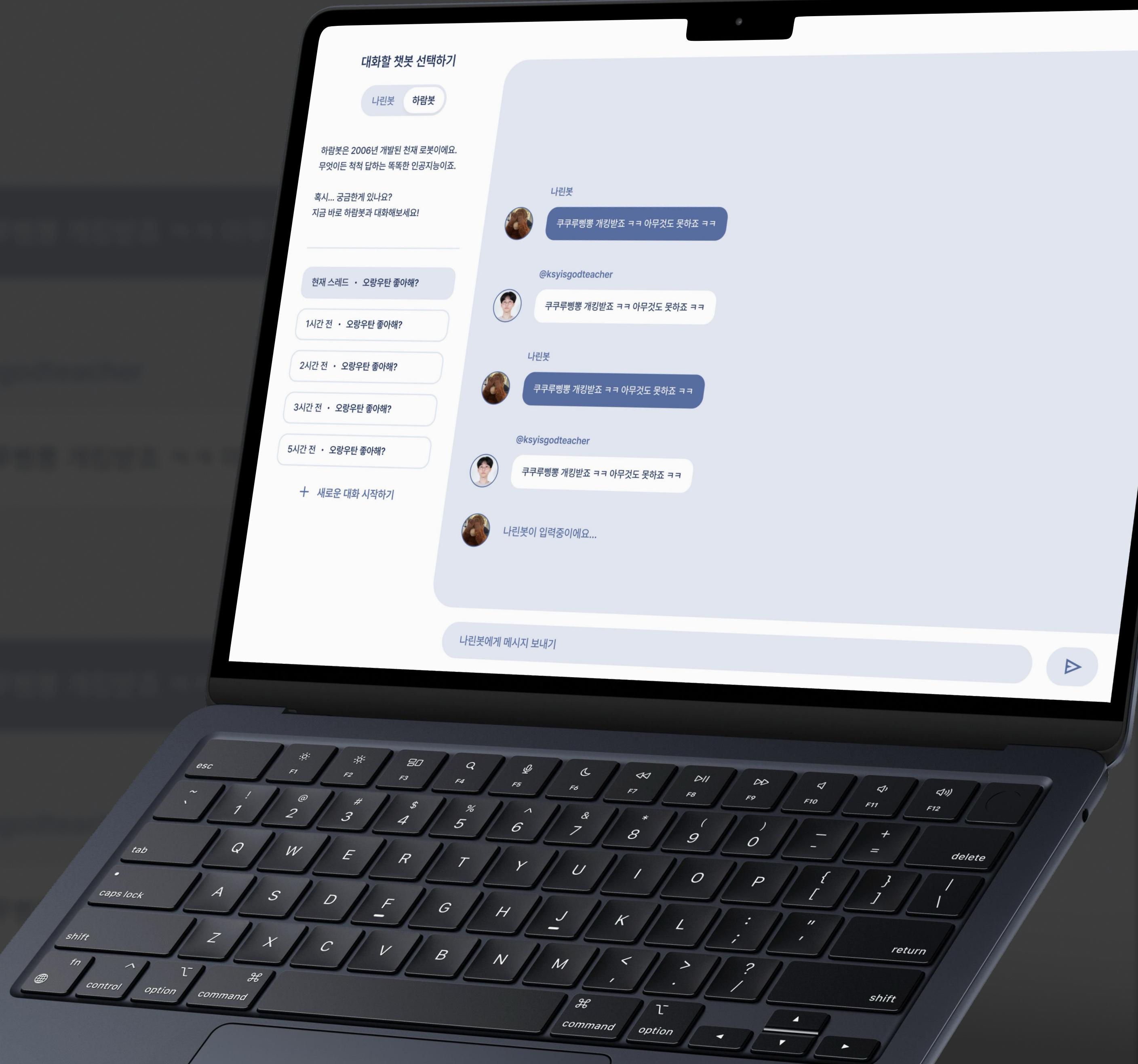


박시원 · 오유성 · 이나린 · 이하람 · 주현명

문장 분석을 통한 채팅봇 서비스 제작



1. 연구 주제와 연구 동기

2. 연구/개발 과정

3. 구현된 서비스 시연

4. 연구 결과

1. 연구 주제와 연구 동기

1-1. 개요

주제 친구들의 DM을 기반으로 한 개인화된 대화형 AI 모델 개발 및 챗봇 서비스 제작

개발 방법 개인의 말투와 특성을 학습시킨 후, 이를 GPT 모델에 적용

연계 교과 커뮤니케이션 문학, 인공지능과 미래사회

목표 유니크하고 개인화된 대화 경험을 제공하는 AI 제작 이후 웹 서비스와 통합

1. 연구 주제와 연구 동기

1-2. 연계 교과와 동기

커뮤니케이션 문학

기술 발전과 커뮤니케이션:

디지털 시대에서 AI의 역할이

커뮤니케이션에 미치는 영향을 이해하고자 합니다.

개인화와 커뮤니케이션의 진화:

개인화된 말투와 스타일을 반영하는 AI가 인간 간

커뮤니케이션을 어떻게 변화시킬 수 있는지 탐구합니다.

인공지능과 미래사회

AI의 사회적 영향:

AI 기술이 미래 사회에

어떤 변화를 가져올지 탐구합니다.

기술 윤리와 책임:

개인 데이터를 사용하는 AI 개발 과정에서의

윤리적 고려사항에 대해 이해하고자 합니다.

1. 연구 주제와 연구 동기

1-3. 연구 교과와 목적

커뮤니케이션 문학

대화 스타일의 개인화:

개인의 특성을 반영하는 대화 AI의 개발을 통해
커뮤니케이션의 새로운 형태를 탐색합니다.

인간-AI 상호작용 이해:

AI와 인간 간의 상호작용이 언어 사용과 커뮤니케이션
스타일에 어떤 영향을 미치는지 연구합니다.

인공지능과 미래사회

AI의 사회적 적용:

개인화된 AI가 사회적 상호작용과
커뮤니케이션에 미치는 영향을 탐구합니다.

윤리적 AI 개발:

개인정보 보호 및 데이터 사용의 윤리적 기준을 설정하고
준수하는 AI 개발 방법을 모색합니다.

1. 연구 주제와 연구 동기

1-4. 연구 교과와 문제

커뮤니케이션 문학

개인화된 AI 모델이 인간의 언어 사용 및
커뮤니케이션 스타일에 어떤 변화를 가져올 수 있을까?
AI와 인간 상호작용이 커뮤니케이션 이론에
어떤 새로운 시사점을 제공할 수 있을까?

인공지능과 미래사회

개인화된 AI 모델이 미래 사회에 어떤 영향을 미칠 것인가?
AI 개발 과정에서 데이터 사용의 윤리적 기준을
어떻게 확립/유지할 수 있을까?

1. 연구 주제와 연구 동기

1-5. 연구 교과와 방법

커뮤니케이션 문학

언어 분석:

수집된 DM 데이터를 분석하여 개인의 언어 스타일과 커뮤니케이션 패턴을 파악합니다.

AI 모델 통합과 상호작용 연구:

분석된 데이터를 기반으로 AI 모델을 학습시켜 개인화된 대화 스타일을 구현합니다. 이후 AI, 사용자의 상호작용을 관찰하고, 그 영향을 분석합니다.

인공지능과 미래사회

윤리적 기준 설정:

데이터 수집 및 사용 과정에서의 윤리적 기준을 마련하고 준수합니다.

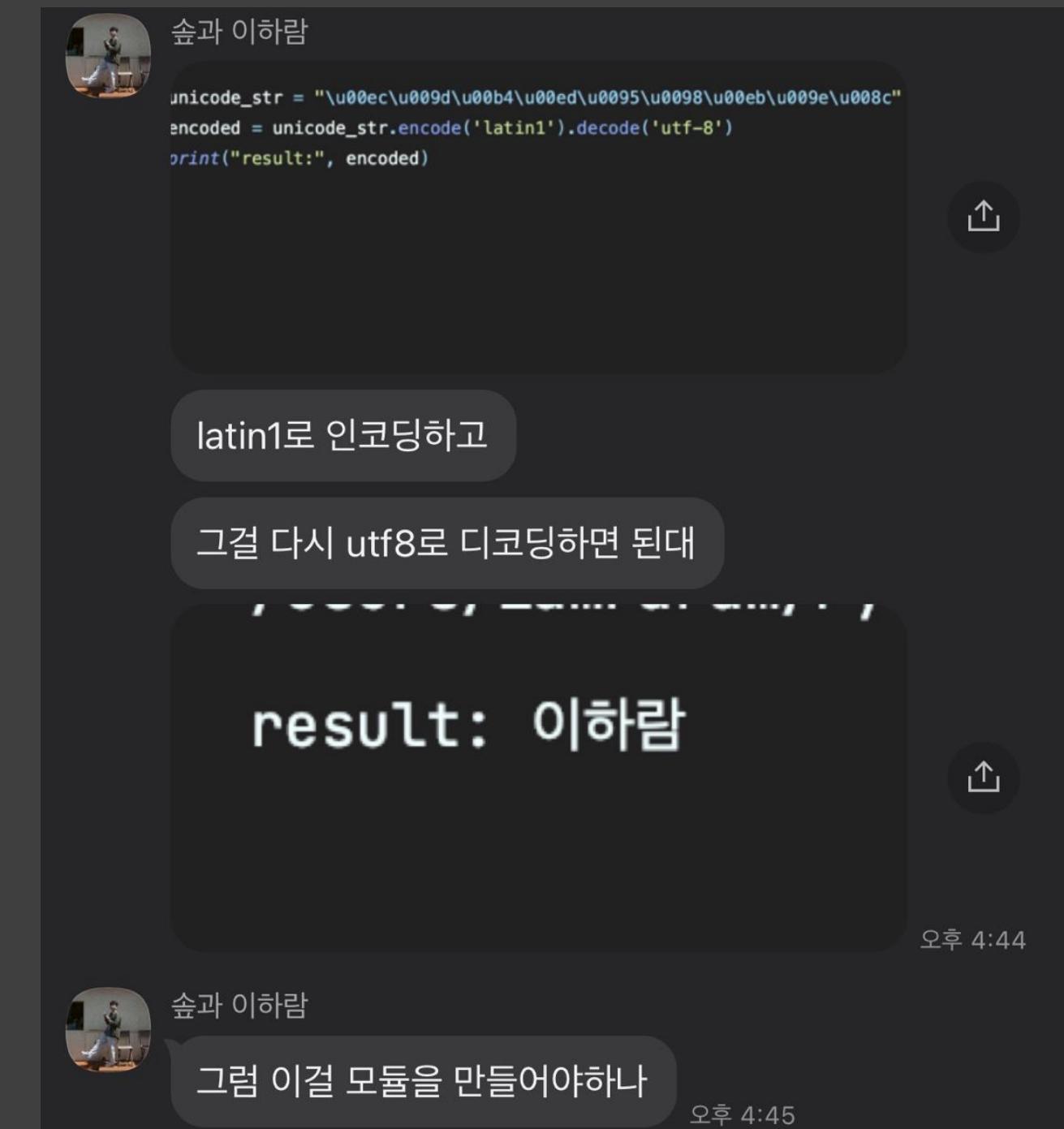
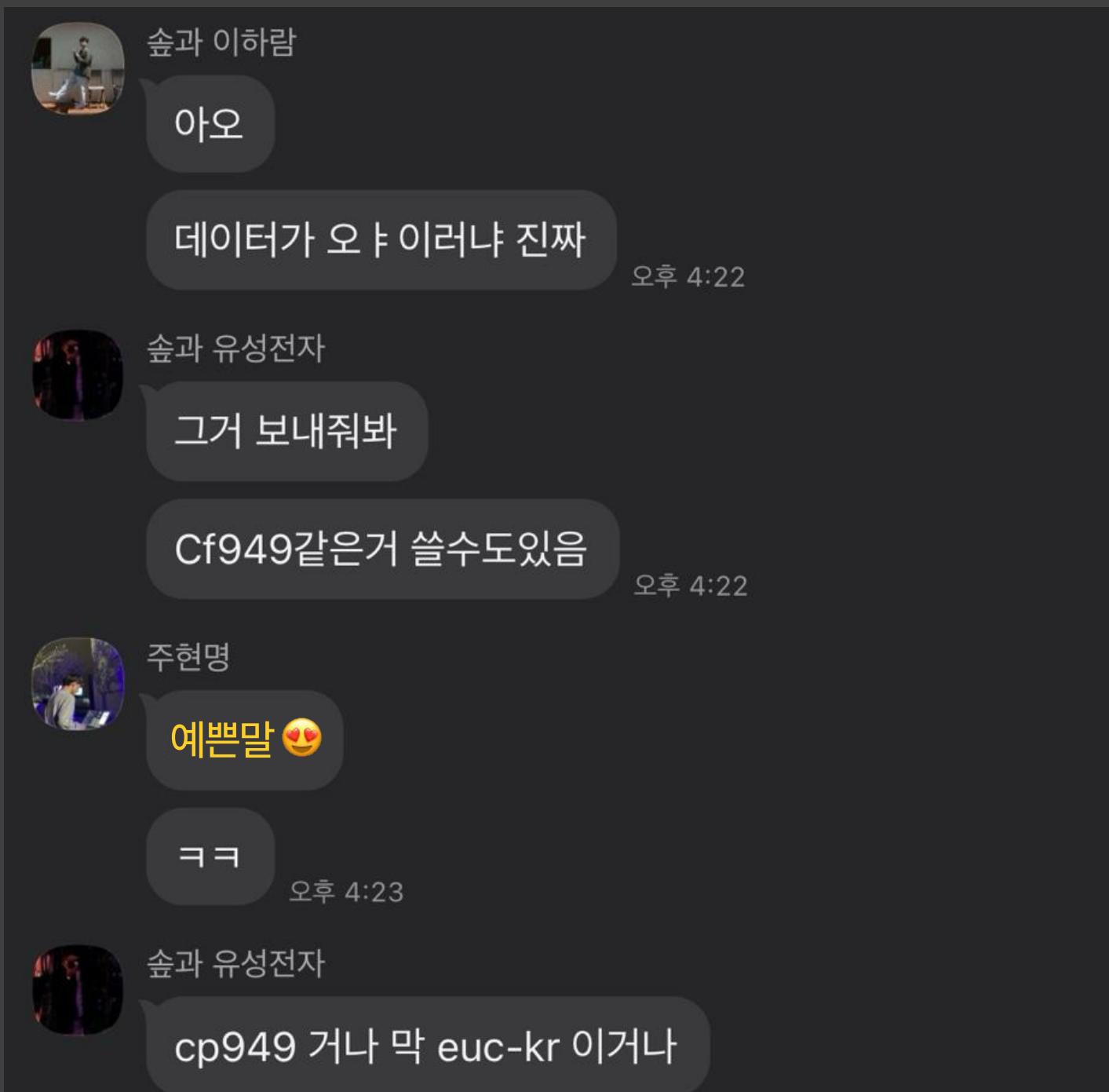
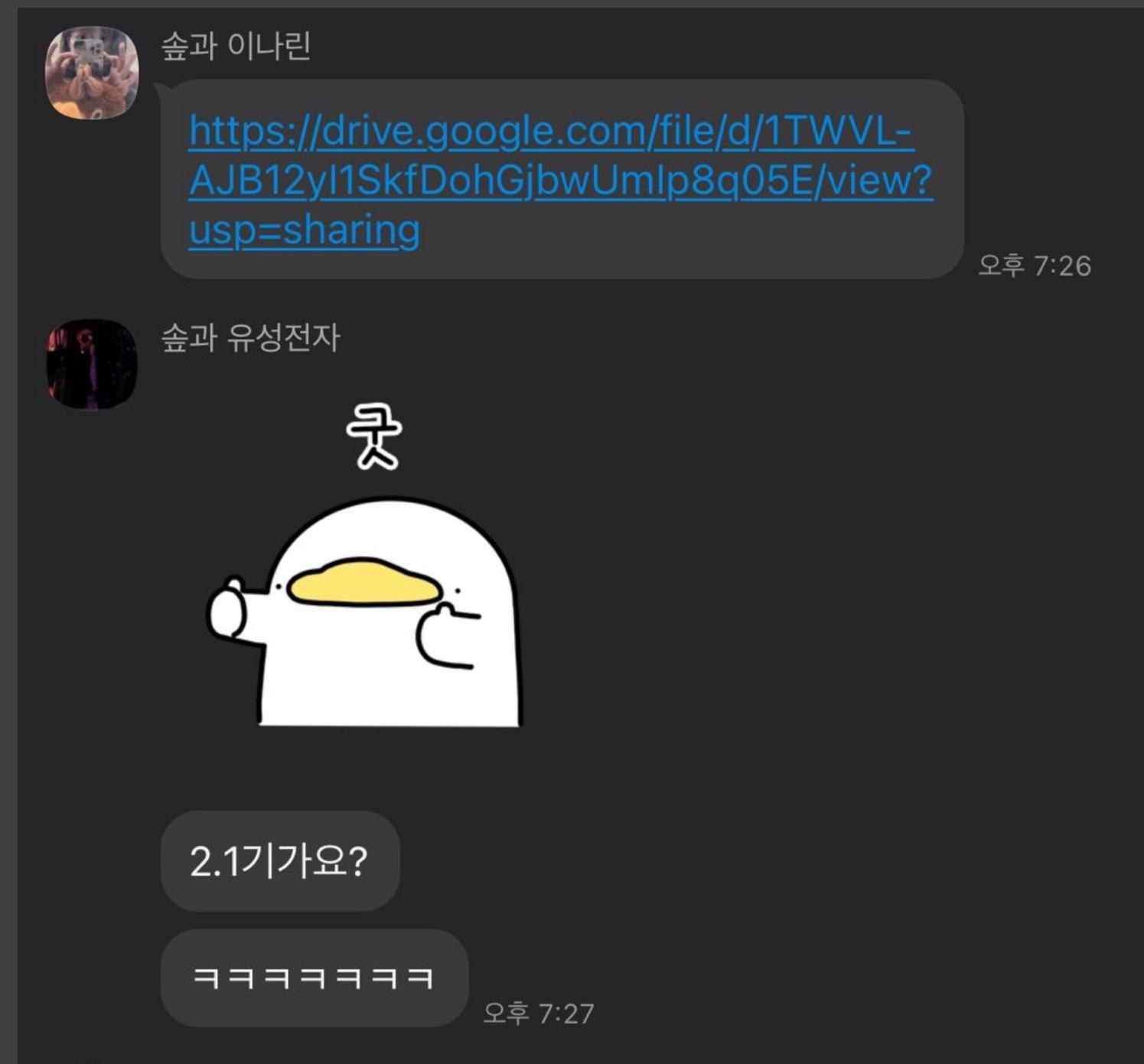
실험 및 설문조사:

AI 모델을 사용한 실험과 사용자의 피드백을 통해 AI의 사회적 영향을 평가합니다.

2. 연구/개발 과정

2-1. 데이터 수집

Step 1. 데이터 다운로드 후 데이터의 형태 파악하기



2. 연구/개발 과정

2-2. 데이터 전처리

Step 2. 더럽게 인코딩된 데이터를 우리가 이해할 수 있도록 자연어로 바꿔주기



```
original_string = "lue@ec\u0098\vooa4\uo@ec\u009c\uooao\uo@ec\u0084\uoob1"
utf8_decoded = original_string.encode ("latin1") .decode ("utf-8")
print (utf8_decoded)
```

2. 연구/개발 과정

2-2. 데이터 전처리

Step 3. 모델 학습에 불필요한 데이터 삭제하기 (삭제된 메시지, 영상통화, 이모티콘 등)

```
● ● ●

def process_message(content):
    # URL 제거
    content = re.sub(r'http\S+', '', content)
    # 특수 문자 및 기호 제거
    content = re.sub(r'[\^\\w\\s]', '', content)
    # 유니코드 정규화
    content = unicodedata.normalize('NFKC', content)
    # 너무 짧거나 긴 메시지 제거
    if 2 <= len(content) <= 200 and not is_repetitive_single_char(content):
        # BOS와 EOS 토큰 추가
        return f"{BOS_TOKEN} {content} {EOS_TOKEN}"

    # 필터링 및 전처리
    content = re.sub(r'삭제된 메세지입니다|이모티콘|사진|동영상|스토리를 공유했습니다|영상 통화를 시작했습니다|영상 통화가 종료되었습니다|메시지를 좋아합니다', '', content)
    content = re.sub(r'(.+)\1{2,}', '', content)
    processed_content = process_message(content)
```

2. 연구/개발 과정

2-2. 데이터 전처리

Step 4. 전처리를 적용해 같은 날의 메시지만 저장하기

```
● ● ●

BOS_TOKEN = '<s>' # 시작 토큰
EOS_TOKEN = '</s>' # 종료 토큰

def is_same_day(date1, date2):
    """ 두 날짜가 같은 날인지 확인 """
    return date1.date() == date2.date()

def process_json_file(file_path):
    print("Processing: ", file_path)
    try:
        with open(file_path, 'r', encoding='latin1') as file:
            data = json.load(file)

            messages_pairs = []
            last_question = None
            last_answer = None

            for message in data['messages']:
                sender = message['sender_name'].encode("latin1").decode("utf-8")
                content = message.get('content', '').encode("latin1").decode("utf-8")
                timestamp = message.get('timestamp_ms', 0)
                date = parse_date(timestamp)
```

2. 연구/개발 과정

2-2. 데이터 전처리

Step 5. 전처리 데이터 확인

haram_messages.txt

Q: <s> 하람모바일 </s> A: <s> 그거 </s>

Q: <s> 혹시 형들중 한명이 운영하시는 </s> A: <s> 몰르는데 </s>

Q: <s> 너 막내아니였 </s> A: <s> 동생이있긴해 </s>

Q: <s> 무섭다 하람이 </s> A: <s> 조신해 </s>

Q: <s> 완전 터져버리는거지 </s> A: <s> 나네 </s>

Q: <s> 너 머리기름 </s> A: <s> 꼈다니 </s>

Q: <s> 주식도 하니 부자네 </s> A: <s> 15년치 세뱃돈이 </s>

2. 연구/개발 과정

2-3. 모델 선정

Step 1. 프로젝트 준비와 조사

The diagram illustrates the GPT-2 model family, showing four variants: Small, Medium, Large, and Extra Large. Each variant is represented by a logo and a title. Below each title is a vertical stack of pink rectangular boxes labeled 'DECODER'. The number of these decoder layers increases progressively from 12 for Small to 48 for Extra Large.

Model	Decoder Layers	Model Dimensionality
GPT-2 SMALL	12, ..., 1	768
GPT-2 MEDIUM	24, ..., 2, 1	1024
GPT-2 LARGE	36, ..., 4, 3, 2, 1	1280
GPT-2 EXTRA LARGE	48, ..., 6, 5, 4, 3, 2, 1	1600

개인 프로젝트: 나처럼 말하는 봇을 만들어보자! - 2

tmddn0311 · 2021년 5월 7일
GPT2 NLP chatbot huggingface

project

▼ 목록 보기

2/2 < >

me: 0F
me: 뭐해
me:
AI: 나
AI: 나 지금
AI: 집이야
AI: ㅋㅋ
me: 뭐야 왜 별써 집임
me:

2. 연구/개발 과정

2-3. 모델 선정

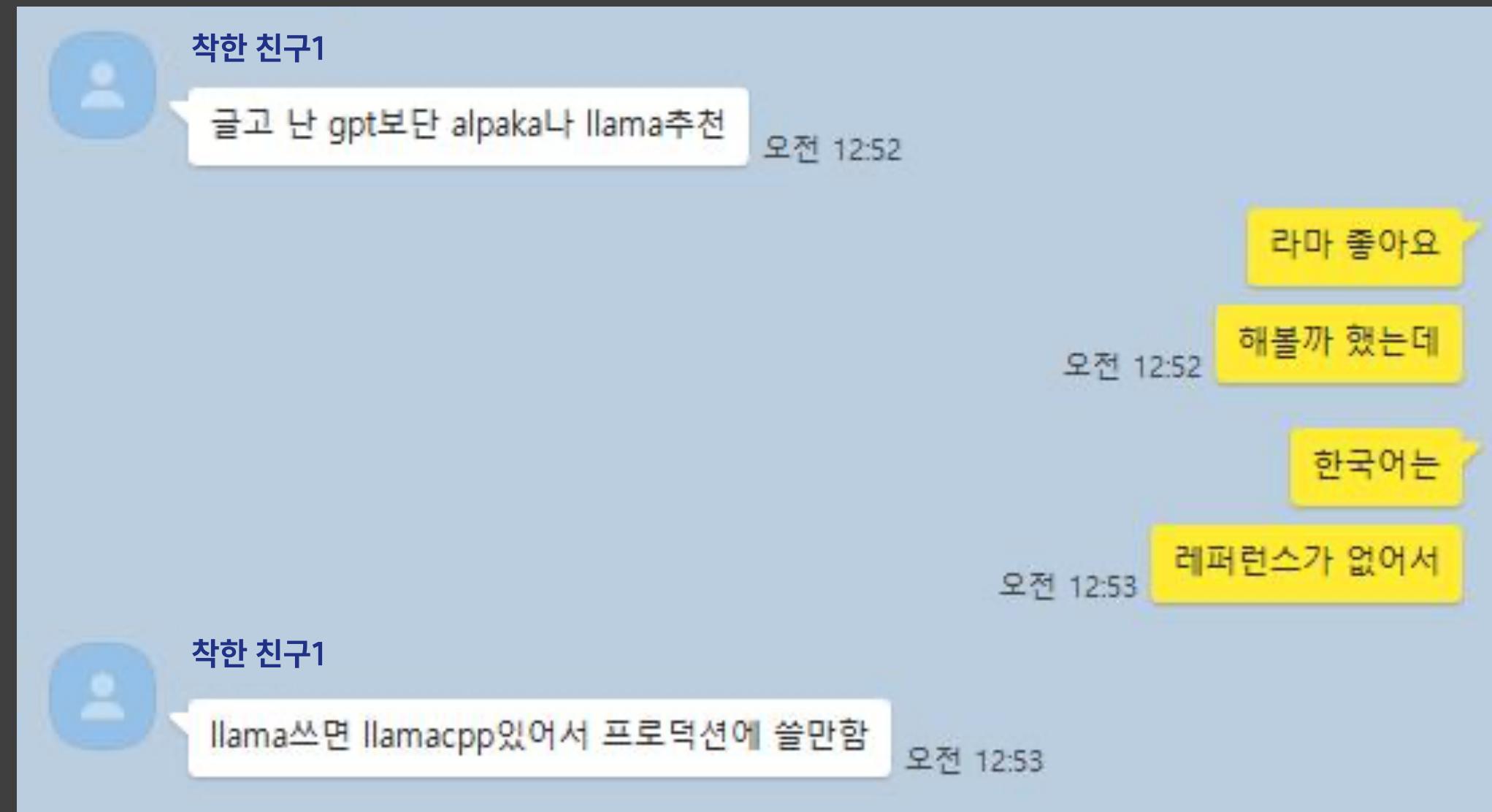
Step 3. GPT에 관한 간단한 고찰



2. 연구/개발 과정

2-3. 모델 선정

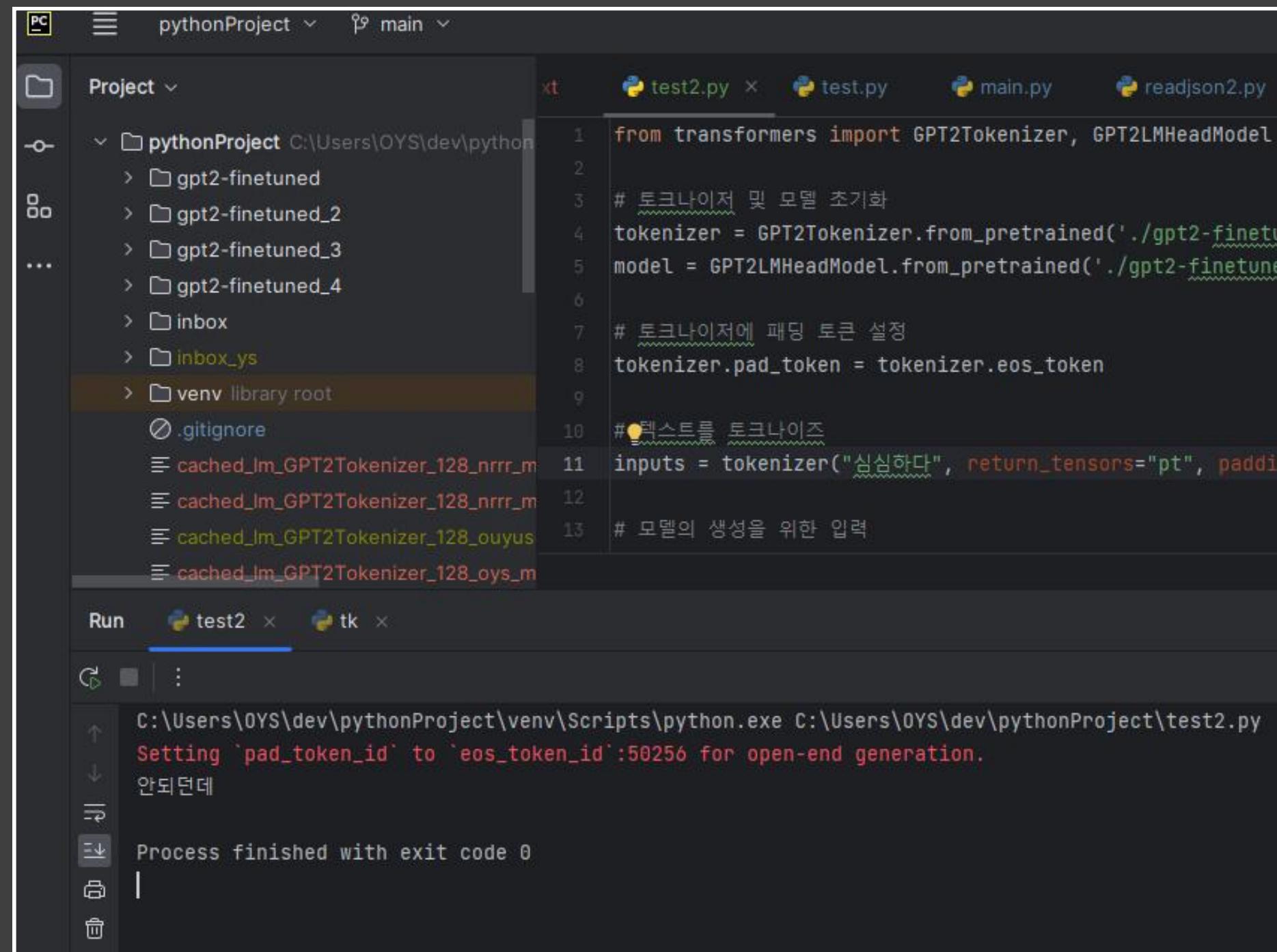
Step 3. 어떤걸 사용해볼까?



2. 연구/개발 과정

2-3. 모델 선정

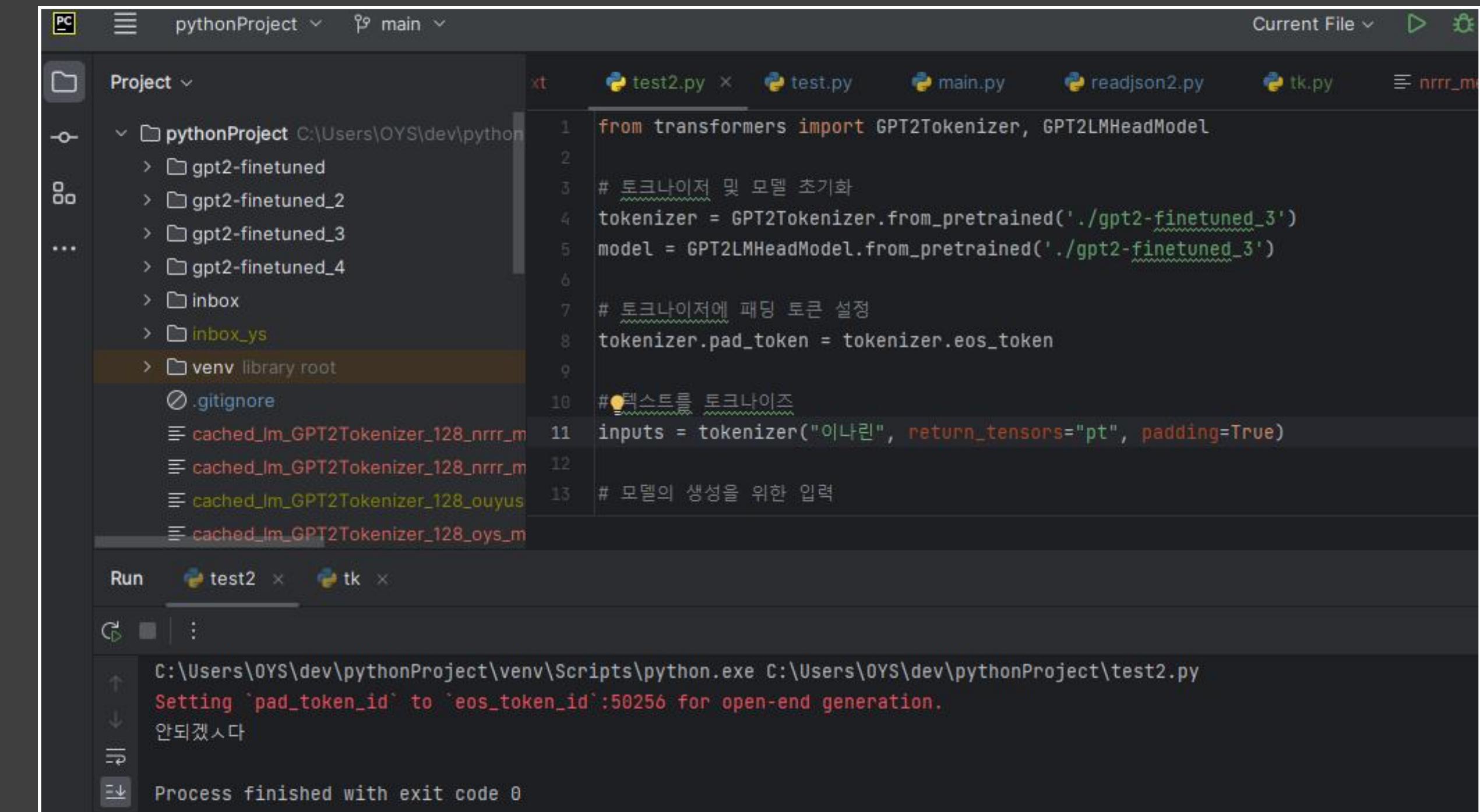
Step 2. GPT 시도



PyCharm IDE screenshot showing a Python project named 'pythonProject'. The current file is 'test2.py'. The code imports GPT2Tokenizer and GPT2LMHeadModel from the transformers library. It initializes a tokenizer and model from pretrained weights. It sets the pad_token to the eos_token. It tokenizes the text "심심하다" and creates tensors. The terminal output shows the command run and the resulting tokens.

```
from transformers import GPT2Tokenizer, GPT2LMHeadModel
# 토크나이저 및 모델 초기화
tokenizer = GPT2Tokenizer.from_pretrained('./gpt2-finetuned')
model = GPT2LMHeadModel.from_pretrained('./gpt2-finetuned')
# 토크나이저에 패딩 토큰 설정
tokenizer.pad_token = tokenizer.eos_token
# 텍스트를 토크나이즈
inputs = tokenizer("심심하다", return_tensors="pt", padding=True)
# 모델의 생성을 위한 입력
cached_lm_GPT2Tokenizer_128_nrrr_m
cached_lm_GPT2Tokenizer_128_nrrr_m
cached_lm_GPT2Tokenizer_128_ouyus
cached_lm_GPT2Tokenizer_128_ouyus
```

C:\Users\OYS\dev\pythonProject\venv\Scripts\python.exe C:\Users\OYS\dev\pythonProject\test2.py
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
안되던데
Process finished with exit code 0



PyCharm IDE screenshot showing a Python project named 'pythonProject'. The current file is 'test2.py'. The code is identical to the one in the first screenshot, but it tokenizes the text "이나린". The terminal output shows the command run and the resulting tokens.

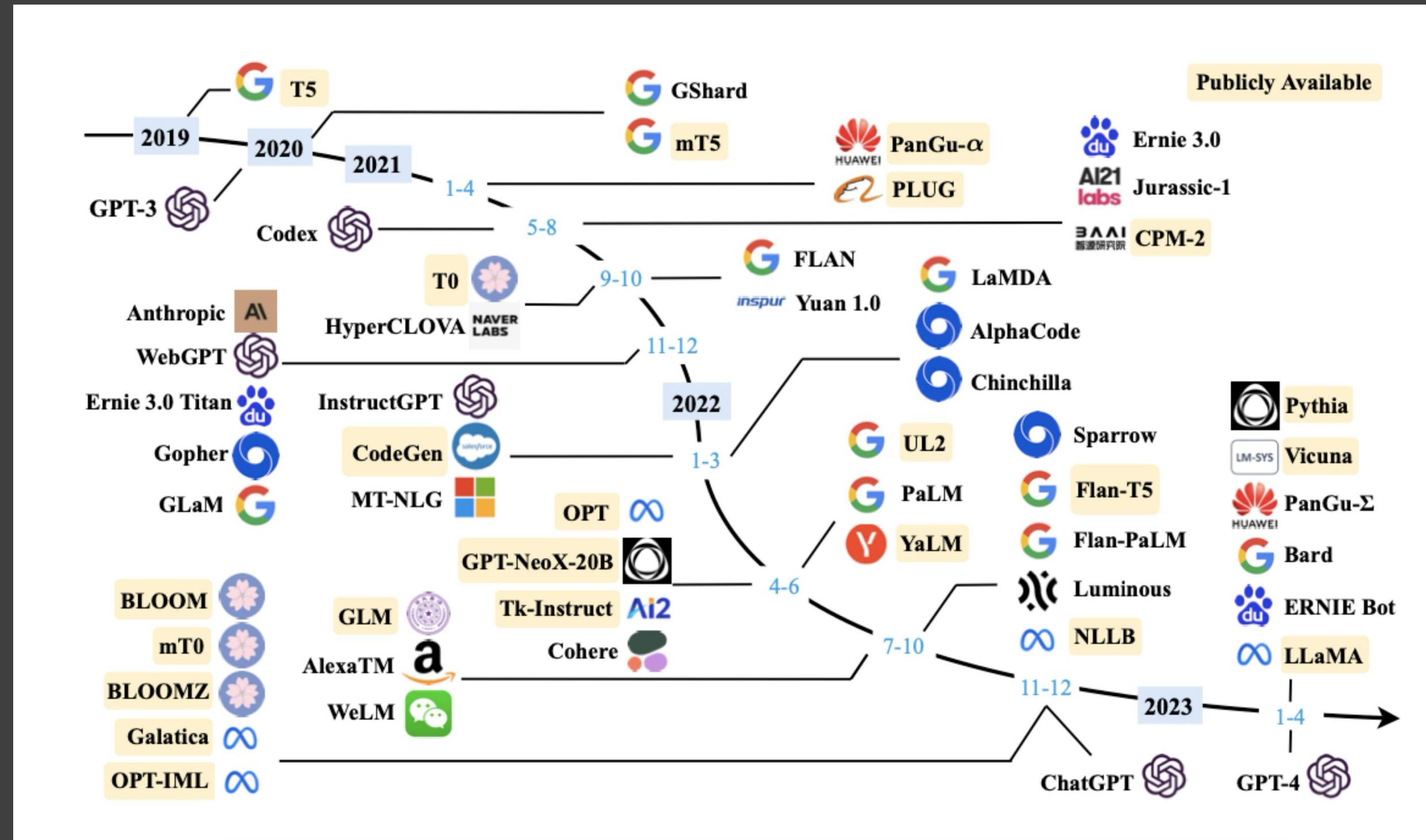
```
from transformers import GPT2Tokenizer, GPT2LMHeadModel
# 토크나이저 및 모델 초기화
tokenizer = GPT2Tokenizer.from_pretrained('./gpt2-finetuned_3')
model = GPT2LMHeadModel.from_pretrained('./gpt2-finetuned_3')
# 토크나이저에 패딩 토큰 설정
tokenizer.pad_token = tokenizer.eos_token
# 텍스트를 토크나이즈
inputs = tokenizer("이나린", return_tensors="pt", padding=True)
# 모델의 생성을 위한 입력
cached_lm_GPT2Tokenizer_128_nrrr_m
cached_lm_GPT2Tokenizer_128_nrrr_m
cached_lm_GPT2Tokenizer_128_ouyus
cached_lm_GPT2Tokenizer_128_ouyus
```

C:\Users\OYS\dev\pythonProject\venv\Scripts\python.exe C:\Users\OYS\dev\pythonProject\test2.py
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
안되겠소다
Process finished with exit code 0

2. 연구/개발 과정

2-3. 모델 선정

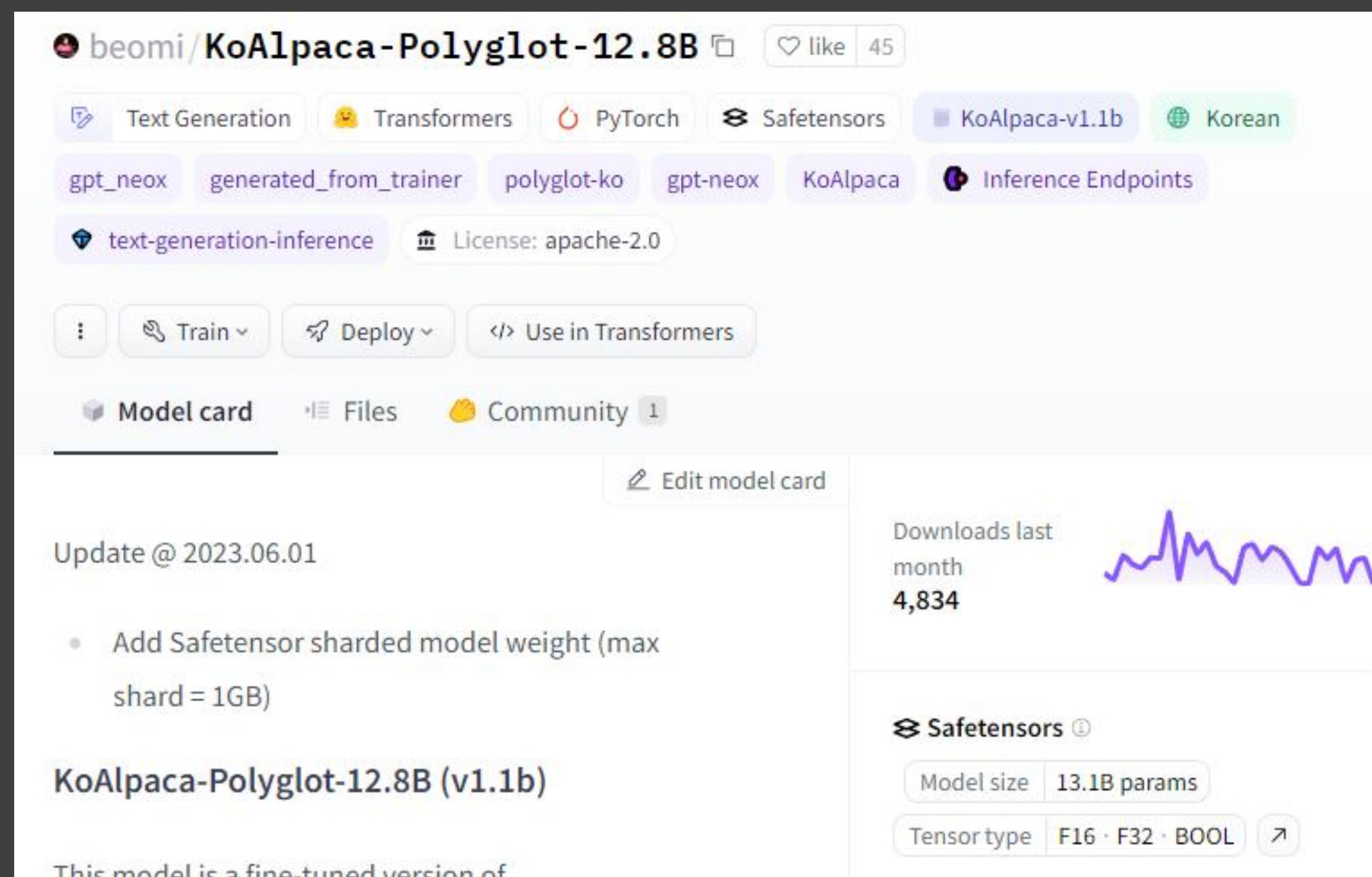
Step 3. LLaMA에 대한 깨달음



2. 연구/개발 과정

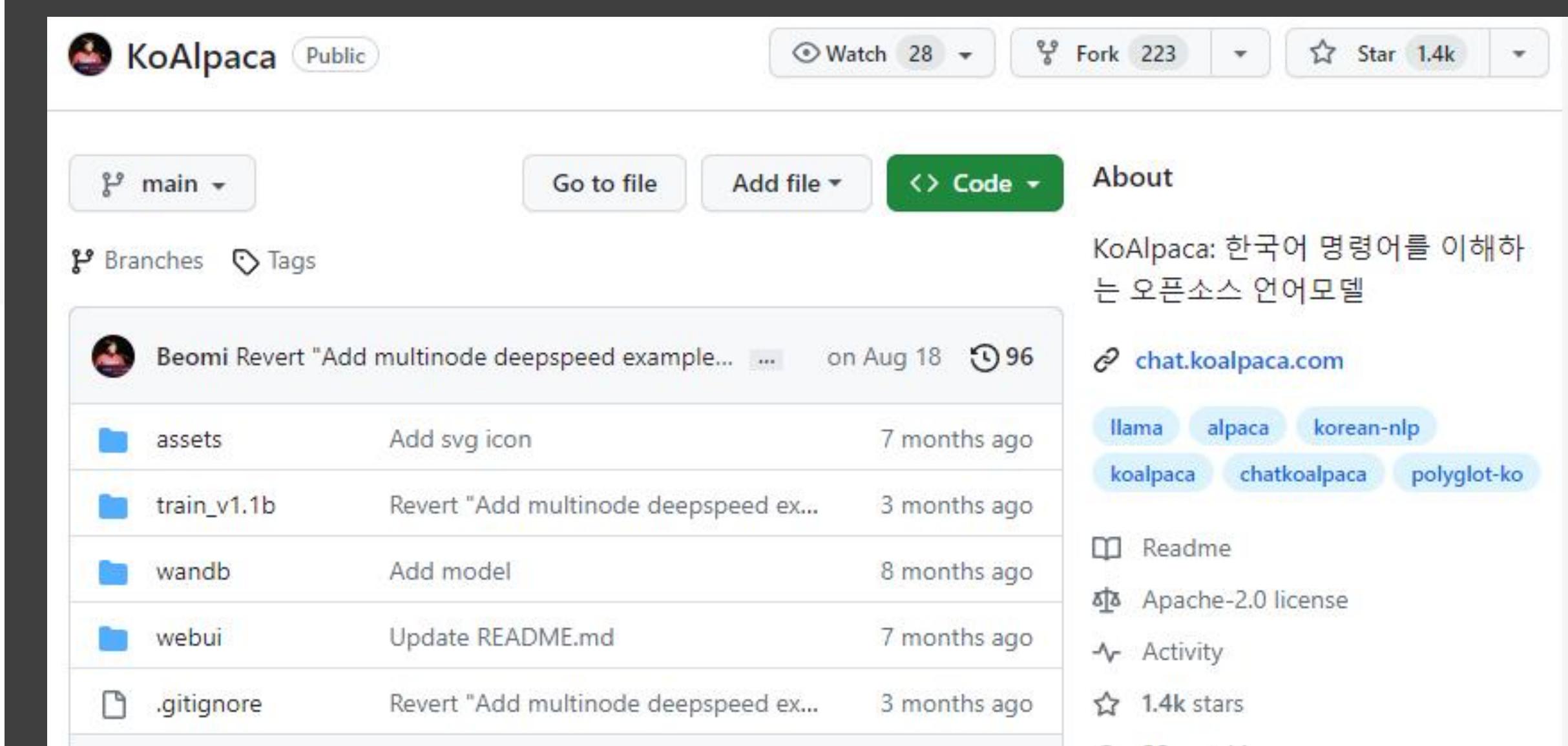
2-3. 모델 선정

Step 3. 자료조사 2

A screenshot of the Hugging Face Model Hub interface. The model card for 'KoAlpaca-Polyglot-12.8B (v1.1b)' is displayed. Key details include:

- Model ID: beomi/KoAlpaca-Polyglot-12.8B
- Downloads last month: 4,834
- Model size: 13.1B params
- Tensor type: F16 · F32 · BOOL

The card also lists recent updates and provides links for training, deployment, and use in Transformers.

A screenshot of a GitHub repository page for 'KoAlpaca'. The repository is public and has 28 stars, 223 forks, and 1.4k stars. The repository details are as follows:

- Language: Korean
- License: apache-2.0
- Code navigation: main, Go to file, Add file, Code
- Branches: main, Tags
- Recent commits:
 - Beomi Revert "Add multinode deepspeed example..." (Aug 18)
 - assets (7 months ago)
 - train_v1.1b (3 months ago)
 - wandb (8 months ago)
 - webui (7 months ago)
 - .gitignore (3 months ago)
- Tags: llama, alpaca, korean-nlp, koalpaca, chatkoalpaca, polyglot-ko
- Links: chat.koalpaca.com, Readme, Apache-2.0 license, Activity, 1.4k stars

2. 연구/개발 과정

2-4. 모델 학습

Step 0. 개발 환경

OS Windows Server 2022 Datacenter

CPU Intel(R) Zeon Gold(TM) 6252 @3.8GHz

GPU NVIDIA Quadro RTX 8000 48GB

RAM DDR4 ECC 3200mhz 128GB

개발 환경 Colab Pro + / ChatGPT plus / Pycharm EnterPrise Edition

2. 연구/개발 과정

2-4. 모델 학습

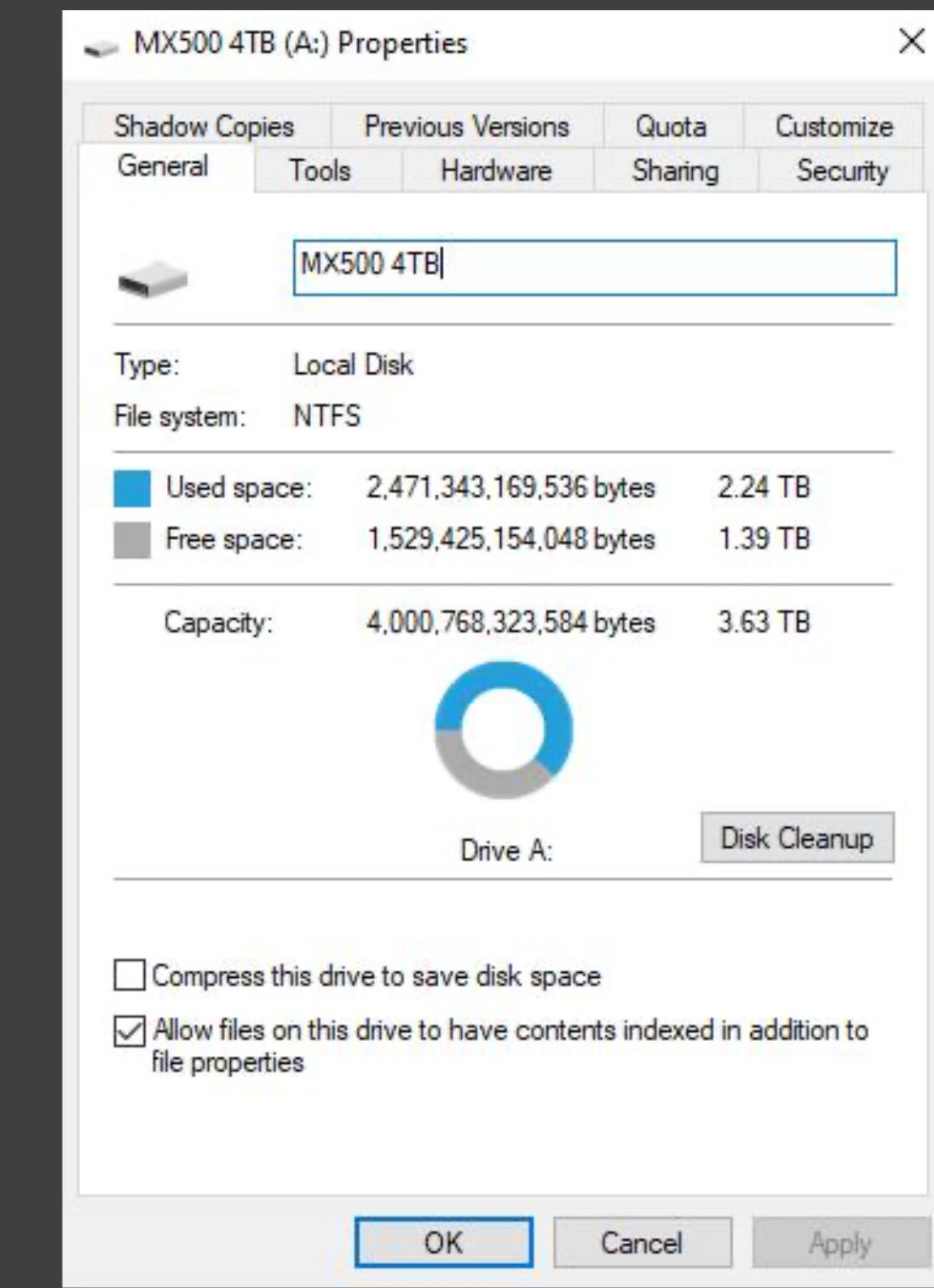
Step 1. TXT 데이터, 원본 모델 데이터 로드

```
# 파일에서 데이터 로드하는 함수
def load_data_from_file(file_path, tokenizer):
    with open(file_path, 'r', encoding='utf-8') as file:
        lines = file.readlines()

    questions = []
    answers = []
    for line in lines:
        parts = line.strip().split(' A: ')
        if len(parts) == 2:
            question, answer = parts
            question = question.replace('Q: ', '')
            questions.append(question)
            answers.append(answer)

    return questions, answers

# 파일로부터 데이터 로드 및 토큰화
file_path = '/content/drive/My Drive/haram_messages_pairs.txt' # Colab 환경에 맞게 경로 설정
tokenizer = AutoTokenizer.from_pretrained("EleutherAI/polyglot-ko-5.8b") # 모델 ID 변경
questions, answers = load_data_from_file(file_path, tokenizer)
```



2. 연구/개발 과정

2-4. 모델 학습

Step 2. 데이터셋 매팅



```
# 데이터셋 형태로 변환
data = {'text': [f"### 질문: {q}\n\n### 답변: {a}" for q, a in zip(questions, answers)]}
dataset = Dataset.from_dict({"text": data['text']})
tokenized_data = dataset.map(lambda samples: tokenizer(samples["text"], padding='max_length',
truncation=True, max_length=512), batched=True)

# 모델 설정
model_id = "EleutherAI/polyglot-ko-5.8b" # 모델 ID 변경
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(model_id, quantization_config=bnb_config, device_map=
{"":0})
model = prepare_model_for_kbit_training(model)
```

2. 연구/개발 과정

2-4. 모델 학습

Step 3. 모델 FineTuning



```
# 파인튜닝 설정
training_args = TrainingArguments(output_dir="/content/outputs",
per_device_train_batch_size=2, max_steps=500,
learning_rate=2e-4, fp16=True,
logging_steps=10,
optim="paged_adamw_8bit" )
data_collator = DataCollatorForLanguageModeling(tokenizer, mlm=False)

# 파인튜닝 실행
trainer = Trainer(model=model,
args=training_args,
train_dataset=tokenized_data,
data_collator=data_collator )
trainer.train()
```

2. 연구/개발 과정

2-4. 모델 학습

Step 4. N차의 최적화

ChatGPT	파인튜닝 team-lucid/llama-ko-1b	ChatGPT	애플 캠: CUDA 오류
모델 파인튜닝 및 테스트	모델 세부 조정 코드	Install Accelerate for Transformer	Device Mismatch Error Fix
파인튜닝 모델 설정	메모리 요구 사항 KoAlpaca-Polyglot	설치 및 디버깅 가이드	GPT-2 Fine-Tuning Troubleshooting
New chat	모델 메모리 관리 개선	파인튜닝 team-lucid/llama-ko-1b	Fix GPT-2 Training Script
GPU 가속 모델 실행	파인튜닝 KoAlpaca DM	모델 세부 조정 코드	챗봇 말투 학습 방법
저장 및 모델 로드	파일 다운로드 방법	메모리 요구 사항 KoAlpaca-Polyglot	Tokenizer Mismatch Warning Fix
GPU 2개의 RAM 합산	파일 읽기 방식 변경	모델 메모리 관리 개선	학습 말투 GPT-3
모델 finetuning 문제 해결	수정된 메시지 쌍	파인튜닝 KoAlpaca DM	훈련 데이터만 사용
Get Current File Directory	제거 빈 줄 코드	파일 다운로드 방법	한국어 GPT-2 모델 사용
Previous 7 Days	메시지 매칭 로직 개선	파일 읽기 방식 변경	학습 루프 및 모델 저장
모델 훈련 및 테스트	CUDA Setup Error Solutions	수정된 메시지 쌍	학습된 GPT 모델 생성
Model Loading OSError Troubleshooting	윈도우에서 CUDA 프로그래밍	제거 빈 줄 코드	인스타 DM 메시지 파인튜닝
해결 방법: 디바이스 일치	CUDA Setup Error Troubleshooting	메시지 매칭 로직 개선	Troubleshooting ultralytics install
모델 파인튜닝 및 테스트	해결 방법: CUDA 오류	CUDA Setup Error Solutions	웹캠 QR 코드 스캔
Install Accelerate for Transformer	Device Mismatch Error Fix	윈도우에서 CUDA 프로그래밍	인스타 DM 파인튜닝
설치 및 디버깅 가이드	GPT-2 Fine-Tuning Troubleshooting	CUDA Setup Error Troubleshooting	인스타 DM 말투 파인튜닝 가능
	Fix GPT-2 Training Script	해결 방법: CUDA 오류	만드는 챗봇법 NLP

우리 지피티 잘한다!

2. 연구/개발 과정

2-4. 모델 학습

Step 5. 최종 모델 완성

하람봇



[500/500 04:22, Epoch 0/1]

Step	Training Loss
10	3.967900
20	2.134600
30	1.902800

나린봇



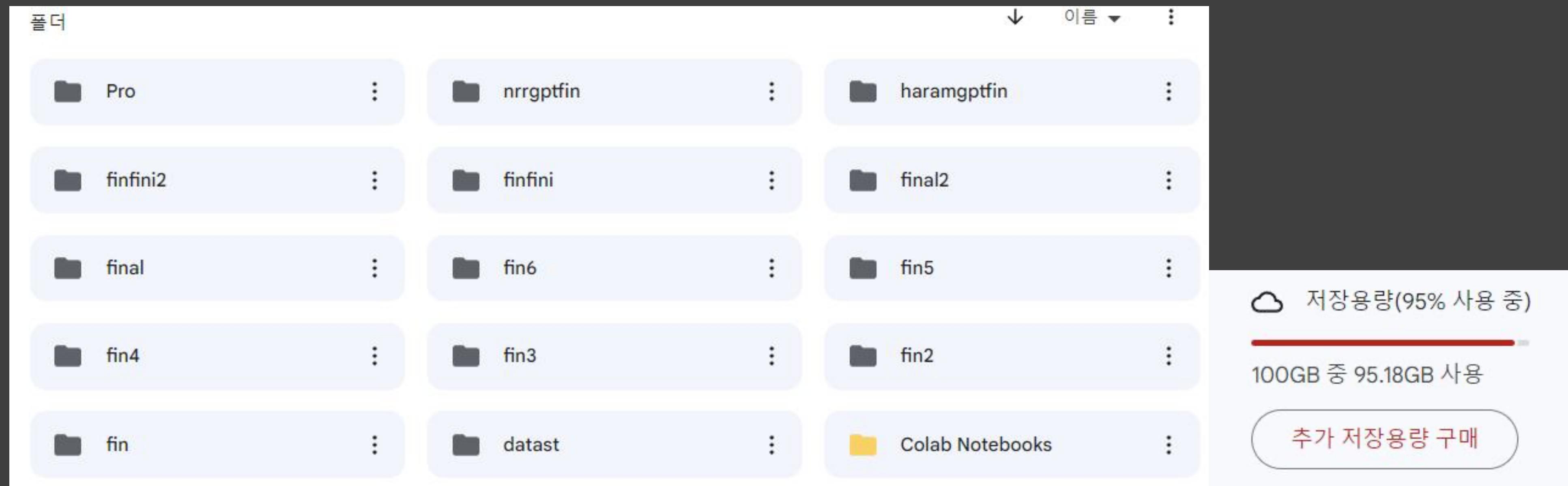
[500/500 04:30, Epoch 0/1]

Step	Training Loss
10	3.959900
20	2.194000
30	2.070700

2. 연구/개발 과정

2-4. 모델 학습

Step 5. 최종 모델 완성



2. 연구/개발 과정

2-5. 모델 불러오기 및 테스트

Step 1. 하람봇 테스트

```
[ ] gen('안녕')
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
### 질문: 안녕
### 답변: 안녕하세요 ㅋㅋㅋㅇ ㅋㅇㅋㅇ♦

[ ] gen('하람아')
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
### 질문: 하람아
### 답변: 어어 맞아 어어맞아 ㄱㅋㄱㅋㄱㅋㄱ♦

[ ] gen('뭐해')
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
### 질문: 뭐해
### 답변: 몰라느 느 내가 뭘 하지?### 답변: 잘 자고 잘 일어남 ♦

[ ] gen('심심해')
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
### 질문: 심심해
### 답변: ㅎㅎㄷㄷㄷㄷㅋㅋ♦

[ ] gen('어쩔티비')
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
### 질문: 어쩔티비
### 답변: 어쩔티비### 답변: 어쩔티비### 답변: 어쩔티비### 답변: 어쩔티비### 답변: 어쩔티비

[ ] gen('부럽다 막북')
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
### 질문: 부럽다 막북
### 답변: 부럽다 너도막북하셈### 답변: 부럽다 나도 막북하고 싶네너도막북하셈#
```

2. 연구/개발 과정

2-5. 모델 불러오기 및 테스트

Step 2. 나린봇 테스트

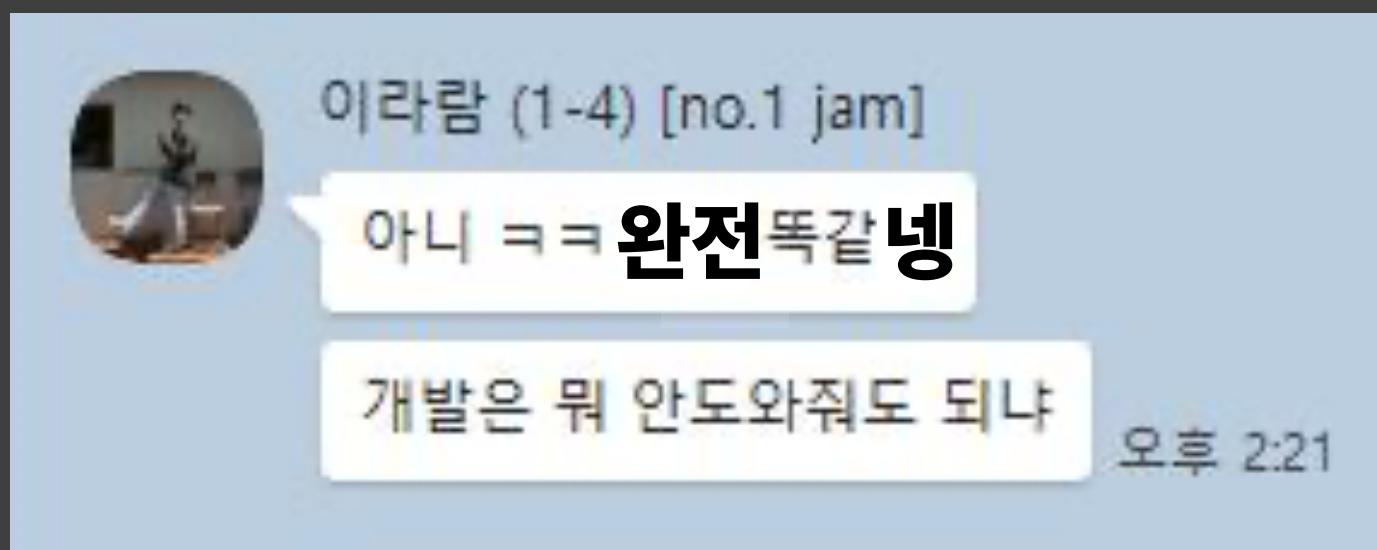
```
[ ] gen('안녕')
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
### 질문: 안녕
### 답변: 저도 이제 잘게요 자고나서 톡해요ㅠㅠㅠㅠㅠ◆

[ ] gen('나린아')
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
### 질문: 나린아
### 답변: 어젠 내가 힘들어서 너무 화난채로 보냈네 너는 괜찮진? 나는 그와중에도 계속 얘기해주고 잘

[ ] gen('뭐해에')
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
### 질문: 뭐해에
### 답변: 그냥 노래듣고 있었엉ㅋㅋㅇㅋㅋㅇㅋ◆

[ ] gen('심심해')
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
### 질문: 심심해
### 답변: 집에서 놀잖아 집에있으면서 무슨 심심ㅅㅇ이야 어디서 놀아 친구들이랑 놀아 나누나
```

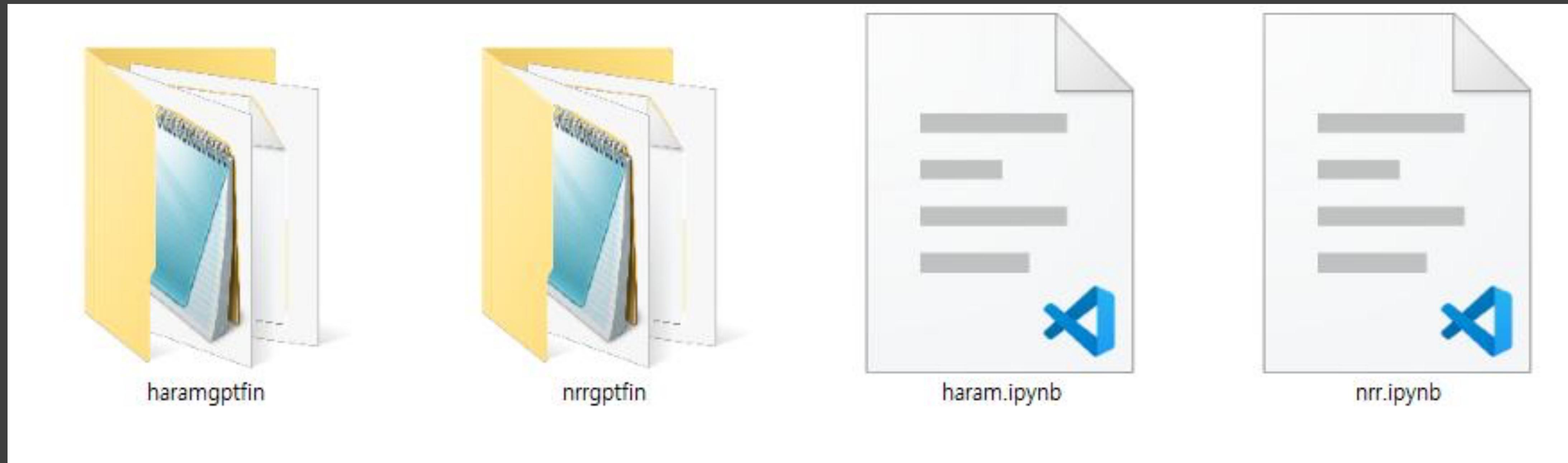
완전
이나린
그자체



2. 연구/개발 과정

2-5. 모델 불러오기 및 테스트

Step 3. 모델 저장



2. 연구/개발 과정

2-6. 프론트엔드 구현 및 연동

Step 1. Next.JS 14 프로젝트 생성 및 필요 라이브러리 세팅

필요 라이브러리 선정

스타일링 라이브러리 styled-components

Data-Fetching Axios

비동기 통신 / 캐싱 Tanstack Query

전역 상태 관리 Recoil

```
yarn create next-app@latest sunringpt
```

NEXT.js 14 프로젝트 생성

```
yarn add styled-components @types/  
styled-components recoil axios  
@tanstack/tanstack-query
```

필요 라이브러리 설치

2. 연구/개발 과정

2-6. 프론트엔드 구현 및 연동

Step 2. 컴포넌트, 타입 설계

타입, 인터페이스 정의

Segement

나린봇	하람봇
나린봇	하람봇

SendButton

Type Pink	
Type Blue	

Chat

나린봇	쿠쿠루삥뽕 개킹받죠 ㅋㅋ 아무것도 못하죠 ㅋㅋ	Type Pink-bot
@ksyisgodteacher	쿠쿠루삥뽕 개킹받죠 ㅋㅋ 아무것도 못하죠 ㅋㅋ	Type Pink
나린봇	쿠쿠루삥뽕 개킹받죠 ㅋㅋ 아무것도 못하죠 ㅋㅋ	Type Blue-bot
@ksyisgodteacher	쿠쿠루삥뽕 개킹받죠 ㅋㅋ 아무것도 못하죠 ㅋㅋ	Type Blue

ChatRecent Component

현재 스레드 · 오랑우탄 좋아해?	Type Pink-selected
1시간 전 · 오랑우탄 좋아해?	Type Pink-normal
현재 스레드 · 오랑우탄 좋아해?	Type Blue-selected
1시간 전 · 오랑우tan 좋아해?	Type Blue-normal

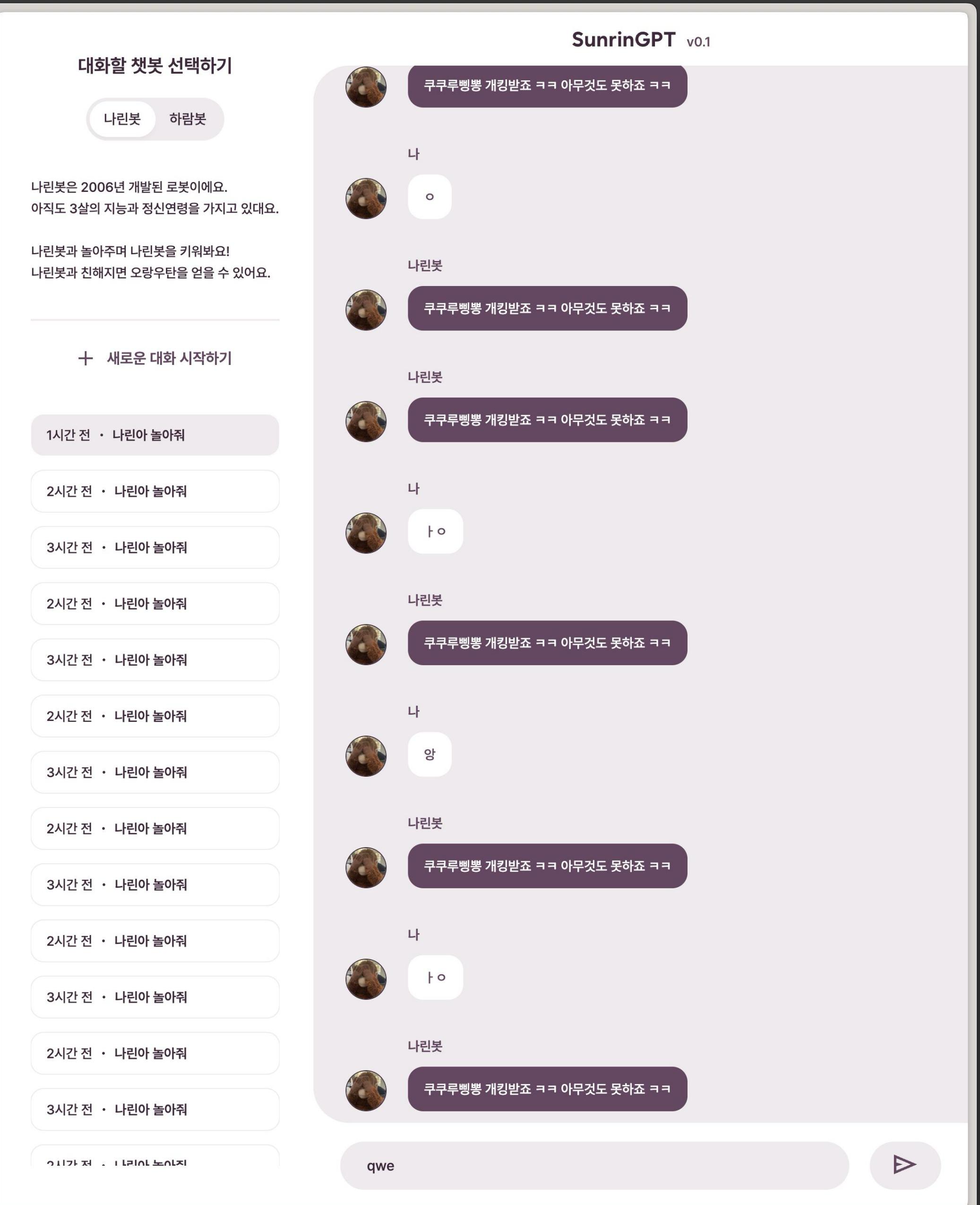
```
● ● ●  
interface ChatProps {  
  sender: string; // 메시지 전송자 이름  
  message: string; // 메시지 내용  
  isMine?: boolean; // 본인이 전송한 채팅인지  
}  
  
interface ChatRecentProps {  
  time: string; // 마지막 전송 시간  
  active?: boolean; // 현재 대화 여부  
  name: string; // 대역 이름  
}  
  
interface Chat {  
  isMine: boolean; // 내가 전송한 채팅인지  
  message: string; // 채팅 내용  
}  
  
type Theme = "narin" | "haram" // 모드 타입
```

2. 연구/개발 과정

2-6. 프론트엔드 구현 및 연동

Step 3. 컴포넌트 배치 및 UI 테스트

채팅 UI 리스트 배치



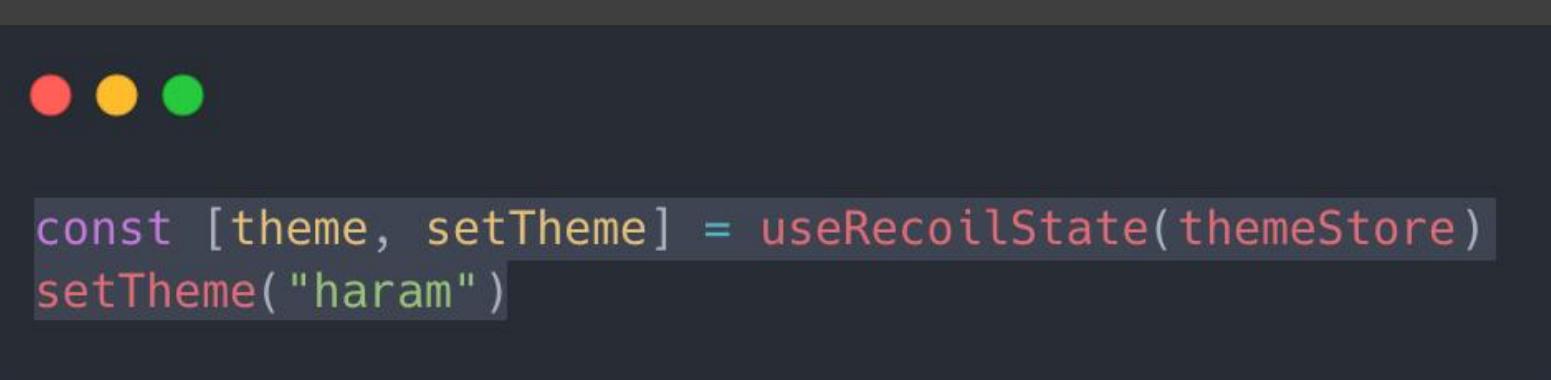
2. 연구/개발 과정

2-6. 프론트엔드 구현 및 연동

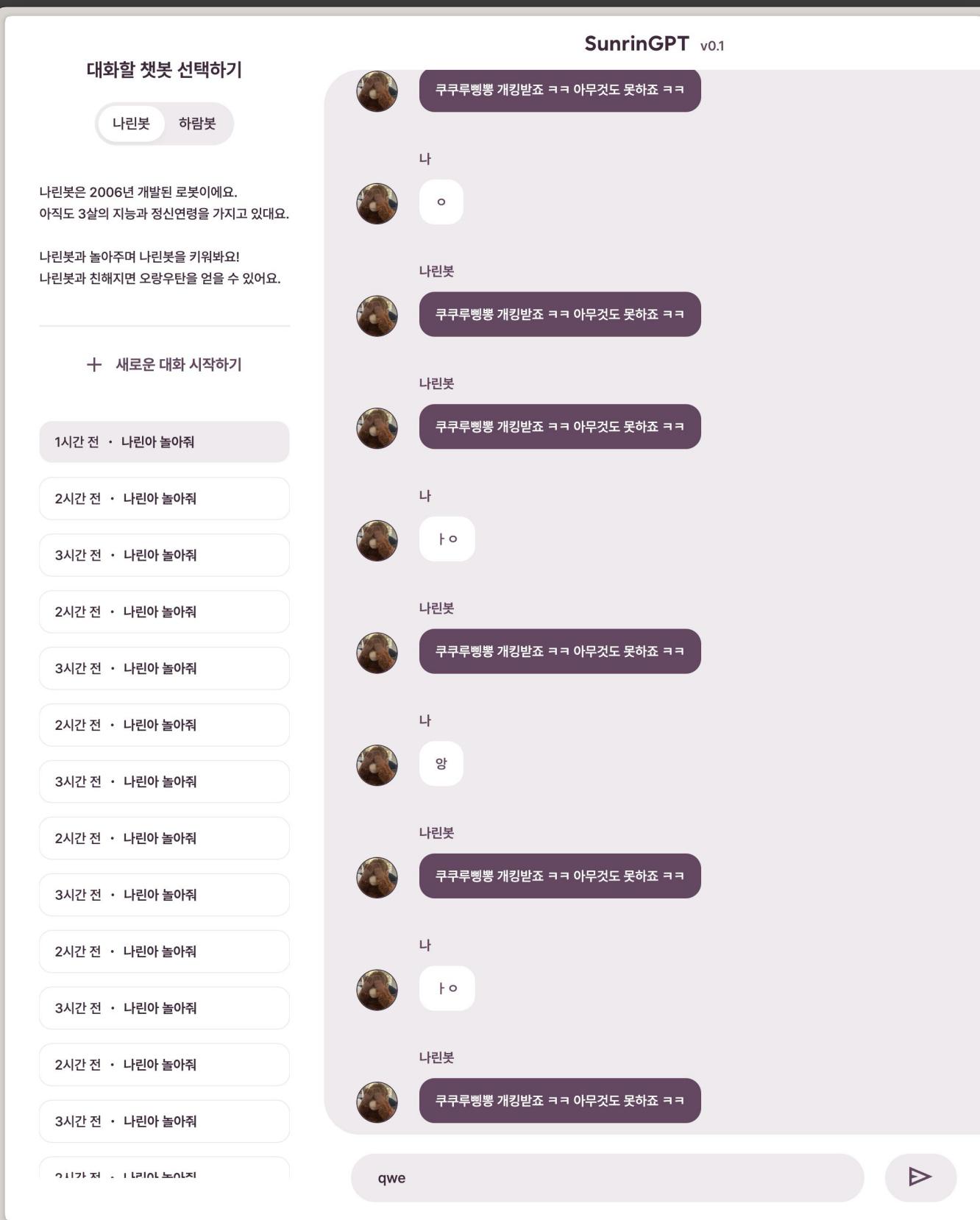
Step 4. 하람봇,나린봇 모드 토글



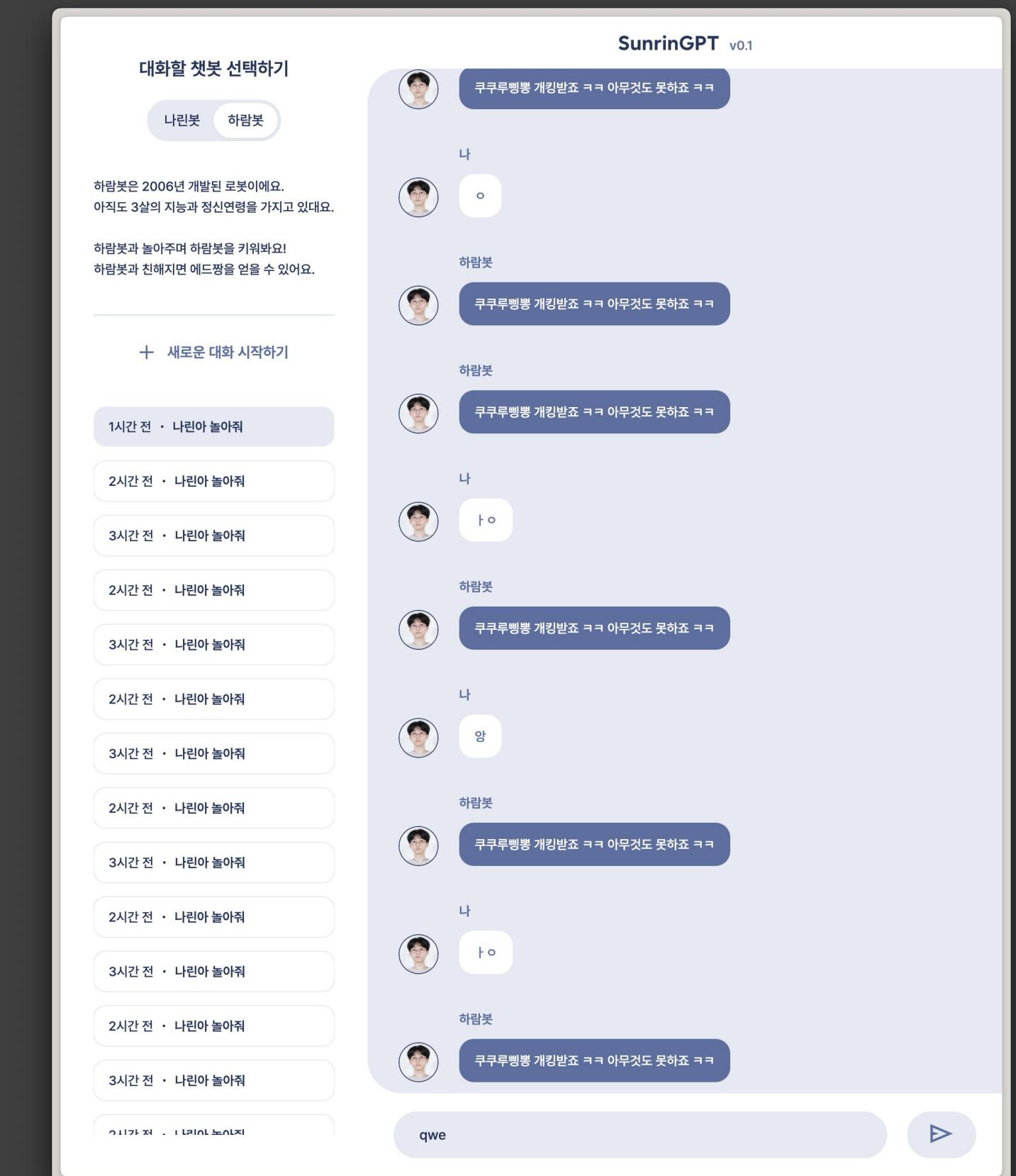
토글 전역 상태 atom으로 선언



useRecoilState로 토글



나린봇



하람봇

2. 연구/개발 과정

2-6. 프론트엔드 구현 및 연동

Step 5. 외부 서버와의 통신 연결



```
import axios from 'axios';

const client = axios.create({
  baseURL: 'https://api.sunringpt.com' // 대충 주소
});

export default client;
```

통신 인스턴스 생성(axios)

```
import client from "@/app/lib/api/client";

export const getChatList = async (token: string) => {
  const response = await client.get('/chat/list', {
    headers: {
      Authorization: `Bearer ${token}`
    }
  });
  return response.data;
}

export const getChat = async (token: string, id: string) => {
  const response = await client.get(`/chat/${id}`, {
    headers: {
      Authorization: `Bearer ${token}`
    }
  });
  return response.data;
}

export const postChat = async (token: string, data: any) => {
  const response = await client.post('/chat', data, {
    headers: {
      Authorization: `Bearer ${token}`
    }
  });
  return response.data;
}

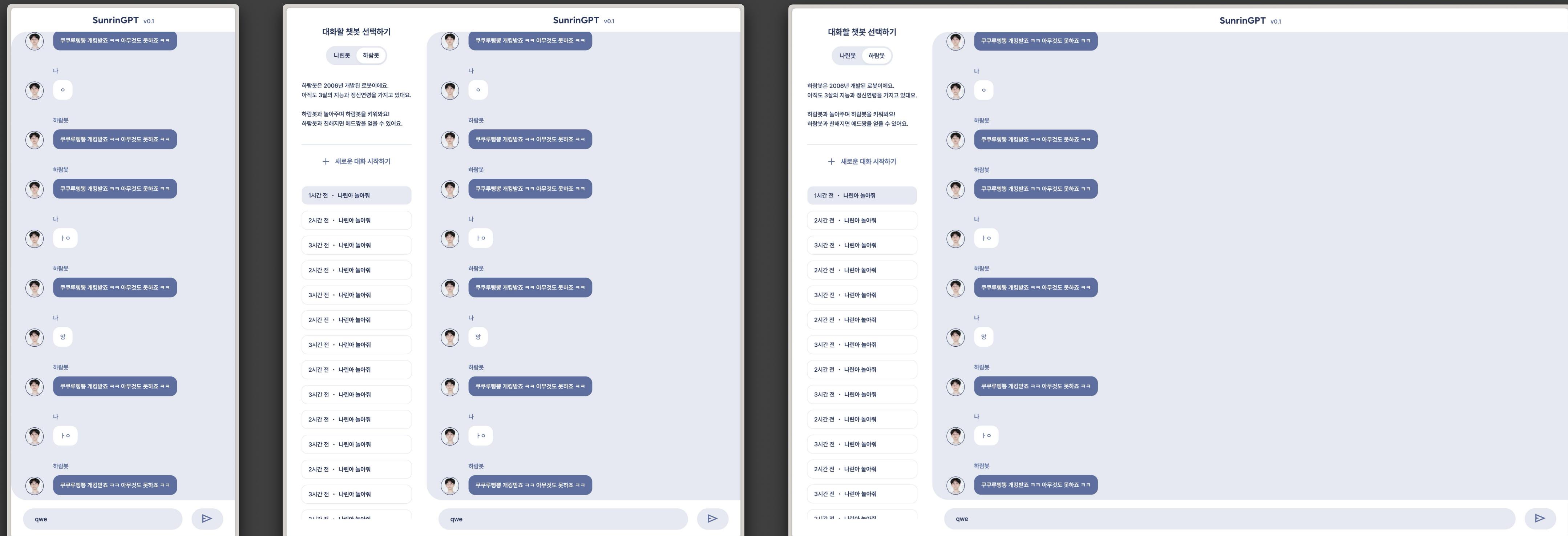
export const deleteChat = async (token: string, id: string) => {
  const response = await client.delete(`/chat/${id}`, {
    headers: {
      Authorization: `Bearer ${token}`
    }
  });
  return response.data;
}
```

인스턴스를 이용한 통신 메서드 정의, export

2. 연구/개발 과정

2-6. 프론트엔드 구현 및 연동

Step 6. 반응형 구현(media query)



mobile

tablet

desktop

4. 연구 결과



하람 모델



나린 모델



나린 코드



하람 코드

감사합니다