

미드로 배우는 Daily Conversation Chatfriends

18기 김나연
18기 이수민
19기 윤성민

CONTENTS

01

Introduction

- 주제선정 배경
- 데이터 소개

02

Process

- PEFT Method
- Model Structure
- Memory Buffer

03

Results

- 결과 공유

04

Future Prospect

- 한계점
- 활용 가능성





01. Introduction

01. 주제 선정 배경

본격적으로 프로젝트 시작에 앞서 논문 스터디 진행

- LoRA: Low-Rank Adaptation of Large Language Models
- Llama 2: Open Foundation and Fine-Tuned Chat Models (2023)
- QLoRA: Efficient Finetuning of Quantized LLMs)
- Mamba: Linear-Time Sequence Modeling with Selective State Spaces
- Jamba: A Hybrid Transformer-Mamba Language Model
- Toolformer: Language Models Can Teach Themselves to Use Tools

→ 챗봇 모델과 Model Lightweight에 대한 논문 위주로

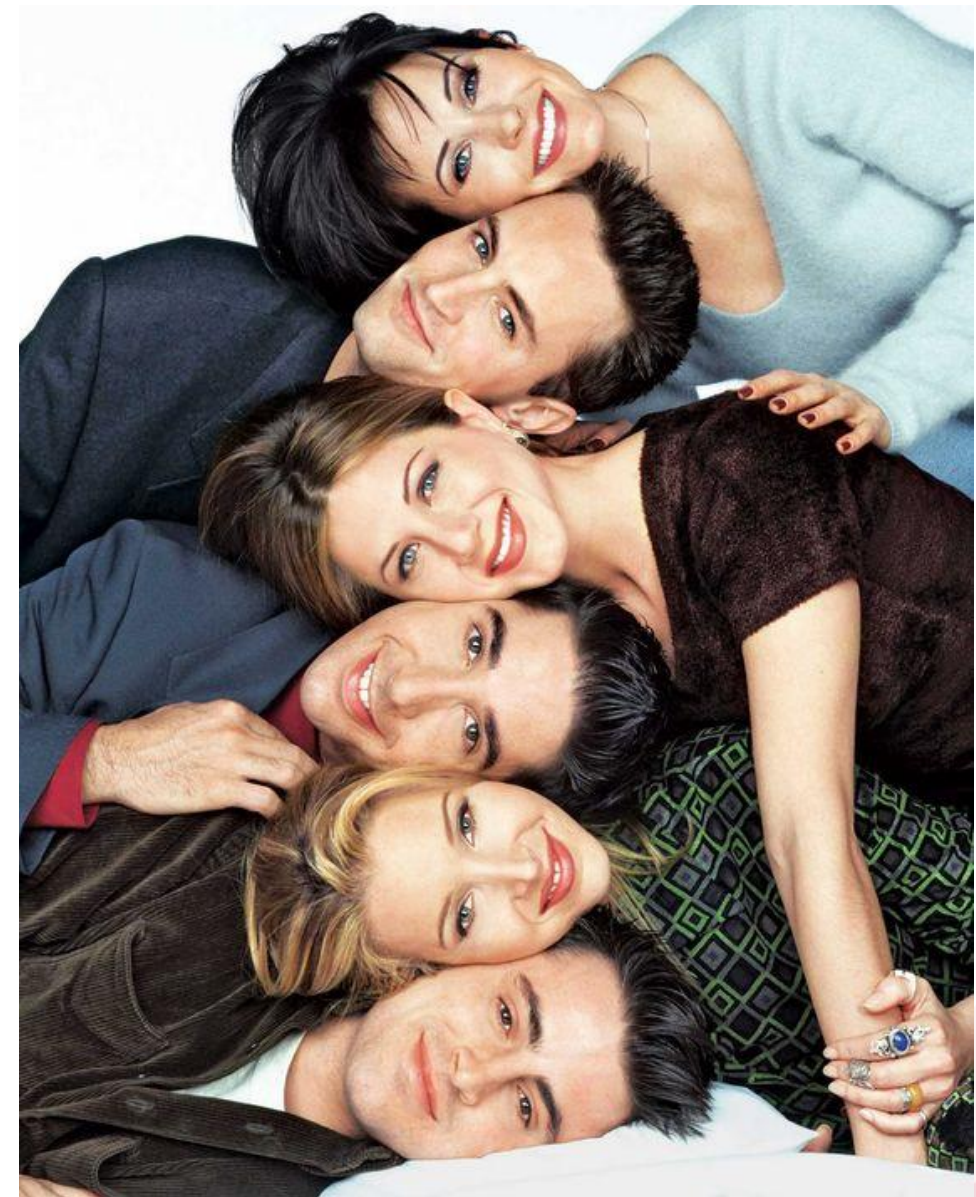
새로운 Architecture인 Mamba를 LoRA를 이용해서 Finetuning 해보자!

01. 데이터 소개

Customed data로 Chatbot을 구축하기

기존 모델들을 Daily conversation에 특화될 수 있도록 구어체 대화를 가능하게 하고,
Chatbot을 통한 비형식적 영어대화를 나눌 수 있도록 하는 것을 프로젝트 목표로 한다

-> 대표적인 미국드라마 Friends를 통해 Customed data를 생성하고 친구와 소통하는
듯한 Chatbot 구축을 하고자 함



01. 데이터 소개

사용 데이터

미드 Friends 대본 (docx 파일)

DataFrame 구축

→ scene, character, dialogue 각각 split

→ 각 대사와 그 대사를 한 인물이 있는 dataframe 생성

→ input, output 형식이 되도록 speaker, dialogue, next speaker,

next dialogue 열 형태의 dataframe 형식으로 csv 파일 생성

데이터 정제

- 정규표현식을 이용한 [Scene], (인물의 행동) 등 괄호 안의 문자 및 기호 삭제
- 여러 줄의 대사일 경우 하나의 row로 처리

```
# Function to parse the extracted text and extract dialogues
def parse_dialogues(text):
    dialogues = []
    lines = text.split("\n")[10:] # 앞부분 제거해주고 lines 지정
    for line in lines:
        # print(line)
        if ":" in line:
            parts = line.split(":", 1)
            speaker = parts[0].strip()
            dialogue = parts[1].strip()
            if dialogue:
                dialogues.append((speaker, dialogue))

        elif (":" not in line) & (line[0] != '[') & (line[0] != '('):
            dialogues.pop()
            dialogue = (dialogue + f' {line}').strip()
            dialogues.append((speaker, dialogue))

    return dialogues

def make_dialogues(f):
    conversation_pairs = []
    for i in range(0, f.shape[0]-1):
        current_speaker = f.loc[i, 'Character']
        current_line = re.sub(r"\\[.*\\]|\\(.*)|\\{.*\\}|-", "", f.loc[i, 'Dialogue']).strip()
        next_speaker = f.loc[i+1, 'Character']
        next_line = re.sub(r"\\[.*\\]|\\(.*)|\\{.*\\}|-", "", f.loc[i+1, 'Dialogue']).strip()
        if (current_speaker.isalpha()) & (next_speaker.isalpha()): # 발화자/응답자 처리
            conversation_pairs.append((current_speaker, current_line, next_speaker, next_line))
    return conversation_pairs

# Function to create conversation pairs from dialogues
def create_conversation_pairs(dialogues):
    conversation_pairs = []
    for i in range(len(dialogues) - 1):
        current_speaker = dialogues[i][0]
        current_line = re.sub(r"\\[.*\\]|\\(.*)|\\{.*\\}|-", "", dialogues[i][1]).strip() # re.sub("
        next_speaker = dialogues[i + 1][0]
        next_line = re.sub(r"\\[.*\\]|\\(.*)|\\{.*\\}|-", "", dialogues[i+1][1]).strip()
        if (current_speaker.isalpha()) & (next_speaker.isalpha()): # 발화자/응답자 처리
            conversation_pairs.append((current_speaker, current_line, next_speaker, next_line))
    return conversation_pairs
```



02. Process

02. PEFT Method

LoRA 모델을 이용해서 경량화 시도

```
5 tokenizer = AutoTokenizer.from_pretrained("state-spaces/mamba-2.8b-hf")
6 model = AutoModelForCausalLM.from_pretrained("state-spaces/mamba-2.8b-hf")
7
8 dataset = load_dataset("csv", data_files="./data/friends_conversation_dataset_0626.csv", field='data')
9 training_args = TrainingArguments(
10     output_dir="./model/results",
11     num_train_epochs=3,
12     per_device_train_batch_size=4,
13     logging_dir='./model/logs',
14     logging_steps=10,
15     learning_rate=2e-3
16 )
17 lora_config = LoraConfig(
18     r=8,
19     target_modules=["x_proj", "embeddings", "in_proj", "out_proj"],
20     task_type="CAUSAL_LM",
21     bias="none"
22 )
```

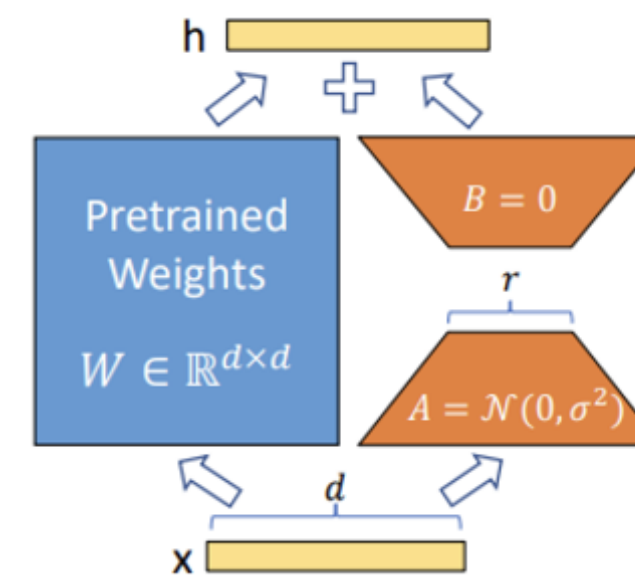



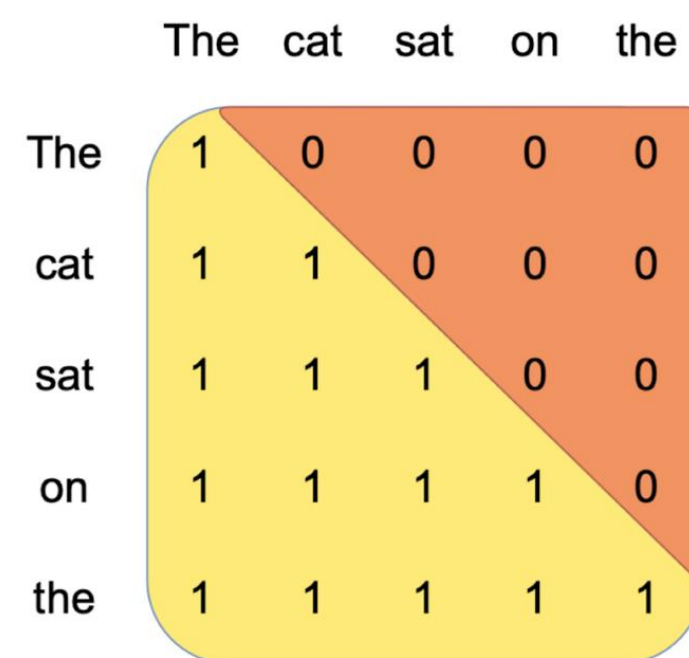
Figure 1: Our reparametrization. We only train A and B .

저차원의 intrinsic rank를 이용해 finetuning하는 방법론을 제시하는 논문

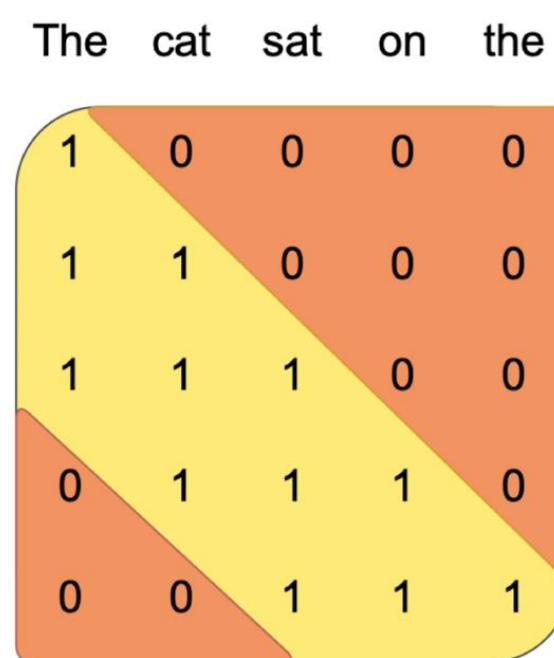
- 하나의 Pre-trained model에 대한 작은 LoRA 모듈들을 만들어 서로 다른 task에 적용할 수 있어 저장용량 측면과 task 전환 측면에서 매우 유용
- 대부분의 parameter에 대해 추가적인 최적화를 진행할 필요가 없고, Fully fine-tuned model에 비교하여 추가적인 inference latency가 없음

02. Model Structure

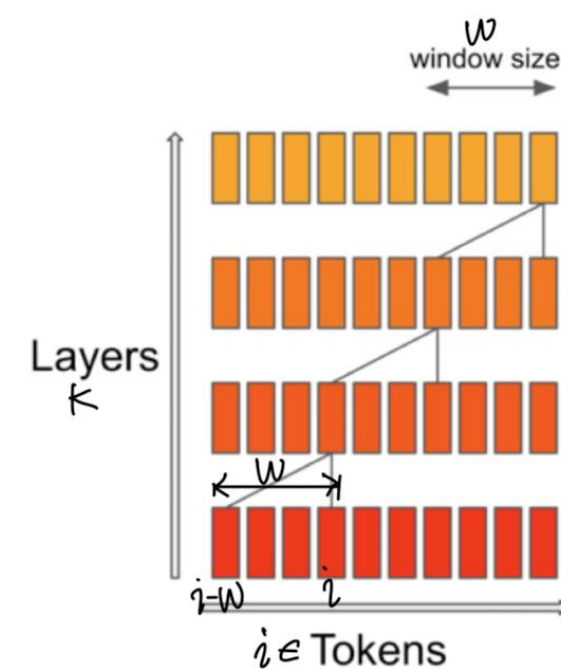
 [mistralai/Mistral-7B-Instruct-v0.3](#)



Vanilla Attention



Sliding Window Attention

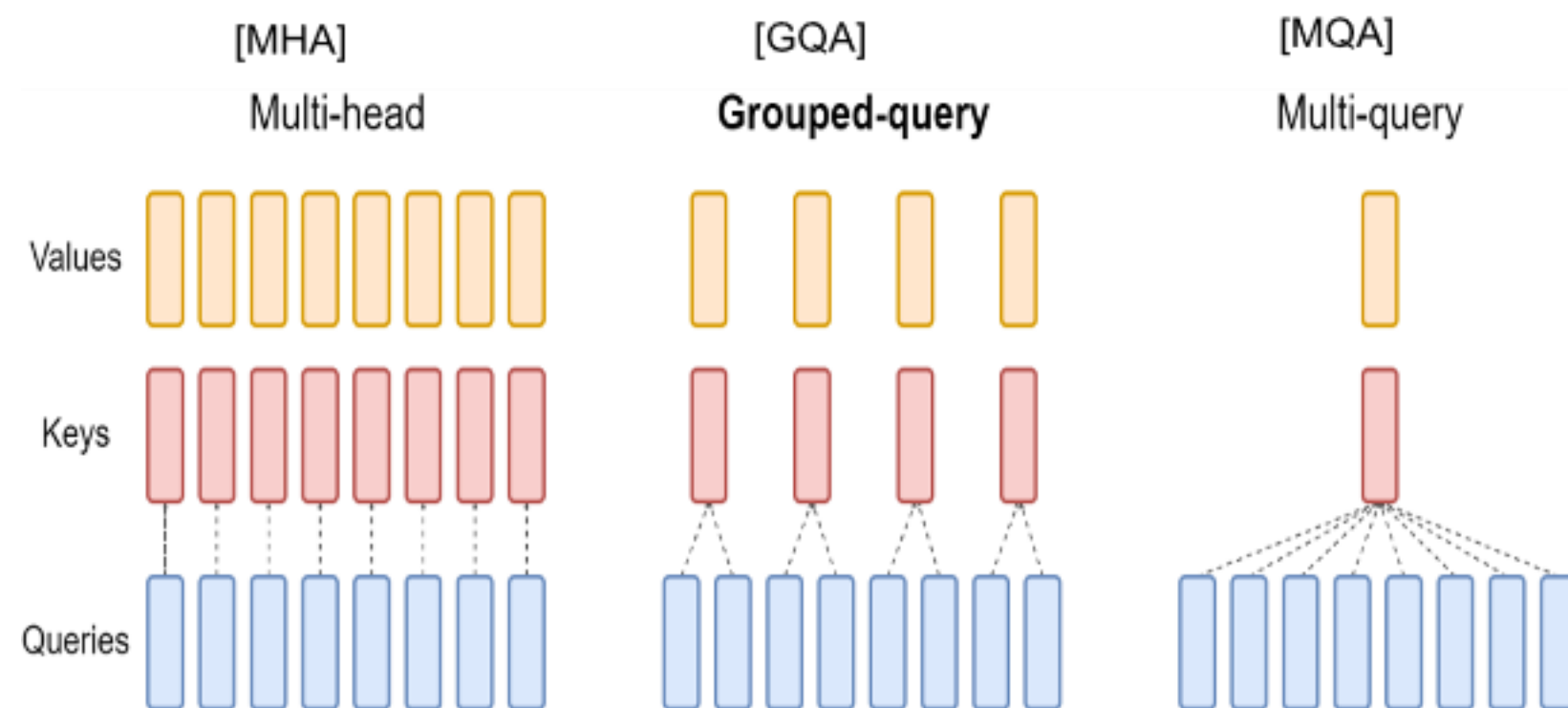


Effective Context Length

Llama2와 비교하여 추론, 이해, STEM에서 3배 이상의 좋은 성능을 보인 Transformer 구조의 모델

02. Model Structure

mistralai/Mistral-7B-Instruct-v0.3



1) Sliding Window Attention

→ Window size 제한으로 정보손실이 적음

2) Rolling Buffer Cache

→ Cache size 제한으로 fixed size 이상이 된 경우 기존 캐시의 자리에 덮어 씌우는 방식

3) Pre-fill and Chunking

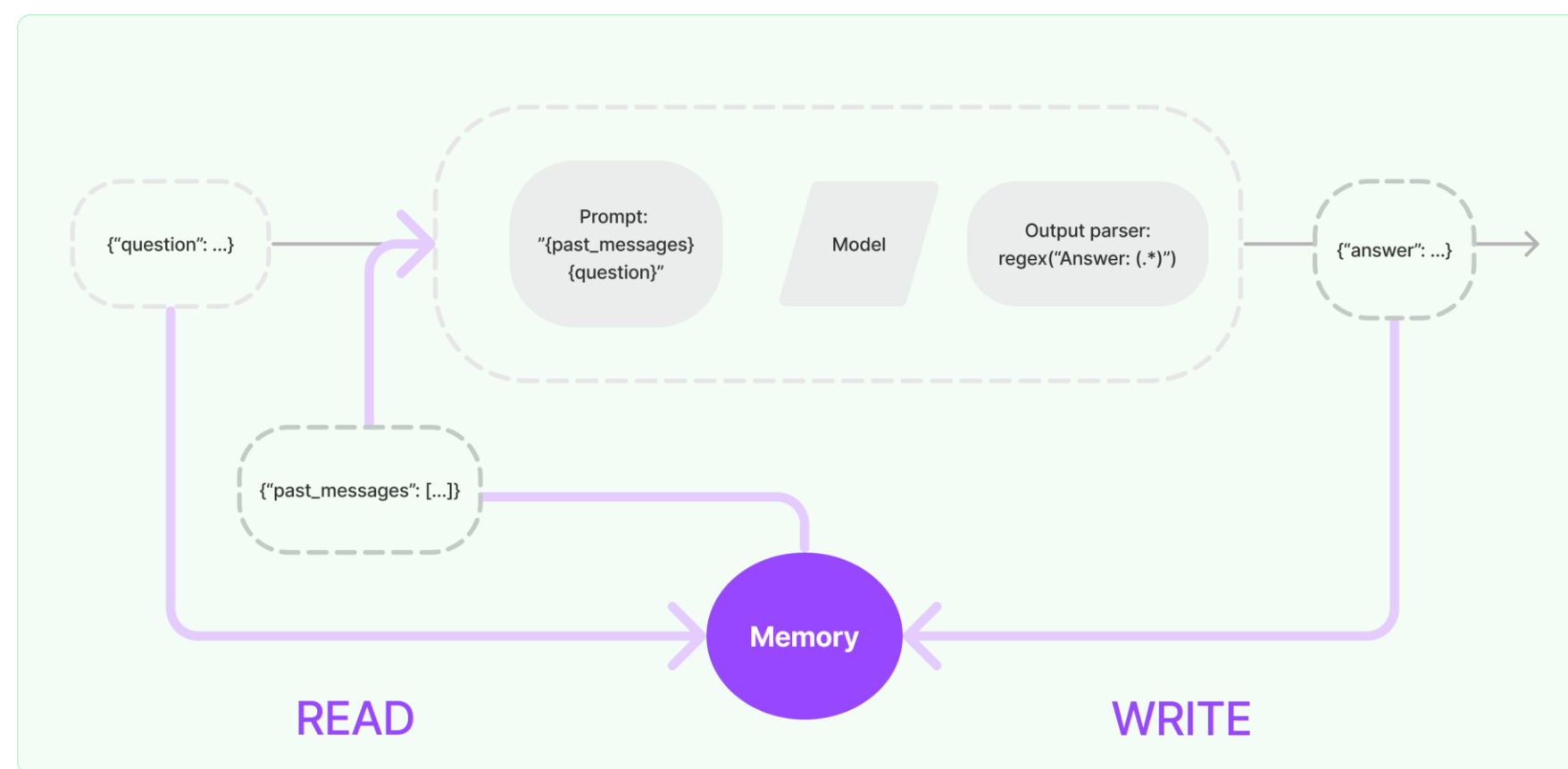
→ Chunk size로 자동으로 window size 지정

4) GQA(Group-Query Attention)

→ MHA와 MQA 사이 유리한 Trade-off

03. Memory Buffer

Langchain을 이용한 Memory Buffer 구현



단순 QA형식의 Chatbot이 아닌 기존의 대화내용과 정보를 기억하는 Chatbot을 구현

LangChain의 Conversation Buffer을 활용

03. Memory Buffer

Langchain을 이용한 Memory Buffer 구현

● Nangni/mistral_friends

```
1 from transformers import AutoTokenizer, AutoModelForCausalLM
2
3 # 모델 및 토크나이저 로드
4 model_path = './mistral3/fine_tuned_model'
5 tokenizer = AutoTokenizer.from_pretrained(model_path)
6 model = AutoModelForCausalLM.from_pretrained(model_path)
7
```

숨겨진 출력 표시

```
1 ## repo
2 MODEL_SAVE_REPO = 'friends_mistral' # ex) 'my-bert-fine-tuned'
3 HUGGINGFACE_AUTH_TOKEN = HF_TOKEN # https://huggingface.co/settings/token
4
5 ## Push to huggingface-hub
6 model.push_to_hub(
7     'Nangni/friends_mistral',
8     use_temp_dir=True,
9     use_auth_token=HUGGINGFACE_AUTH_TOKEN
10 )
11 tokenizer.push_to_hub(
12     'Nangni/friends_mistral',
13     use_temp_dir=True,
14     use_auth_token=HUGGINGFACE_AUTH_TOKEN
15 )
```

```
6 llm = HuggingFaceEndpoint(
7     repo_id=repo_id,
8     max_new_tokens=20,
9     temperature=0.1,
10     #callbacks=[StreamingStdOutCallbackHandler()],
11     streaming=False,
12 )
13
14
15 memory = ConversationSummaryBufferMemory(
16     llm=llm,
17     max_token_limit=80,
18     memory_key="chat_history",
19     return_messages=True,
20 )
21
22 memory.clear()
23
24 def load_memory():
25     return memory.load_memory_variables({})["chat_history"]
26
27 prompt = ChatPromptTemplate.from_messages([
28     ("system", "You are a helpful AI talking to human"),
29     MessagesPlaceholder(variable_name="chat_history"),
30     ("human", "{question}"),
31     ("ai", "answer"),
32 ])
33
34 llm_chain = LLMChain(prompt=prompt, llm=llm)
35
36 def invoke_llm_chain(question, chat_history=None):
37     if chat_history is None:
38         chat_history = load_memory()
39     result = llm_chain.invoke({"question": question, "chat_history": chat_history})
40     # result_splited = result['text'].split('\n')[0]
41     # memory.save_context({"input": question}, {"output": result_splited})
42     print(result)
```



03. Results

03. 결과 공유

Baseline fine-tuned GPT2

```
User: Hi how are you doing ?  
Chatbot: Hi how are you doing?",Chandler,"Oh, I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm  
fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine. I'm fine  
User: why are you repeating I'm fine that much?  
Chatbot: why are you repeating I'm fine that much? I mean, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I  
'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine, I'm fine  
User: ... okay, then talk about what you're doing these days.  
Chatbot: ... okay, then talk about what you're doing these days. Season 1"  
Ross,"Okay, okay, then talk about what you're doing these days. Season 1",Rachel,"Okay, okay, then talk about what you're doing these days.  
Season 1",Ross,"Okay, okay, then talk about what you're doing these days. Season 1",Rachel,"Okay, okay, then talk about what you're doing
```

Base-model Mistral Inference

LS hello how are you ?

Hello! I'm an assistant, so I don't have feelings, but I'm here and ready to help you. How can I assist you today?

22:02 · Large



LS do you plan tonight ?

I don't have personal plans as I am an assistant and don't have personal experiences or needs. However, I can help you with your plans for tonight. Do you need suggestions for activities, help with making reservations, or something else? Just let me know how I can assist you.

03. 결과 공유

Mistral Model + Langchain MemoryBuffer

User: how are you doing ?
Chatbot: how are you doing ?

i am doing well, thank you. how are you?

User: i'm good to do you have any plan tonight?
Chatbot: i'm good to do you have any plan tonight?

no, i don't have any plans for tonight. i'm just going to relax and watch some tv. how about you? do you have any plans?

```
invoke_llm_chain("My name is Yun")
```

: Nice to meet you, Yun! How can I assist you today?

```
invoke_llm_chain("What's my name?")
```

: Your name is Yun, as you mentioned earlier in our conversation.
None

```
invoke_llm_chain("My job is a bartender.")
```

: That's interesting! As a bartender, you have the opportunity to interact with a variety of people and help create a welcoming atmosphere for your customers. You may also have the chance to learn
None

```
invoke_llm_chain("What's my job and my name?")
```

: Your job is a bartender, and your name is Yun. I'm here to help you with any questions or topics you'd like to discuss. Is there something specific you'd like to know or talk about?



04. Future Prospect

04. Future Work

한계점

- 한정된 Memory, Computing 자원 문제로 더 다양한 시도를 하지 못했던 점
Ex) finetuning parameter 조절, 기존에 사용하고자 했던 Mamba structure 사용 등
- transcript 사용으로 반복되는 단어나 말이 중간에 끊기는 경우를 처리하기 까다로웠던 점

활용가능성

- 구어체의 형식으로 영어를 포함한 제2외국어를 일상대화형식으로 배우는 것이 가능
- 인물의 특성을 반영한 model을 각각 생성하여 실제 인물과 대화하는 듯한 시뮬레이션으로 활용가능
- Prototype 제작 및 배포로 깔끔한 interface 지원



Thank You