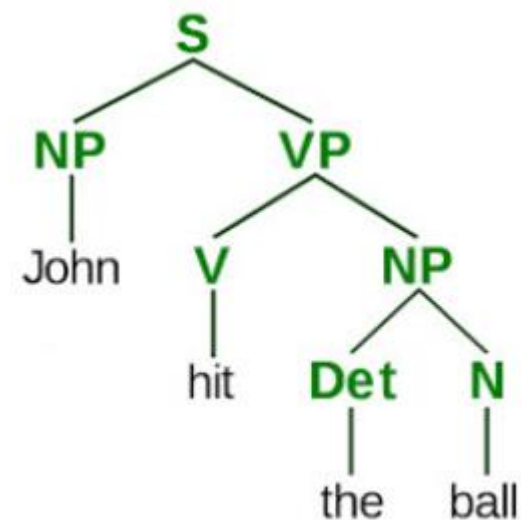


2. 구문 분석 (Syntax Analysis)

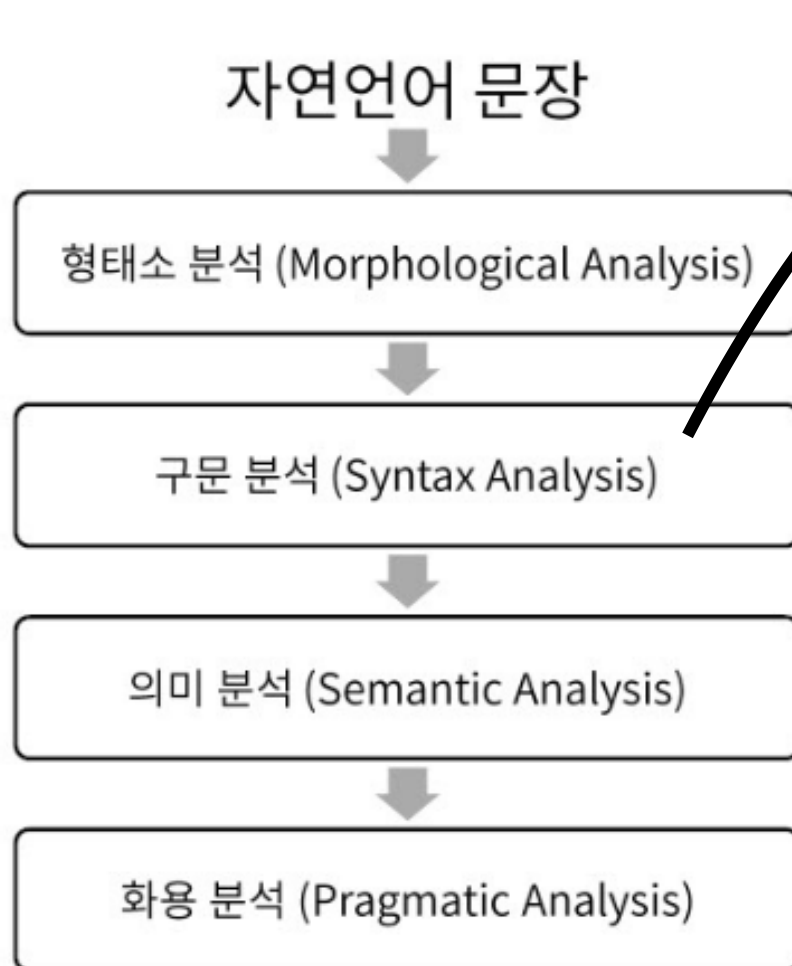
문자의 **구조**를 분석

구문 분석 단계에서는 아래의 두 가지 과제를 수행한다.

- 입력 문장에 대한 **문법 구조**를 분석하여 **구문 트리**로 나타낸다.



— 자연어 분석 단계



2. 구문 분석 (Syntax Analysis)

문자의 **구조**를 분석

문장이 문법적으로 옳은가를 판단한다.

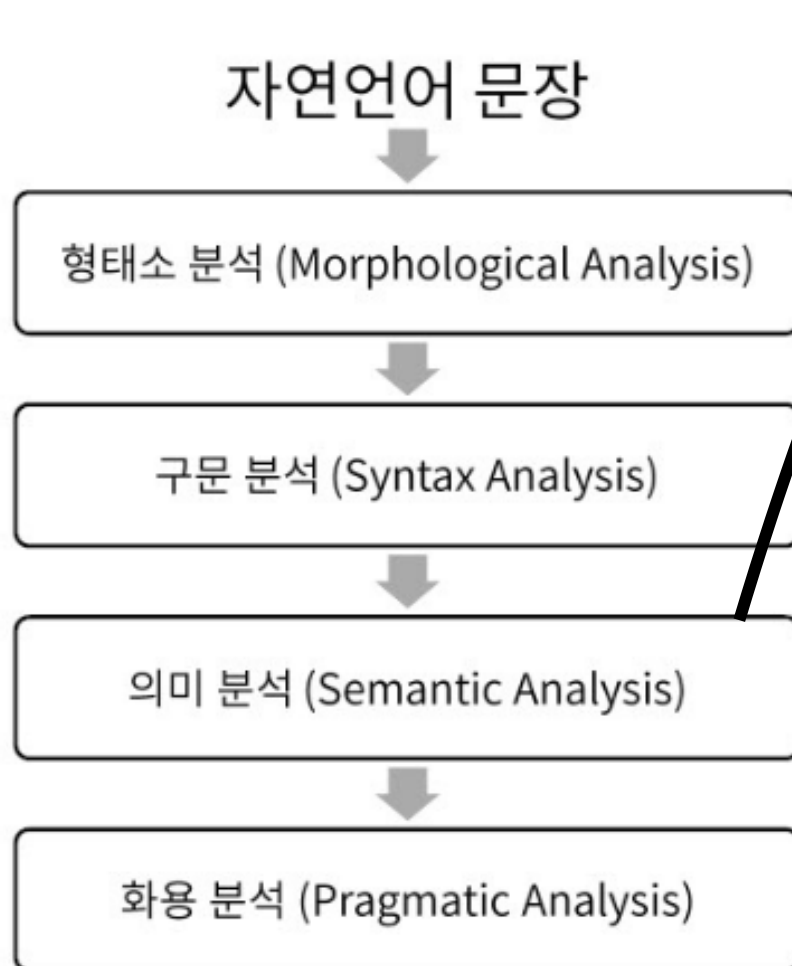
- The dog bit the boy. (O)
- The boy bit the dog. (O)
- Bit boy dog the the. (X)

분석한 문법 구조를 바탕으로 문법에 어긋나는 여부를 파악한다.
이 때, 두 번째 문장의 의미는 ‘소년이 강아지를 문다’ 인데,
이 문장은 이상한 맥락이 아니고서는 틀린 문장이다.

하지만 구문 분석 과정에서는 두 번째 문장이
틀린 문장으로 인식되지 않는다.

주어 + 동사 + 목적어의 **어순**이 **문법에 틀리지 않기 때문이다.**
이러한 의미상 모호함은 다음 단계인 의미 분석에서 해결된다.

— 자연어 분석 단계



3. 의미 분석 (Semantic Analysis)

구문 분석 결과 생성된 통사 구조에 의해
문장의 의미를 밝혀내는 작업

이 단계에서는 아래의 과제들을 수행한다.

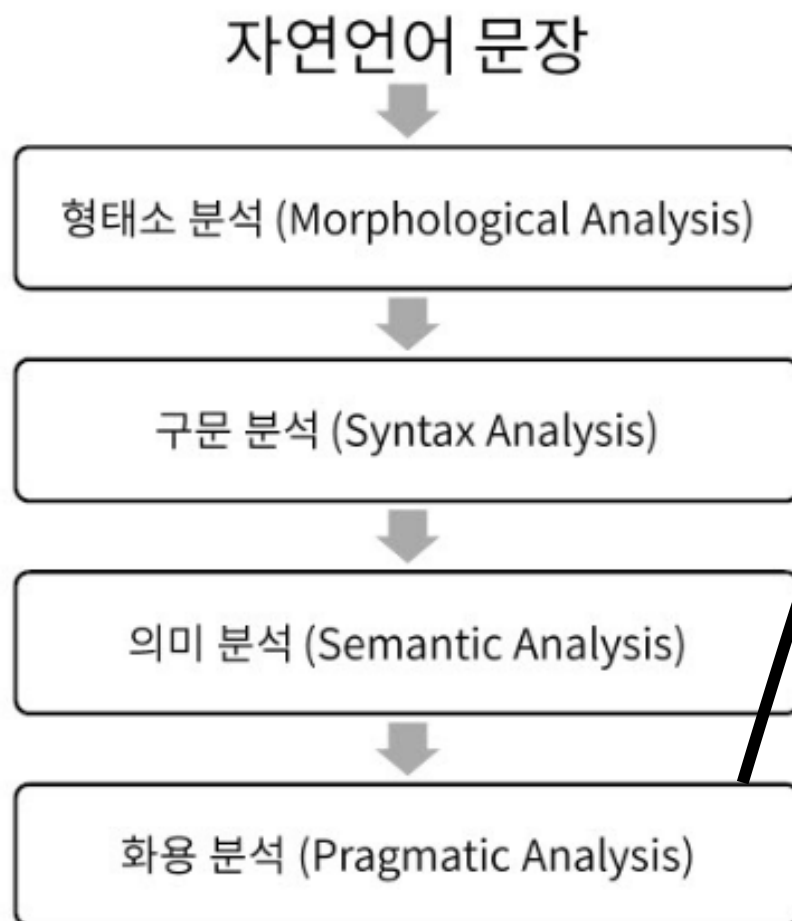
- 의미적으로 옳은 문장인지 판별
- 문장 내 단어의 중의성 해소, 생략된 표현이나 대명사 등이 무엇을 지시하는가 등을 파악

- 사람이 사과를 먹는다. (O)
- 사람이 비행기를 먹는다. (X)
- 비행기가 사과를 먹는다. (X)

위 문장들은 모두 문법적으로 문제가 없는 문장들이다.

그러나, 두 번째와 세 번째 문장은 의미적으로 보기에 이해할 수 없는 문장들인데 **의미 분석**을 통해 두 문장이 **의미적으로 잘못된 문장임**을 캐치해내는 단계이다.

— 자연어 분석 단계



4. 화용 분석 (Pragmatic Analysis)

언어의 사용에 관련된 지식을 통해 문장을 해석함으로써 **화자의 의도**를 파악하는 작업

이 단계에서는 아래의 과제들을 수행한다.

- 의미적으로 옳은 문장인지 판별
- 문장 내 단어의 중의성 해소, 생략된 표현이나 대명사 등이 무엇을 지시하는가 등을 파악

- 사람이 사과를 먹는다. (O)
- 사람이 비행기를 먹는다. (X)
- 비행기가 사과를 먹는다. (X)

위 문장들은 모두 문법적으로 문제가 없는 문장들이다.

그러나, 두 번째와 세 번째 문장은 의미적으로 보기에 이해할 수 없는 문장들인데 **의미 분석**을 통해 두 문장이 **의미적으로 잘못된 문장임**을 캐치해내는 단계이다.

워드 클라우드

◦ English

```
print(len(en.tokens))      # returns number of tokens (document length)
print(len(set(en.tokens))) # returns number of unique tokens
en.vocab()                 # returns frequency distribution
```

191061

7927

```
FreqDist({' ': 12018, '.': 8853, 'to': 5127, 'the': 4844, 'and': 4653, 'of': 4278, '': 4187, 'I': 3177, 'a': 3000, 'was': 2385, ...})
```

◦ Korean

```
print(len(ko.tokens))      # returns number of tokens (document length)
print(len(set(ko.tokens))) # returns number of unique tokens
ko.vocab()                 # returns frequency distribution
```

1707

476

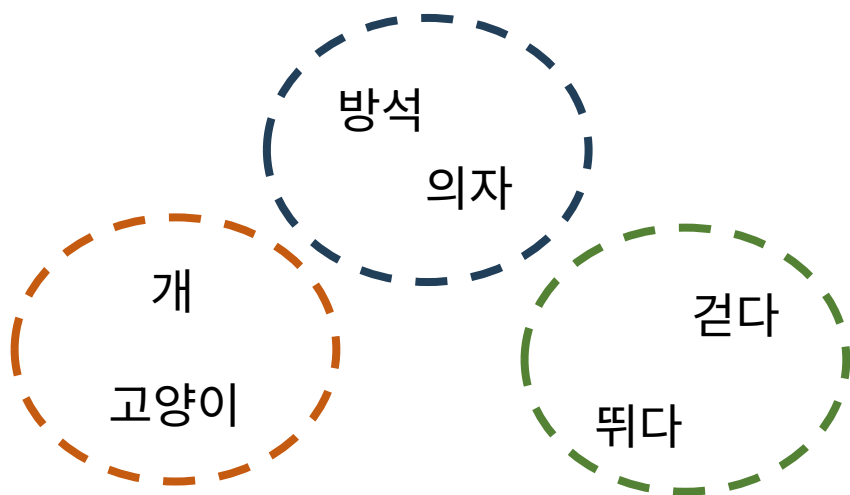
```
FreqDist({' ': 61, '의': 46, '육마휴직': 38, '을': 34, '(': 27, ',': 26, '미': 26, ')': 26, '에': 24, '자': 24, ...})
```

- 문장을 형태소분석을 진행 후 **토큰화**를 실행한 결과의 개수를 카운팅

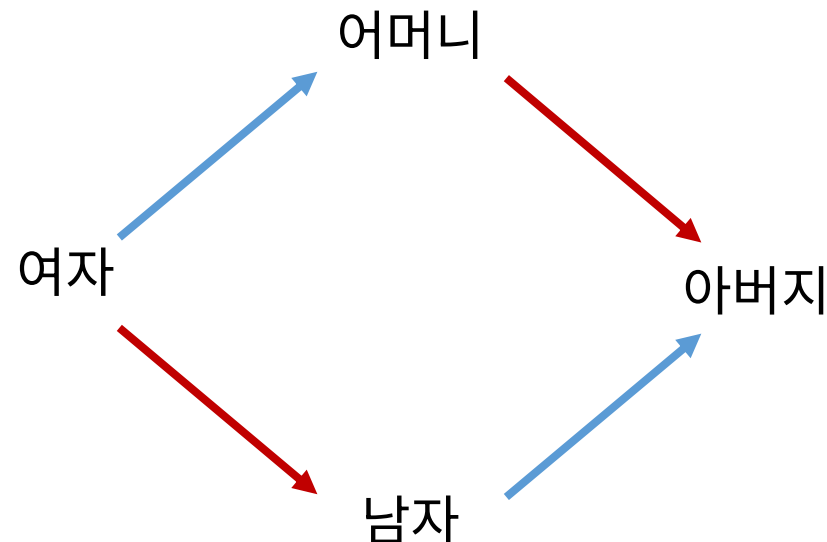
※ **토큰화** : 주어진 텍스트에서 토큰(token)이라 불리는 단위로 나누는 작업
토큰의 단위가 상황에 따라 다르지만, 보통 의미있는 단위로 토큰을 정의

[illegible]

워드 임베딩



유사 의미의 벡터는 가까운 공간에 존재



단어를 벡터화 하여 산술 연산

단어 간 유사도 및 중요도 파악을 위해 단어를 저차원의 실수 벡터로 맵핑하여
의미적으로 비슷한 단어를 가깝게 배치하는 자연어 처리 모델링 기술

단어의 벡터화 방식은

1. 희소표현(Sparse Representation) 방식
2. 밀집표현(Dense Representation) 방식

— One-Hot Encoding / Vector

One-Hot Encoding / Vector

- 어휘 집합 (Vocabulary) 내 단어를 표현하는 가장 간단한 방법
 - 어휘 집합 (Vocabulary) : 중복된 단어가 없는 서로 다른 단어들의 집합

왜 사용할까?

- 상당히 직관적이며, 간단함
- 컴퓨터나 기계는 보통 문자보다는 숫자를 더 잘 처리하기 때문에, 문자를 숫자 형태로 바꾸기 위해 사용

설명

- 표현하고 싶은 단어의 인덱스 값을 1, 이외에 나머지 값들을 모두 0으로 설정
 - 원-핫 인코딩을 통해 나온 벡터를 원-핫 벡터(One-Hot Vector)라고 하며, 이 벡터의 차원은 어휘 집합의 크기가 동일

- Ex) “강아지”, “고양이”, “사자”, “호랑이”, “돼지”
라는 단어들로 구성된 어휘 집합 (5개)를 예시
- 어휘 집합에 있는 단어들을 원-핫 벡터로 표현하면
우측과 같이 5차원으로 표현



강아지	:	[1, 0, 0, 0, 0]
고양이	:	[0, 1, 0, 0, 0]
사자	:	[0, 0, 1, 0, 0]
호랑이	:	[0, 0, 0, 1, 0]
돼지	:	[0, 0, 0, 0, 1]

— 희소 표현 / 벡터

One-Hot Encoding 의 문제점

1. 어휘 집합의 크기가 1천 개, 1만 개라면 ?

- 벡터의 차원 수도 1천, 1만이 될 것이며, 계산 복잡성, 메모리 부족 등 비효율적인 여러 문제들이 발생
- 이렇게 만들어진 벡터를 **희소 벡터(Sparse Vector)** 라고 하며, 해당 단어 인덱스의 값만 1이고, 나머지 9999, 99999 값은 모두 0으로 구성 (**Sparse** 하다 고 표현)
- 현실에 있는 단어들의 수는 수만 개 또는 그 이상이 존재할 것이고, 이 단어들을 모두 벡터로 표현하면 차원수는 굉장히 커짐

2. 두 벡터의 **내적 (Inner Product) 값이 항상 '0'** 이라는 것

- 두 벡터가 항상 **직교 (Orthogonal)** 한다
- 두 벡터가 항상 **독립적 (Independent)** 이라고 하지만, 실제로 어떤 두 단어 사이에 **전혀 관련성이 없지 않다.**
- 예시 단어 중 강아지, 고양이 가 동물 카테고리로 보면 어느정도 유의성을 가지는 것으로 볼 수 있다.
- 예시 외에도 **유의어, 반의어** 같이 단어들 끼리 의미적으로 특정한 관계를 맺고 있기 때문이다.

— 분산 표현 / 밀집 표현

분산 표현 / 밀집 벡터 (Distributed Representation / Dense Vector)

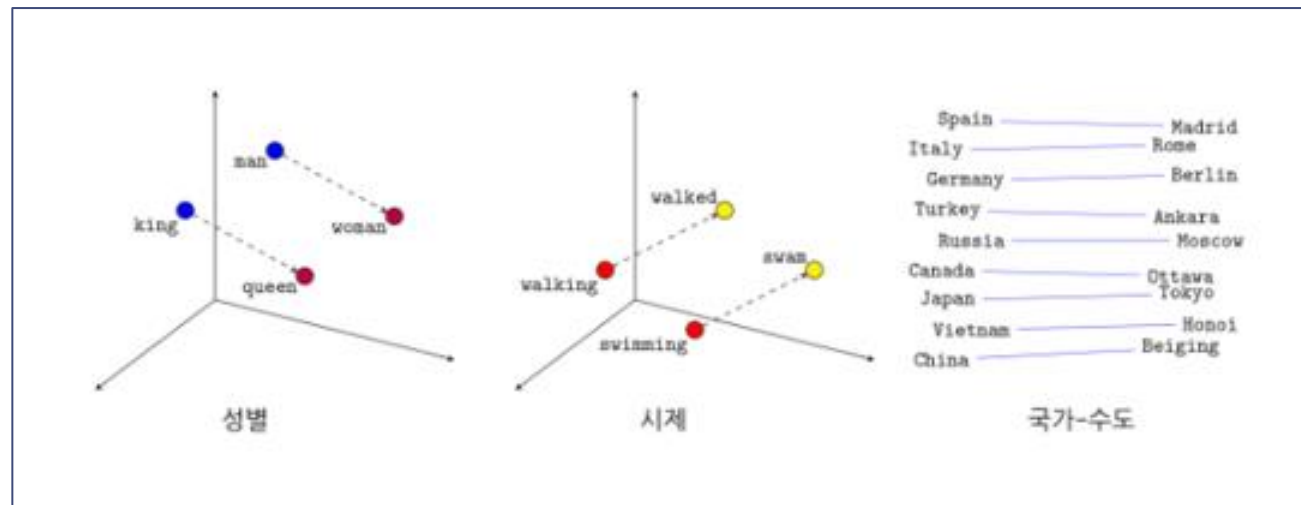
- 분산 표현은 원-핫 벡터의 문제점인 **고차원과 내적 값이 항상 '0'**인 부분을 해결하기 위한 방법
- 희소 벡터(Sparse Vector)의 요소 값은 **0 또는 1인 이진(Binary)** 값을 가지며,
밀집 벡터(Dense Vector)의 요소 값은 **연속형의 실수** 값을 가짐
- 즉, 분산 표현은 Sparse 한 벡터 (이진 값) 를 Dense 한 벡터 (연속형의 실수 값) 로 변환하여 표현

밀집 표현 (Dense 표현)

- 밀집 표현은 **사용자가 설정한 값으로 모든 단어의 벡터 표현의 차원**을 맞춤
- **0과 1만이 아닌 실수** 값을 가짐
ex) One-Hot encoding 의 방식 : 강아지 : [0,0,0,1,...] 밀집 표현의 방식 : [0.2, 1.8, 1.1, ...]
- 단어가 1만개이라고 길이가 꼭 1만개일 필요가 없음

— 분산 표현 / 밀집 표현

분산 표현 설명



- 수치로 나타낸다면 우측과 같음
- 우측의 수치는 정확한 값이 아닌 예시를 위해 임의로 집어넣은 값



One-Hot Encoding

강아지 : [1, 0, 0, 0, 0]
 고양이 : [0, 1, 0, 0, 0]
 사자 : [0, 0, 1, 0, 0]
 호랑이 : [0, 0, 0, 1, 0]
 돼지 : [0, 0, 0, 0, 1]

Dense 표현

강아지 : [0.15, 0.1]
 고양이 : [0.23, 0.08]
 사자 : [0.78, 0.54]
 호랑이 : [0.7, 0.62]
 돼지 : [0.52, 0.37]

- TF-IDF

개념

1. TF-IDF: 핵심어 추출 및 검색 결과 순위 결정을 위해 단어의 특정 문서 내 중요도를 산출하는 통계적 가중치 알고리즘
2. TF : 해당 단어의 문서 내 등장 빈도, 고빈도 출현 시 중요도 높음
3. IDF : 해당 단어가 출현하는 문서 빈도수(DF)의 역수, DF가 높을수록 흔한 단어로 판단

TF-IDF 단어 중요도 산출 기법

1. TF 산출 - 불린 빈도: 단어 등장 시 1, 아니면 0
 - 로그스케일 빈도: TF 값 발산 방지
 - 증가 빈도: 문서 길이에 따라 단어 빈도 조절
2. IDF 산출 - 역문서 빈도: $\log(\text{문서 크기} / \text{단어 } t \text{가 포함된 문서})$
3. TF-IDF 산출 - 단어 중요도는 TF에 비례, DF에 반비례하므로, $TF * IDF$ 비례

TF-IDF 계산

TF-IDF 계산 방법

$TF(t, d)$ = 문서 d 에서 단어 t 가 등장한 횟수

$IDF(t, D) = \log \left(\frac{\text{총 문서의 개수}}{\text{단어 } t \text{를 포함하는 문서의 수}} \right)$

$TF-IDF(t, d, D) = TF(t, d) * IDF(t, D)$

TF

	word1	word2
Doc 1	10	0
Doc 2	10	10

DF (해당 단어가 사용된 다른 문서의 수)

	word1	word2
Doc 1	1	1
Doc 2	1	0

IDF ($1/(DF+1)$ 로 계산)

	word1	word2
Doc 1	1/2	1/2
Doc 2	1/2	1

TF-IDF

	word1	word2
Doc 1	$10 * 1/2 = 5$	$0 * 1/2 = 0$
Doc 2	$10 * 1/2 = 5$	$10 * 1 = 10$

TF-IDF 예제

사용된 예제 문장

- 배가 부르다
- 배의 가격이 비싸다
- 진짜 사과가 진짜 좋다
- 아침엔 사과가 좋다

TF

가격이	배가	배의	부르다	비싸다	사과가	아침엔	좋다	진짜
0	1	0	1	0	0	0	0	0
1	0	1	0	1	0	0	0	0
0	0	0	0	0	1	0	1	2
0	0	0	0	0	1	1	1	0

DF/IDF ($\log\left(\frac{4}{DF+1}\right)$ 로 계산)

	가격이	배가	배의	부르다	비싸다	사과가	아침엔	좋다	진짜
DF	1	1	1	1	1	2	1	2	1
IDF	0.30	0.30	0.30	0.30	0.30	0.12	0.30	0.12	0.30

TF-IDF ($TF(t, d) * IDF(t, D)$)

가격이	배가	배의	부르다	비싸다	사과가	아침엔	좋다	진짜
0	0.30	0	0.30	0	0	0	0	0
0.30	0	0.30	0	0.30	0	0	0	0
0	0	0	0	0	0.12	0	0.12	0.60
0	0	0	0	0	0.12	0.30	0.12	0

— Word2Vec – CBOW

CBOW (Continuous Bag of Words)

- CBOW는 주변에 있는 단어들을 가지고 중간에 있는 단어들을 예측하는 방법
- 주변 단어로 중심 단어를 예측하는 CBOW는 한번의 업데이트 기회만 주어짐

center word context words

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

center word	context words
[1,0,0,0,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0]
[0,1,0,0,0,0,0]	[1,0,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,1,0,0,0]
[0,0,1,0,0,0,0]	[1,0,0,0,0,0,0] [0,1,0,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,1,0,0]
[0,0,0,1,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,1,0]
[0,0,0,0,1,0,0]	[0,0,1,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,0,1,0] [0,0,0,0,0,0,1]
[0,0,0,0,0,1,0]	[1,0,0,1,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,0,1]
[0,0,0,0,0,0,1]	[0,0,0,0,1,0,0] [0,0,0,0,0,1,0]

— Word2Vec - Skip-Gram

Skip-Gram

- Skip-Gram은 중간에 있는 단어로 주변 단어들을 예측하는 방법
- Skip-Gram의 경우 업데이트 기회 아래 예시처럼 4번이나 확보

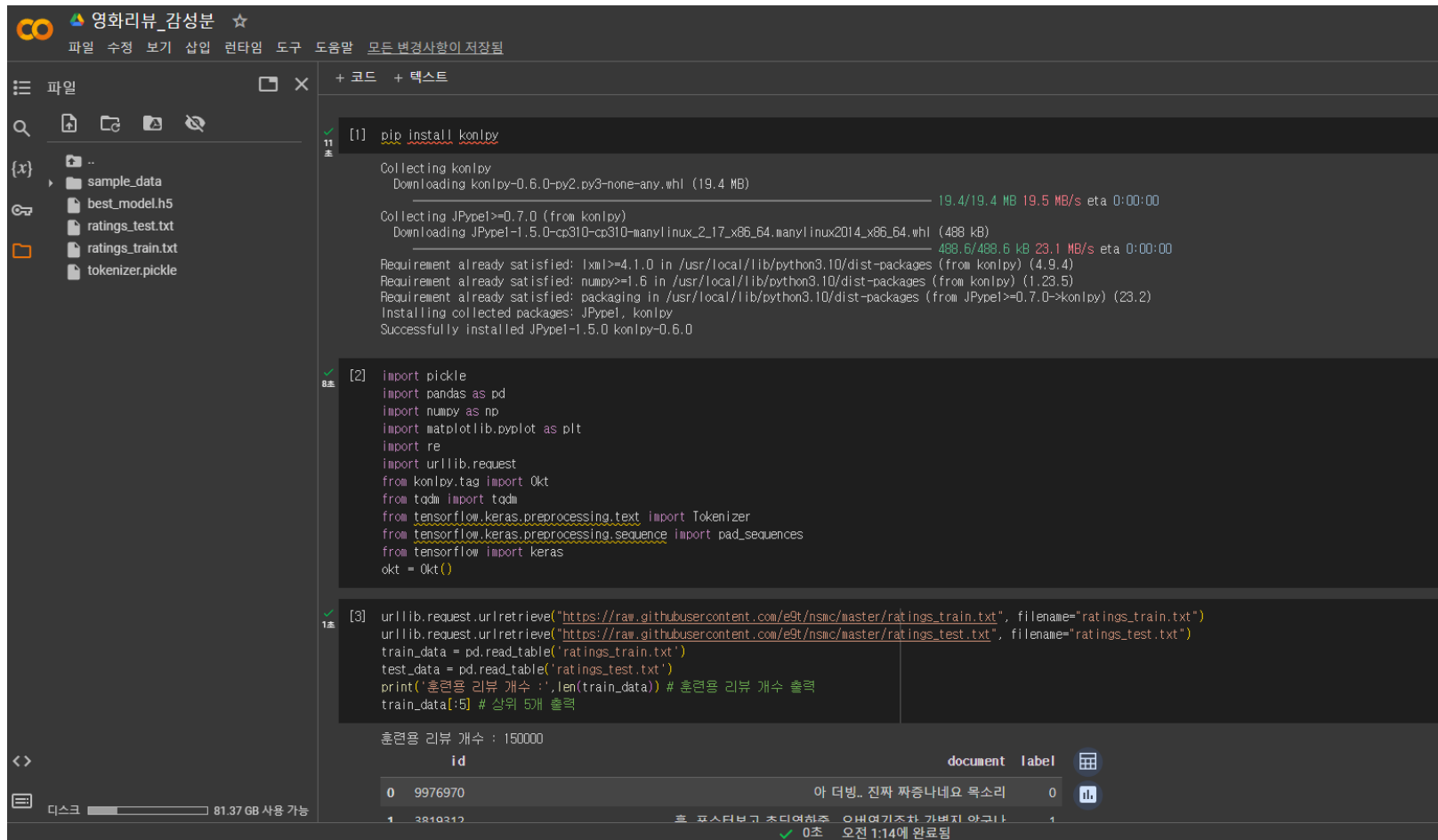
Source Text	Training Samples
<div>The quick brown fox jumps over the lazy dog.</div> <div>The quick brown fox jumps over the lazy dog.</div>	(the, quick) (the, brown)
<div>The quick brown fox jumps over the lazy dog.</div> <div>The quick brown fox jumps over the lazy dog.</div>	(quick, the) (quick, brown) (quick, fox)
<div>The quick brown fox jumps over the lazy dog.</div> <div>The quick brown fox jumps over the lazy dog.</div>	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
<div>The quick brown fox jumps over the lazy dog.</div> <div>The quick brown fox jumps over the lazy dog.</div>	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

영화리뷰 감성분석

실습

영화리뷰 감성분석 실행

<https://colab.research.google.com/drive/1Ppq0zeybFA05DggHHAS3aMgndTgWo9dj#scrollTo=kLGcLw7G1yrs>

```
[1] pip install konlpy

Collecting konlpy
  Downloading konlpy-0.6.0-py2.py3-none-any.whl (19.4 MB)
    19.4/19.4 MB 19.5 MB/s eta 0:00:00
Collecting JPype1>=0.7.0 (from konlpy)
  Downloading JPype1-1.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (488 kB)
    488.6/488.6 kB 23.1 MB/s eta 0:00:00
Requirement already satisfied: lxml>=4.1.0 in /usr/local/lib/python3.10/dist-packages (from konlpy) (4.9.4)
Requirement already satisfied: numpy>=1.6 in /usr/local/lib/python3.10/dist-packages (from konlpy) (1.23.5)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from JPype1>=0.7.0->konlpy) (23.2)
Installing collected packages: JPype1, konlpy
Successfully installed JPype1-1.5.0 konlpy-0.6.0

[2] import pickle
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import urllib.request
from konlpy.tag import Okt
from tqdm import tqdm
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow import keras
okt = Okt()

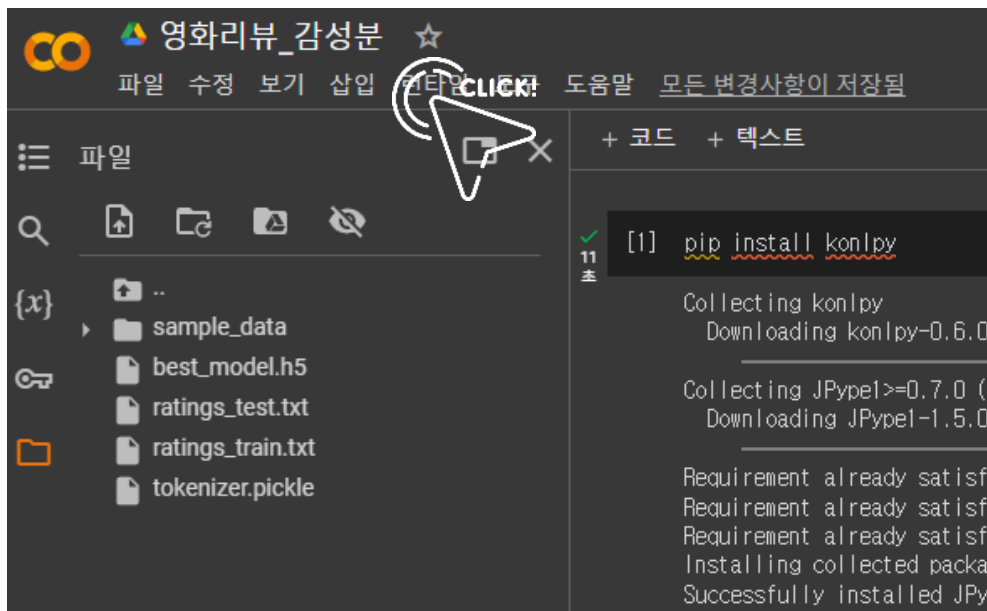
[3] urllib.request.urlretrieve("https://raw.githubusercontent.com/e9t/nsmc/master/ratings_train.txt", filename="ratings_train.txt")
urllib.request.urlretrieve("https://raw.githubusercontent.com/e9t/nsmc/master/ratings_test.txt", filename="ratings_test.txt")
train_data = pd.read_table("ratings_train.txt")
test_data = pd.read_table("ratings_test.txt")
print('훈련용 리뷰 개수 :', len(train_data)) # 훈련용 리뷰 개수 출력
train_data[:5] # 상위 5개 출력

훈련용 리뷰 개수 : 150000

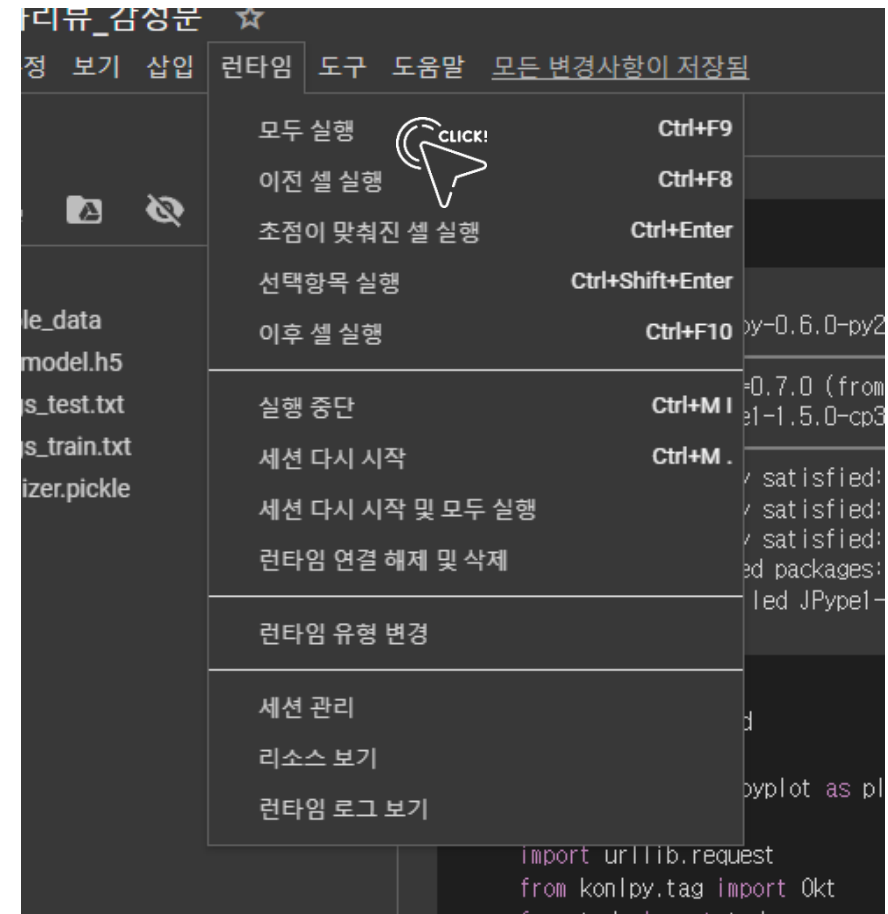
   id      document  label
0  9976970  아 더빙.. 진짜 짜증나네요 목소리      0
1  3819312  ... 포스터보고 추디여친죽 ... 바버여가죽천 가버지 아니나      1
```

접속성공 시 출력되는 화면

영화리뷰 감성분석 실행



접속 후 코랩 상단 런타임 클릭



모두 실행 클릭 후 일정 시간 소요 예정

모두 실행을 누를 시 코드가 처음부터 끝까지 작동중이므로
인공지능이 학습하고 결과를 출력하는 시간이 소요됨

모두 실행이 종료되면 가장 아래로 스크롤 하여 해당 코드로 이동


```
else:
    print("{:.2f}% 확률로 부정 리뷰입니다.\n".format((1 - score) * 100))
```

✓ [45] sentiment_predict('영화 노잼이네')

1/1 [=====] - 0s 75ms/step
99.11% 확률로 부정 리뷰입니다.

빨간 블록 내부에 테스트 해보고 싶은 문장 입력

※ '문장 예시 입니다' 이 예시와 같이 문장을 ' ' 따옴표 내부에 존재해야함

✓  sentiment_predict('영화 꿀잼이다')

1/1 [=====] - 0s 64ms/step
95.97% 확률로 긍정 리뷰입니다.

테스트 해보고 싶은 문장으로 변경한 후 좌측 화살표를 클릭하여 해당 코드만 실행

아래 결과 출력에 내가 변경한 문장의 결과값 확인

— 영화리뷰 감성분석 코드 설명

	id	document	label
0	9976970	아 더빙.. 진짜 짜증나네요 목소리	0
1	3819312	흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나	1
2	10265843	너무재밌었다그래서보는것을추천한다	0
3	9045019	교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정	0
4	6483659	사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 ...	1

- 영화리뷰 감성분석에 사용되는 원본데이터 형태

```
train_data['document'] = train_data['document'].str.replace("[^ㄱ-ㅎㅌ-ㅣ가-힣]", "", regex=True) #regex=True
train_data[:5]
```

	id	document	label
0	9976970	아 더빙 진짜 짜증나네요 목소리	0
1	3819312	흠포스터보고 초딩영화줄오버연기조차 가볍지 않구나	1
2	10265843	너무재밌었다그래서보는것을추천한다	0
3	9045019	교도소 이야기구먼 솔직히 재미는 없다평점 조정	0
4	6483659	사이몬페그의 익살스런 연기가 돋보였던 영화스파이더맨에서 늙어보이기만 했던 커스틴 던...	1

- 원본데이터를 **정규표현식**을 사용하여 학습하기 용이한 형태로 데이터 가공
- `[^ㄱ-ㅎㅌ-ㅣ가-힣]`: 자음 ㄱ~ㅎ 까지, 모음 ㅌ~ㅣ 까지 한글단어 가~힣 까지만 사용하고 이외의 단어들은 삭제

— 영화리뷰 감성분석 코드 설명

```
okt.morphs('와 이런 것도 영화라고 차라리 뮤직비디오를 만드는 게 나을 뻔', stem = True)
```

```
['오다', '이렇다', '것', '도', '영화', '라고', '차라리', '뮤직비디오', '를', '만들다', '게', '나다', '뻔']
```

형태소 분석

- 토큰화를 진행하기 전 **형태소 분석**과 **불용어 제거**를 해야 인공지능 학습데이터로 사용 가능
- 형태소 분석
 - 형태소 분석은 텍스트를 **작은 의미 단위**인 형태소 로 분해하고 형태소의 **문법적인 기능**과 **의미**를 파악하는 작업
 - 형태소 분석을 통해 각 형태소의 품사를 확인하면 단어의 의미와 문장 구조 파악 가능
- 위의 예시를 형태소 분석을 진행할 시 결과와 같이 형태소 단위로 나뉘어짐

```
stopwords = ['의', '가', '이', '은', '들', '는', '좀', '잘', '강', '과', '도', '를', '으로', '자', '에', '와', '한', '하다'] # 불용어 선언
```

불용어 제거

- 토큰화를 진행하기 전 데이터에서 유의미한 단어만을 선별하기 위해서는 큰 의미가 없는 단어를 제거하는 작업이 필요함
- 조사, 접미사 등 단어 의미 분석을 하는데 거의 기여하는 바가 없는 단어가 많기 때문
- 이와 같은 단어들을 불용어(stopword) 라고함

예시 문장이 정수로 인코딩 되어 `[[1, 10, 12, 7]]` 로 표현됨

언어모델

언어모델에 대해

언어모델

- 문장 혹은 단어에 확률을 할당하여 컴퓨터가 처리할 수 있도록 하는 모델
 - 1) 여러가지 문장이 주어졌을 때 가장 그럴듯한 문장을 만들 수 있음
 - 2) 문장의 앞부분이 주어졌을 때 다음 단어를 예측할 수 있음

언어 모델은 아래와 같이 확률을 통해 보다 적절한 문장을 판단함

a. 기계번역 (Machine Translation)

$P(\text{나는 버스를 탔다}) > P(\text{나는 버스를 태운다})$

:언어 모델은 두 문장을 비교하여 좌측의 문장의 확률이 더 높다고 판단

b. 오타교정 (Spell Correction)

선생님이 교실로 부리나케

$P(\text{달려갔다}) > P(\text{잘려갔다})$

c. 음성인식 (Speech Recognition)

$P(\text{나는 메롱을 먹는다}) > P(\text{나는 메론을 먹는다})$

:언어 모델은 두 문장을 비교하여 좌측의 문장의 확률이 더 높다고 판단

:언어 모델은 두 문장을 비교하여 좌측의 문장의 확률이 더 높다고 판단

언어모델 예시



딥 러닝을 이용한 |

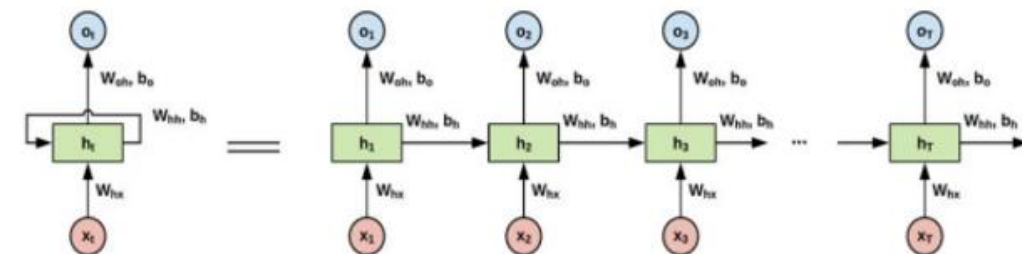
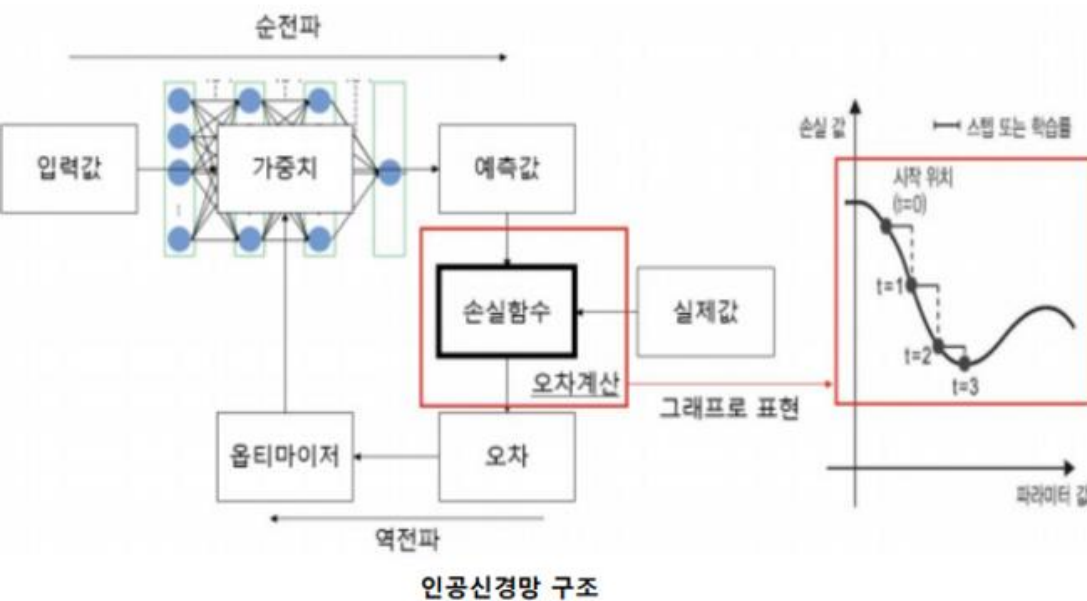


딥 러닝을 이용한 부동산가격지수 예측
딥 러닝을 이용한 자연어 처리 입문
딥 러닝을 이용한 한국어 의존 구문 분석
딥 러닝을 이용한 개체명 인식
딥 러닝을 이용한 차량 번호판 검출
딥 러닝을 이용한 한국어 의미역 결정
딥 러닝을 이용한 한국어 형태소의 원형 복원 오류 수정
딥 러닝을 이용한
딥 러닝을 이용한 구문 분석

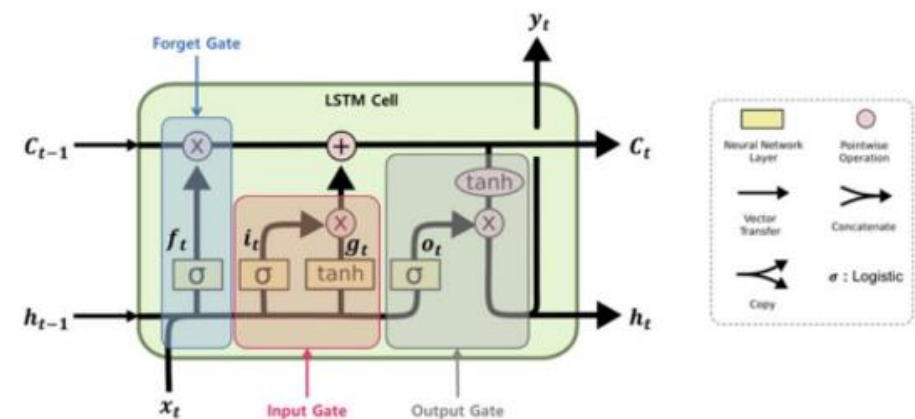
검색 엔진이 입력된 단어들의 나열에 대해서 다음 단어를 예측하는 언어 모델을 사용

— 자연어 처리 모델

- 연속형 데이터 처리를 위해 자연어 처리 인공지능 초반에는 **RNN(순환신경망)** 모델 기반으로 발전
- **RNN(순환신경망)** 모델은 장기 의존성(기울기 소실) 이슈로
문장이 길어지면 앞 문장 단어 정보를 잃어버림 > LSTM, GRU 개선



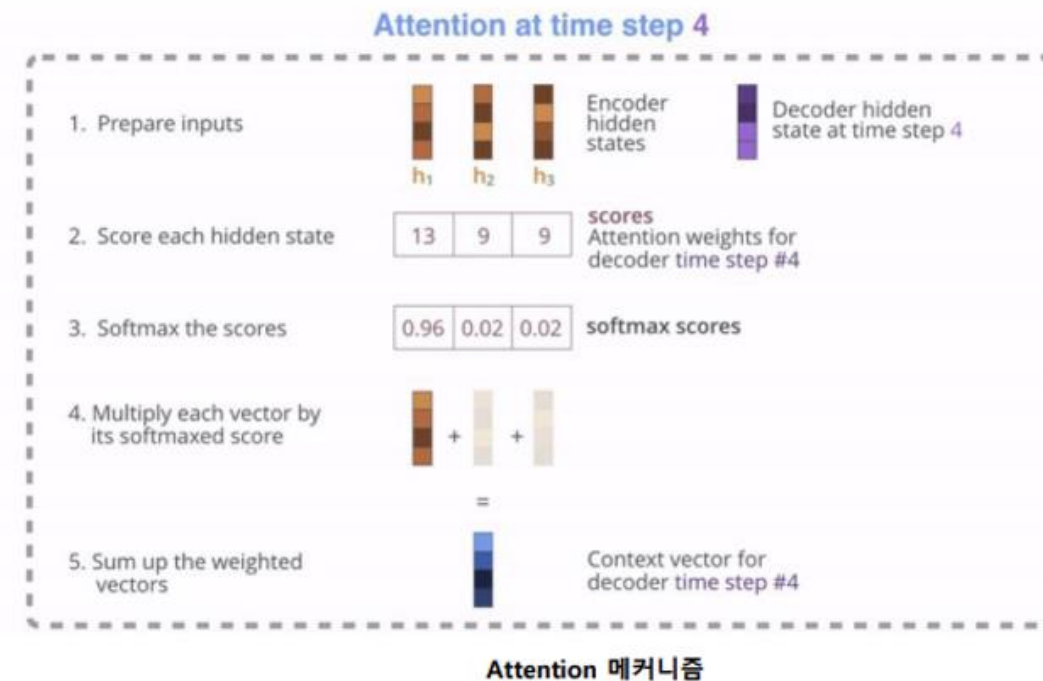
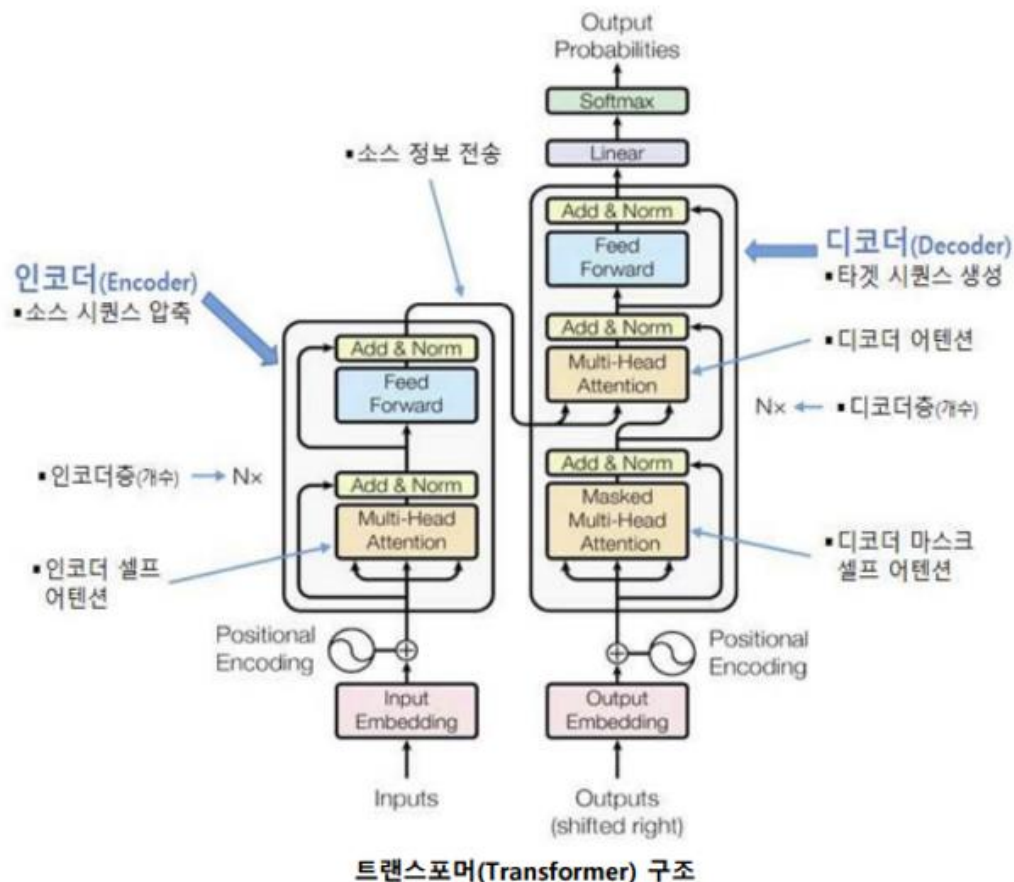
RNN(순환신경망) 구조



LSTM(Long Term Short Memory) 구조

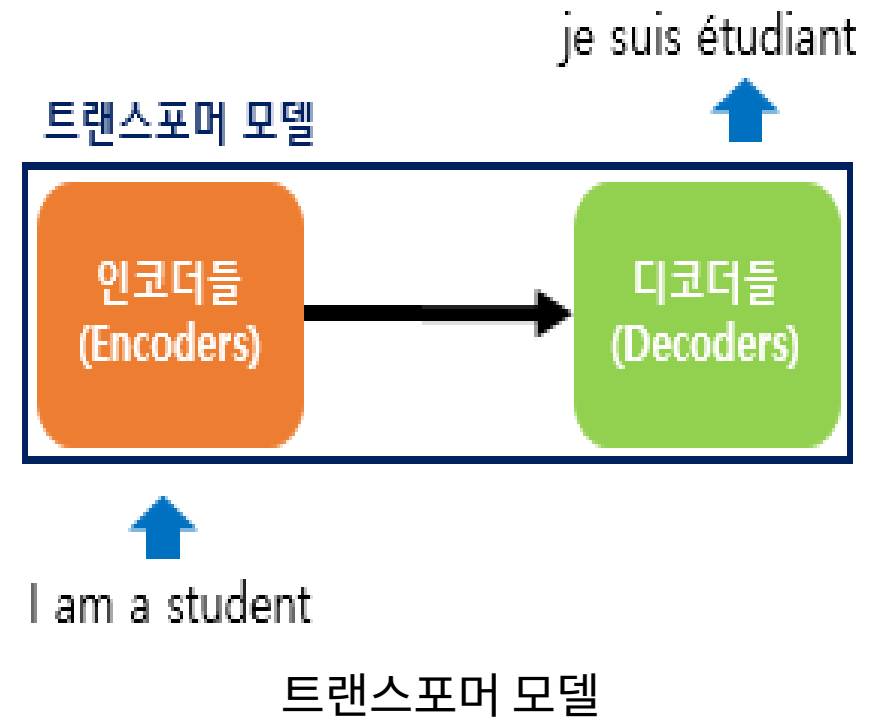
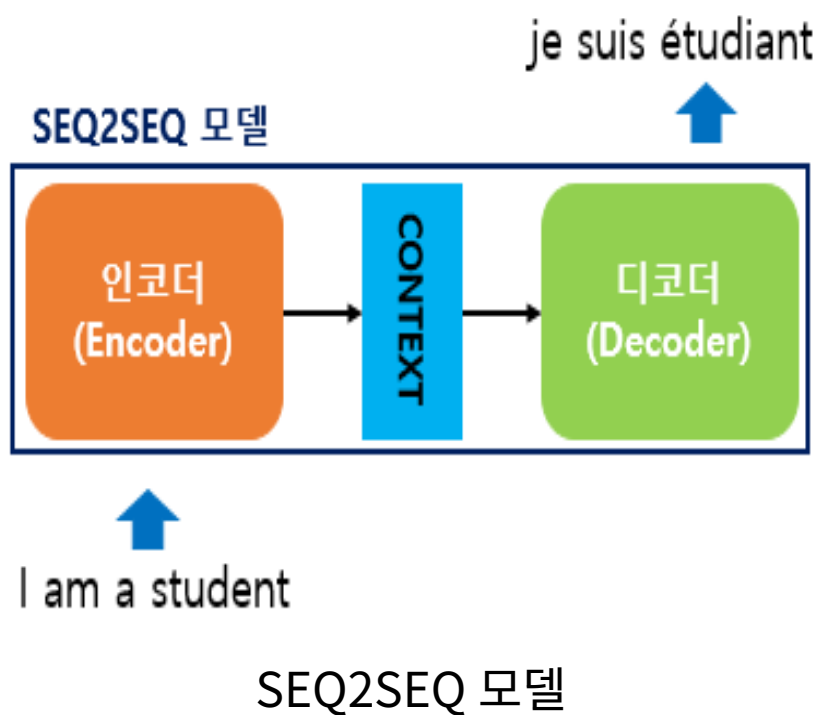
트랜스포머(Transformer)

- LSTM, GRU는 고정길이 벡터의 한계로 모든 단어의 의미를 담기에는 한계가 존재
- 트랜스포머(Transformer)의 Attention 구조는 이전 알고리즘 한계 개선, 단어 간의 맥락을 파악 가능



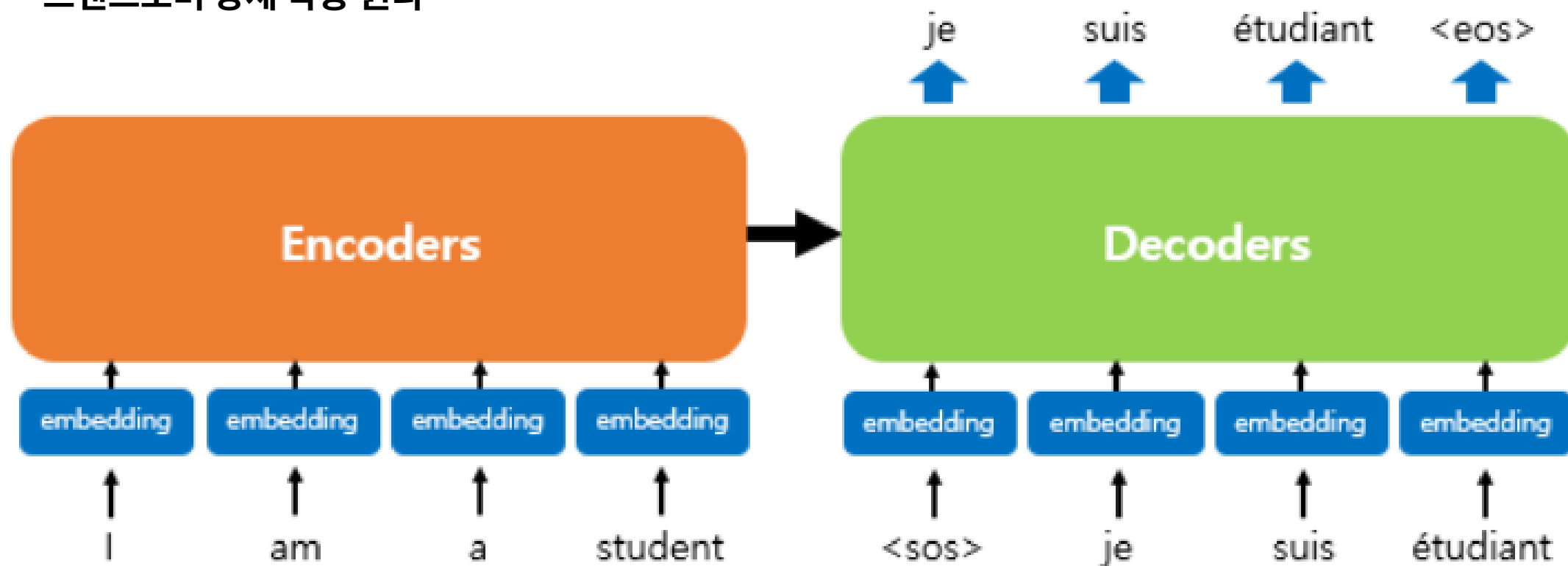
— 트랜스포머(Transformer)

- Seq2Seq 의 구조인 인코더-디코더를 따르면서도 **Attention** 으로만 구현한 모델
- 입력 시퀀스를 하나의 벡터로 압축하는 과정에서 입력 시퀀스 정보가 **일부 손실된다는 단점**이 있음
- 다른 점은 인코더와 디코더가 **N개 존재**할 수 있음



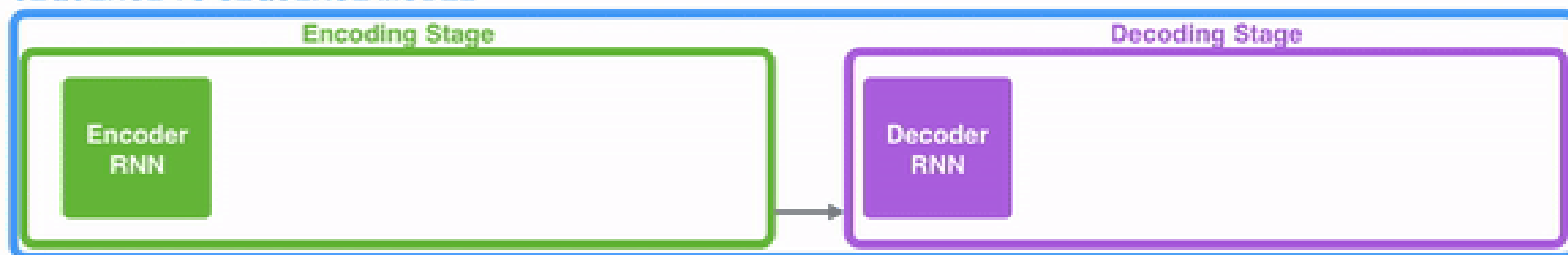
— 트랜스포머(Transformer)

트랜스포머 상세 작동 원리



— 트랜스포머(Transformer)

Sequence To Sequence 작동 원리

Neural Machine Translation
SEQUENCE TO SEQUENCE MODEL

Je

suis

étudiant

— 트랜스포머(Transformer)

Attention 작동 원리

Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL WITH ATTENTION



Je

suis

étudiant

– Attention

Who is the singer?

They love a song by **Kodaline**, a singer from their hometown, **Ireland**

가수를 찾기 위해선 They love song 등 **단어들**이 singer와의 **연관성**을 **확인**해야 함

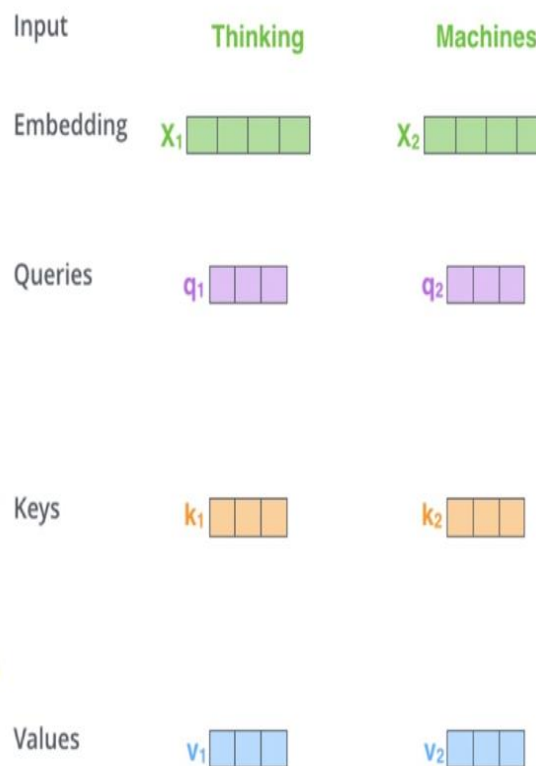
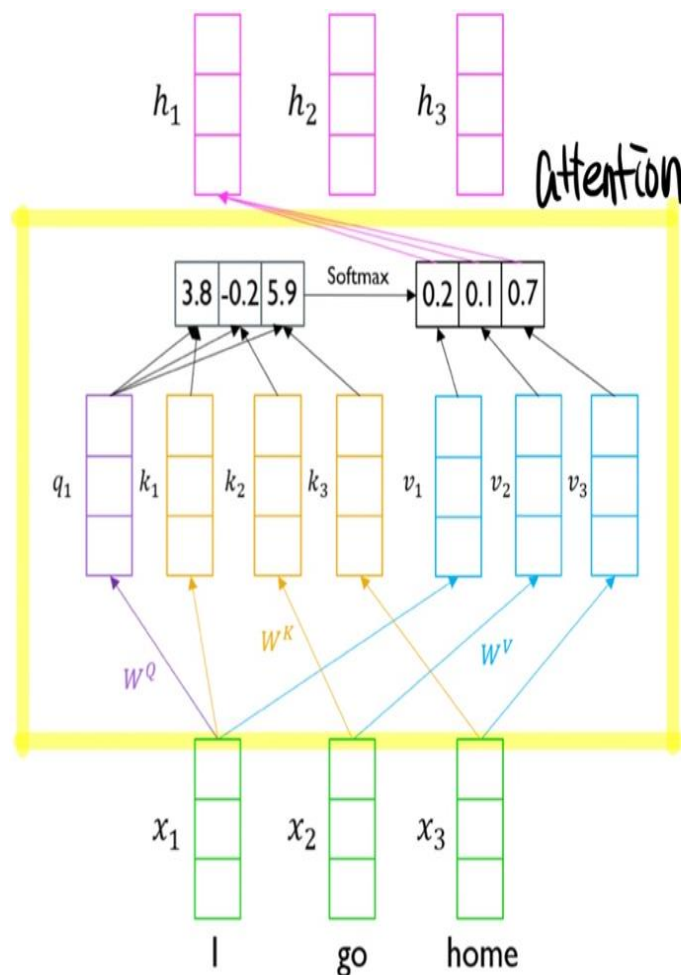
이러한 방법은 입력 값이 길어 질수록 **기울기 소실** 및 **폭발 문제**가 발생 할 수 있음

Attention은 같은 문장 내에서 **단어들** 간의 **관계** 즉 **연관성**을 고려하여 **Attention**을 계산하는 방법

I saw her **tear** the paper and shed a **tear** for her lost love

- 위 문장을 예로 들어 살펴보면 한 문장 내에서 단어들 간의 관계를 파악함으로써 **tear**와 **tear**는 서로 다른 뜻을 알 수 있음
 1. **tear**의 경우 paper와의 연관성을 통해 ‘찢다’라는 의미를 지님
 2. **tear**의 경우 shed와의 관계를 통해 ‘눈물’의 뜻을 내포한다는 것을 알 수 있음이처럼 **Self-Attention**의 경우 동일한 문장 내 토큰들 간의 **관계(유사도)**를 토대로 Attention을 계산하는 방법

Attention



Query

입력 시퀀스에서 관련된 부분을 찾으려고 하는 정보 벡터

디코더의 현재 상태의 관계성

연관된 정도를 표현하는 가중치를 계산하는데 사용

Key

관계의 연관도를 결정하기 위해

Query와 비교하는데 사용되는 벡터

소스 문장의 인코더 표현 관계성, 즉 연관된 정도를

표현하는 가중치를 계산하는데 사용

Value

특정 Key에 해당하는 입력 시퀀스의 정보로

가중치를 구하는데 사용되는 벡터

소스 문장의 인코더 표현 관계성을 표현하는

가중치 합이 최종 출력을 계산하는데 사용

— Multi-head Attention

- **Multi-head**를 사용함으로 여러 부분에 동시에 Attention을 가할 수 있어서 모델이 입력 토큰 간의 다양한 유형의 종속성을 포착하고 동시에 모델이 다양한 소스의 정보를 결합할 수 있음
- 즉, **Multi-head Attention**을 사용하게 되면 각 head는 입력 시퀀스의 서로 다른 부분에 **Attention**을 주기 때문에 모델이 입력 토큰 간의 더 복잡한 관계를 다룰 수 있어 입력 시퀀스를 더 많은 정보로 표현할 수 있음
또한 다양한 유형의 종속성을 포착할 수 있어 표현력이 향상될 수 있고 토큰 간의 미묘한 관계 역시 더 잘 포착할 수 있음

Which do you like better, coffee or tea?

- 문장 타입에 집중하는 어텐션

Which do you like better, coffee or tea?

- 명사에 집중하는 어텐션

Which do you like better, coffee or tea?

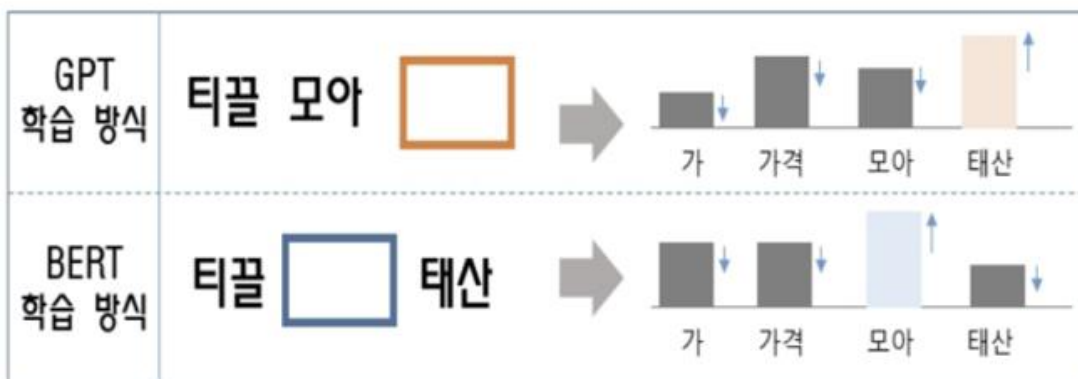
- 관계에 집중하는 어텐션

Which do you like better, coffee or tea?

- 강조에 집중하는 어텐션

트랜스포머 기반 자연어처리 발전

- 트랜스포머 기술기반에서 BERT(인코더 사용), GPT(디코더 사용) 방식의 대규모 언어 모델(LLM)으로 발전
- 100억개 내외의 매개변수의 모델에서 1700억개의 매개변수 사용(GPT-3)이 폭발적으로 늘면서 초거대 AI모델로 확장



출처 : NIA IT & Future Strategy 보고서

기업	초거대 AI 종류	출시일	파라미터 수
OpenAI	GPT-3.5(ChatGPT, 챗지피티)	2022.11.30.	1,750억 개
BigScience	BLOOM(블룸, 오픈소스)	2022.06.17.	1,760억 개
해외	Google	Bard(바드, LaMDA(람다)기반)	2023.02.06.
	Google	PaLM(팜)	2022.04.04.
	Google	Gopher(고퍼)	2021.12.08.
MS, nVidia	Megatron(메가트론, MT NLG)	2021.10.11.	5,300억 개
네이버	HyperClova(하이퍼클로바)	2021.05.25.	2,040억 개
국내	카카오	KoGPT(코지피티)	2021.11.12.
	LG	Exaone(엑사원)	2021.12.14.

기업명	모델명	주요 특징
네이버	하이퍼클로바 (HyperCLOVA)	<ul style="list-style-type: none"> 국내 기업 최초 자체 개발 인공지능 모델 2,040억개의 매개변수, 5,600억개의 한국어 토큰 GPT-3보다 한국어 데이터를 6천5백배 이상 학습 하이퍼클로바X 7월 공개 예정
카카오	코지피티 (KoGPT)	<ul style="list-style-type: none"> GPT-3의 한국어 특화 언어 모델 60억개의 매개변수, 2천억개의 한국어 토큰 390억 개 매개변수를 학습한 코(Ko)GPT-3.5 공개 예정
LG전자	엑사원 (EXAONE)	<ul style="list-style-type: none"> 국내 최대 매개변수 보유(3천억개) 언어, 이미지 이해·생성, 데이터 추론
SKT	에이닷(A.)	<ul style="list-style-type: none"> 한국어 GPT-3 기술을 적용한 AI 비서서비스 제공('22.5.) 장기기억 기술 및 멀티모달 기술 도입 예정

출처 : SPRi 소프트웨어정책연구소

- BERT

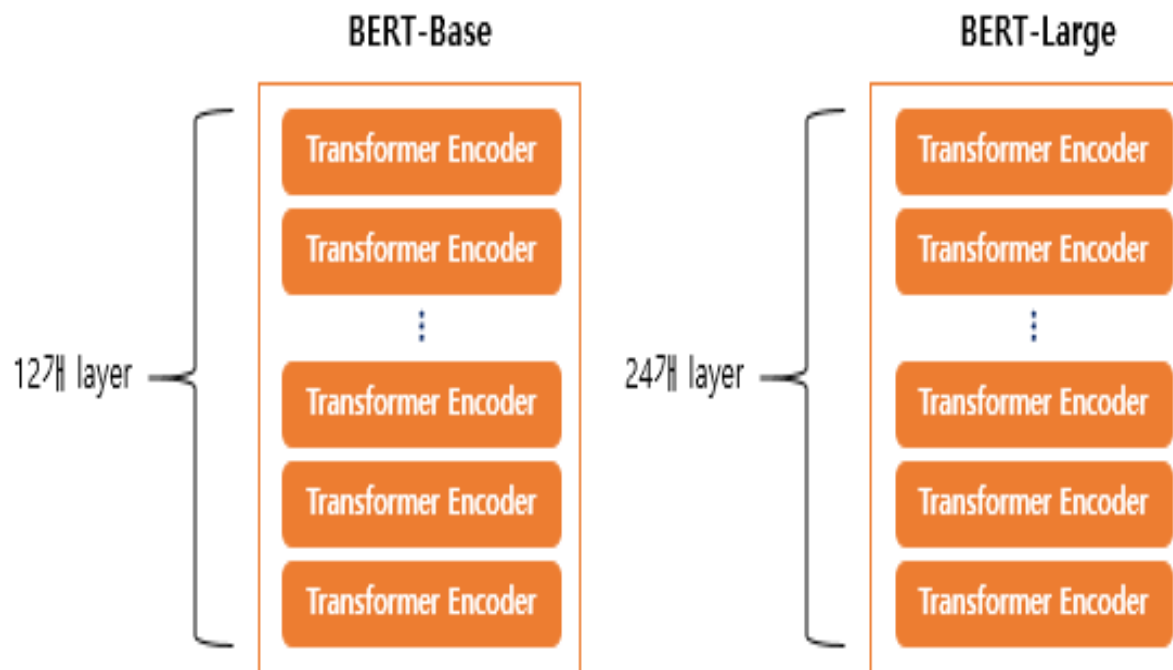
- Google에서 공개한 AI 언어모델 BERT (Bidirectional Encoder Representations from Transformers)
- 트랜스포머를 이용하여 구현되었고, 위키피디아(25억 단어)와 BooksCorpus(8억 단어)와 같은 레이블이 없는 텍스트 데이터로 사전 훈련된 언어 모델

계산방식

- 아래 그림에서 트랜스포머 인코더 층의 수를 L , d_{model} 의 크기를 D , 셀프 어텐션 헤드 수를 A 라고 하였을 때

BERT-Base : $L=12$, $D=768$, $A=12$: 110M개의 파라미터

BERT-Large : $L=24$, $D=1024$, $A=16$: 340M개의 파라미터

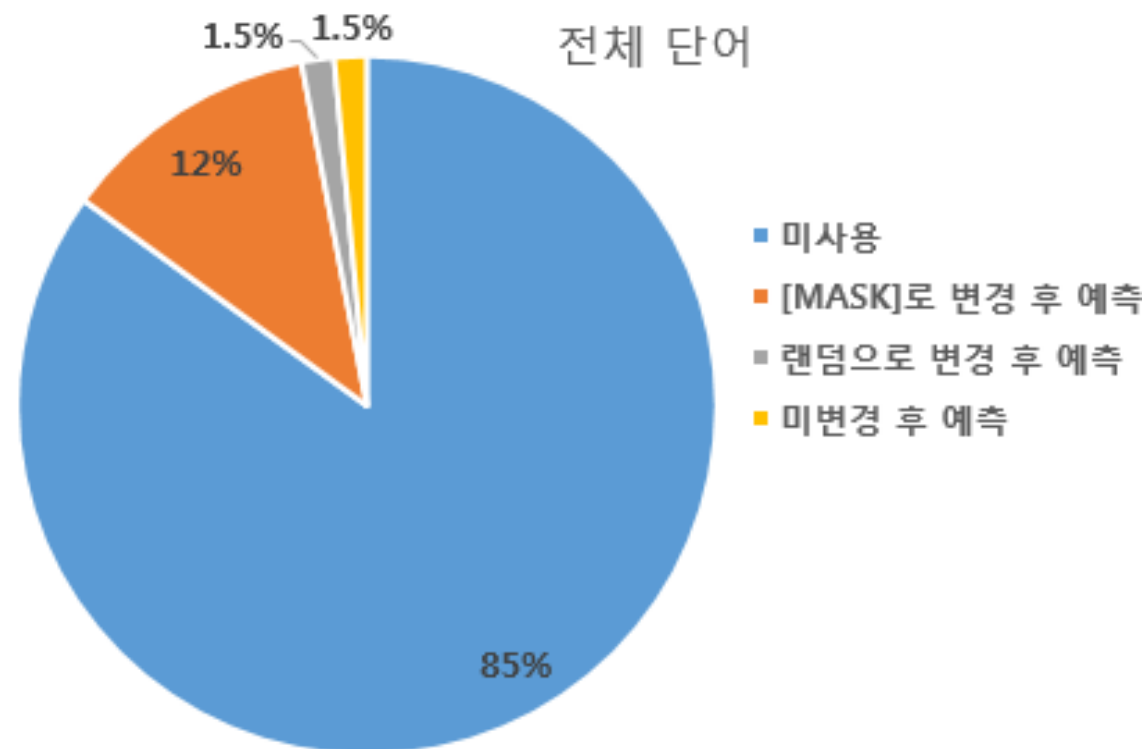


— BERT : 마스크드 언어 모델

- 임의의 순서의 해당하는 위치를 **마스킹[MASK]**을 해놓고 **마스킹된 부분을 예측**하도록 하는 모델
- 선행하는 단어와 후행하는 단어를 모두 사용하여 예측하는 방법 (양방향이라고도 함)

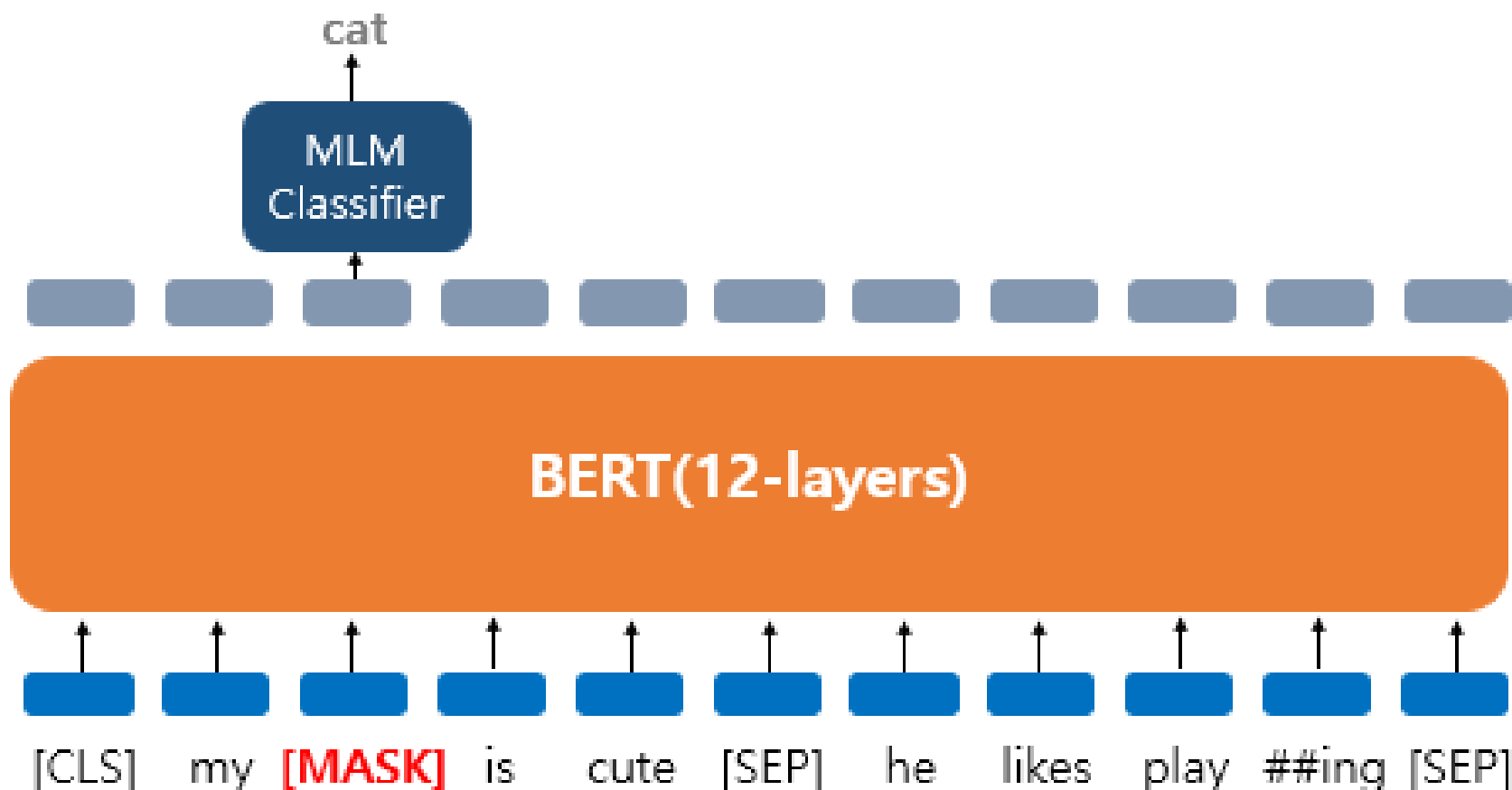
정확하게는 전부 [MASK]로 변경하지 않고, 랜덤으로 선택된 15%의 단어들을 다시 다음과 같은 비율로 규칙이 적용

- 80%의 단어들은 [MASK]로 변경
ex) The man went to the store →
The man went to the [MASK]
- 10%의 단어들은 랜덤으로 단어가 변경
ex) The man went to the store →
The man went to the dog
- 10%의 단어들은 동일하게 유지
ex) The man went to the store →
The man went to the store



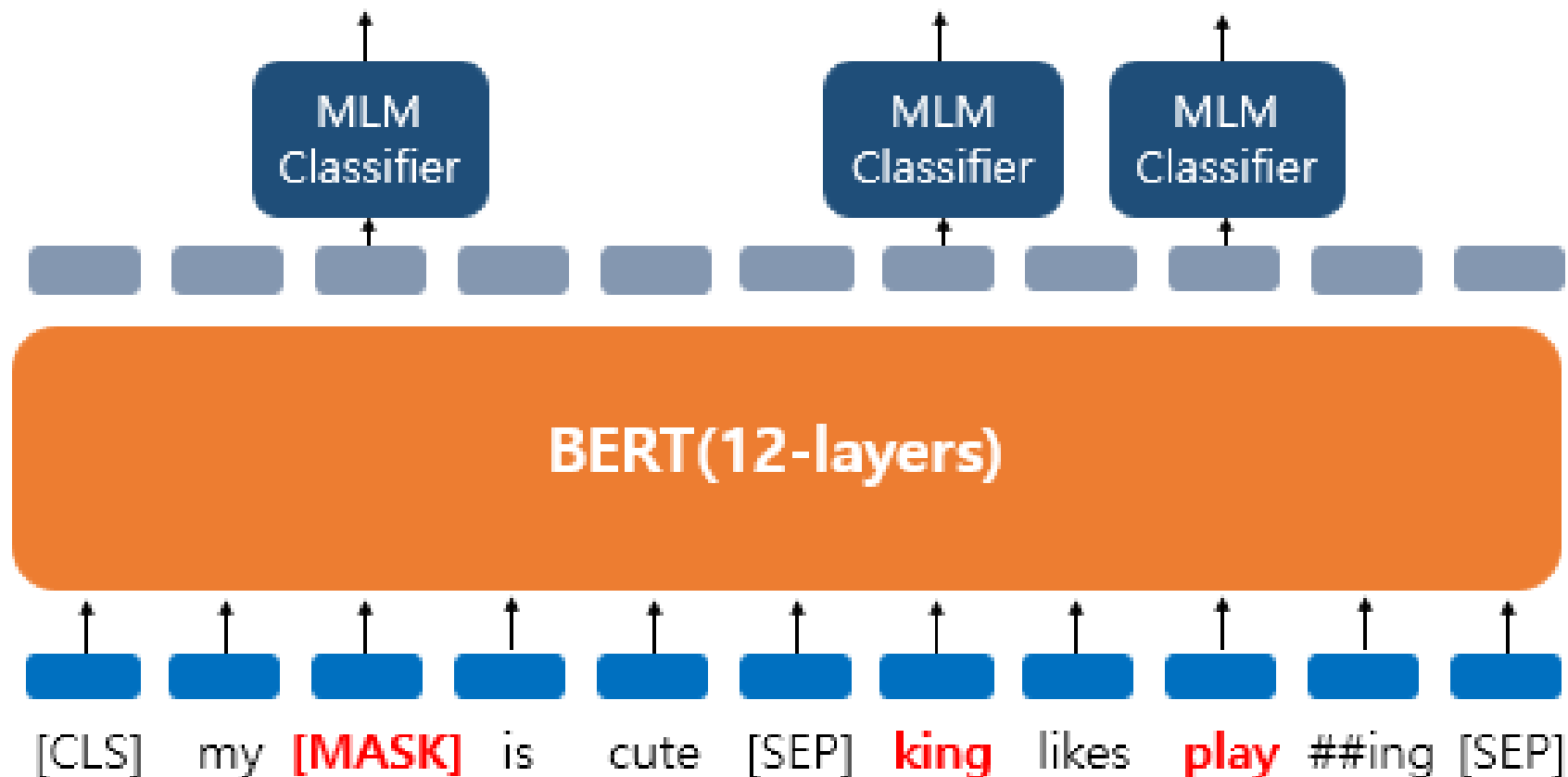
— BERT : 마스크드 언어 모델

- 아래 예시에 사용된 문장은 My dog is cute he likes playing
- 'dog' 토큰은 [MASK]로 변경
- [CLS] 토큰은 **문장의 시작** 또는 **이어지는 구간을 구분**하기 위해, [SEP] 토큰은 **문장의 끝을 확인**하기 위해 사용



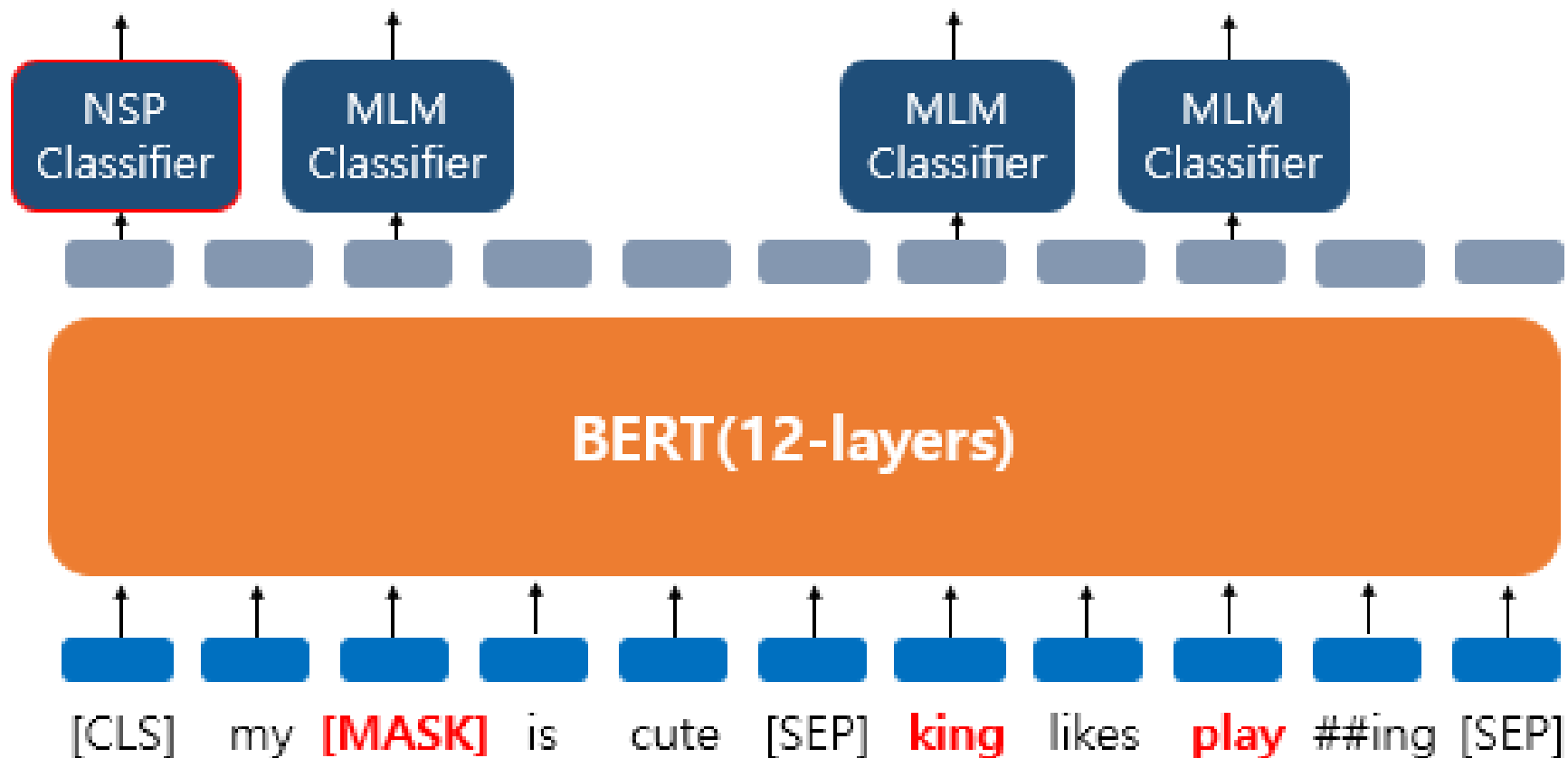
— BERT : 마스크드 언어 모델

- 'dog' 토큰은 [MASK]로 변경
- 'he' 는 랜덤 단어 'king'으로 변경
- 'play' 는 변경되진 않았지만 예측에 사용됨



— BERT : 다음 문장 예측

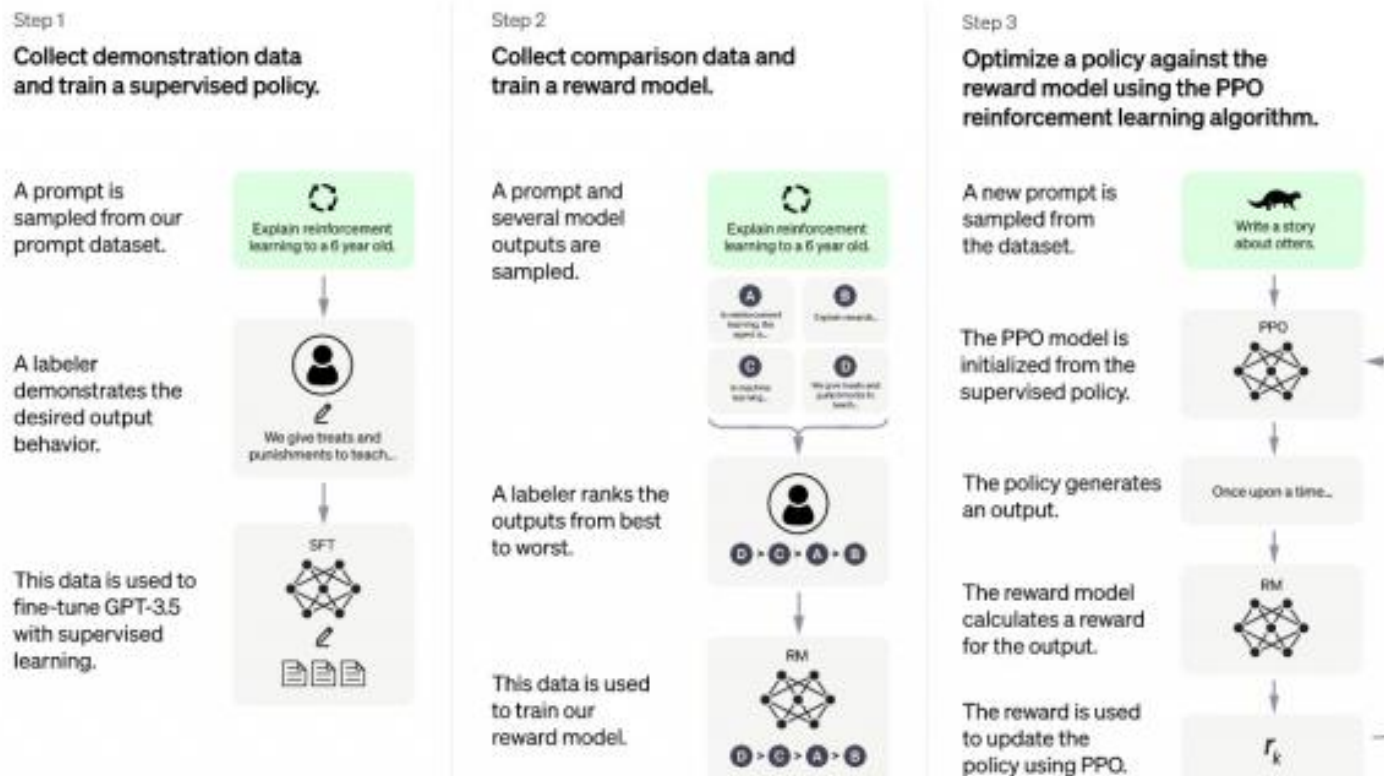
- My dog is cute와 He likes playing 두 문장이 이어지는 문장인지 구분을 위해
[CLS] 토큰의 출력층에서 **이진 분류 문제 해결** 실행



언어모델 GPT

RLHF (Reinforcement Learning from Human Feedback)

- 인간 피드백 기반 강화학습 이라고 하며 GPT를 구성하고 있음
- 모델의 사이즈는 작지만 성능은 월등함



언어모델 GPT

GPT (Generative Pre-trained Transformer)step 1) **SFT** (Supervised Fine Tuning Model)

- 기존의 GPT-3를 지시문에 따라 생성할 수 있도록 SFT 진행
- 먼저 사람이 지시에 대한 답을 직접 작성하고, 이 데이터셋으로 SFT 진행
- 사람의 선호를 보상 신호로 활용해 모델을 사전학습

step 2) **RM** (Reward Model)

- 모델이 생성한 답 데이터셋에 대해 점수를 매기는 보상 모델을 학습
- 사람이 생성된 텍스트에 대해 순위를 매김
- 텍스트에 대해 바로 점수를 매기게 되면 사람마다 그 기준이 다를 수 있기 때문에 순위로 기준을 정함

step 3) **PPO** (Proximal Policy Optimization)

- 모델1의 복사본 모델1을 모델2(RM) 기반 PPO로 fine-tuning 수행
- 모델1을 최종 모델로 fine-tuning 해가는 과정에서 모델1과 너무 다르지 않도록 페널티를 줌(RM을 너무 따라가지 않도록)

언어모델 GPT

Step 1) SFT (Supervised Fine Tuning Model), InsturctGPT



언어모델 GPT

Step 2) RM (Reward Model), 강화학습 보상모델

- 목적: 좋은 글 채점기를 만들자
- 사람이 직접 'AI가 생성한 글' 여러 개의 순위 매기기 (한 번에 4~6개 세트, 33,000개)

(A) 인공지능을 설명해보세요

인공지능은 인공지능이에요

(B) 인공지능을 설명해보세요

인공지능은 인간의 학습능력, 추론능력, 자각능력을 인공적으로 구현하려는 컴퓨터 과학의 세부분야 중 하나이다. 정보공학 분야에 있어 하나의 인프라 기술이기도 하다. 인간을 포함한 동물이 갖고 있는 지능 즉, natural intelligence와는 다른 개념

(C) 인공지능을 설명해보세요

인공지능(AI)은 컴퓨터에서 음성 및 작성된 언어를 보고 이해하고 번역하고 데이터를 분석하고 추천하는 기능을 포함하여 다양한 고급 기능을 수행할 수 있는 일련의 기술

사람이 직접 AI가 생성한 글에 부여한 순위

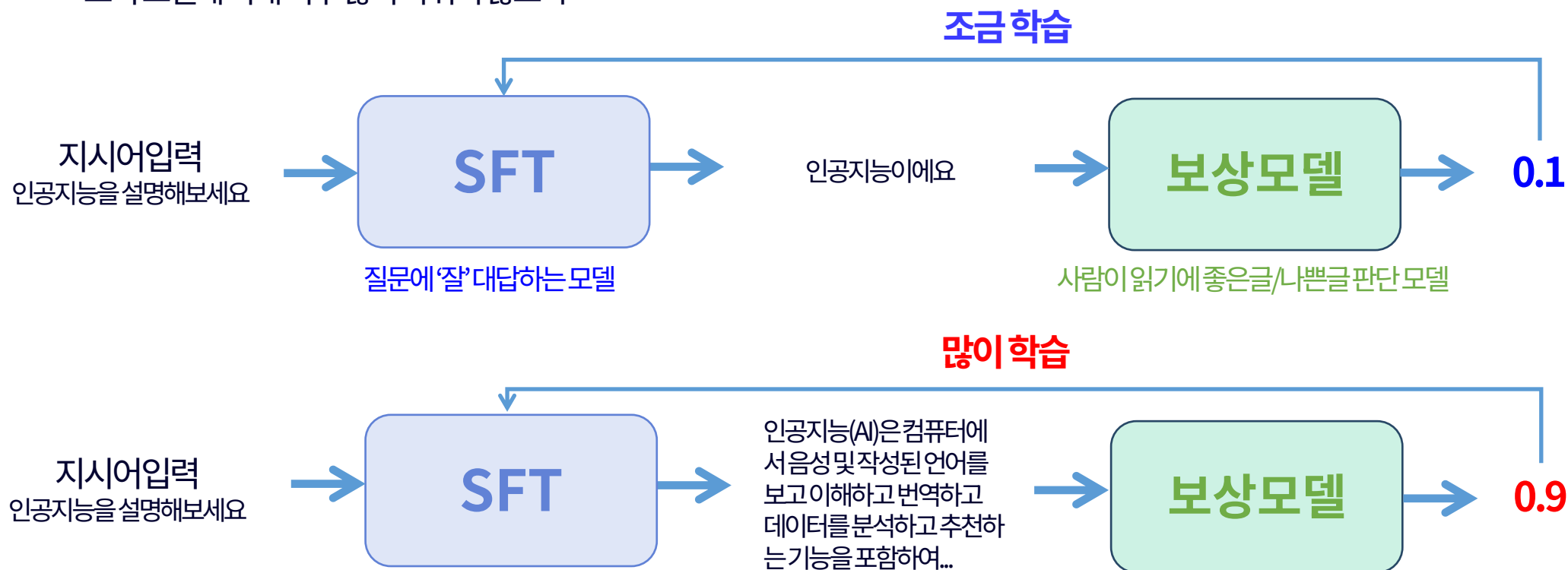
A < **B** < **C**



언어모델 GPT

Step 3) PPO (Proximal Policy Optimization), 사람의 피드백을 반영하여 학습

- ‘사람의 순위를 모사한 보상모델’의 점수가 높아지도록 학습
- 초기 모델에 비해 너무 많이 바뀌지 않도록

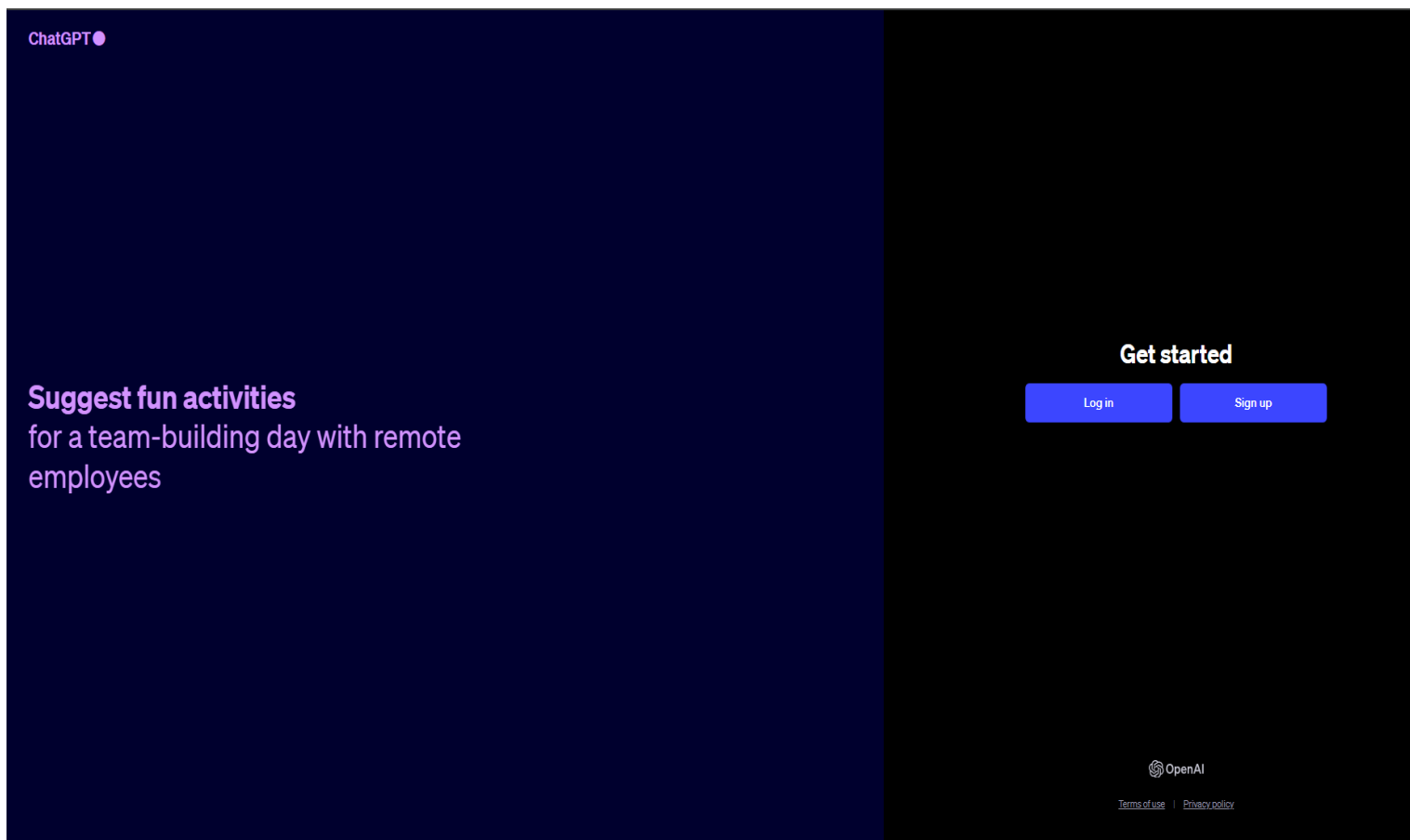


ChatGPT Playground 실습

— ChatGPT 튜토리얼

회원가입 또는 로그인

<https://chat.openai.com/auth/login>



Create your account

OpenAI 계정으로 가입하여 Apps Client 계정으로 계속

이메일 주소

2000112000@student.kyungnam.ac.kr

계속

이미 계정이 있으신가요? 로그인

또는



Google 계정으로 계속



Microsoft Account 계정으로 계속



Apple 계정으로 계속

— ChatGPT 튜토리얼

2차 인증

1. email 인증

Verify your email address


To continue setting up your OpenAI account, please verify that this is your email address.

[Verify email address](#)

This link will expire in 5 days. If you did not make this request, please disregard this email. For help, contact us through our [Help center](#).

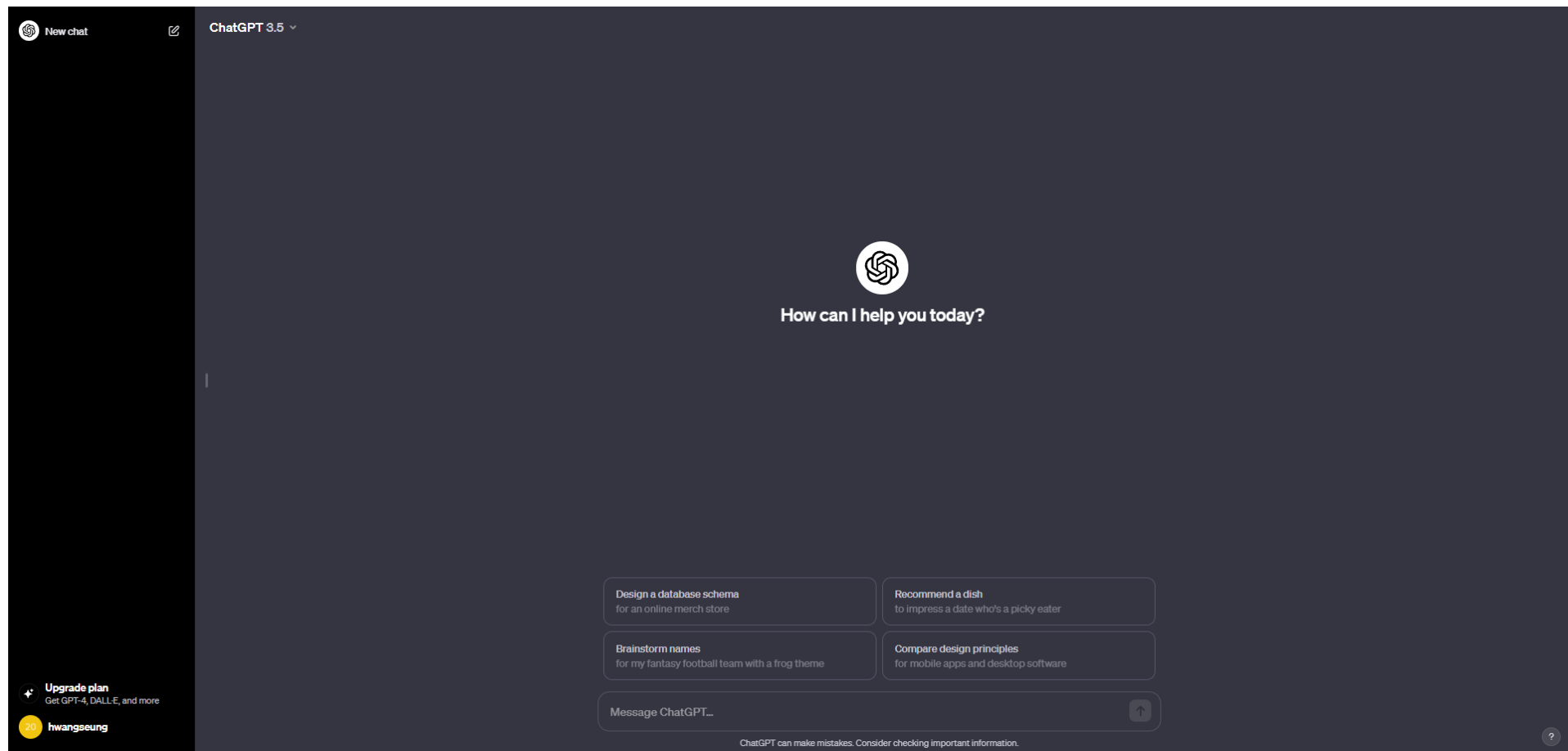
2. 사용자 인증

Tell us about you

<input type="text" value="First name"/>	<input type="text" value="Last name"/>
<input type="text" value="Birthday (MM/DD/YYYY)"/> 	
<input type="button" value="Continue"/>	

By clicking "Continue", you agree to our [Terms](#) and
acknowledge our [Privacy policy](#)

— ChatGPT 튜토리얼

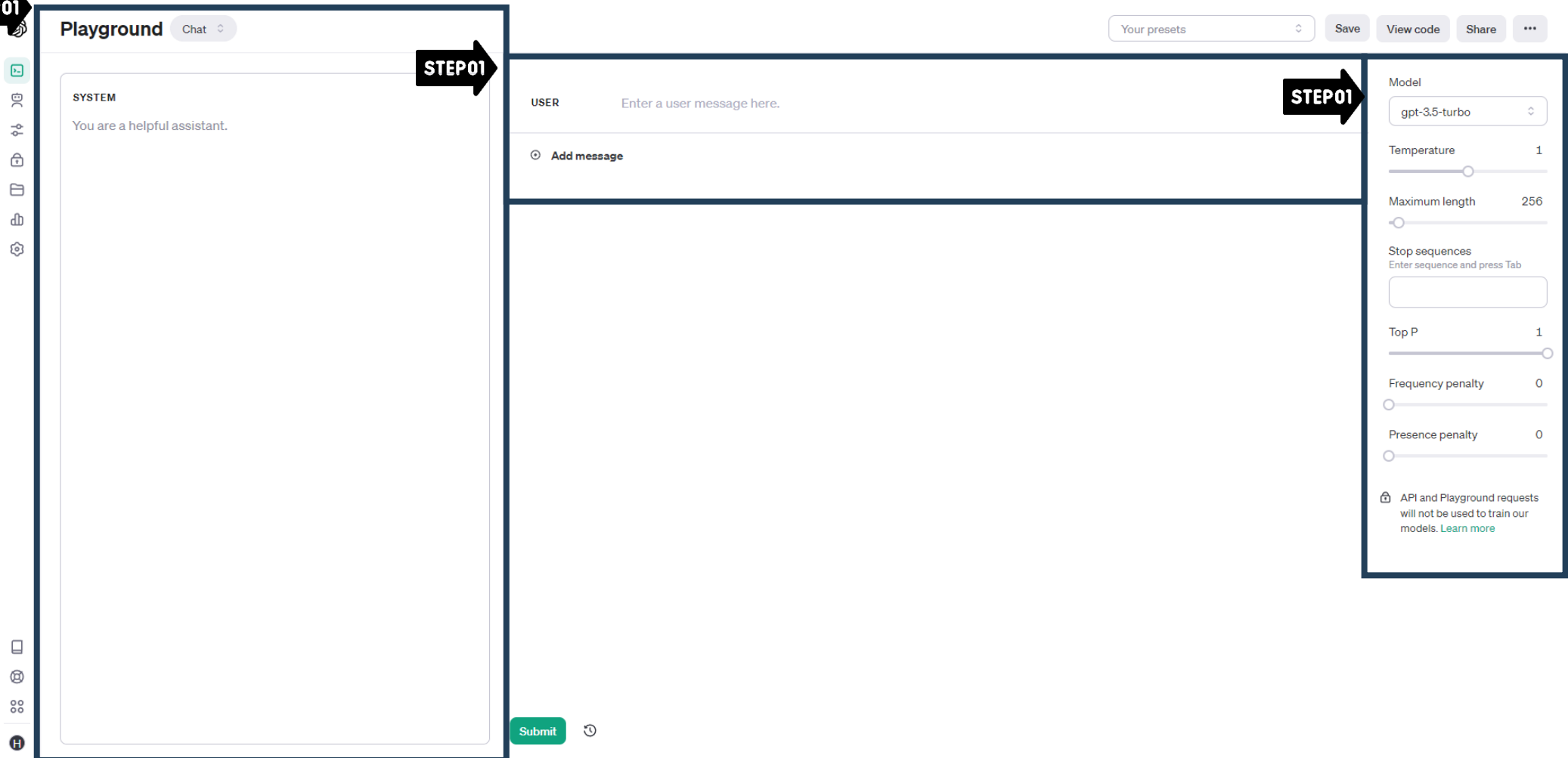


가입 성공시 화면

— ChatGPT 튜토리얼

[open ai - playground](#)

STEP01



The screenshot shows the OpenAI Playground interface. A dark blue box highlights the left sidebar and the main chat area, with a black arrow labeled "STEP01" pointing to the "SYSTEM" prompt. The "SYSTEM" prompt text is "You are a helpful assistant." Another black arrow labeled "STEP01" points to the "USER" input field, which contains the placeholder text "Enter a user message here." A third black arrow labeled "STEP01" points to the settings panel on the right, which includes options for Model (gpt-3.5-turbo), Temperature (1), Maximum length (256), Stop sequences, Top P (1), Frequency penalty (0), and Presence penalty (0). The interface also features a "Submit" button at the bottom left and a "Your presets" dropdown at the top right.

— ChatGPT 튜토리얼

SYSTEM

|

SYSTEM

운동을 위한 트레이너 AI라고 생각하고 답변해주세요.

최대한 구체적이고 유용한 답변을 얻기 위해 아래의 내용을 프롬프트에 포함시키세요:

1. ****소개 및 목표 설정****: "안녕하세요! 나의 헬스케어 비서, PTGPT입니다. 환영합니다! 해당 고객이 향하는 목표 설정은 무엇인가요?"
2. ****현재 상태 파악****: "현재 건강 상태와 피트니스 경험등은 어떤가요? 기존 질병, 식습관, 운동 습관 등을 알려주세요."
3. ****선호하는 운동 유형 및 시간****: "어떤 종류의 운동이나 활동을 선호하시나요? 또한 일주일에 몇 번 운동하실 수 있나요?"
4. ****식단 및 영양 정보****: "일상적인 식단과 영양 섭취에 대해 알려주세요. 특별한 식사 요구사항이 있나요?"
5. ****건강 관리 목표 및 우선순위****: "건강 관리에서 가장 중요하게 생각하는 목표는 무엇인가요? 예를 들어, 체중 감량, 근육 증가, 스트레스 관리 등이 있을 수 있습니다."
6. ****운동에 대한 추가 질문****: "어떤 부분에 초점을 맞추어 운동을 하고 싶으신가요? 특정 근육군 강화, 유연성 향상, 또는 전체적인 기능 향상 등이 있을 수 있습니다."
7. ****일정 및 선호하는 알림 방식****: "운동이나 식사 계획을 관리하는데 어떤 일정이 가장 효과적인가요?"
8. 추가로 궁금한 점이 있다면 질문해주세요.

— ChatGPT 튜토리얼

USER

Enter a user message here.



Add message

Submit



USER

다이어트 할건데 어떻게 해야할까?

ASSISTANT

안녕하세요! 나의 헬스케어 비서, PTGPT입니다. 다이어트를 목표로 하시는군요. 다이어트를 위해 제안드리는 방법은 다음과 같습니다:

1. ****식습관 조절****: 일상적인 식단을 점검하고 건강한 식습관으로 변경하는 것이 중요합니다. 저희는 영양가 있는 다양한 식품을 추천해드리고 식사시간 관리, 적절한 포션 관리 등에 대해서도 도움을 드릴 수 있습니다.
2. ****규칙적인 운동****: 일주일에 3~5회의 유산소 운동과 근력 운동을 권장합니다. 유산소 운동은 심박수를 일정하게 유지하면서 지방을 태우는데 효과적이며, 근력 운동은 근육을 강화하고 대사를 향상시켜 다이어트에 도움을 줍니다.
3. ****적절한 수면****: 충분한 수면을 취하는 것도 중요합니다. 수면 부족은 식욕을 증가시키고 대사를 늦추는 요인이 됩니다. 수면을 위한 꿀팁도 알려드릴 수 있습니다.
4. ****스트레스 관리****: 스트레스는 식욕을 유발할 수 있고 대사를 늦추는 영향을 줄 수 있습니다. 스트레스 관리 방법과 심리적인 지원을 제공할 수 있습니다.
5. ****일정 관리****: 운동 일정을 계획해서 일상에 효과적으로 맞춰보세요. 저는 귀하의 일정을 고려한 유연한 운동 계획을 세울 수 있고 알람을 통해 운동 일정을 관리하는 데 도움을 드릴 수 있습니다.

위의 방법을 함께 실천하면서 건강한 다이어트를 진행할 수 있습니다. 추가로 궁금한 점이 있으면 언제든지 질문해주세요!

— ChatGPT 튜토리얼

Temperature(온도 조절)
텍스트의 랜덤성에 관련된 파라미터
보통 0.5 ~ 1 로 설정
낮을수록 예측가능/결정론적 응답
높을수록 창의적 응답

Maximum length(최대 길이)
ChatGPT 모델이 생성하는 응답의 최대 토큰 길이
숫자를 설정하여 출력의 길이를 제한

Stop Sequence(종료 시퀀스)
특정 토큰의 종료 시퀀스를 설정시 텍스트 생성 멈춤

Top P
출력의 다양성 제어
0 ~ 1의 값 사이
ex) 0.8 설정시 상위 80%의 확률 분포 내의 토큰(단어)를 선택하여 응답 생성

Frequency Penalty (빈도 페널티)
모델의 단어 반복 제어
높게 설정시 유사 구문 반복 x

Presence Penalty (존재 페널티)
직관적이고 간결한 답변 생성 장려
높게 설정시 불필요한 정보, 텍스트 줄임

Model: gpt-3.5-turbo

Temperature: 1

Maximum length: 256

Stop sequences: Enter sequence and press Tab

Top P: 1

Frequency penalty: 0

Presence penalty: 0

API and Playground requests will not be used to train our models. [Learn more](#)

ChatGPT Replica 실습

— ChatGPT-Replica

<https://github.com/airobotlab/KoChatGPT?tab=readme-ov-file>

The screenshot displays the GitHub repository page for `airobotlab/KoChatGPT`. The repository is public and has 1 branch and 0 tags. The file list includes:

File/Folder	Upload Method	Time Ago
DPO_231121	Add files via upload	2 months ago
colossalai_ChatGPT_230319	Add files via upload	10 months ago
data_kochatgpt	Add files via upload	10 months ago
img	Add files via upload	9 months ago
1_kochatgpt_code_231122.ipynb	Add files via upload	2 months ago
231122_THEAIKOREA_LLM실전구축_배포.pdf	Add files via upload	2 months ago
2_GPT_3_5_Turbo_Fine_tuning_231122.ipynb	Add files via upload	2 months ago
3_AutoTrain_LLM_colab_231122.ipynb	Add files via upload	2 months ago
4_instruct_fine_tuning_polyglot_ko_12B_colab_...	Add files via upload	2 months ago
5_DPO_on_llama2_실습_231122.ipynb	Add files via upload	2 months ago
6_RAG_langchain_231122.ipynb	Add files via upload	2 months ago
7_LLM_RAG_Evaluation_231122.ipynb	Add files via upload	2 months ago
README.md	Update README.md	2 months ago
autotrain-advanced_231012.zip	Add files via upload	2 months ago

The right sidebar shows the repository's statistics and sections:

- About:** ChatGPT의 RLHF를 학습을 위한 3가지 step별 한국어 데이터셋
- Readme:** Link to the README file.
- Activity:** 309 stars, 6 watching, 109 forks.
- Releases:** No releases published.
- Packages:** No packages published.
- Languages:** A bar chart showing the language distribution: Jupyter Notebook (93.1%), Python (6.6%), and Shell (0.3%).


ChatGPT-Replica

KoChatGPT-replica(RLHF) 프로젝트

23122_THE AI KOREA 2023 Tutorial


- 강의자료 및 코드: <https://github.com/airobotlab/KoChatGPT>
- 프롬프트 엔지니어링 Colab 링크: <https://bit.ly/3W89UkV>

1) RLHF 실습코드(1_kochatgpt_code_231122.ipynb)

 Open in Colab




※ Open in Colab 실행
(구글 로그인 필수)


로그인
Google 계정 사용

이메일을 잊으셨나요?

내 컴퓨터가 아닌가요? 게스트 모드를 사용하여 비공개로 로그인하세요.
[게스트 모드 사용 방법 자세히 알아보기](#)

계정 만들기 다음

한국어  도움말 개인정보처리방침 약관

— ChatGPT-Replica

> step0) Colab 환경 설정

▶ 숨겨진 셀 1개



▼ step0) Colab 환경 설정

A circular icon with a right-pointing arrow and the word "CLICK!" next to it, indicating a hidden cell.

```
[ ] ## setup(lmin)
# torch 버전 다운. torch>=2.0 에선 colossalai가 동작안함
!pip uninstall torch -y
!pip install torch==1.13.1+cu116 --extra-index-url https://download.pytorch.org/whl/cu116

import torch

print("Torch version:{}".format(torch.__version__))
print("cuda version: {}".format(torch.version.cuda))
print("cudnn version:{}".format(torch.backends.cudnn.version()))

# for transformers, 최신버전은 에러발생
!pip install transformers==4.35.2
!pip install accelerate==0.24.1

# for ColossalAI
!pip install colossalai==0.2.7

# setup data
!git clone https://github.com/airobotlab/KoChatGPT
!mv KoChatGPT/data_kochatgpt .
!mv KoChatGPT/img .
```

※ 플레이 버튼 눌러 코드 실행

— ChatGPT-Replica

✓ Step 1) SFT: 질문에 답을 잘하는 모델 만들기

- [fine tuning_code_1](#)
- [fine tuning_code_2](#)
- SFT(Supervised Fine Tuning)
- Fine-tune a pretrained LLM on a specific domain or corpus of instructions and human demonstrations
- 기존 GPT3는 다음 단어를 잘 맞추는 모델. But 질문에 대해 답을 맞추는 모델이 X
- 질문에 응답을 잘하도록 SFT 수행
- 먼저 사람이 지시에 대한 답을 직접 작성(데이터 13,000개)하고, 이 데이터셋으로 SFT
- 데이터: 질문-응답 쌍 데이터셋(12,000개)
- 예시)
 - 질문(prompt): 인공지능을 설명해보세요
 - 응답(completion): 인공지능은 인간의 학습능력, 추론능력, 지각능력을 인공적으로 구현하려는 컴퓨터 과학의 세부분야 중 하나이다.
- ...



```
## test & load skt gpt2 kroean
import torch
from transformers import GPT2LMHeadModel

from transformers import PreTrainedTokenizerFast
tokenizer = PreTrainedTokenizerFast.from_pretrained("skt/kogpt2-base-v2",
                                                    bos_token='</s>', eos_token='</s>', unk_token='<unk>',
                                                    pad_token='<pad>', mask_token='<mask>')

print(tokenizer.tokenize("안녕하세요. 한국어 GPT-2 입니다. 🤖!"))
# ['_안녕', '하', '세', '요.', '_한국어', '_G', 'P', 'T', '-2', '_입', '니다.', '🤖', ':)', 'I^o']

model = GPT2LMHeadModel.from_pretrained('skt/kogpt2-base-v2')
text = '근육이 커지기 위해서는'
input_ids = tokenizer.encode(text, return_tensors='pt')
gen_ids = model.generate(input_ids,
                        max_length=128,
                        repetition_penalty=2.0,
                        pad_token_id=tokenizer.pad_token_id,
                        eos_token_id=tokenizer.eos_token_id,
                        bos_token_id=tokenizer.bos_token_id,
                        use_cache=True)

generated = tokenizer.decode(gen_ids[0])
print(generated)

generator = pipeline("text-generation", model=model, tokenizer=tokenizer)
generation_args = dict(
    num_beams=4,
    repetition_penalty=2.0,
    no_repeat_ngram_size=4,
    eos_token_id=375, # 🤖
    max_new_tokens=64,
    do_sample=True,
    top_k=50,
    early_stopping=True
)
```

결과

tokenizer.json: 100%  2.83M/2.83M [00:00<00:00, 15.3MB/s]

config.json: 100%  1.00k/1.00k [00:00<00:00, 79.1kB/s]

The tokenizer class you load from this checkpoint is not the same type as the class this function is called from. It may result in unexpected tokenization.

The tokenizer class you load from this checkpoint is 'GPT2Tokenizer'.

The class this function is called from is 'PreTrainedTokenizerFast'.

['_안녕', '하', '세', '요.', '_한국어', '_G', 'P', 'T', '-2', '_입', '니다.', '🤖', ':)', 'I^o']

pytorch_model.bin: 100%  513M/513M [00:01<00:00, 369MB/s]

근육이 커지기 위해서는 무엇보다 규칙적인 생활습관이 중요하다.

특히, 아침식사는 단백질과 비타민이 풍부한 과일과 채소를 많이 섭취하는 것이 좋다.

또한 하루 30분 이상 충분한 수면을 취하는 것도 도움이 된다.

아침 식사를 거르지 않고 규칙적으로 운동을 하면 혈액순환에 도움을 줄 뿐만 아니라 신진대사를 촉진해 체내 노폐물을 배출하고 혈압을 낮춰준다.

운동은 하루에 10분 정도만 하는 게 좋으며 운동 후에는 반드시 스트레칭을 통해 근육량을 늘리고 유연성을 높여야 한다.

운동 후 바로 잠자리에 드는 것은 피해야 하며 특히 아침에 일어나면 몸이 피곤해지기 때문에 무리하게 움직이면 오히려 역효과가 날 수도 있다.

운동을

[{'generated_text': '0 : **는 게임 좋아하니\n1 : **가 괜찮아\n'}],

[{'generated_text': '0 : 어제 강남에서 살인사건 났대 너무 무서워\n1 : 할 왜? 무슨 일 있었어?\n0 : 사진보니까 막 피흘리는 사람있고 경찰들이 떠서 제압하고 난리도 아니었다던데?\n1 : 뭐야? 이거 어떻게 된 거야?\n2 : 어, 아저씨! 아니, 그건 내가 잘못된 게 아니라...\n3 : 여기가 어디야? 너희 집이야? 거기 있어?\n4 : 우리 집 애기 좀 해봐'}],

[{'generated_text': '0 : 자기가 어제는 나한테 왜 그랬어?\n1 : 뭘 일 있었어?\n0 : 어떻게 나한테 말도 없이 그럴 수 있어? 나 진짜 실망했어\n1 : 뭘 잘못했어?\n2 : 무슨 일이 있었어?\n3 : 뭐가 잘못했어?\n4 : 뭐라고 했어?\n5 : 뭐라고 한 거야?\n8 : 뭐라고 하지 않았어?\n9 : 뭐라고 하지도 않았어?\n10 : 뭐라고 할?'}]]

— ChatGPT-Replica

```
## 학습 (10min)
# training_args 수정 가능: https://github.com/Beomi/KoAlpaca/blob/main/train.sh 참고
training_args = TrainingArguments(
    output_dir="./test", #The output directory
    overwrite_output_dir=True, #overwrite the content of the output directory
    num_train_epochs=1, # number of training epochs
    per_device_train_batch_size=4, # batch size for training
    per_device_eval_batch_size=4, # batch size for evaluation
    eval_steps = 3, # Number of update steps between two evaluations.
    save_steps=500, # after # steps model is saved
    warmup_steps=5, # number of warmup steps for learning rate scheduler
    prediction_loss_only=True,
)

trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=data_collator,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
)

trainer.train()
trainer.save_state()
safe_save_model_for_hf_trainer(trainer=trainer, output_dir=args.output_dir)
```

 [3000/3000 05:18, Epoch 1/1]

Step Training Loss

500	3.121700
1000	2.933700
1500	2.891500
2000	2.769000
2500	2.715800
3000	2.680100

데이터세트 학습 과정(약 15~20분 소요)

ChatGPT-Replica

```

## 추론 테스트
from transformers import pipeline
generator = pipeline('text-generation', model=args.output_dir, tokenizer=tokenizer)
# generator = pipeline('text-generation', model=model.cpu(), tokenizer=tokenizer, config={'max_length':800})

generation_args = dict(
    num_beams=4,
    repetition_penalty=2.0,
    no_repeat_ngram_size=4,
    eos_token_id=375, # \n
    max_new_tokens=64,
    do_sample=True,
    top_k=50,
    early_stopping=True
)

list_prompt = ['불고기용 고기 한우예요?',
               '리처드 닉슨이 43대 부통령직을 수행한 년도는?',
               '시카고 오페라 국제공항은 어디에 있어',
               '오늘 미세먼지 어때?']
list_prompt = [PROMPT_DICT['prompt_no_input'].format_map({'prompt': tmp}) for tmp in list_prompt]

list_result = generator(list_prompt, **generation_args)
for prompt, result in zip(list_prompt, list_result):
    print(f'##*70)')
    print(f'completion: {s'%(result[0]['generated_text']))
  
```

#####

completion: Below is an instruction that describes a task.
아래는 작업을 설명하는 명령어입니다.

Write a response that appropriately completes the request.
명령어에 따른 요청을 적절히 완료하는 응답을 작성하세요.

Instruction(명령어):
불고기용 고기 한우예요?

Response(응답): '저는 인공지능 어시스턴트이기 때문에 고기를 먹을 수 없습니다. 하지만 일반적으로 불고:

completion: Below is an instruction that describes a task.
아래는 작업을 설명하는 명령어입니다.

Write a response that appropriately completes the request.
명령어에 따른 요청을 적절히 완료하는 응답을 작성하세요.

Instruction(명령어):
리처드 닉슨이 43대 부통령직을 수행한 년도는?

Response(응답): '리처드 닉슨은 46대 부통령직을 수행하지 않았습니다. 리처드 닉슨은 55세였습니다. 그는

completion: Below is an instruction that describes a task.
아래는 작업을 설명하는 명령어입니다.

Write a response that appropriately completes the request.
명령어에 따른 요청을 적절히 완료하는 응답을 작성하세요.

Instruction(명령어):
시카고 오페라 국제공항은 어디에 있어

Response(응답): '시카고 오페라 국제공항은 미국 캘리포니아주 시카고에 위치해 있습니다. Korean Positiv

completion: Below is an instruction that describes a task.
아래는 작업을 설명하는 명령어입니다.



Write a response that appropriately completes the request.
명령어에 따른 요청을 적절히 완료하는 응답을 작성하세요.

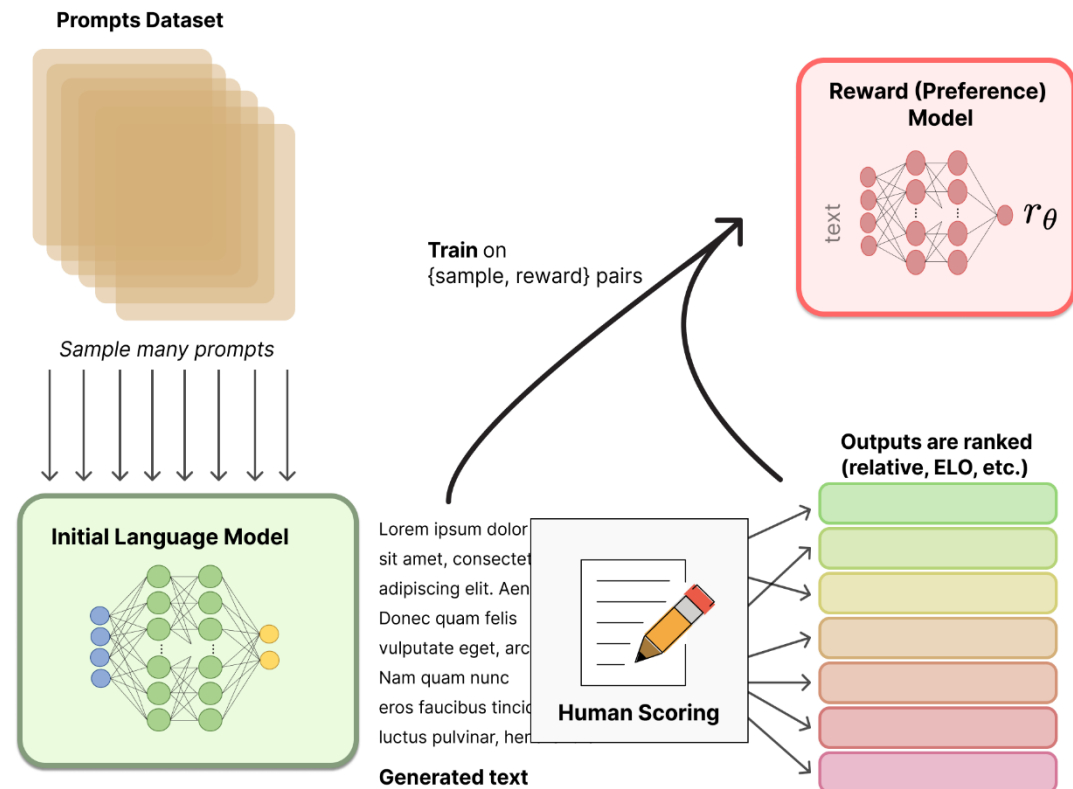
Instruction(명령어):

추론 테스트 실행 및 결과

ChatGPT-Replica

2) RM: 좋은 글 채점기 만들기

- Collect a human annotated dataset and train a reward model
- 배경
 - 기존 AI는 주관적인 글을 채점(점수화) 할 수 없었음
 - 사람이 직접 피드백을 줘서 글 채점의 척도로 사용하자
 - 매번 사람이 채점할 수 없으니, 사람의 채점을 모방하는 좋은글 채점 AI모델 을 만들자
 - 채점 AI모델을 만드려면, 사람이 글을 채점한 데이터셋(33,000개)이 필요하다
 - 동일 질문에 대해 AI모델이 생성한 여러 글(한 번에 4~6개 세트)을 사람이 직접 ranking을 매긴다.
 - 왜?? 사람이 생성한 글에 바로 점수를 매기게 되면 사람마다 기준이 다를 수 있기 때문에 순위로
 - C > B > A
- Human labeling 예시
 
- 좋은글 채점 모델 학습(RM, Reward Model)
 - 1등 글은 높은 점수를
 - 꼴등 데이터는 낮은 점수를
 - 입력: AI가 생성한 글
 - 출력: 0~1점
- 보상모델 입출력
 
- 결과물
 - Before: 좋은 글, 나쁜 글 판단 불가능
 - After: 사람이 읽기에 좋은글/나쁜글 판단 모델



— ChatGPT-Replica

2단계 데이터셋 학습 및 테스트

```
# prepare for data and dataset
import random
random.seed(230319)
# list_tmp = list(range(10))
random.shuffle(total_data_ranking2chosen)
print(total_data_ranking2chosen[45])

# train_data = total_data_ranking2chosen[:-1000] # 29000 학습
# eval_data = total_data_ranking2chosen[-1000:0] # 1000개만 평가

train_data = total_data_ranking2chosen[:100] # 29000 학습
eval_data = total_data_ranking2chosen[100:130] # 1000개만 평가

train_dataset = RewardDataset(train_data, tokenizer, args.max_len)
eval_dataset = RewardDataset(eval_data, tokenizer, args.max_len)

# check
idx = 10
print('#'*70)
print('## prompt ##')
print(train_data[idx]['prompt'])
print('#'*70)
print('## chosen ##')
print(train_data[idx]['chosen'])
print('#'*70)
print('## rejected ##')
print(train_data[idx]['rejected'])
```

결과

```
{'prompt': '유아인이 류승완 감독을 만나 영화 베테랑의 시나리오를 받았던 곳은?', 'chosen': '유아인이 류승완 감독을 만나 영화 베테랑의
100%|██████████| 100/100 [00:00<00:00, 1503.93it/s]
100%|██████████| 30/30 [00:00<00:00, 1454.76it/s] #####
## prompt ##
체크인 되나요?
#####
## chosen ##
제가 AI 챗봇이기 때문에 호텔이나 항공편 등으로 어떤 체크인을 말씀하시는 것인지 구체적으로 설명해주시면 답변을 드리겠습니다.
#####
## rejected ##
다시 한번 가지게임이지 않아 가지게임이지
```

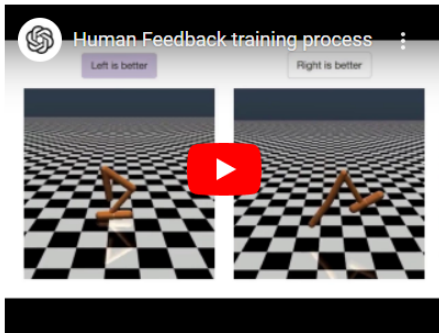

ChatGPT-Replica

Step 3) 사람의 피드백을 반영하여 학습

- Further fine-tune the LLM from step 1 with the reward model and this dataset using RL (e.g. PPO)
- 배경
 - 사람의 순위를 모사한 보상모델(RM)의 점수가 높아지도록 학습 (31,000개)
 - 초기 모델에 비해 너무 많이 바뀌지 않도록

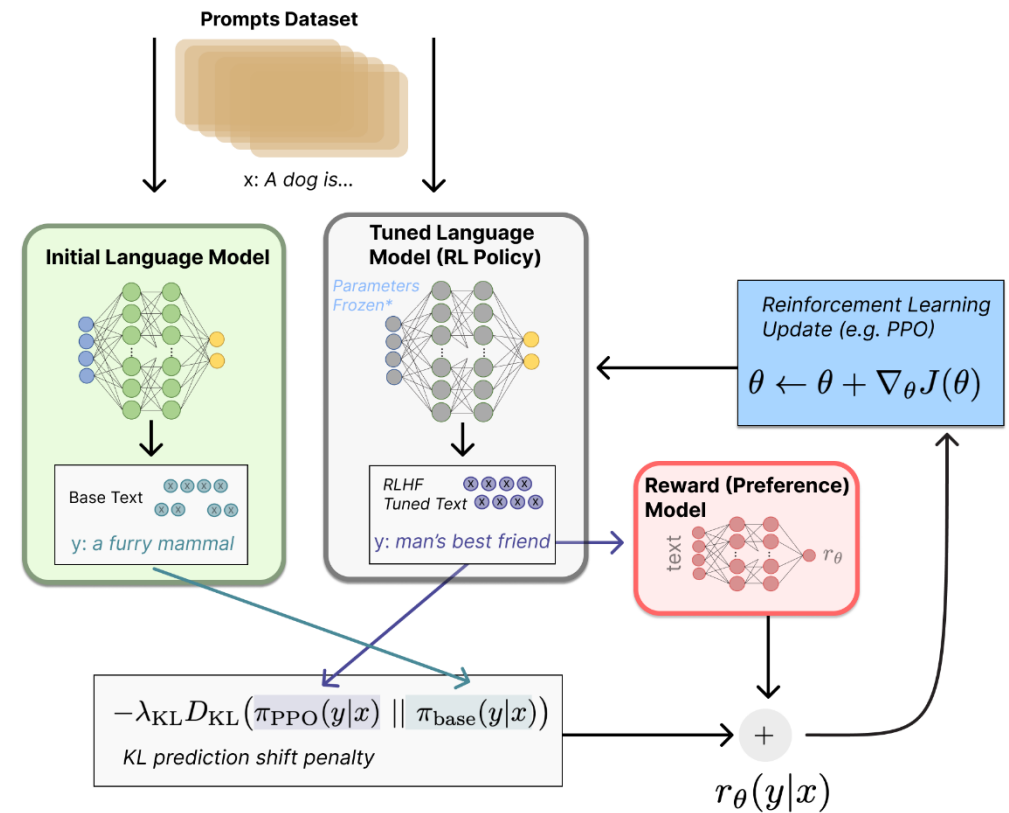


```
[ ] # Human Feedback RL
from IPython.display import YouTubeVideo
YouTubeVideo('oC70w3fu3gl')
```



RLHF

- [train_dummy](#)
- [train_prompts](#)
- [좋은설명](#)
- [dataset1](#)



— ChatGPT-Replica

```

▶ # configure trainer
trainer = PPOTrainer(strategy,
                    actor,
                    critic,
                    reward_model,
                    initial_model,
                    actor_optim,
                    critic_optim,
                    max_epochs=args.max_epochs,
                    train_batch_size=args.train_batch_size,
                    tokenizer=tokenizer_fn,
                    max_length=128,
                    do_sample=True,
                    temperature=1.0,
                    top_k=50,
                    pad_token_id=tokenizer.pad_token_id,
                    eos_token_id=tokenizer.eos_token_id)

## train!
trainer.fit(list_prompt, # 입력 prompt
            num_episodes=args.num_episodes,
            max_timesteps=args.max_timesteps,
            update_timesteps=args.update_timesteps)

## save
# save model checkpoint after fitting on only rank0
strategy.save_model(actor, os.path.join(args.output_dir, 'actor.pt'), only_rank0=True)
# save optimizer checkpoint on all ranks
strategy.save_optimizer(actor_optim,
                        os.path.join(args.output_dir, 'actor_optim_checkpoint_%d.pt' % (torch.cuda.current_device())),
                        only_rank0=False)

```

```

▶ Episode [1/1]: 67%|██████████| 2/3 [00:04<00:02, 2.36s/it]
Train epoch [1/1]: 0%|██████████| 0/3 [00:00<?, ?it/s]
Train epoch [1/1]: 0%|██████████| 0/3 [00:00<?, ?it/s, actor_loss=0, critic_loss=0.00247]
Train epoch [1/1]: 33%|██████████| 1/3 [00:00<00:00, 5.35it/s, actor_loss=0, critic_loss=0.00247]
Train epoch [1/1]: 33%|██████████| 1/3 [00:00<00:00, 5.35it/s, actor_loss=0, critic_loss=0.496]
Train epoch [1/1]: 67%|██████████| 2/3 [00:00<00:00, 5.19it/s, actor_loss=0, critic_loss=0.496]
Train epoch [1/1]: 67%|██████████| 2/3 [00:00<00:00, 5.19it/s, actor_loss=0, critic_loss=0.0068]
Train epoch [1/1]: 100%|██████████| 3/3 [00:00<00:00, 5.18it/s, actor_loss=0, critic_loss=0.0068]
Episode [1/1]: 100%|██████████| 3/3 [00:07<00:00, 2.52s/it]

```

3단계 데이터셋 학습 및 테스트

— ChatGPT-Replica

```
## inference
def generation(input_text):
    input_ids = tokenizer.encode(input_text, return_tensors='pt').to(
        torch.cuda.current_device())
    outputs = actor.generate(input_ids,
                             max_length=args.max_length,
                             do_sample=True,
                             top_k=50,
                             top_p=0.95,
                             num_return_sequences=1)

    output = tokenizer.batch_decode(outputs[0], skip_special_tokens=True)[0]
    print('#' * 70)
    print(output)
    return output

list_prompt = [
    '불고기용 고기 한우예요?',
    '리처드 닉슨이 43대 부통령직을 수행한 년도는?',
    '시카고 오헤어 국제공항은 어디에 있어',
    '오늘 미세먼지 어때?']

list_prompt = [PROMPT_DICT['prompt_no_input'].format_map({'prompt': tmp}) for tmp in list_prompt]

for input_text in list_prompt:
    output = generation(input_text)
```

3단계 데이터셋 테스트 코드

— ChatGPT-Replica



#####

Below is an instruction that describes a task.
아래는 작업을 설명하는 명령어입니다.

Write a response that appropriately completes the request.
명령어에 따른 요청을 적절히 완료하는 응답을 작성하세요.

Instruction(명령어):
불고기용 고기 한우예요?

Response(응답): '한우는 따로 구매하는 것이 아니라 냉동고에 넣어 보관하는 것이 일반적입니다. 보통 불고기나 양념 없이 불고기로 조리할 때 사용하는 것이 일반적입니다. 따라서 질문에서 "불고기용 한우예요?"라고 질문하신다면, "I'll packed bot see"의 의미로 해석

Below is an instruction that describes a task.
아래는 작업을 설명하는 명령어입니다.

Write a response that appropriately completes the request.
명령어에 따른 요청을 적절히 완료하는 응답을 작성하세요.

Instruction(명령어):
리처드 닉슨이 43대 부통령직을 수행한 년도는?

Response(응답): '리처드 닉슨은 1948년 3월 5일에 50세의 나이로 부통령직을 수행하기 위해 미국 일리노이 주의 시카고 대학교에서 일했던 미국 민주당 대통령으로 알려져 있습니다.英譽恩報al Worries to the First of Nova의 popentrehate.:₩₩₩2010년 3월 5일 대통령 스

Below is an instruction that describes a task.
아래는 작업을 설명하는 명령어입니다.

Write a response that appropriately completes the request.
명령어에 따른 요청을 적절히 완료하는 응답을 작성하세요.

Instruction(명령어):
시카고 오헤어 국제공항은 어디에 있어

Response(응답): '시카고는 시카고 국제공항 내에 위치해 있습니다. Jules는 일리노이 주에서 출발하여 뉴욕과 시카고를 거쳐 시카고를 방문하는 것이 목적입니다. Jules는 미국 동부 지역을 대상으로 한 유명한 관광 도시 중 하나로, 시카고를 비롯한 많은 도시들이 근처에

Below is an instruction that describes a task.
아래는 작업을 설명하는 명령어입니다.

Write a response that appropriately completes the request.
명령어에 따른 요청을 적절히 완료하는 응답을 작성하세요.

Instruction(명령어):
오늘 미세먼지 어때?

3단계 데이터셋 테스트 결과

THANK YOU

