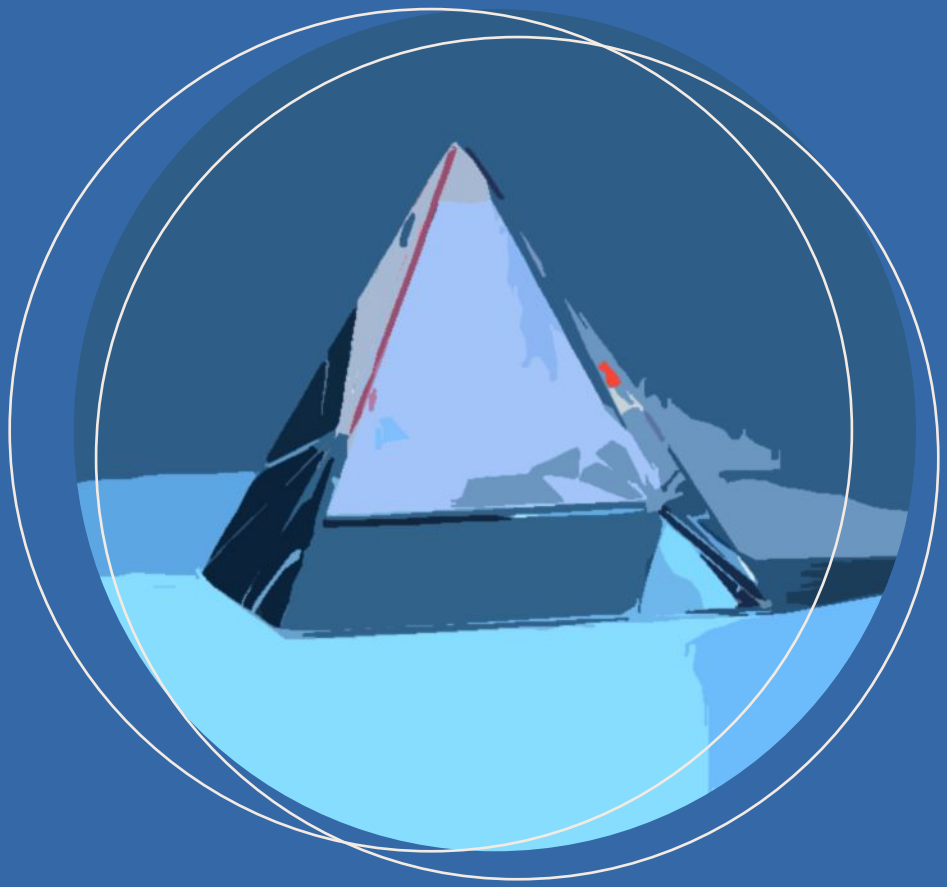


CodeStates AI 12기 + WhoCaresKorea

김윤희 박석환 이대웅

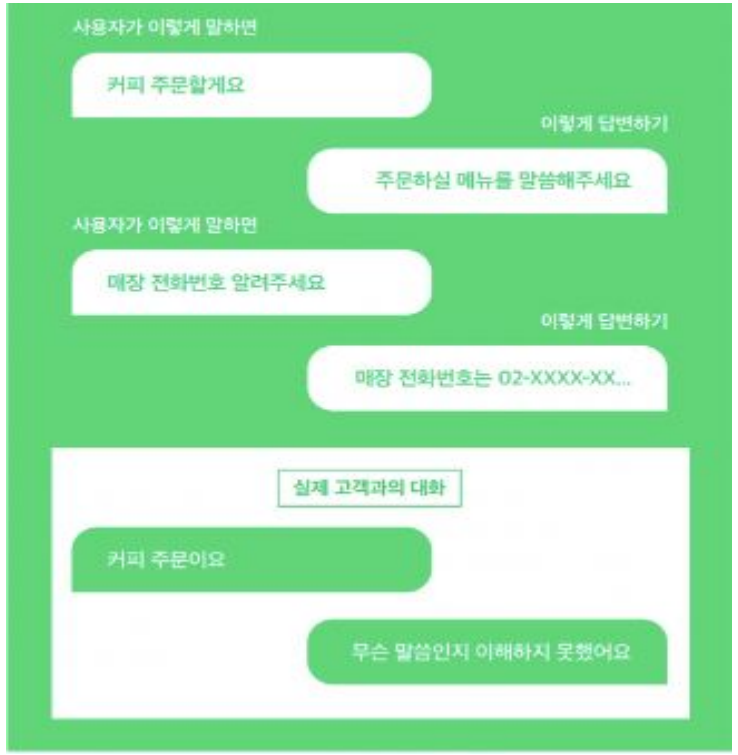
목차 Table of Contents

Part 1
챗봇이란

1 챗봇이란

비인공지능형 vs. 인공지능형

AI나리오형 (객관식형, 트리형)



- 목적에 맞춰 탈출구가 없도록 설계
- 로직 구조가 하나하나 짜여져 있음
- 사용자가 저장된 값과 다른 입력값을 인풋할 경우 알아듣지 못함

비인공지능형

VS.

인공지능형

주제형, 목적형, 결합형

- 사용자가 저장된 값과 다른 입력값을 인풋하여도 의미 캐치 가능

1 챗봇이란

인공지능형 - 주제형, 목적형

주제형

1. 대화가 연속적이며, 1회성 답변을 통한 대화 완결
2. 고객이 궁금한 주제 질문, 챗봇이 질문에 대해 답
3. 목적 없이 단순 단발성 대화도 가능

목적형

1. 고객이 특정 결과를 얻기 위해 필요한 데이터를 대화를 통해 추출
2. 챗봇이 목적을 위한 질문, 고객이 질문에 대답
3. 정보를 수집하고 저장하는 과정이 필수

Part 2
서비스 기획

챗봇 서비스 기획

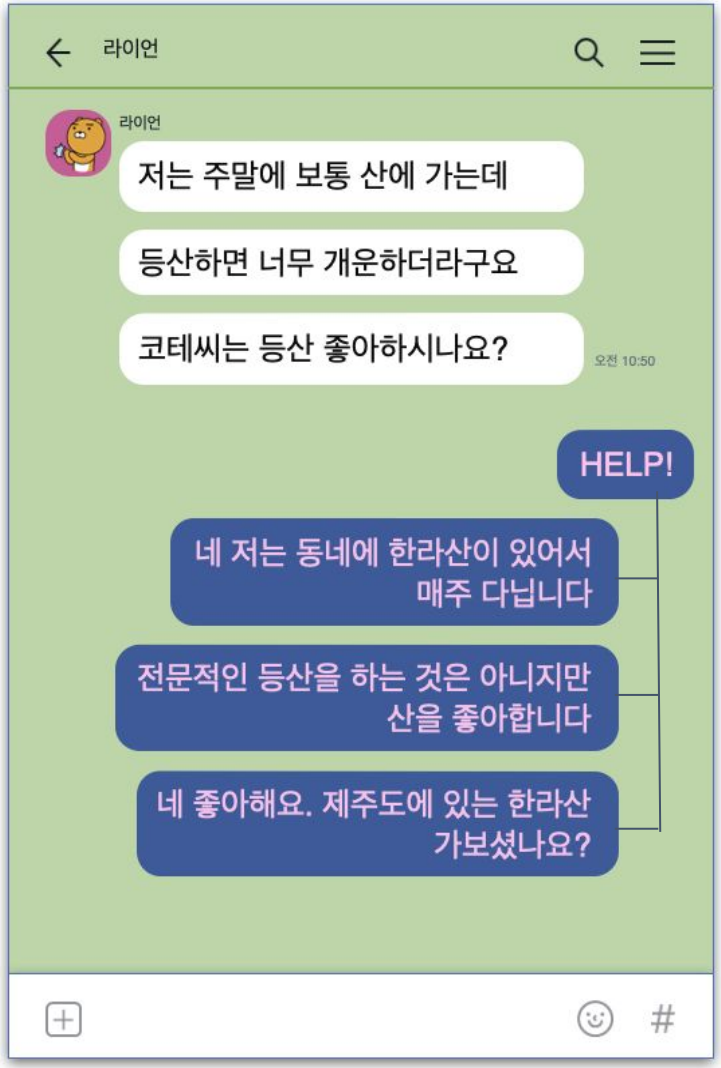
서비스 구조

1. 해야 할 대답을 생각하지 못할 때에 도움 표시 클릭

2. 도움 표시 클릭 시 상대 대화자의 마지막 대화뭉치 추출

3. 마지막 대화를 기준으로 3가지 스크립트 추천

INPUT

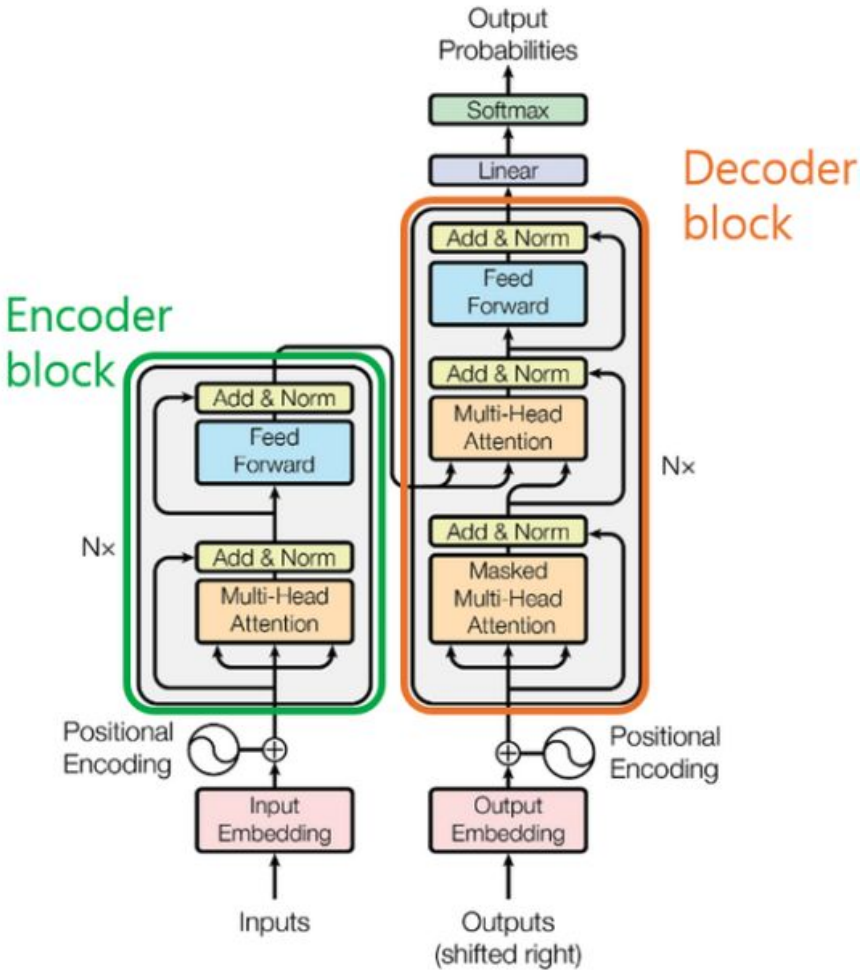


Part 3
모델 선정

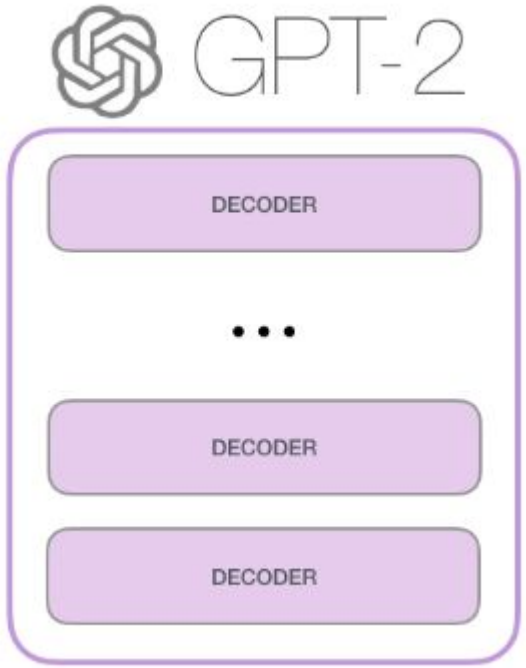
3 모델 선정

Chatbot base model - BERT vs GPT2

1 BERT



2 GPT 2



3 모델 선정

Chatbot base model - BERT vs GPT2

항목	GPT	BERT
Transformer block	Decoder block	Encoder block
Attention 방향	Uni-directional	Bi-directional
문장 생성 활용 여부	문장 생성 가능	직접 생성 불가능

GPT2

- Token 생성 후 입력 시퀀스에 더해지는 방식으로 작동되어 직접 문장 생성 가능
- one shot/few shot learning 가능하여, 데이터가 많지 않은 프로젝트에 적절

Bert

- 양 방향으로부터 Context 활용이 가능해 상대적으로 나은 성능 제공
- 데이터가 충분한 경우에 주로 사용

학습할 데이터 부족한 상황에서 자동으로 문장을 생성해주는

GPT2 Model 이 적합하다고 판단

Part 4
모델 구조

4 모델 구조

Define Token

Q_TKN : 질문 데이터 토큰

A_TKN : 답변 데이터 토큰

BOS (Beginning of Sentence) : 문장 시작 토큰

EOS (End of Sentence) : 문장 끝맺음 토큰

PAD : 패딩 토큰

MASK : 마스킹 토큰

SENT : 분류(라벨링) 토큰

```
## 문장 토큰, 마스킹 설정

Q_TKN = "<usr>"
A_TKN = "<sys>"
BOS = "</s>"
EOS = "</s>"
PAD = "<pad>"
MASK = "<unused0>"
# SENT = '<unused1>'
```

4 모델 구조

Data Preprocessing

특수문자 모두 제거

why?
특수문자에 따른 문장의 유형을 구분하지 않고 문장을 생성

```
def __getitem__(self, idx):  # 로드한
    turn = self._data.iloc[idx]
    q = turn["Q"]  # 질문을 가져온다.
    q = re.sub(r"([?.!,])", r" ", q)

    a = turn["A"]  # 답변을 가져온다.
    a = re.sub(r"([?.!,])", r" ", a)
```

	Q	A	label
0	가출할까?	무모한 결정을 내리지 마세요.	0
1	가출해도 갈 데가 없어	선생님이나 기관에 연락해보세요.	0
2	간만에 떨리니까 좋더라	떨리는 감정은 그 자체로 소중해요.	0
3	간만에 쇼핑 중	득템했길 바라요.	0
4	간만에 휴식 중	휴식도 필요하죠.	0

	Q	A	label
0	가출할까	무모한 결정을 내리지 마세요	0
1	가출해도 갈 데가 없어	선생님이나 기관에 연락해보세요	0
2	간만에 떨리니까 좋더라	떨리는 감정은 그 자체로 소중해요	0
3	간만에 쇼핑 중	득템했길 바라요	0
4	간만에 휴식 중	휴식도 필요하죠	0

4 모델 구조

Tokenization

```
koGPT2_TOKENIZER = PreTrainedTokenizerFast.from_pretrained(
    "skt/kogpt2-base-v2",
    bos_token=BOS, eos_token=EOS, unk_token="<unk>", pad_token=PAD, mask_token=MASK,)
```

```
Q        정치적 음모 관련 내용이었어요
A            킬러 내용인가 보네요
label                    0
```

```
q_toked = self.tokenizer.tokenize(self.q_token + q)
# q_toked = self.tokenizer.tokenize(self.q_token + q + self.sent_token + sent)
q_len = len(q_toked)

a_toked = self.tokenizer.tokenize(self.a_token + a + self.eos)
a_len = len(a_toked)
```

Tokenization
(MeCab + SentencePiece)

q_toked

```
['<usr>', '_정치적', '_음모', '_관련', '_내용', '이었', '어', '요', '<unused1>']
```

a_toked

```
['<sys>', '_킬', '러', '_내용', '인가', '_보', '네', '요', '</s>']
```

4 모델 구조

Maximum Length

토큰 길이 조절

```
if q_len > self.max_len:
    a_len = self.max_len - q_len        #답변의 길이를
    if a_len <= 0:       #질문의 길이가 너무 길어 질문
        q_toked = q_toked[-(int(self.max_len / 2)) :]
        q_len = len(q_toked)
        a_len = self.max_len - q_len              #답
    a_toked = a_toked[:a_len]
    a_len = len(a_toked)

#질문의 길이 + 답변의 길이가 최대길이보다 크면
if q_len + a_len > self.max_len:
    a_len = self.max_len - q_len        #답변의 길이를
    if a_len <= 0:       #질문의 길이가 너무 길어 질문
        q_toked = q_toked[-(int(self.max_len / 2)) :]
        q_len = len(q_toked)
        a_len = self.max_len - q_len              #답
    a_toked = a_toked[:a_len]
    a_len = len(a_toked)
```

만약, 질문 토큰 길이 + 답변 토큰 길이가 설정 길이보다 크다면

질문 토큰의 길이를 줄여 토큰 구성

Q_toked 길이 : 9

```
['<usr>', '_정치적', '_음모', '_관련',
'_내용', '이었', '어', '요', '<unused1>']
```

A_toked 길이 : 9

```
['<sys>', '_킬', '러', '_내용',
'인가', '_보', '네', '요', '</s>']
```

전체 길이 : 18

4 모델 구조

Labeling. Masking. Padding

```
# 답변 labels = [mask, mask, ...., mask, ..., <bos>,..답변.. <eos>, <pad>....]
labels = [self.mask,] * q_len + a_toked[1:]
```

```
['<usr>', '_응', '_다', '들', '_합', '이', '_잘', '맞', '아', '<unused1>']
['<sys>', '_맞아', '_척', '척', '_다', '들', '</s>']
```

labels

```
['<unused0>', '<unused0>', '<unused0>', '<unused0>', '<unused0>'
, '<unused0>', '<unused0>', '<unused0>', '<unused0>', '<unused0>'
'_맞아', '_척', '척', '_다', '들', '</s>']
```

4 모델 구조

Labeling. Masking. Padding

```
# mask = 질문길이 0 + 답변길이 1 + 나머지 0
mask = [0] * q_len + [1] * a_len + [0] * (self.max_len - q_len - a_len)
```

```
['<unused0>', '<unused0>', '<unused0>', '<unused0>', '<unused0>',
 '<unused0>', '<unused0>', '<unused0>', '<unused0>', '<unused0>']
['<unused0>', '<unused0>', '<unused0>', '<unused0>', '_그럼', '_엄마', '라면',
'_고추', '장', '_닭', '_날개', '_조', '림에', '_곁들', '일', '래', '_', '</s>']
```


mask

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

4 모델 구조

Label Ids & Token Ids

label_token_index

```
# 답변 labels을 index 로 만든다.
labels_ids = self.tokenizer.convert_tokens_to_ids(labels)
# 최대길이만큼 PADDING
while len(labels_ids) < self.max_len:
    labels_ids += [self.tokenizer.pad_token_id]
```

mask

```
[9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
```

labels

```
27532, 13444, 9468, 10393, 6872, 7098, 8084, 739, 1,
```

padding

```
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

chat_token_ids

```
# 질문 + 답변을 index 로 만든다.
token_ids = self.tokenizer.convert_tokens_to_ids(q_toked + a_toked)
# 최대길이만큼 PADDING
while len(token_ids) < self.max_len:
    token_ids += [self.tokenizer.pad_token_id]
```

Q

```
[2, 9873, 468, 11316, 7692, 9023, 9163, 15531, 9347, 12356, 9078, 7208, 9329, 8084, 739, 10]
```

A

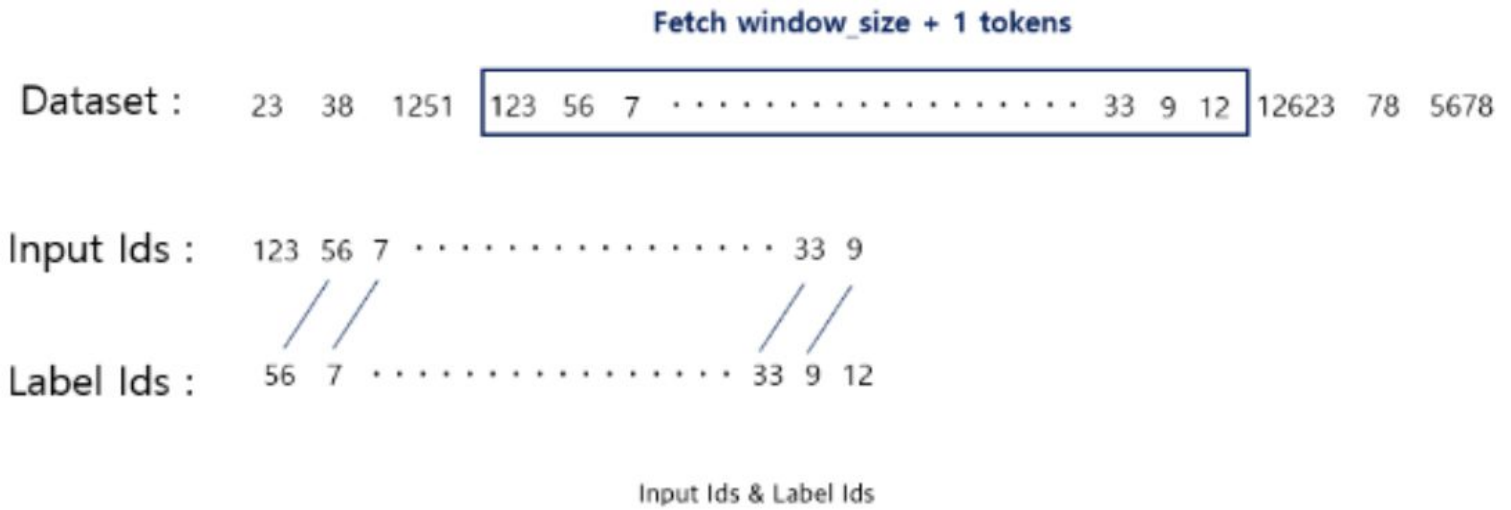
```
[4, 27532, 13444, 9468, 10393, 6872, 7098, 8084, 739, 1]
```

padding

```
[3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

4 모델 구조

Label Ids & Token Ids



Input Ids & Label Ids

chat_token_ids A : `[4, 27532, 13444, 9468, 10393, 6872, 7098, 8084, 739, 1]`

label_token_ids labels : `27532, 13444, 9468, 10393, 6872, 7098, 8084, 739, 1,`

4 모델 구조

Model

1. 사용 모델 : skt/kogpt2

```
GPT2LMHeadModel.from_pretrained('skt/kogpt2-base-v2')
```

2. 사용 파라메타

```
batch_size=32
learning_rate = 3e-5
criterion = torch.nn.CrossEntropyLoss(reduction="none")
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
epoch = 100
Sneg = -1e18
```

- 손실함수 : CrossEntropyLoss(다중분류함수 사용)
- 옵티마이저 : Adam 사용

Part 5
서비스 구조

5 서비스 구조

Repository Architecture

ChatBot12

- Data — 적재되는 채팅 DB 중 날짜 기준 갱신된 채팅 내역 Data (질문-답변)
- Model — 매일 갱신된 채팅 내역을 바탕으로 Model 재학습
- batch.py — 서버 스케줄러를 통해 구동되는 Data, Model Batch 관련 실행 파일
- kogpt2_model.py — Batch 파일을 통해 구동되는 Model 재학습 관련 실행 파일
- main.py — 채팅 서버로부터 Request 요청 시 학습 Model을 통해 Response 생성 실행 파일
- gpt2_modeling.ipynb — Model 학습 및 생성 Base Code

Part 6
발전 가능성 및 추후 상용화

6 발전 가능성

Development Roadmap

Option 1

labeling 칼럼을 사용하여 감정 분석 모델을 추가하여 공감 능력 강화

Option 2

labeling 칼럼을 사용하여 관심사 분류를 강화하여 대화 주제에 더욱 관련 있는 스크립트를 생성 및 추천하도록 개선

Option 3

나쁜말 데이터셋을 통해 라벨링 전 전처리에서 문장 스크립트 정제

Option 4

실제 AILI어들의 채팅 데이터 확보 후 정제하여 인풋해주어 더욱 AILI어층만을 위한 서비스로 개선

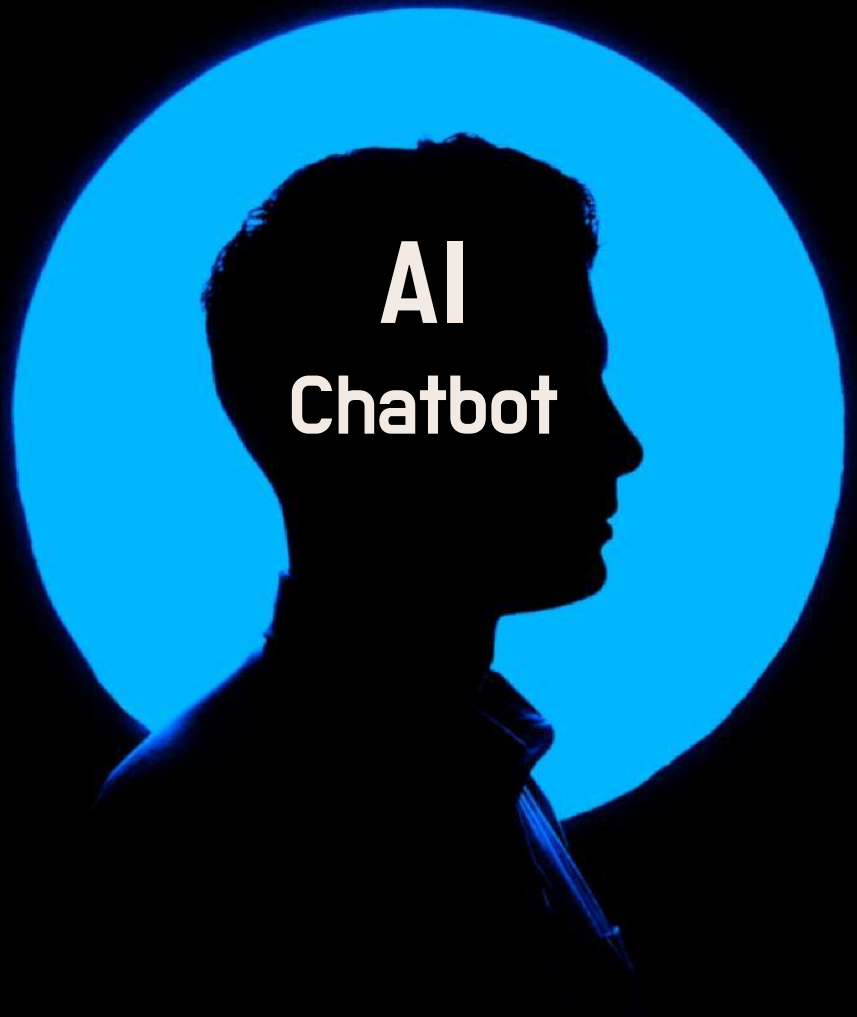
6 추후 상용화

Productization

늘어나는 시니어 인구층을 위한 새로운 니즈들 파악

이에 맞춰, 시니어들을 위한 서비스 플랫폼이 증가하고 있는 현황과 더불어 시니어 매칭 플랫폼 개발

상대적으로 채팅문화를 어려워하는 시니어들에게 서비스 제공을 통한 만족도 향상



THANK YOU