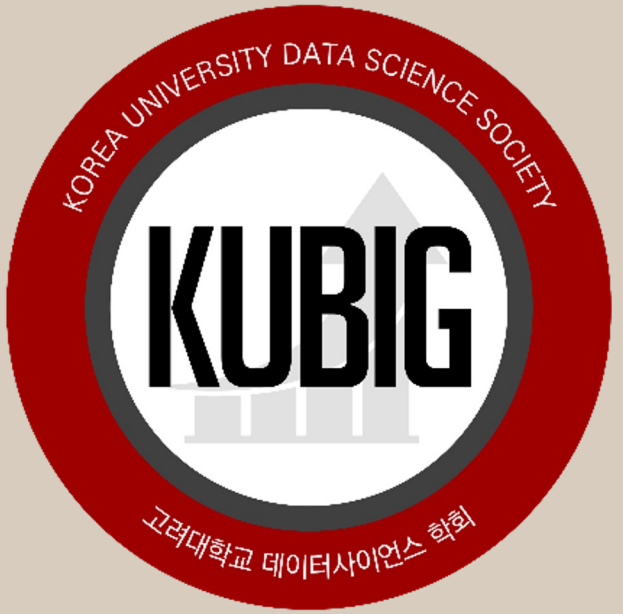
KUBIG 24-W
겨울방학 BASIC STUDY SESSION

NLP SESSION
WEEK6



Today's Session Leader: 17기 홍여빈

CONTENTS

01 Announcement, 우수 복습과제 Review

1-2. 우수 복습과제 코드 Review

주희님

5주차
복습과제1

BERT classification fine-tuning

화면공유 하셔서 3분 내외로 가볍게 리뷰해주시면 됩니다!

02 GPT series

Text Generation

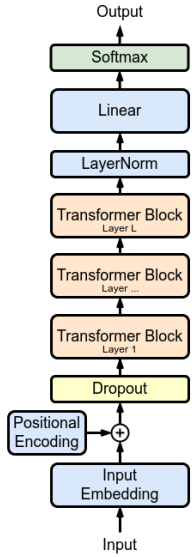
	날짜	특징	한계
GPT-1	2018.6	Auto-regressive pre-training objective	Zero-shot x
GPT-2	2019.2	• prompt를 추가하면 zero-shot 생성 • WebText (번역 등 다양한 downstream task)	
GPT-3	2020.5	In-context learning	Alignment Problem (hallucination, toxic, not helpful)
GPT-3* (InstructGPT)	2022.5	GPT-3 + RL	
GPT-3.5 (ChatGPT)	2022.11	InstructGPT의 연장선	
GPT-4 (ChatGPT+)	2023.3	Multi-modal	

GPT-1

Unlabeled pre-training + Labeled fine-tuning

- annotated resources가 부족
- fine-tune하기 위해 minimal change만 듦

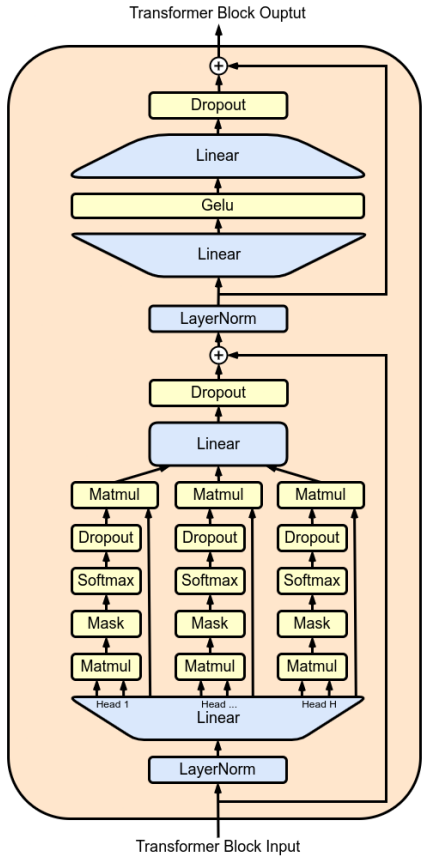
pre-training을 통해 universal representation을 학습함으로써 better generalized (initialized) model이 됨



Model Architecture

pre-training: multi-layer Transformer decoder

fine-tuning: pre-trained model → linear output layer → softmax



$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

$$L_1(\mathcal{U}) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)$$

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

Self-supervised pre-training

- unlabeled data
- auto-regressive
- standard language modeling objective(next token prediction) : maximize L1

long-range dependency를 처리

Supervised fine-tuning

- labeled data
- task 별 objective + language modeling objective : maximize L3
- language modeling objective은 model의 generalization 성능을 높이고, 빠르게 수렴하도록 하는 장점

개별 task에 적용 가능

Limitation of GPT-1

- fine-tuning 없이 zero-shot으로 downstream task를 다루지 못함

🤔 특정 data나 task에만 뛰어난 narrow expert model이 아닌, competent generalist를 만들고 싶다!

Zero-shot generalization

- 이전에 multi-task training이 제시된 적 있음. 하지만 당시에 multitask training은 아직 대규모 데이터가 확보되지 않은 초기 상태.
- GPT-2는 sufficiently large language model은 대규모의 unlabeled dataset에서 학습했을 때 implicit하게 내재되어 있는 task를 학습할 수 있다는 것을 보임.

Model

- Transformer decoder만 사용한 GPT-1의 모델 구조를 사용
  - layer normalization을 먼저 하는 등 몇가지 변화
- batch size, input length(512→1024), vocab size 증가
  - model size 증가

Datasets

WebText dataset: Reddit에서 링크된 글만 필터링하여 사용. 데이터에서 위키피디아 글 삭제, 중복 제거 등 전처리.

- prompt를 추가하면 zero-shot으로 생성 가능
- explicit supervision 없이도 zero-shot generalization 가능성을 제시

reading comprehension을 비롯한 task에서 supervised baseline보다 우수한 성능을 보임

- summarization과 같은 task에서 성능이 좋지 않음
- sufficient model capacity일 때에만 baseline을 넘음

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← example
cheese =>                           ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← examples
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>                           ← prompt
```

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

Limitation of fine-tuning

1.많은 라벨링된 데이터가 필요

2.out-of-distribution data에 대해 낮은 generalization 성능

pre-training 때 대량의 지식을 흡수 → fine-tuning 시에 작은 태스크 분포를 학습

훈련 데이터의 분포로 한정된 모델이 그 외의 영역은 잘 일반화하지 못하며 학습 데이터로부터 과적합되는 경향

- GPT-3에서는 이러한 fine-tuning의 한계를 극복하기 위해 meta-learning의 일종인 in-context learning을 사용.
- In-context learning은 fine-tuning과 다르게 gradient update를 하지 않으며, supervised dataset이 필요하지 않음.

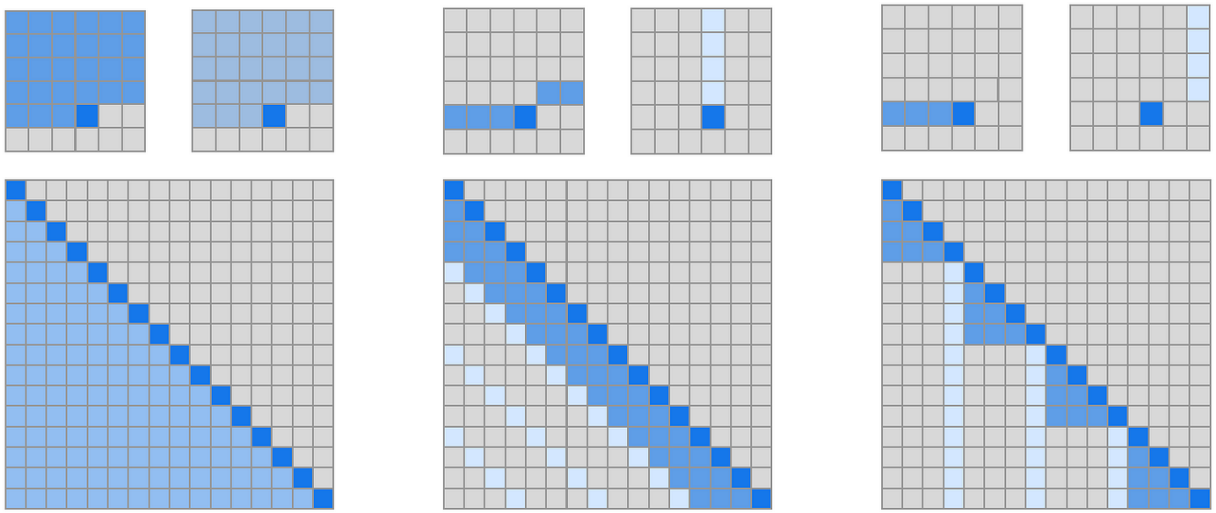
In-context learning

- model이 task에 대한 정보를 참고해서 inference 할 수 있도록 input에 예제(demonstrations) 추가
- inner loop를 같은 task로 구성해서 model이 다양한 task에 대해 multi-task learning 하는 효과

KUBIG DATA SCIENCE & AI

Model

- GPT-2의 모델 구조를 사용
- Dense, locally banded sparse attention을 번갈아 사용



(a) Transformer (b) Sparse Transformer (strided) (c) Sparse Transformer (fixed)

Results

- 단어 풀이 등 task는 one ~ two shot으로도 우수한 성능 달성

parameter가 클 수록 성능이 좋음

Limitation

- NLI 등 task는 few shot 이후에도 성능이 좋지 않음
  - 동어 반복 현상, 일관성 부족, 내용에 모순 등
- auto-regressive, 모든 token에 대한 가중치가 같게 pre-training
  - 비용

Limitation of GPT-3
: Alignment problem

LM의 objective인 NTP(Next Token Prediction)이 user intention과 같지 않은 문제
없는 사실을 만들거나(not truthful), 편향적이고(biased), 유해한 텍스트를 만들거나(toxic, harmful)하여 사용자의 의도대로 사용되지 않을 수 있음(not in accordance with user's intention)

Objective of GPT-3*

1. 사용자의 instruction을 따르는 것 => explicit하게 학습
2. Helpful (user가 task를 달성하게 도움), honest (거짓 정보 없이), harmless => implicit하게 학습

How?

Fine-tuning by RLHF

- 인간의 피드백을 통한 강화학습으로 사용자의 광범위한 지시사항에 따를 수 있도록 하는 것
- 인간의 평가(preference)를 reward로 활용

평가를 하는 인간은?

- 데이터 구축을 위해 40명의 labeler를 고용 : screening test를 통해 상위 rank된 사람들
- 연구에 참여한 저자들

2-4. GPT-3* (InstructGPT)

최적이 될 때까지 step 2와 step 3를 반복

Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

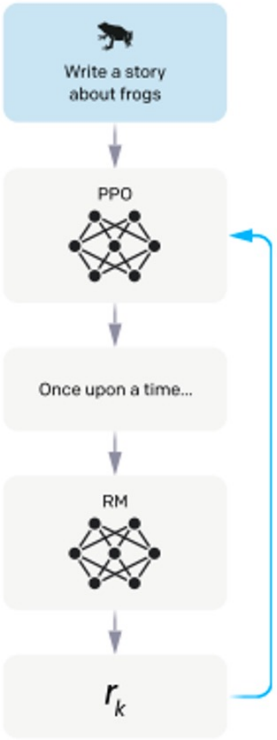
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

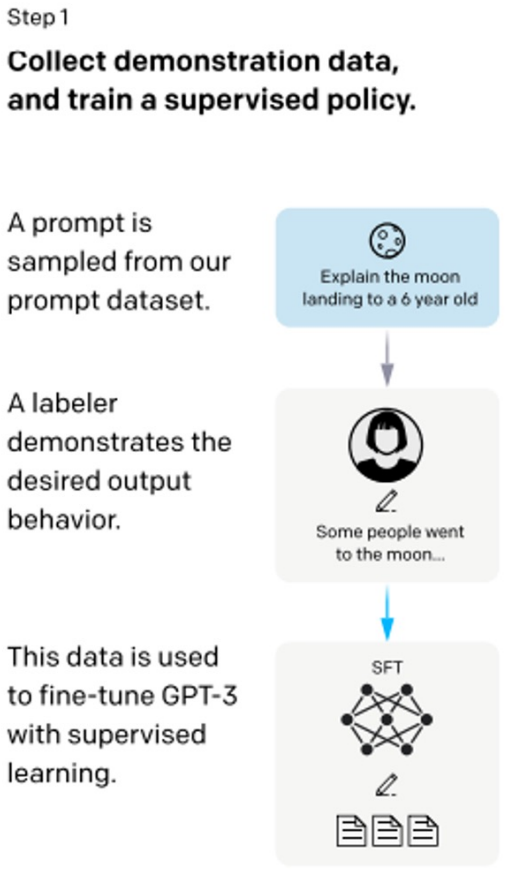
The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



2-4. GPT-3* (InstructGPT)



step 1: demonstration data 구축, supervised fine-tuning(SFT)

1) collect demonstration data

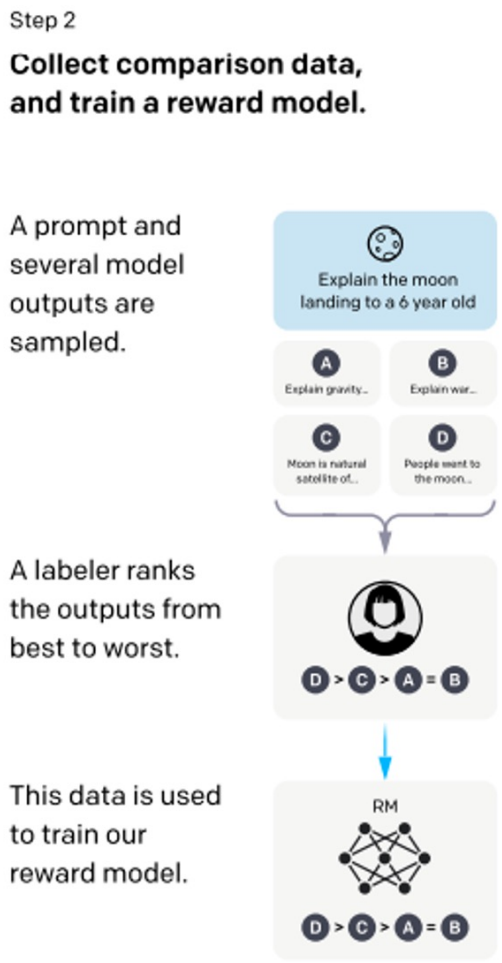
demonstration: prompt - response 쌍

- prompt: labeler 작성 + open AI를 통해 수집된 사용자들이 작성
- response: labeler들이 주어진 prompt에 대해 직접 작성

2) supervised fine-tuning (SFT)

구축한 demonstration data로 16 epochs 학습

최종 SFT model은 validation set에 대해 RM score를 기준으로 선정



step 2: 인간의 선호도를 반영한 comparison data 구축, Reward Model 학습

1) collect comparison data

comparison data: 각 prompt에 대응하는 4~9개의 response 생성 결과물을 대상으로 labeler가 선호도 순위를 매긴다 (labeler ranking)

이 때 사용된 33k개의 prompt: 마찬가지로 API + labeler 작성 prompt

2) train a Reward Model

reward model: labeler가 선호하는 답변을 예측하는 모델

comparison data를 이용해서 RM을 학습

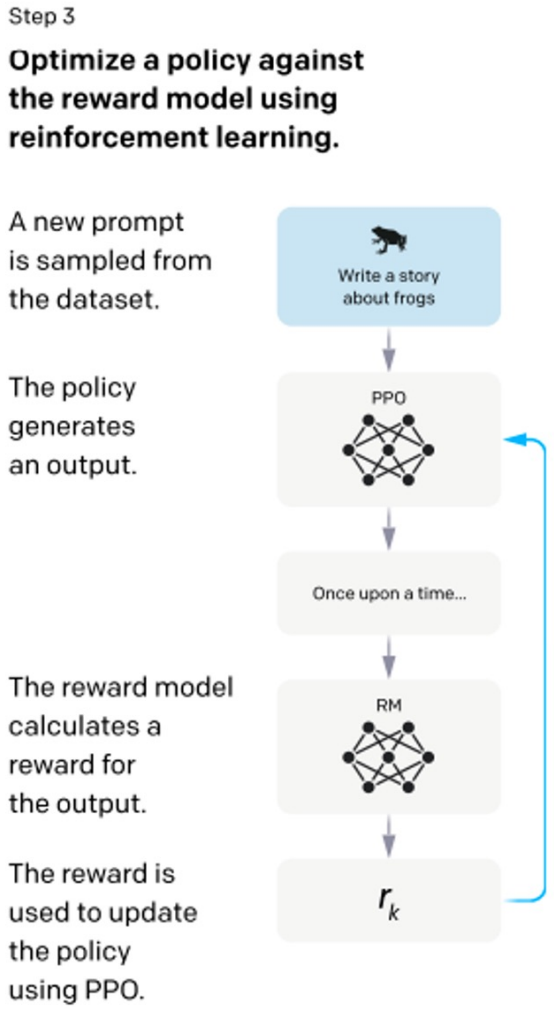
input: prompt, response 2개 -> output: reward 값 (scalar)

하나의 prompt에 대한 k개의 response 중 2개씩 1:1 비교하여 선호도가 더 높은 response를 y_w, 낮은 거를 y_l로 설정. 둘의 차이가 클수록 loss가 감소하도록 학습. 1 epoch만 학습하여 overfitting을 방지.

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right]$$

reward

dataset prompt response



step 3: RM을 활용하여 PPO 방식의 RL로 GPT-3를 fine-tuning

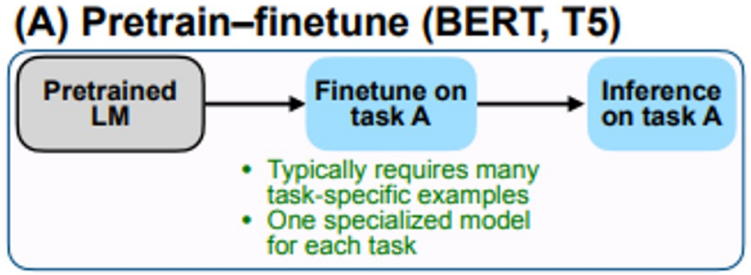
1) LM이 prompt에 대한 output을 생성

2) RM이 output의 reward (선호도)를 계산

3) 이 reward를 policy update에 사용. policy update는 PPO 알고리즘으로 진행

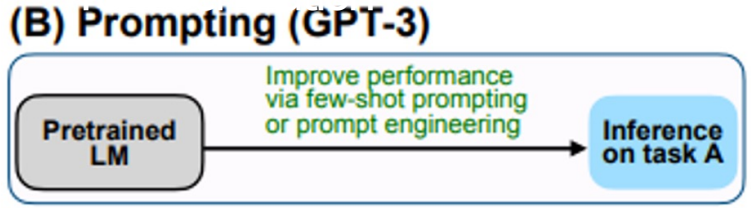
03 Alpaca

Human Alignment



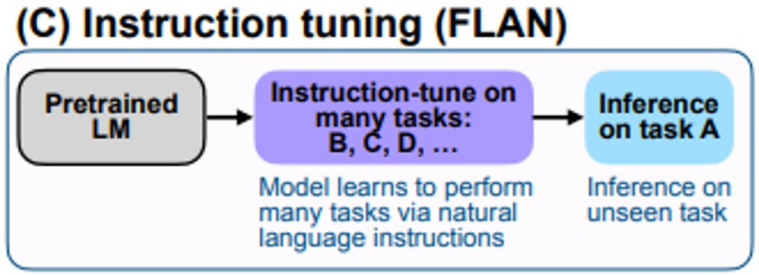
fine tuning

- pretrain된 언어 모델을 downstream task에 맞춰 supervised 방식으로 데이터셋을 학습
- pretrain 단계에서의 weight가 task-specific한 데이터셋을 학습하면서 업데이트됨



prompt learning

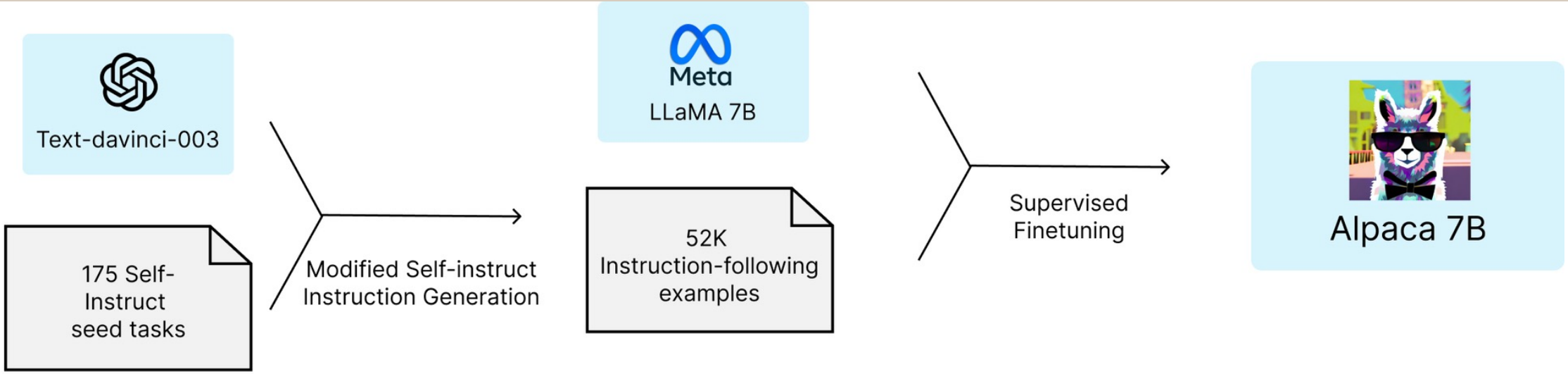
- task-specific한 데이터를 추가로 학습시키지 않음
- 사전학습을 진행할 때 모델에게 prompt (문제에 대한 설명)를 제공하여 학습
- pretrained 모델의 추가적인 가중치 업데이트가 이뤄지지 않음



Instruction tuning

- prompt learning의 prompt 방식과 finetuning의 가중치 업데이트를 결합한 방법
- 모델에게 instruction이 있는 task를 supervised하게 학습하여 모델이 instruction을 따르는 방식을 학습
- Instruction을 따르는 방식을 이미 배웠기 때문에 unseen task의 instruction에 따라 잘 추론할 수 있을 것이라는 아이디어

3-2. Alpaca



Example seed task

Instruction: Brainstorm a list of possible New Year's resolutions.

Output:
- Lose weight
- Exercise more
- Eat healthier

Example Generated task

Instruction: Brainstorm creative ideas for designing a conference room.

Output:
... incorporating flexible components, such as moveable walls and furniture ...

Datasets

- self-instruct seed set에서 사람이 작성한 instruction-output pairs175개
- Seed sets를 활용하여 text-davinci-003을 prompting하여 더 많은 instructions를 생성

Evaluation

text-davinci-003과 Alpaca 7B 간의 paiwise comparison 결과:
모델의 성능은 매우 유사. Alpaca는 text-davinci-003과의 비교에서 90대 89

Alpaca의 size가 작고 data 양이 적은 것에 비해 놀라운 성과

Limitation

- Alpaca는 언어 모델의 몇 가지 일반적인 결함을 보임: hallucination, toxicity, stereotypes 등
- Alpaca에는 underlying language model data 관련된 다른 많은 문제점이 있을 수 있음

04 LangChain

LLM python library

4. LangChain

- LangChain은 OpenAI, Huggingface 등 여러 대형 LLM 공급업체와의 상호작용을 간소화하기 위해 설계된 python 라이브러리
- 여러 모델을 순서대로 상호 작용해야하는 복잡한 AI 응용 프로그램을 만들 때 유용
- 체인을 사용하면 여러 구성 요소를 결합하여 일관된 단일 애플리케이션을 만들 수 있어서 복잡한 애플리케이션의 구현을 대폭 단순화하고 모듈화하여 애플리케이션을 디버그, 유지 관리 및 개선하는 것을 훨씬 쉽게 만듦

langchain을 통해 openAI API 등을 이용한 서비스 개발을 쉽게 할 수 있습니다.

```
1   from langchain.llms import OpenAI
2
3   llm = OpenAI(openai_api_key="...")
```

4-1. LLMChain

LLMChain의 구성 요소:

1. LLM

1-1. LLMs: 문자열을 입력으로 받아 문자열을 반환하는 언어 모델

1-2. ChatModels: list of messages을 입력으로 사용하고 ChatMessage를 반환하는 언어 모델

ChatMessage의 구성 요소:

a) content: message의 내용

b) role: ChatMessage가 전송되는 개체의 역할

HumanMessage: 사람/사용자가 보내는 ChatMessage

AIMessage: AI/어시스턴트가 보내는 ChatMessage

SystemMessage: 시스템에서 전송되는 ChatMessage

역할을 수동으로 지정

2. Prompt Templates

언어 모델에게 instructions를 제공. 언어 모델의 출력을 제어하므로 프롬프트와 다양한 프롬프트 전략을 구성하는 방법을 이해하는 것이 중요.

3. Output Parsers

LLM의 raw response를 보다 실행 가능한 형식으로 변환하여 출력을 쉽게 사용

Ex) LLM에서 텍스트를 구조화된 정보(예: JSON)으로 변환

Ex) ChatMessage를 문자열로 변환

Ex) 메시지 외에 호출에서 반환된 추가 정보를 문자열로 변환

4-1. LLMChain

```python
from langchain.prompts import PromptTemplate
from langchain.llms import HuggingFace
from langchain.chains import LLMChain

prompt = PromptTemplate(
    input_variables=["city"],
    template="Describe a perfect day in {city}?",
)

llm = HuggingFace(
        model_name="gpt-neo-2.7B",
        temperature=0.9)

llmchain = LLMChain(llm=llm, prompt=prompt)
llmchain.run("Paris")
```

4-1. LLMChain

```python
from langchain.output_parsers.json import SimpleJsonOutputParser

json_prompt = PromptTemplate.from_template(
    "Return a JSON object with an `answer` key that answers the following question: {question}"
)
json_parser = SimpleJsonOutputParser()
json_chain = json_prompt | model | json_parser
```

4-2. SimpleSequentialChain

SimpleSequentialChain은 하나의 출력이 다음 입력으로 사용되는 LLM을 논리적으로 연결
체인 사이에 단일 입력과 단일 출력이 있는 경우에 사용되며, SequentialChain은 여러 입력과 출력이 있는 경우에 사용됨
SimpleSequentialChain의 input_variables 및 output_variables을 명시적으로 언급하지 않음. 이는 chain 1의 출력이 chain 2의 입력으로 전달된다는 가정에 기반함.

Ex) 사용자가 선호하는 장르에 따라 영화 추천 생성
Chain #1 - 사용자가 좋아하는 영화 장르에 대해 묻는 LLM 체인
Chain #2 - Chain 1의 결과를 기반으로 선호하는 장르를 사용하여 해당 장르의 영화를 추천하는 다른 LLM 체인

```
# chain 1
template = '''당신은 영화 장르 설문 조사입니다. 사용자에게 선호하는 장르에 대해 물어보세요.

묻기:'''
prompt_template = PromptTemplate(input_variables=[], template=template)
chain_one = LLMChain(llm=llm, prompt=prompt_template)

# chain 2
template = '''당신은 영화 추천 프로그램입니다. 사용자가 선호하는 장르: {genres}, 이 장르에 해당하는 영화를 추천하세요.

추천:'''
prompt_template = PromptTemplate(input_variables=["genres"], template=template)
chain_two = LLMChain(llm=llm, prompt=prompt_template)

```

4-2. SimpleSequentialChain

```python
# chain 1, 2 연결
from langchain.chains import SimpleSequentialChain

overall_chain = SimpleSequentialChain(
    chains=[chain_one, chain_two],
    verbose=True)

overall_chain.run('당신이 좋아하는 영화 장르는 무엇인가요?')

# 결과
# > Entering new SimpleSequentialChain chain...
# > Entering new LLMChain...
# 좋아하는 장르: 액션, 드라마, 코미디#> 완료된 체인.

# 좋아하는 장르: 액션, 드라마, 코미디
# 액션: 다이 하드, 매드 맥스: 분노의 도로, 다크 나이트
# 드라마: 쇼생크 탈출, 위대한 개츠비, 유령
# 코미디: 굳세어라, 방황하는 카메라, 삼촌
#> 완료된 체인.
```

05 KUBIG Contest 중간발표

6-1. 우수 예습과제 Review

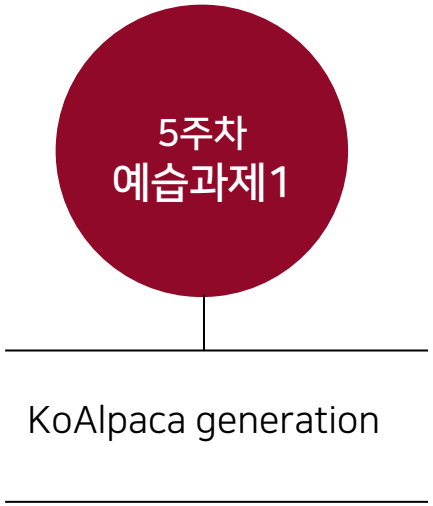
1 -> 2 -> 3 -> 4 팀 순서로 화면공유 하셔서
3분 내외로 가볍게 발표해주시면 됩니다!

06 Announcement

Week5 예습과제 Review, week6 복습 과제 안내, week7진도 안내

6-1. 우수 예습과제 Review

민아님



화면공유 하셔서 3분 내외로 가볍게 리뷰해주시면 됩니다!

6-2. Week6 예,복습과제 안내, Week7 진도 안내

코드과제의 파일형식은 ipynb로, KUBIG 24-1 **Github** repo에 업로드 될 예정입니다!
Colab 환경에서 제작된 과제들이므로 google colab에서 실행하시는 것을 권장드립니다.

WEEK6
복습과제1

생성모델의 Decoding 전략

WEEK7 진도

- 응용 분야 소개

E.O.D
수고하셨습니다!