강화학습

기말 PROJECT 중간발표

20195237 장윤성
20215124 김수현

CONTENTS

01
프로젝트 목표

02
팀원 소개

03
서론

문제설명, 딥러닝, 강화학습

04
관련 연구 및 기술 동향

딥러닝, 강화학습

05
방법

강화학습 시스템 정의

06
실험

실험환경, 평가방법,
강화학습 학습 로깅 결과,
체크포인트 성능 비교

07
토의

현 강화학습 시스템의 한계점,
개선사항 제안

08
기대효과

프로젝트 결과물의 활용 예

01. 프로젝트 목표

각 사용자에게 맞는 개인화된 음식 추천을 위한 강화학습 알고리즘 개발

---

02. 팀원 소개

팀장 : 빅데이터학과 20195237 장윤성
팀원 : 빅데이터학과 20215124 김수현

03. 서론

문제 설명 :

메뉴 선택을 쉽게 하지 못하는 사람들을 위해 생각해낸 프로젝트입니다.

딥러닝으로 해결이 어려운 이유 :

개개인의 음식 취향이 다르고 당장의 상황에 따라 데이터가 동적으로 달라지기 때문에 딥러닝으로 학습하기 어렵습니다.

강화학습으로 도전하는 이유 :

각 사용자의 기분과 선호도가 달라 동적으로 변하기 때문에 레이블이 정해진 딥러닝으로 학습이 어렵고,
에이전트가 환경과 상호작용하며 연속적으로 학습할 수 있습니다.
또한, 사용자로부터 얻은 피드백을 보상으로 설정하여 에이전트가 사용자의 선호를 더 잘 이해하고 예측할 수 있게 됩니다.

03. 서론 - 학습 과정

초기학습 :

보편적으로 인정되는 메뉴로 Dqn 알고리즘을 사용하여 강화학습 모델을 초기 학습 합니다.

배포 :

초기 학습된 모델을 사용자에게 배포하여 사용자의 피드백을 수집합니다.

개인화된 학습 :

사용자 피드백을 수집하여 각 사용자에게 맞는 음식 추천을 위해 모델 파라미터를 업데이트합니다.

세트별 업데이트 :

아침, 점심, 저녁, 야식으로 구분된 4묶음으로 하루를 한 에피소드로 설정합니다.
즉, 하루가 끝나면 업데이트합니다.

평가 :

사용자 피드백에 따라 모델의 성능을 평가합니다.

04. 관련 연구 및 기술 동향

딥러닝으로 시도한 사례들 :

[음식 추천 시스템]

많은 음식 추천 시스템들이 사용자의 클릭이나 구매 기록을 기반으로 CNN, RNN을 사용해 추천을 해왔습니다.

하지만 이 방식은 사용자의 변화하는 기분이나 감정을 반영하기 어렵습니다.

강화학습으로 시도한 사례들 :

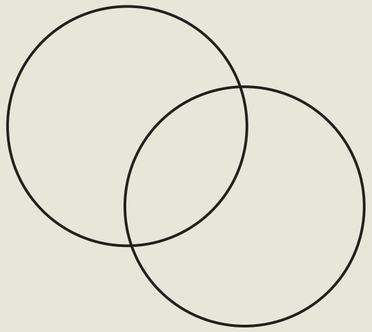
[개인화 추천 시스템]

Netflix, YouTube 등이 사용하는 강화학습 기반 추천 시스템은 사용자의 상호작용을 통해 지속적으로 학습하고,

사용자 경험을 개선합니다.

05. 방법

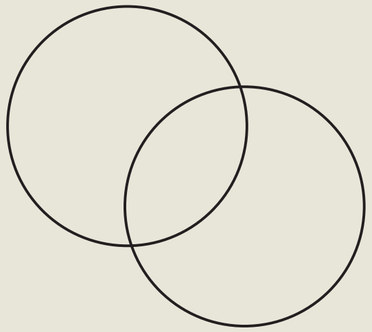
강화학습 시스템 정의



Agent	메뉴를 추천하는 주체로, 사용자의 기분, 이전에 선택한 메뉴, 현재 상태 등을 고려하여 다음 메뉴를 추천합니다.
Environment	사용자의 기분, 선택 가능한 메뉴 리스트, 시간 등이 포함된 환경입니다. 사용자의 기분에 영향을 미치는 요인들 또한 환경이 됩니다.
State	Agent가 결정을 내리는 데 사용하는 정보로, 사용자의 현재 기분(1-10점), 온도(1-10), 날씨(맑음, 구름, 흐림, 비, 눈), 인원 수(1-4명, 5명 이상), 성별(남, 녀) 시간(아침, 점심, 저녁, 야식)으로 구성됩니다.
Action	Agent가 할 수 있는 행동으로, 특정 메뉴를 추천하는 것입니다. 저희는 설문조사를 통해 메뉴리스트를 만들어 그 리스트 안에서 추천을 진행합니다.
Reward	사용자의 선택으로 계산되며, Agent의 추천이 얼마나 좋은지 평가하는 지표입니다. 만족도가 높을수록 더 큰 보상을 받습니다. 현재는 학습데이터에 있는 label 즉 menu값이 추천메뉴와 같다면 1점, 그렇지 않으면 -1점을 주기로 하였습니다.
강화학습 시스템 정의	Off - Policy Network
가치기반 강화학습 알고리즘 (off-policy)	뒷 페이지에 짧은 의사코드 있습니다.

05. 방법

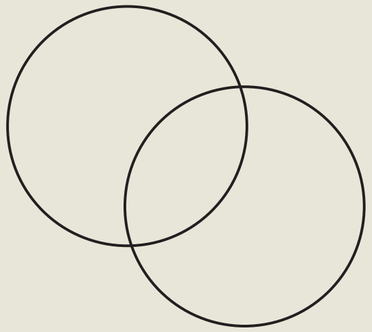
Dqn 수도코드



```
import numpy as np
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from collections import deque
import random

# 데이터 로딩 및 전처리
data = pd.read_csv("/content/강화학습을 위한 개인별 선호 음식 데이터 수집.csv")
data = data.drop(columns=['unnecessary_columns'])
data['encoded_column'] = pd.factorize(data['category_column'])[0]

# DQN 모델 정의
class DQN(nn.Module):
    def __init__(self, input_size, output_size):
        super(DQN, self).__init__()
        self.layer1 = nn.Linear(input_size, 50)
        self.layer2 = nn.Linear(50, output_size)

    def forward(self, x):
        x = torch.relu(self.layer1(x))
        return self.layer2(x)
```

05. 방법

Dqn 수도코드

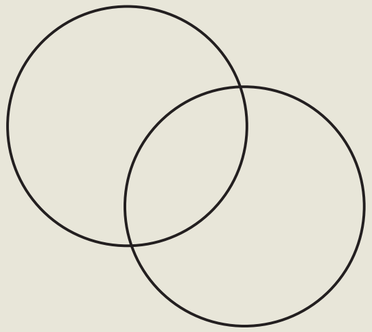


```
# 리플레이 버퍼 정의
class ReplayBuffer:
    def __init__(self):
        self.buffer = deque(maxlen=1000)

    def push(self, state, action, reward, next_state, done):
        self.buffer.append((state, action, reward, next_state, done))

    def sample(self, batch_size):
        return random.sample(self.buffer, batch_size)

# 에이전트 클래스 정의
class Agent:
    def __init__(self, state_dim, action_dim):
        self.model = DQN(state_dim, action_dim)
        self.buffer = ReplayBuffer()
        self.optimizer = optim.Adam(self.model.parameters(), lr=0.001)

    def update(self, batch_size=32):
        batch = self.buffer.sample(batch_size)
        # 손실 계산 및 역전파 코드는 생략
```

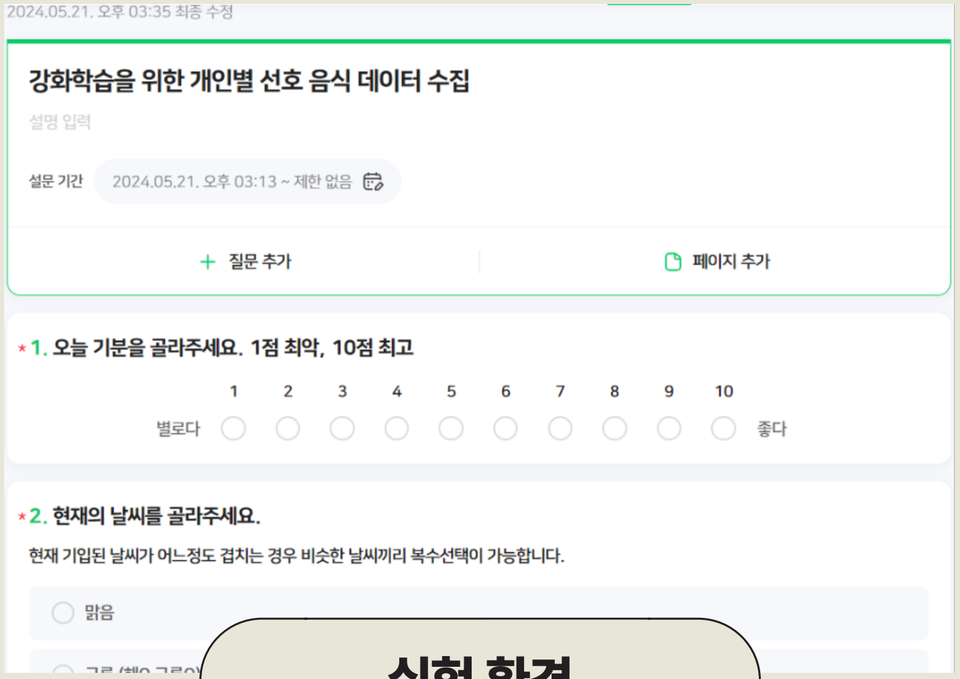
05. 방법

Dqn 수도코드

```
# 메인 학습 루프
def main():
    agent = Agent(state_dim=10, action_dim=len(menu_list)) # action_dim은 메뉴 리스트의 길이로 수정
    state = np.random.rand(10) # 초기 상태 (랜덤)

    for _ in range(1000): # 에피소드 또는 반복 횟수
        for one_day in day_info:   # day_info는 아침, 점심, 저녁, 야식을 나타냄
            state = zip(mood, temperature, weather, people_count, gender, time_of_day)
            action = user_pick(list_menu)
            next_state = zip(next_mood, next_temperature, ... , next_time_of_day)  # 다음 상태
            reward = 1 or -1   # 보상 1(True) 아니면 -1(False)
            done = if final? true or false  # 종료 여부 True 아니면 False

            agent.buffer.push(state, action, reward, next_state, done)

            if(done) -> agent.update
```



06. 실험



실험 환경

Intel Core i5-8250U CPU

google colab T4 GPU

데이터 수집 :

네이버 폼을 활용하여 사용자의 기분, 날씨, 시간, 배고픔 정도, 인원 수, 성별, 나이대 등의 정보 수집

학습 환경 :

하루의 식사 시간(아침, 점심, 저녁, 야식)을 하나의 에피소드로 묶어 학습합니다.

모델이 정확히 맞출 경우 +1의 보상을, 틀릴 경우 -1의 보상을 받습니다.

이를 통해 모델은 더 정확하게 예측하도록 학습합니다.

알고리즘 선택 :

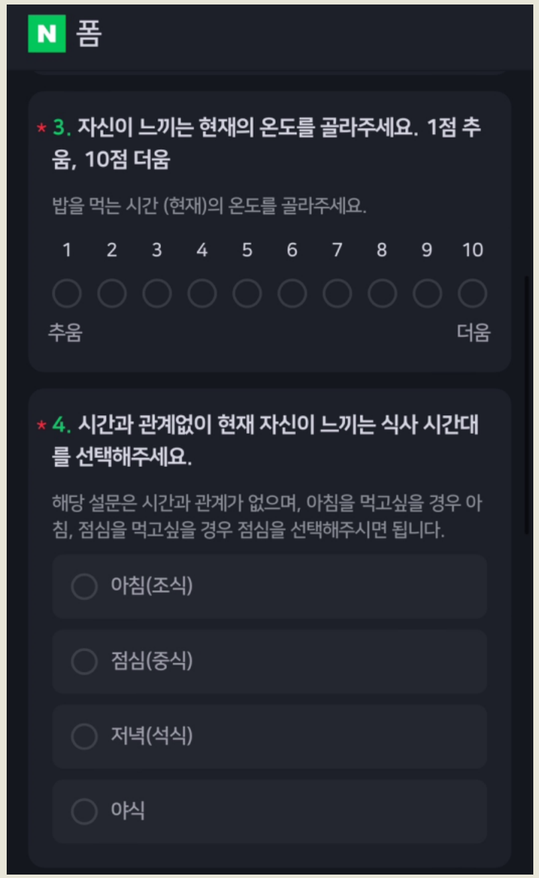
DQN 알고리즘 선택

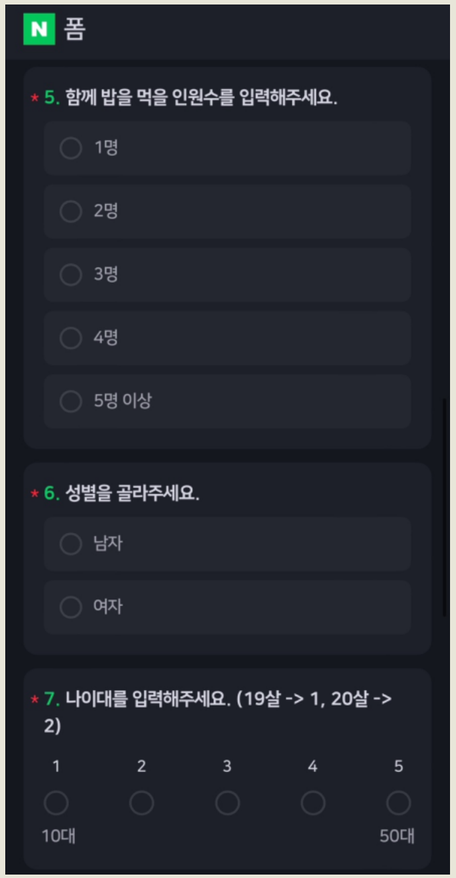
보상함수 설계 :

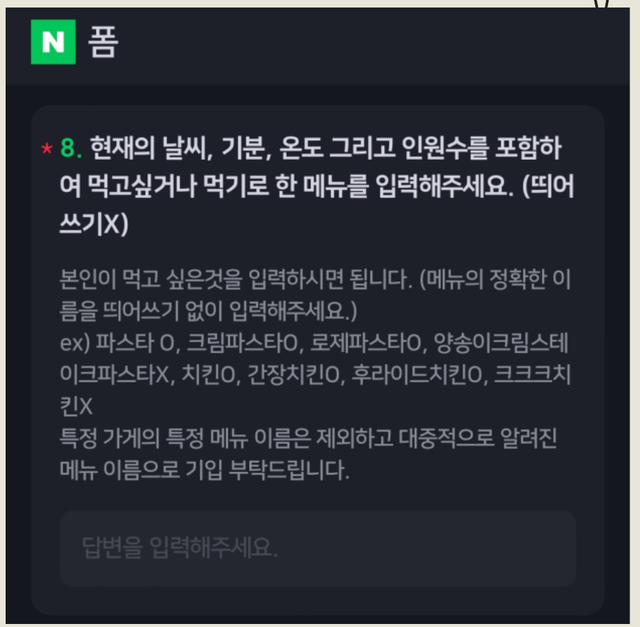
사용자 피드백을 반영하여 보상을 조정.

사용자가 좋아하는 음식에 대한 평가는 높은 보상을 부여하고, 만족하지 못한 음식에 대한 평가는 낮은 보상을 부여 (별점 1~5 까지 설정 예정)

06. 실험





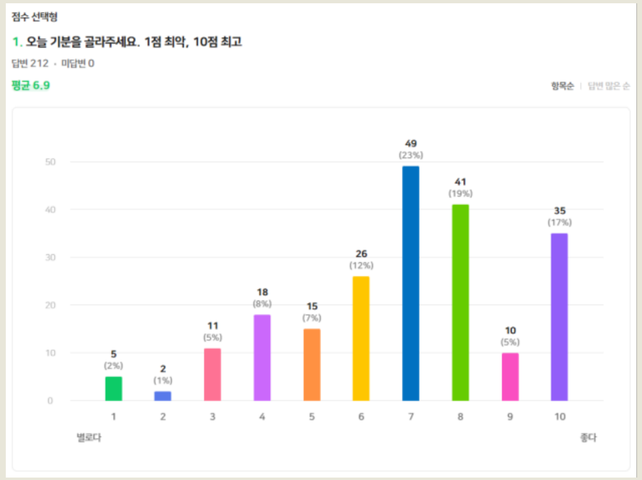


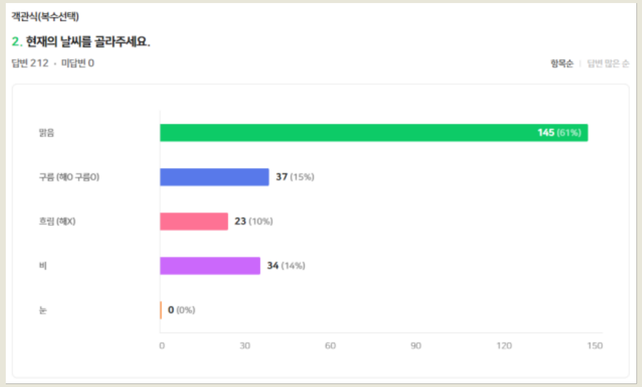
설문조사

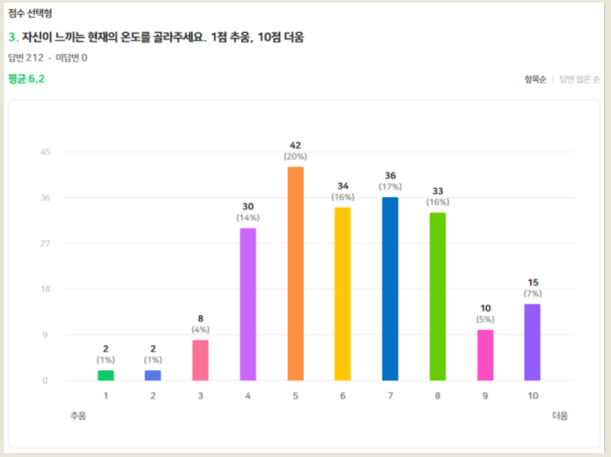
https://naver.me/xLJuRxYz

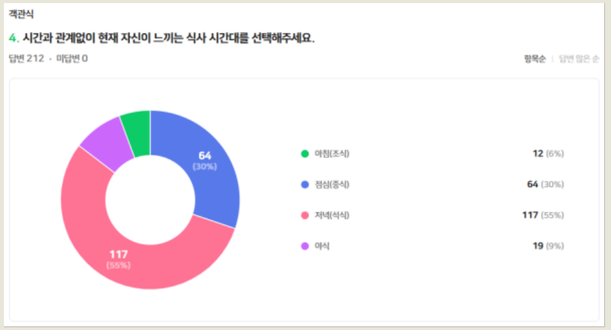
06. 실험


진행 중
강화학습을 위한 개인별 선호 음식 데이터 수집
2024. 05. 21. 오후 03:13 ~ 제한 없음 • 총 참여 212

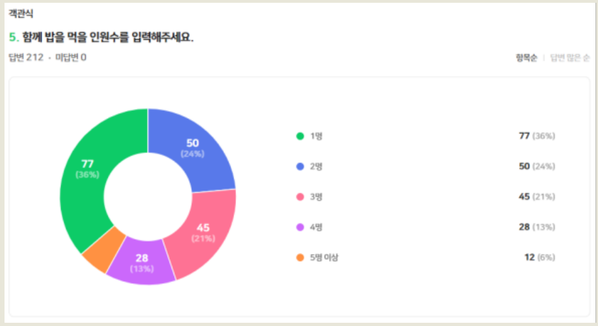

점수 선택형
1. 오늘 기분을 골라주세요. 1점 최악, 10점 최고
답변 212 · 미답변 0
평균 6.9
5 (2%)
2 (1%)
11 (5%)
18 (8%)
15 (7%)
26 (12%)
49 (23%)
41 (19%)
10 (5%)
35 (17%)
별로다
좋다

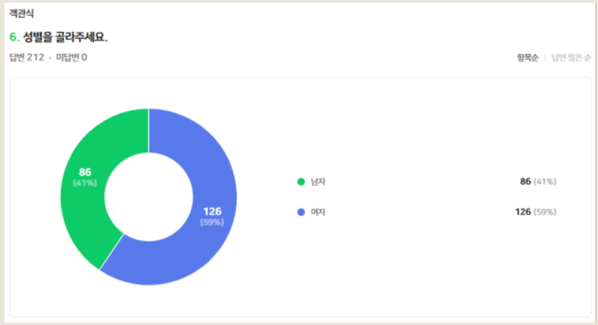

객관식(복수선택)
2. 현재의 날씨를 골라주세요.
답변 212 · 미답변 0
맑음 145 (61%)
구름 (해O 구름O) 37 (15%)
흐림 (해X) 23 (10%)
비 34 (14%)
눈 0 (0%)

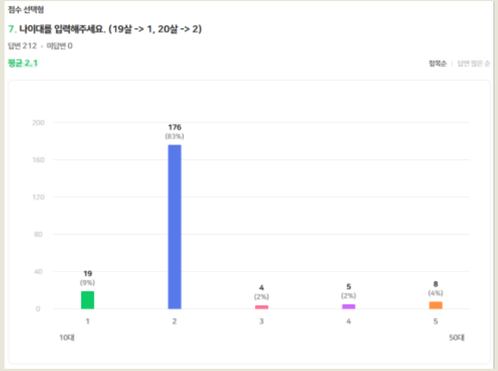

점수 선택형
3. 자신이 느끼는 현재의 온도를 골라주세요. 1점 추움, 10점 더움
답변 212 · 미답변 0
평균 6.2
2 (1%)
2 (1%)
8 (4%)
30 (14%)
42 (20%)
34 (16%)
36 (17%)
33 (16%)
10 (5%)
15 (7%)
추움
더움


객관식
4. 시간과 관계없이 현재 자신이 느끼는 식사 시간대를 선택해주세요.
답변 212 · 미답변 0
64 (30%)
117 (55%)
아침(조식) 12 (6%)
점심(중식) 64 (30%)
저녁(석식) 117 (55%)
야식 19 (9%)

데이터 수집


객관식
5. 함께 밥을 먹을 인원수를 입력해주세요.
답변 212 · 미답변 0
77 (36%)
50 (24%)
45 (21%)
28 (13%)
1명 77 (36%)
2명 50 (24%)
3명 45 (21%)
4명 28 (13%)
5명 이상 12 (6%)


객관식
6. 성별을 골라주세요.
답변 212 · 미답변 0
86 (41%)
126 (59%)
남자 86 (41%)
여자 126 (59%)


점수 선택형
7. 나이대를 입력해주세요. (19살 -> 1, 20살 -> 2)
답변 212 · 미답변 0
평균 2.1
19 (9%)
176 (83%)
4 (2%)
5 (2%)
8 (4%)
10대
50대

06. 실험

주관식 단답형

8. 현재의 날씨, 기분, 온도 그리고 인원수를 포함하여 먹고싶거나 먹기로 한 메뉴를 입력해주세요. (띄어쓰기X)

답변 212 · 미답변 0

제육볶음	답변 3개
마라탕	답변 13개
떡볶이	답변 9개
닭볶음탕	답변 3개
등심스테시크피자치즈스파게티	답변 1개
항정살덮밥	답변 1개
족발	답변 2개
맑음882돈까스	답변 1개
삼겹살	답변 9개
파스타	답변 4개
하와이안피자	답변 1개

데이터 수집

파스타	답변 4개
하와이안피자	답변 1개
크림파스타	답변 1개
피자	답변 3개
김치찌개	답변 5개
불고기피자	답변 1개
회	답변 2개
덮밥	답변 1개
타코야끼	답변 2개
닭강정, 야끼소바	답변 1개
초밥	답변 5개
짜장면	답변 2개
고구마피자	답변 1개

06. 실험

평가방법

1. 사용자 만족도 조사

직접 피드백 : 사용자로부터 직접 받은 피드백 점수를 분석하여 평가합니다.

설문 조사 : 사용자 경험에 대한 설문 조사를 통해 모델에 대한 만족도를 추가로 평가합니다.

2. 반복적 학습 및 평가

학습 곡선 : 각 Episode마다 모델의 성능 변화를 추적합니다.

피드백 루프 : 사용자 피드백을 모델에 반영하고, 그 결과를 다시 평가합니다.

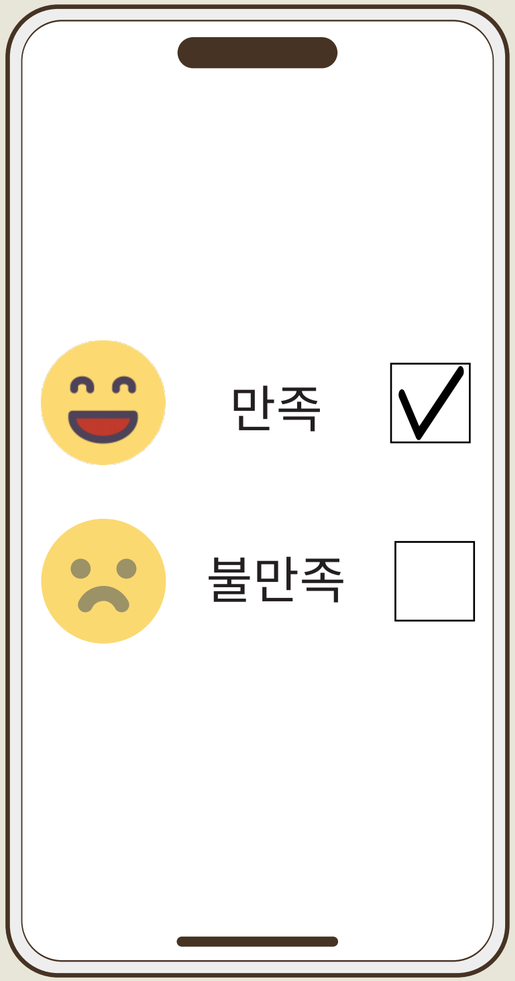
3. 평가 지표 - 정확성

평균 보상: 사용자가 제공한 피드백 점수의 평균을 측정합니다. 이는 모델이 사용자의 기대에 얼마나 부합하는지를 간단하게 보여줍니다.

정확률 (Precision): 추천 시스템이 제안한 메뉴가 사용자가 실제로 선택하거나 선호하는 메뉴인지를 확인합니다. 아침, 점심, 저녁, 야식 각각의 보상을 합산하여 총 보상(total reward)을 계산합니다.

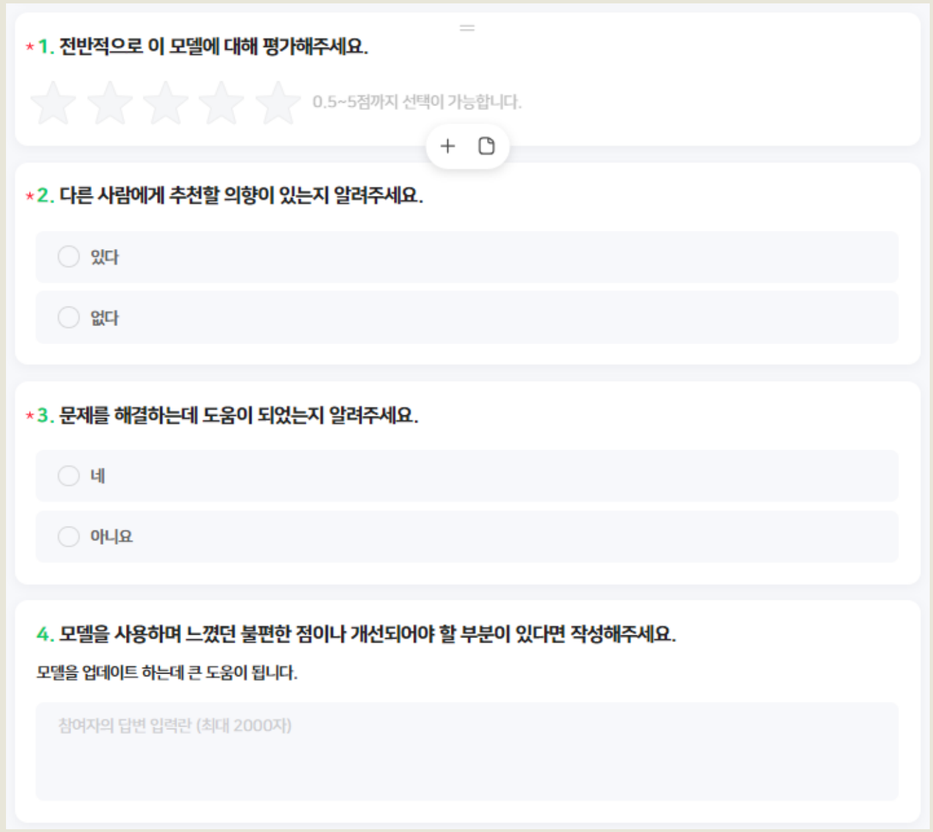
06. 실험

직접 피드백



예시화면

설문조사



06. 실험

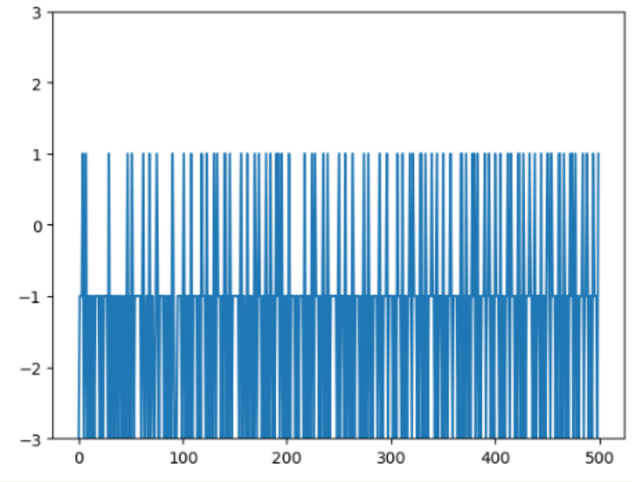
각 에피소드 별 총 보상(total reward)은 후반부로 갈수록 향상되는 경향을 보입니다.
이는 아침, 점심, 저녁, 야식 시간대별로 추천한 메뉴를 정확히 맞추는 경우가 증가한다는 것을 의미합니다.
따라서, 모델의 성능이 점차 개선되고 있음을 나타냅니다.

```
np.mean(reward_arr[500:1000])
```

```
-1.296
```

```
plt.ylim(-3, 3)
plt.plot(reward_arr[500:1000])
```

```
[<matplotlib.lines.Line2D at 0x7be068698550>]
```

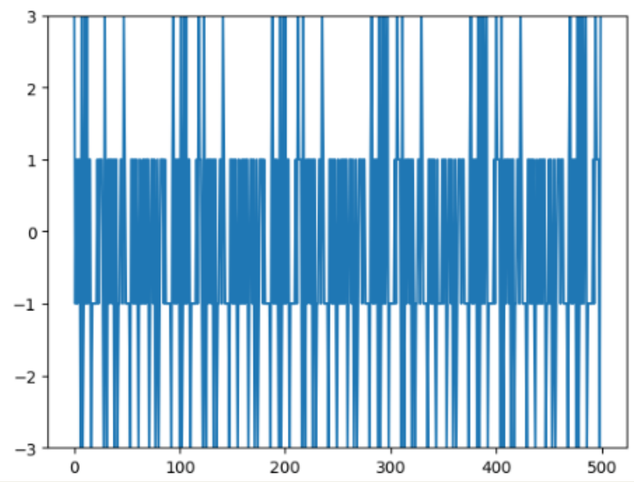



```
np.mean(reward_arr[3500:4000])
```

```
-0.196
```

```
plt.ylim(-3, 3)
plt.plot(reward_arr[3500:4000])
```

```
[<matplotlib.lines.Line2D at 0x7be0687149d0>]
```

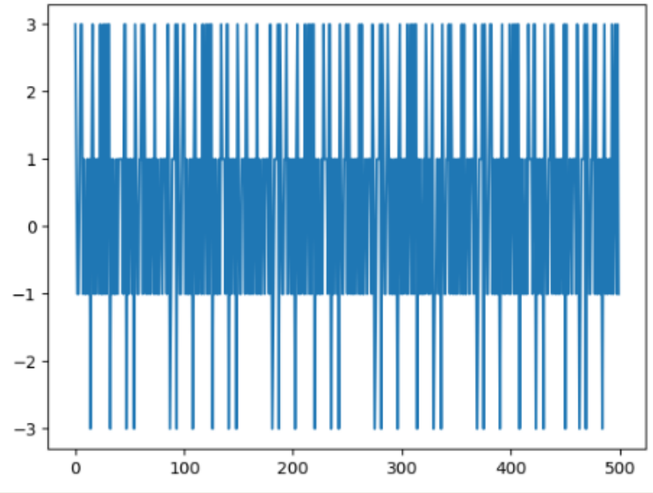



```
np.mean(reward_arr[9500:10000])
```

```
0.432
```

```
plt.plot(reward_arr[9500:10000])
```

```
[<matplotlib.lines.Line2D at 0x7be068581960>]
```



강화학습 학습 로깅 결과

06. 실험

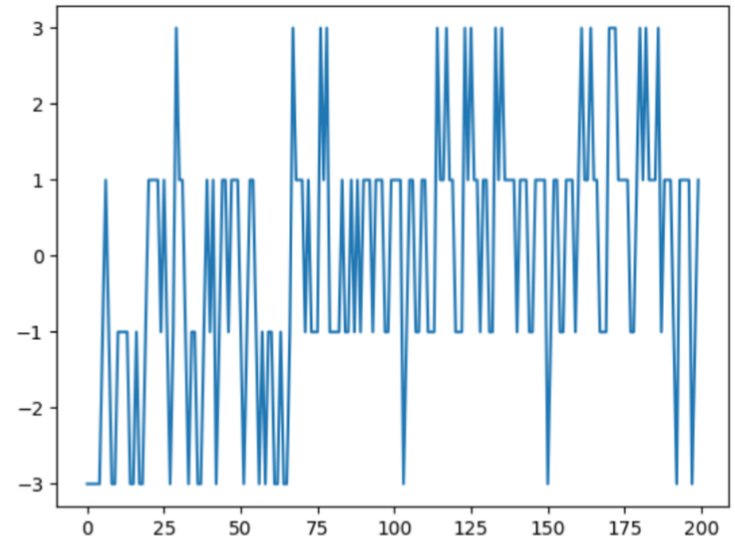
체크포인트 성능 비교

```
# 모델 체크포인트 로드 및 평가
evaluation_results = []

for checkpoint in checkpoints:
    checkpoint_path = f'model_checkpoint_{checkpoint}.pth'
    checkpoint_data = torch.load(checkpoint_path)

    agent.model.load_state_dict(checkpoint_data['model_state_dict'])
    agent.optimizer.load_state_dict(checkpoint_data['optimizer_state_dict'])
    agent.epsilon = 0.01 # 평가 시 탐색 확률을 최소화

    total_correct = 0
    total_samples = 0

    for day_data in d_episodes:
        for i in range(len(day_data) - 1):
            state = day_data[i][:-1]
            target_menu = day_data[i][-1]

            action = agent.select_action(state)
            if action == target_menu:
                total_correct += 1
            total_samples += 1

    accuracy = total_correct / total_samples
    evaluation_results.append((checkpoint, accuracy))
```

```
# 평가 결과 출력
for checkpoint, accuracy in evaluation_results:
    print(f"Checkpoint {checkpoint}: Accuracy = {accuracy * 100:.2f}%")
```

06. 실험

체크포인트 성능 표




```
Checkpoint 1000: Accuracy = 37.59%
Checkpoint 2000: Accuracy = 39.36%
Checkpoint 3000: Accuracy = 40.43%
Checkpoint 4000: Accuracy = 44.33%
Checkpoint 5000: Accuracy = 51.06%
Checkpoint 6000: Accuracy = 57.80%
Checkpoint 7000: Accuracy = 58.16%
Checkpoint 8000: Accuracy = 59.57%
Checkpoint 9000: Accuracy = 60.64%
```

Checkpoint	Accuracy
1000	**37.59%**
2000	39.36%
3000	40.43%
4000	44.33%
5000	51.06%
6000	57.80%
7000	58.16%
8000	59.57%
9000	60.64%

07. 토의

현재 강화학습 시스템의 한계점

1. 각 시간대별로 하나의 메뉴만을 추천받기 때문에, 메뉴를 맞출 경우 성공률이 100%,

   틀릴 경우 0%가 됩니다.

2. 현재 평균 보상에 대한 사용자 피드백 시스템은 아직 구현되지 않았습니다.

   예를 들어, 별점과 같은 피드백을 통해 보상을 더 세밀하게 조정하는 기능이 빠져 있습니다.

3. 처음에는 각 시간대별로 세 가지 메뉴를 추천하고자 했으나,

   데이터 수집 과정에서 메뉴를 단 하나씩만 받게 되어 이 계획에 어려움이 있었습니다.

07. 토의

개선사항 제안

1. 1차 혹은 2차 모델을 파인튜닝하여 각 시간대별로 3개의 메뉴를 추천할 수 있도록 할 예정입니다.

2. 2차 모델 배포 단계에서는 사용자의 만족도를 평가할 수 있는 별점 시스템을 도입하는 것을 고려하고 있습니다.

3. 현재의 에피소드 구성이 너무 짧다고 판단되어, 하루가 아닌 2일, 3일 또는 일주일 등으로 에피소드 기간을 늘릴 수 있는 방안도 검토 중입니다.

08. 기대효과

프로젝트 결과물의 활용 예

1. 음식점 및 카페 추천

2. 미디어 콘텐츠 추천 시스템

3. 사용자의 니즈를 파악한 개인별 마케팅 자료 제공

4. 건강 관리 식단 어플

THANK YOU