



LLM을 사용한 AI 챗봇 연구

2023 후기 졸업과제

낙원에가고싶어 조

201824423 김도훈

201924655 이보원



목차

- 1 연구 배경
- 2 연구 배경 지식
- 3 연구 내용
- 4 평가 및 결론
- 5 시연 영상

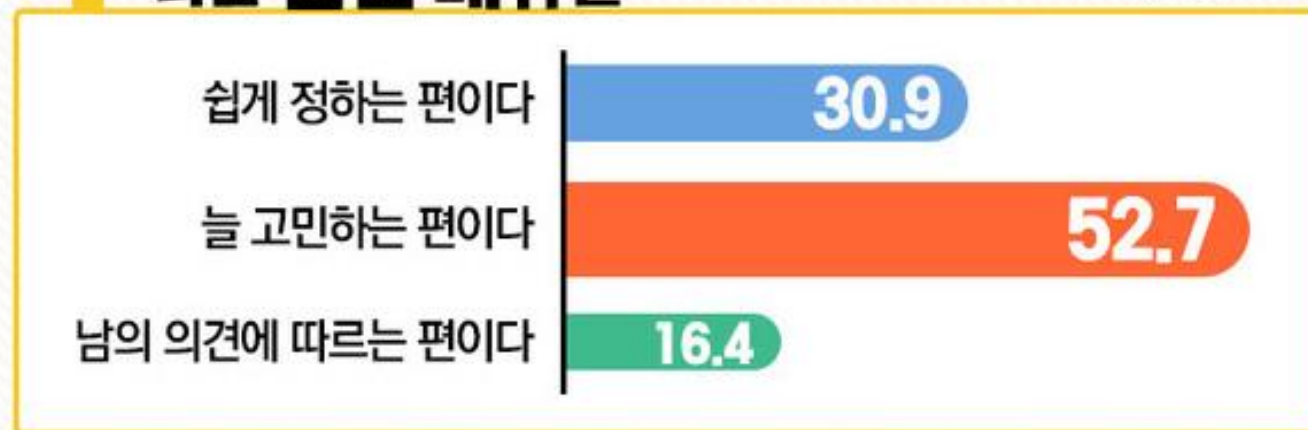
Part 1

연구 배경

1

나는 점심 메뉴를...

(단위: %)



출처: 매경헬스 2021.08

- 많은 사람들이 바쁜 일상에서 업무 중간에 식사를 챙기기 어렵고 메뉴 선택에 어려움을 겪는 경우가 많음
- 이로 인해 균형 잡힌 식사를 하거나 새로운 요리를 시도하지 못하게 됨

Part 1

연구 배경: 기존 시스템 문제점



- 기술 개발·서비스 기업인 업스테이지가 AskUp이라는 챗봇 서비스 출시
- 이미지의 텍스트를 읽고 요약하거나 음식 사진을 올리면 영양 정보를 분석하고 식단에 대한 조언을 들을 수 있음
- 그러나 하루에 던질 수 있는 질문의 개수가 한정돼있고 2021년까지의 데이터로 학습된 chatGPT에 의존하므로 잘못된 정보를 제공할 수있음

Part 1

연구 배경: 목적



- 바쁜 현대인들에게 영양 균형과 개인의 취향을 고려한 적절한 식사 메뉴 제안
- 메뉴를 결정하는 데 걸리는 시간을 줄이고 더 쉽고 빠르게 메뉴를 선택할 수 있게 함

Part 2 연구 배경 지식

LLM(Large Language Model)

- 방대한 양의 데이터를 학습하고 자연어를 비롯한 다양한 유형의 콘텐츠를 이해하고 생성하여 광범위한 작업을 수행하는 모델
- 인간의 텍스트를 이해하고 사람이 입력한 것처럼 자연스러운 문장을 생성해 내기 때문에 대화를 기반으로 하는 AI 서비스에 적합함
- 언어 번역, 고객 지원, 창작 활동 등 다양한 분야에서 활용되고 있고 점점 정교해져 소비자와 더욱 자연스러운 대화를 나눌 수 있게 됨

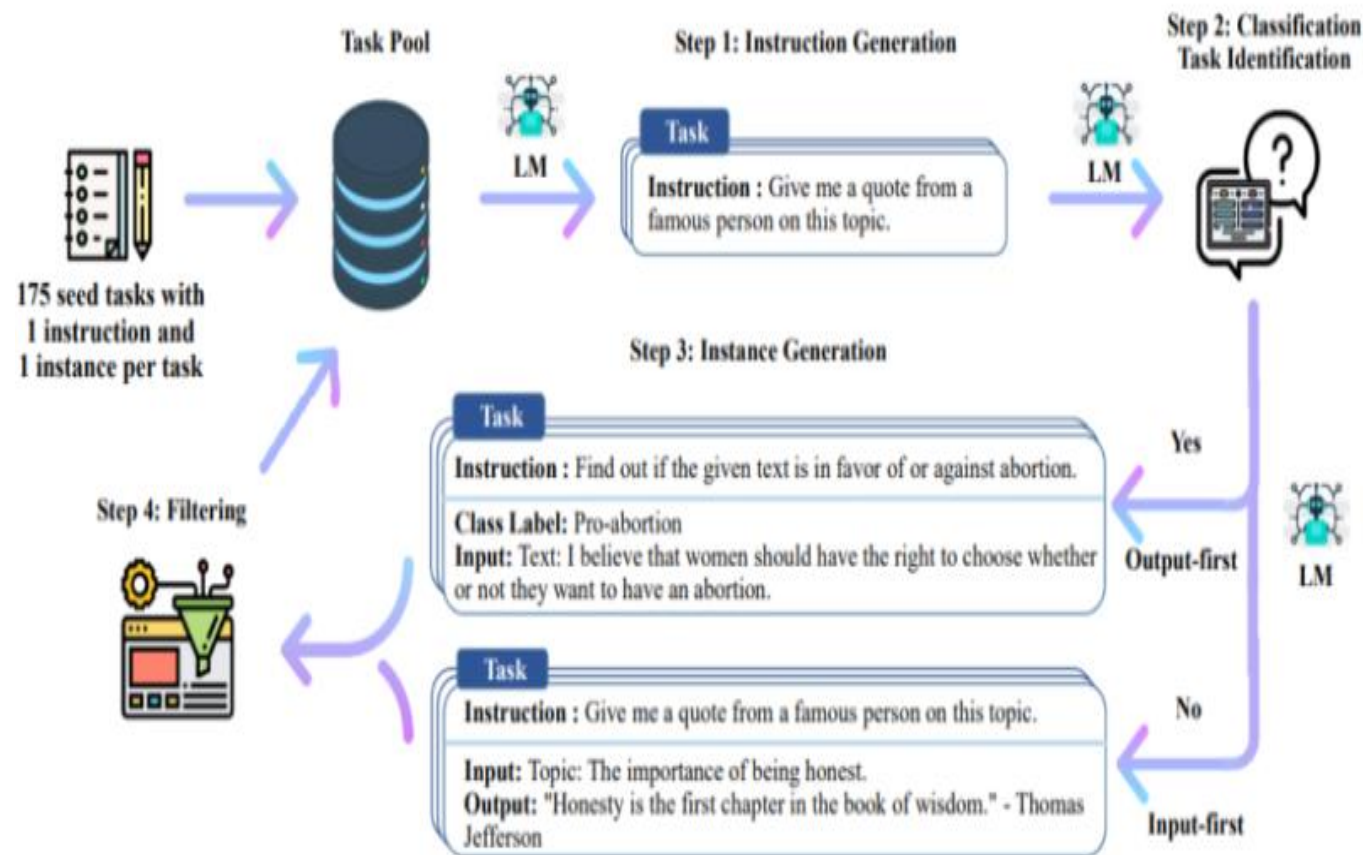


Part 2

연구 배경 지식: 데이터 수집

Self instruct

- 모델이 스스로 학습 데이터를 생성하고 그 데이터를 사용하여 훈련하는 방식으로 대량의 instruction 데이터셋을 생성하기 위한 목적으로 사용하는 프레임워크
- 사람이 만든 데이터의 품질과 양에 의존하여 다양성과 창의성을 제한받을 수 있다는 문제점을 극복할 수 있음
- 본 연구에서는 가장 효과적이고 비용 효율적인 gpt-3.5-turbo 모델 사용

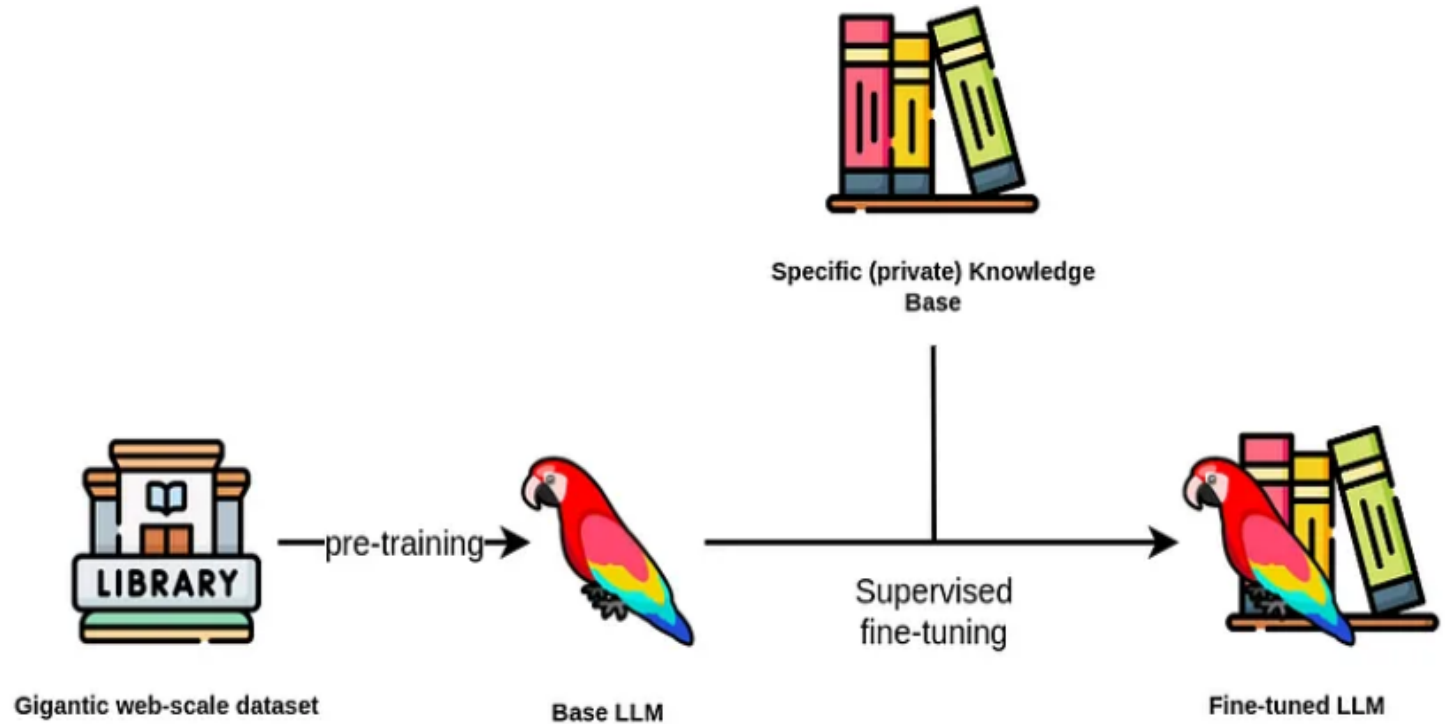


Part 2

연구 배경 지식: Fine Tuning

Fine Tuning

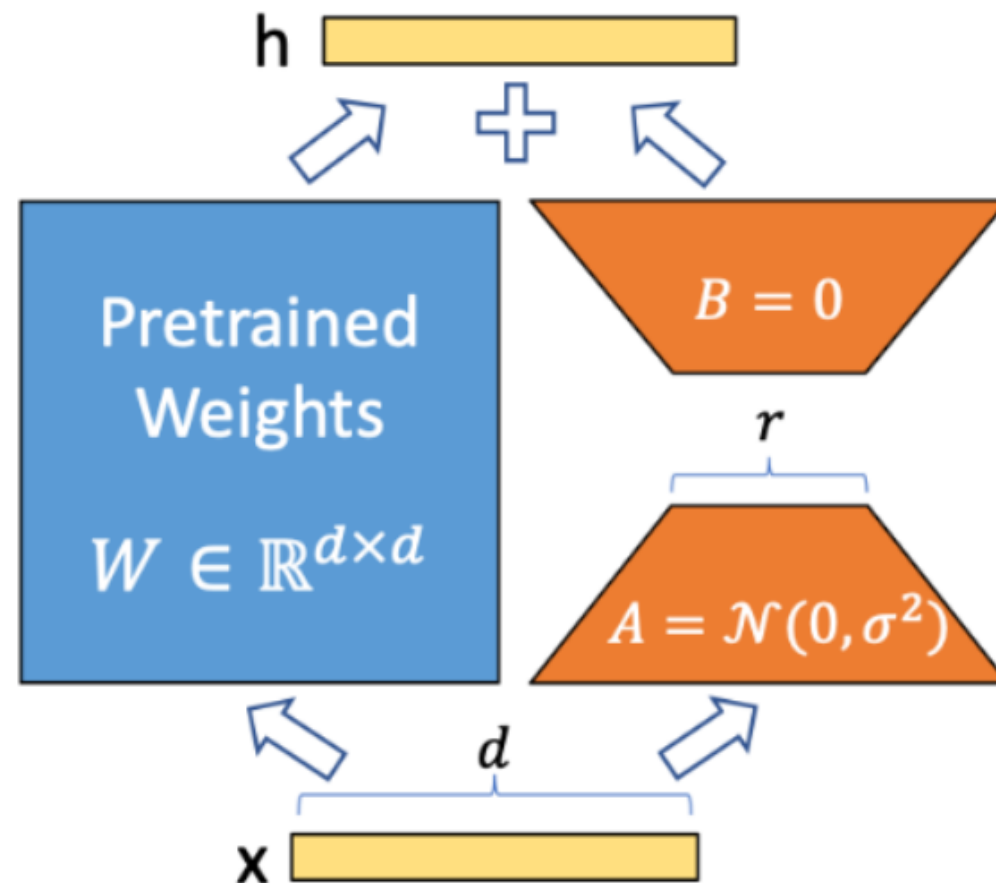
- 특정 작업이나 도메인에 특화된 추가 학습 데이터를 사용하여 사전 학습된 언어모델의 매개변수를 업데이트하는 프로세스를 의미
- 작은 양의 데이터와 짧은 학습시간으로도 좋은 성능을 얻을 수 있으며 이로 인해 자원과 시간을 절약할 수 있게 됨
- 모델이 새로운 데이터에 더 잘 적응하게 하고 예측 또는 추론 성능을 향상시킬 수 있음



Part 2 연구 배경 지식: Fine Tuning

QLoRA

- 메모리 사용량을 크게 줄이는 fine tuning 접근법으로 전체 16비트 fine tuning 성능을 유지하면서 낮은 RAM을 가진 GPU에서도 거대한 매개변수를 가진 모델을 fine tuning 할 수 있음



Part 2

연구 배경 지식: 사용자 인터페이스

Gradio

- 머신러닝 모델 사용자 인터페이스를 만드는데 사용되는 Python 오픈 소스 패키지로서 서버를 직접 만들지 않고도 배포가 가능함
- 머신러닝에 최적화된 UI 컴포넌트를 제공하여 쉽고 빠르게 인터페이스를 생성할 수 있고 앱을 빌드하면 최대 72시간 동안 사용할 수 있는 public URL을 제공함



Part 3

연구 내용: 시스템 구성도



Part 3

연구 내용: 모델 선정

KoAlpaca

- Stanford Alpaca 모델을 학습한 방식과 동일한 방식으로 한국어 데이터를 기반으로 하여 학습시킨 Alpaca 모델이며 백본 모델로 Polyglot-ko 모델을 사용
- Polyglot-ko 12.8B 기반 모델과 polyglot-ko 5.8B 기반 모델을 제공하고 있지만 Colab 저장 용량과 GPU RAM을 고려하여 Polyglot-ko 5.8B 기반 모델 사용



KoAlpaca

Part 3

연구 내용: 데이터 수집

Self instruct

-Instruction set 작성

```
"instruction": "편의점 간식 추천해줘", "instances": [{"input": "", "output": "젤리나 초콜릿은 어떠세요?"}], "is_classification": true,
"instruction": "약간 출출한데 뭘 먹을까", "instances": [{"input": "", "output": "토스트나 씨리얼을 추천드려요."}], "is_classification": false,
"instruction": "중식이 먹고싶어", "instances": [{"input": "", "output": "탕수육은 어떠세요? 탕수육은 돼지고기에 녹말 반죽을 묻혀서 튀겨낸 음식입니다."}], "is_classification": false,
"instruction": "면 요리 중에 뭘 먹을까", "instances": [{"input": "", "output": "소고기 쌀국수는 어떠세요? 쌀국수는 고기 국물에 면을 넣어 끓여낸 음식입니다."}], "is_classification": false,
"instruction": "디저트 추천해줘", "instances": [{"input": "", "output": "아메리카노와 생크림 와플은 어떠세요?"}], "is_classification": true,
"instruction": "매운 거 먹고싶어", "instances": [{"input": "", "output": "매콤한 떡볶이나 닭볶음탕을 추천드려요."}], "is_classification": false,
"instruction": "달달한 거 추천해줘", "instances": [{"input": "", "output": "도넛은 어떠세요? 도넛은 소맥분에 설탕, 버터, 계란 등을 넣어 튀겨낸 음식입니다."}], "is_classification": false,
"instruction": "한식 뭘 먹을까", "instances": [{"input": "", "output": "비빔밥은 어떠세요? 비빔밥은 밥에 각종 나물을 넣고 장을 넣어拌匀한 음식입니다."}], "is_classification": false,
"instruction": "담백한 음식 추천해줘", "instances": [{"input": "", "output": "소고기 뭇국은 어떠세요? 소고기 뭇국은 육수에 무와 대파 등을 넣어 끓여낸 음식입니다."}], "is_classification": false,
"instruction": "혼자 먹을 배달음식 추천해줘", "instances": [{"input": "", "output": "혼자 먹기에 적당하고 맛있는 햄버거는 어떠세요?"}], "is_classification": false,
"instruction": "아침으로 뭘 먹을까", "instances": [{"input": "", "output": "아침에 먹기에 부담스럽지 않은 죽이나 씨리얼은 어떠세요?"}], "is_classification": false,
"instruction": "시원한 거 마시고 싶어", "instances": [{"input": "", "output": "딸기스무디나 밀크셰이크를 추천합니다."}], "is_classification": false,
"instruction": "뭘 먹을까", "instances": [{"input": "", "output": "소불고기 전골은 어떠세요?"}], "is_classification": false,
"instruction": "반찬 추천해줘", "instances": [{"input": "", "output": "조리법도 간단하고 맛있는 계란말이는 어떠신가요?"}], "is_classification": false,
"instruction": "가족끼리 뭘 먹으면 좋을까", "instances": [{"input": "", "output": "찜닭은 어떠세요? 찜닭은 소금간을 한 닭을 푼 채소와 함께 끓여낸 음식입니다."}], "is_classification": false,
"instruction": "떡볶이는 어떤 음식이야?", "instances": [{"input": "", "output": "떡볶이는 떡을 양념에 볶거나 끓여서 먹는 한국 음식입니다."}], "is_classification": false,
"instruction": "매운 한식 추천해줘", "instances": [{"input": "", "output": "김치찌개는 어떠신가요?"}], "is_classification": false,
"instruction": "국물있는 거 먹고싶어", "instances": [{"input": "", "output": "갈비탕은 어떠신가요? 갈비탕은 한국의 국물 요리입니다."}], "is_classification": false,
"instruction": "간단하게 만들 수 있는 음식 뭐가 있을까", "instances": [{"input": "", "output": "프렌치 토스트는 어떠세요? 프렌치 토스트는 빵을 튀겨낸 음식입니다."}], "is_classification": false,
"instruction": "어떤 빵 먹을까", "instances": [{"input": "", "output": "베이글은 어떠세요? 베이글은 밀가루 반죽을 물에 데쳐서 굽은 빵입니다."}], "is_classification": false,
"instruction": "아무거나 추천해줘", "instances": [{"input": "", "output": "단백질이 풍부한 두부를 추천드립니다."}], "is_classification": false,
"instruction": "우울한데 뭘 먹을까", "instances": [{"input": "", "output": "우울할 때는 스트레스 수치를 낮추는 달콤한 초콜릿을 추천드립니다."}], "is_classification": false,
"instruction": "칼로리 적은 음식 추천해줘", "instances": [{"input": "", "output": "칼로리가 적은 음식에는 계란, 오이, 닭가슴살을 추천합니다."}], "is_classification": false,
"instruction": "기름진 거 먹고싶어", "instances": [{"input": "", "output": "새우튀김, 돈까스, 치킨, 삼겹살을 추천합니다."}], "is_classification": false,
```

Part 3

연구 내용: 데이터 수집

Self instruct

-프롬프트 인코딩

```
def encode_prompt(prompt_instructions):  
    """Encode multiple prompt instructions into a single string."""  
    prompt = open("./prompt.txt").read() + "\n"  
  
    for idx, task_dict in enumerate(prompt_instructions):  
        (instruction, input, output) = task_dict["instruction"], task_dict["input"], task_dict["output"]  
        instruction = re.sub(r"\s+", " ", instruction).strip().rstrip(":")  
        input = "<noinput>" if input.lower() == "" else input  
        prompt += f"###\n"  
        prompt += f"{idx + 1}. Instruction: {instruction}\n"  
        prompt += f"{idx + 1}. Input: {input}\n"  
        prompt += f"{idx + 1}. Output: {output}\n"  
    prompt += f"###\n"  
    prompt += f"{idx + 2}. Instruction: "  
    return prompt
```

Part 3

연구 내용: 데이터 수집

Self instruct

-instruction 생성

```
results = utils.openai_completion(  
    prompts=batch_inputs,  
    model_name=model_name,  
    batch_size=request_batch_size,  
    decoding_args=decoding_args,  
    logit_bias={"50256": -100},  
)  
request_duration = time.time() - request_start  
  
process_start = time.time()  
instruction_data = []  
for result in results:  
    new_instructions = post_process_gpt3_response(num_prompt_instructions, result)  
    instruction_data += new_instructions
```

Part 3

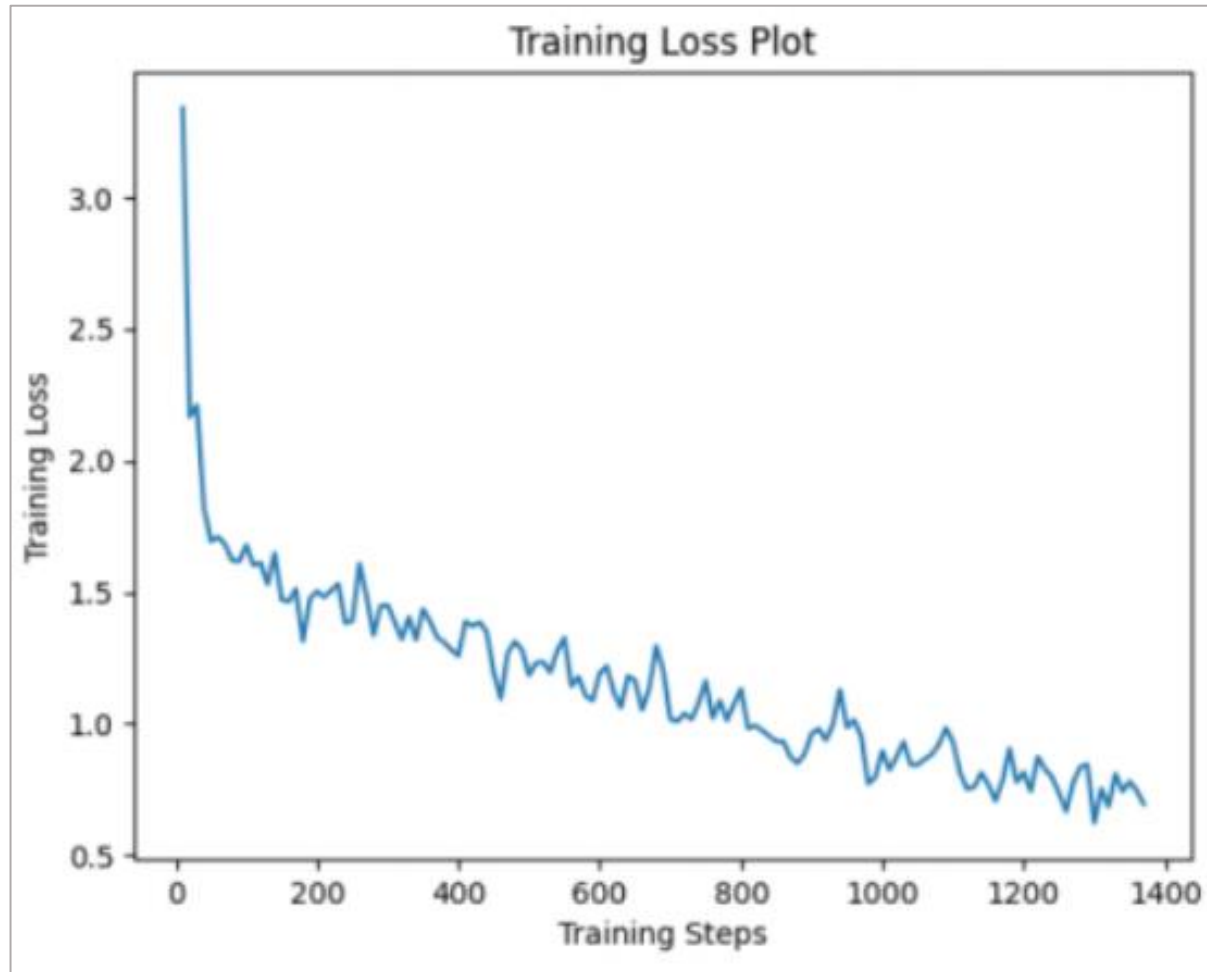
연구 내용: Fine Tuning

- Google Colab에서 제공하는 Tesla T4 GPU 런타임 환경에서 진행
- GPU RAM을 고려하여 QLoRA 방식으로 학습
- Self instruct로 수집한 약 1100여개의 데이터셋을 사용하여 학습

| 하이퍼 파라미터 | 설정값 |
|-----------------------------|------|
| per_device_train_batch_size | 8 |
| gradient_accumulation_steps | 1 |
| max_steps | 1372 |
| learning_rate | 1e-4 |
| logging_steps | 10 |

Part 3

연구 내용: Fine Tuning



loss 값에 대한 그래프

Part 3

연구 내용: 사용자 인터페이스

- 어플리케이션의 이름은 “**메추리**”이며 사용자가 “매운 음식 추천해줘”, “국물 요리 추천해줘” 등의 자신의 상황과 선호에 맞게 메시지를 입력하면 챗봇이 그에 맞는 음식을 추천해줌
- 답변이 마음에 들지 않으면 다시 요청하여 다른 메뉴를 추천 받을 수 있음

 **메추리** 

Chatbot

매운 음식 추천해줘

떡볶이는 어떠세요? 쫄깃한 떡과 매콤한 소스가 어우러져 매운맛을 즐기기에 딱 좋습니다. 간단하면서도 맛있는 한끼 식사로 제격이에요

음식 종류나 끼니를 입력하세요. 예: 점심, 저녁, 디저트

보내기

챗봇에게 원하는 음식 종류나 끼니를 입력하면 추천해 드립니다!

Part 4

평가 및 결론

- 베이스 모델의 낮은 파라미터 값으로 높은 추론 능력을 기대할 순 없지만 어느 정도 자연스러운 답변을 생성함
- 그러나 복잡하고 세밀한 답변을 요구하는 질문에는 엉뚱한 답변을 생성하거나 불완전한 문장을 만들어내기도 함
- 답변의 최대 길이를 200으로 설정했을 때 평균 응답 시간은 13-15초를 기록함

Part 4

평가 및 결론

- Self instruct 방식으로 데이터셋을 구축하여 기반 모델인 KoAlpaca에 QLoRA 방식 Fine-Tuning을 실행함
- 모델을 웹 어플리케이션 형태로 빌드하고 배포하기 위해 Gradio 라이브러리를 활용
- 처음 계획했던 대로 음식 추천 전문 챗봇을 구현하기에는 무리가 있었으나 음식 추천에 대한 질문에 한해서는 꽤 자연스러운 답변을 생성하는 성과를 거둠
- 자원과 시간의 한계로 낮은 파라미터 모델과 작은 데이터셋을 사용했기 때문에 파라미터가 더 많은 모델과 더 거대한 데이터셋을 사용한다면 더 자연스럽고 전문적인 챗봇을 만들 수 있을 것으로 기대됨

Load_and_UI_run.ipynb - Colab

colab.research.google.com/drive/1SVQgVwmiqY4a0Qt_4Xfst2DhoRuhKgj0#scrollTo=8JaD2vpPfkU

Load_and_UI_run.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

파일

sample_data

README.md

anscombe.json

california_housing_test.csv

california_housing_train.csv

mnist_test.csv

mnist_train_small.csv

코드를 실행할 때 발생하는 에러를 표시합니다.

8초

[2] from transformers import AutoTokenizer, AutoModelForCausalLM, BitsAndBytesConfig

5초

[3] from peft import PeftModel, PeftConfig

3분

▶

peft_model_id = "Dohun99/mechuri-koalpaca-polyglot-5.8b-2140step"

config = PeftConfig.from_pretrained(peft_model_id)

bnb_config = BitsAndBytesConfig(
 load_in_4bit=True,
 bnb_4bit_use_double_quant=True,
 bnb_4bit_quant_type="nf4",
 bnb_4bit_compute_dtype=torch.bfloat16
)

model = AutoModelForCausalLM.from_pretrained(config.base_model_name_or_path, quantization_config=bnb_config, device_map={"":0})

model = PeftModel.from_pretrained(model, peft_model_id)

tokenizer = AutoTokenizer.from_pretrained(config.base_model_name_or_path)

model.eval()

Downloading shards: 100%

13/13 [03:01<00:00, 17.22s/it]

model-00011-of-00013.safetensors: 100%

952M/952M [01:07<00:00, 10.4MB/s]

model-00012-of-00013.safetensors: 100%

948M/948M [01:20<00:00, 15.6MB/s]

model-00013-of-00013.safetensors: 100%

515M/515M [00:29<00:00, 16.9MB/s]

Loading checkpoint shards: 100%

13/13 [00:49<00:00, 3.41s/it]

generation_config.json: 100%

116/116 [00:00<00:00, 8.95kB/s]

adapter_model.safetensors: 100%

14.7M/14.7M [00:00<00:00, 93.1MB/s]

tokenizer_config.json: 100%

210/210 [00:00<00:00, 16.0kB/s]

tokenizer.json: 100%

1.65M/1.65M [00:00<00:00, 3.88MB/s]

special_tokens_map.json: 100%

185/185 [00:00<00:00, 14.1kB/s]

Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.

PeftModelForCausalLM(
 (base_model): LoraModel(
 (model): GPTNeoXForCausalLM(
 (gpt_neox): GPTNeoXModel(
 (embed_in): Embedding(30080, 4096)
 (emb_dropout): Dropout(p=0.0, inplace=False)
 (layers): ModuleList(
 (0-27): 28 x GPTNeoXLayer(
 (input_layernorm): LayerNorm((4096,), eps=1e-05, elementwise_affine=True)
 (post_attention_layernorm): LayerNorm((4096,), eps=1e-05, elementwise_affine=True)
 (post_attention_dropout): Dropout(p=0.0, inplace=False)
 (post_mlp_dropout): Dropout(p=0.0, inplace=False)
 (attention): GPTNeoXAttention(
 (rotary_emb): GPTNeoXRotaryEmbedding()
 (query_key_value): Lora.Linear4bit(
 (base_layer): Linear4bit(in_features=4096, out_features=12288, bias=True)
 (lora_dropout): ModuleDict(
 (default): Dropout(p=0.05, inplace=False)
)
)
 (lora_A): ModuleDict(

27초 오전 12:31에 완료됨

6.8

노트북

공유

도움

녹화가 시작되었습니다.

디스크

36.06 GB 사용 가능

찾기

오전 12:34 2024-06-21

PPTBIZCAM



2:00 PM

100%



감사합니다
