

Movie Recommendation

4조 RECCAR



김성연, 배성재, 양승훈, 조수연, 홍재형, 황선태

목차

table of contents

- 1 EDA
- 2 SASRec
- 3 HyperParameter Tuning
- 4 Ensemble

1

EDA

```
sum(train[['user', 'item']].duplicated())
```

0

같은 유저가 같은 영화를
여러번 시청하지 않음

>

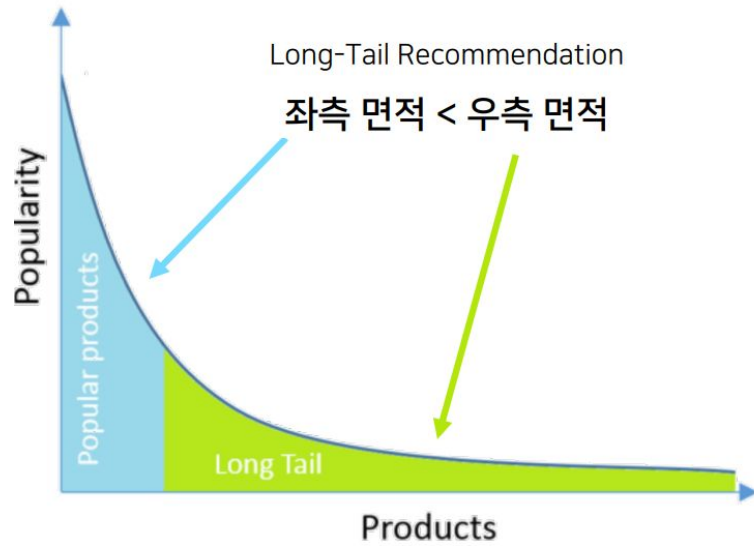
```
train['user'].value_counts()
```

✓ 0.1s

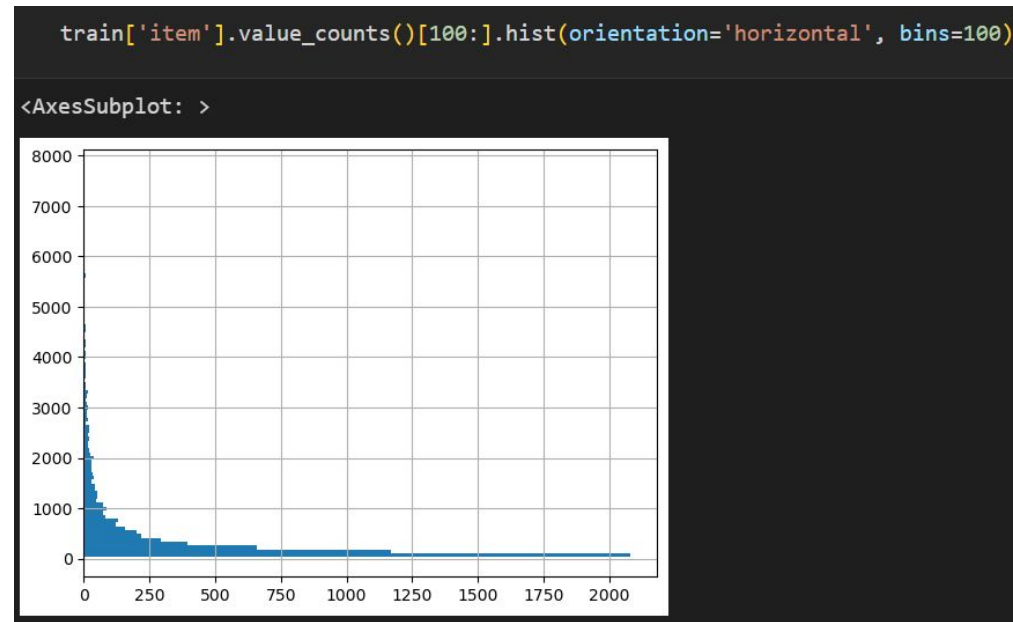
8405	2912
54465	1980
24219	1842
32514	1830
91349	1795
	...
11211	32
128756	28
105578	22
68606	19
53188	16

유저의 영화 시청 횟수
-> 콜드 스타트 문제가 없는 것 확인

Long-tail



The Long-tail of popularity of items



인기있는 영화에 많이 쏠려있는 것을 확인

Rule-base

인기있는 10개 동일 추천

Public Recall
0.0673

```
for user in tqdm(user_lst):
    cnt = 0 # 반복문마다 세는 변수.
    # 영화 시청 기록 숫자만큼 반복문 돌기(최대한으로 잡아놓은 것 뿐 이만큼 안돌음)
    # idx : 0, 1, 2, ...
    for idx in range(num_items):
        # 이번 영화가(top_items[idx], 영화 인기 순으로 나옴)
        # 해당 유저가 본 영화(user_item_lst[user])가 아닌 영화라면.
        if top_items[idx] not in user_item_lst[user]:
            cnt += 1 # 해당 유저에게 추천할 영화가 하나 늘었다.
            # 해당 영화(top_items[idx])를 해당 유저에게 추천한다. 즉 데이터 프레임에 user, item을 추가한다.
            test = test.append(pd.DataFrame([[user, top_items[idx]]], columns=['user', 'item']))
            if cnt == 10: # 한 사람당 추천할 영화는 10개 이므로.
                break
```

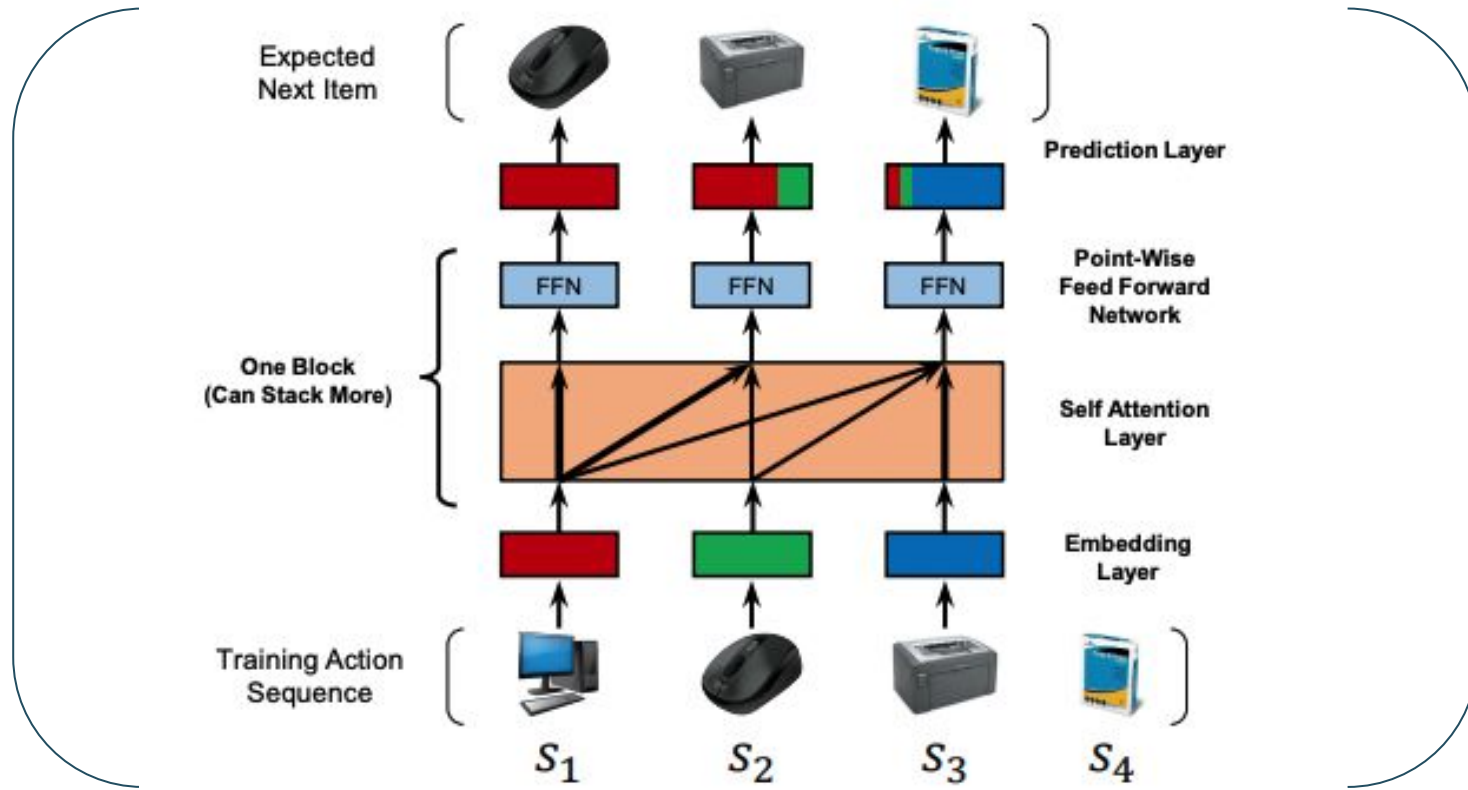
if



2

SASRec

Part 2 SASRec



Transformer decoder를 이용한 sequential recommendation 모델

Part 2 Indexing

```
train['item'].nunique(), train['item'].min(), train['item'].max()
```

✓ 0.8s

```
(6807, 1, 119145)
```

6807개 영화 중 영화번호는 1부터 119145번까지 맵핑되어 있습니다.

```
item = random.randint(1, item_size - 2)
# item_set 안에 없는 item이 나올 때 까지 계속 반복
while item in item_set:
    item = random.randint(1, item_size - 2) |
```

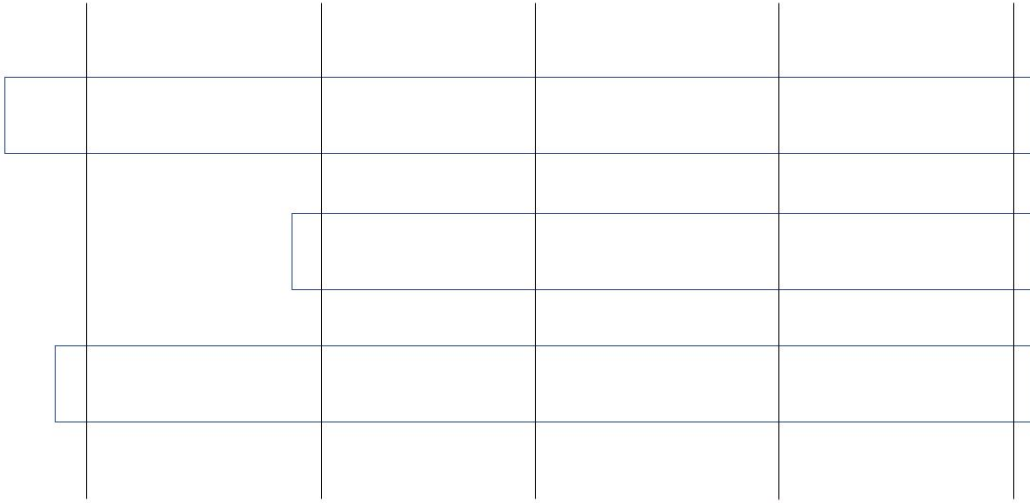
이 경우 Negative Sampling 진행 시 실제로 존재하지 않는 영화번호가 추출될 확률이 높습니다.

Part 2 **Max_len** 변수

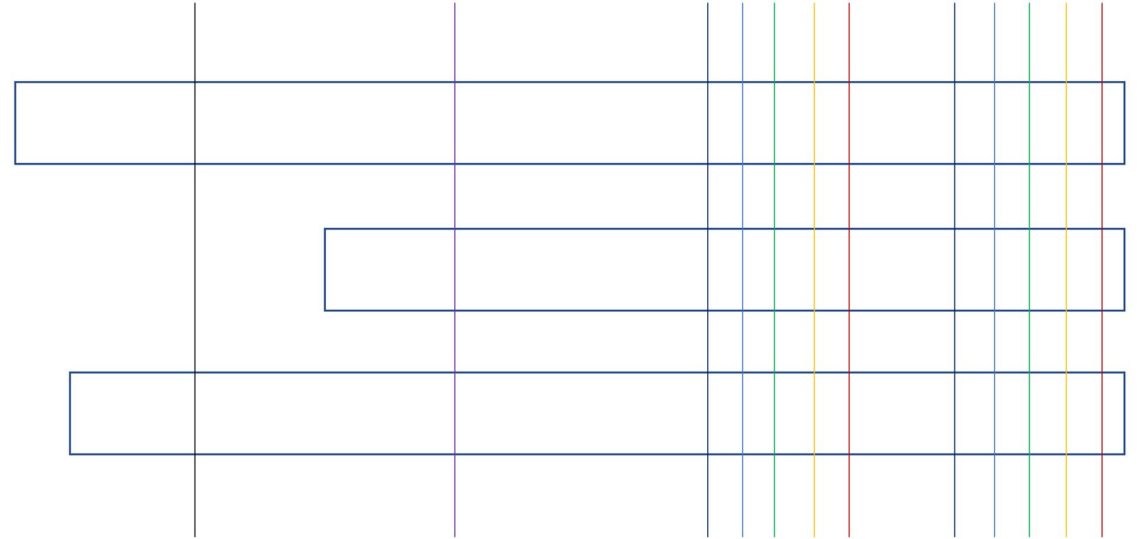
다음 영화를 예측하기 위해 사용하는 영화 기록의 최대 길이인
Max_len 변수의 경향성

Max_len 값	Public Recall@K
20	0.0982
50(Default)	0.1014
100	0.1064

Part 2 Data Augmentation



<단순 증강>



<최근 시점 슬라이싱 윈도우 적용>

유저 시청기록을 겹치지 않게 단순 증강한 것은 성능과 크게 연관이 없었습니다.

최근 데이터 시퀀스를 서로 겹치는 슬라이싱 윈도우방식을 적용한 결과 성능이 상승했습니다.


Part 2 Test Set 구성


W


이원성_마스터 16일 전


- 사용자 당 추천할 아이템은 10개를 넣어주셔야
알맞게 평가가 됩니다. 이력이 하나라도 있는
user는 어떤 식으로든 10개의 추천 아이템을 생
성할 수 있습니다. train에 없는 item_id는 test
set에서도 제외됩니다. 즉, 평가와 무관합니다.
- sequence 중간에 마스킹 된 아이템과 next
item 의 비율은 동일합니다.

(편집됨)

 1

 1






김성연_T4040 16일 전


빠르게 답변주셔서 정말 감사합니다.
혹시 시청기록이 많은 유저는 test item이 더 많을 확
률이 높은지 추가로 여쭙볼 수 있을까요??
감사합니다.


W

이원성_마스터 16일 전

모든 유저는 동일한 개수의 test item을 가집니다.

 1

 1



모든 유저는 동일한 개수의 Test item을 가짐

마스킹 된 item과 Next item 비율은 동일

성능을 극대화하기 위해 시퀀셜한 모델과
시퀀셜하지 않은 모델을 앙상블해야 함

시퀀셜한 모델은 시퀀셜함에 더 집중하면 좋음

SASRec Valid Set 구성 설명

```
elif self.data_type == "valid":  
    input_ids = items[:-1]  
    target_pos = items[1:]  
    answer = [items[-1]]
```

그러므로 유저 기준 마지막 영화만을 Valid Set으로 하는 것은 정당합니다.

```
user_set = set(user_item_list[:-1]) # valid에 있는 데이터는 네거티브에 안걸림.  
# input_ids 길이만큼 target_neg에 negative samples 생성  
for _ in input_ids:  
    target_neg.append(neg_sample(user_set, self.args.item_size))
```

네거티브 샘플링에서 valid set에 있는 영화 또한 제외됩니다.

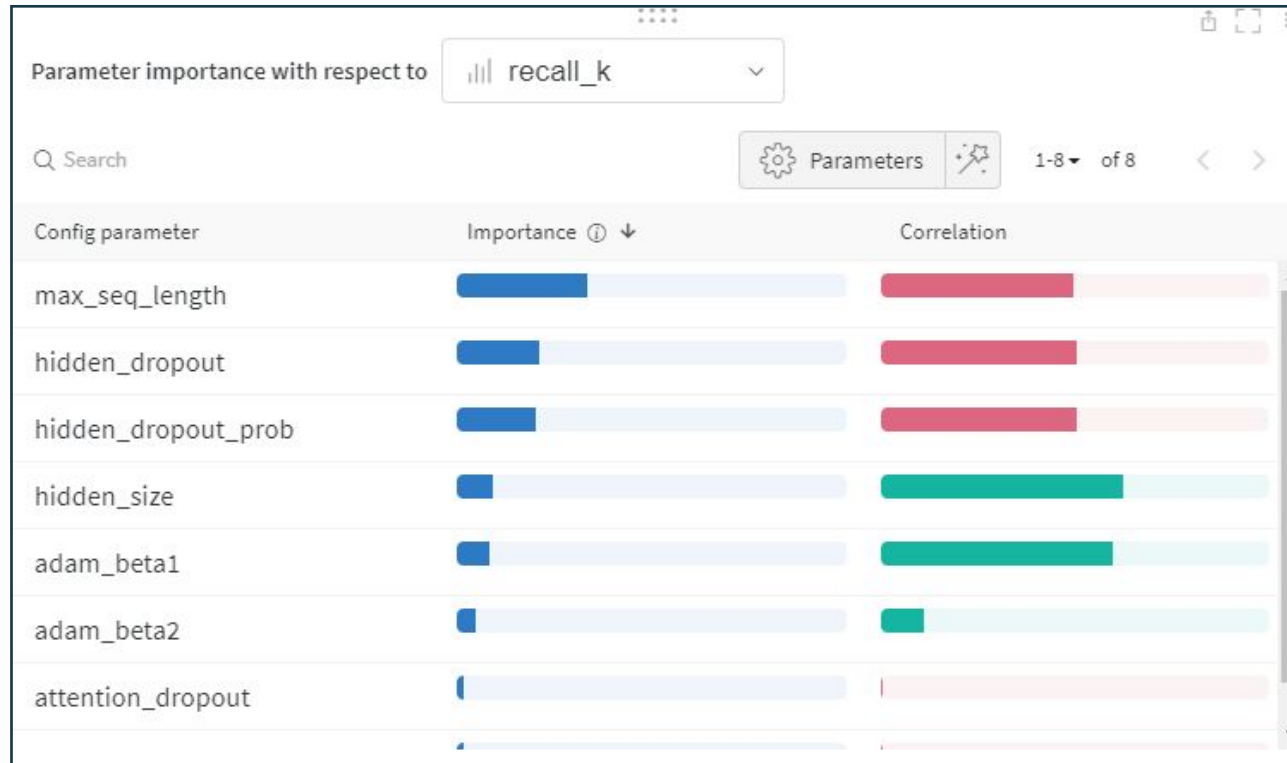
하지만 test set은 네거티브 샘플링에서 제외되지 않기 때문에 에폭이 늘어날 수록 valid recall은 증가하나 test recall은 감소할 수 있습니다.

test와 동일한 환경 조성을 위해 valid set 내 아이템에도 네거티브 샘플링을 진행했습니다.

3

Hyperparameter Tuning

Part 3 Wandb 사용법



importance

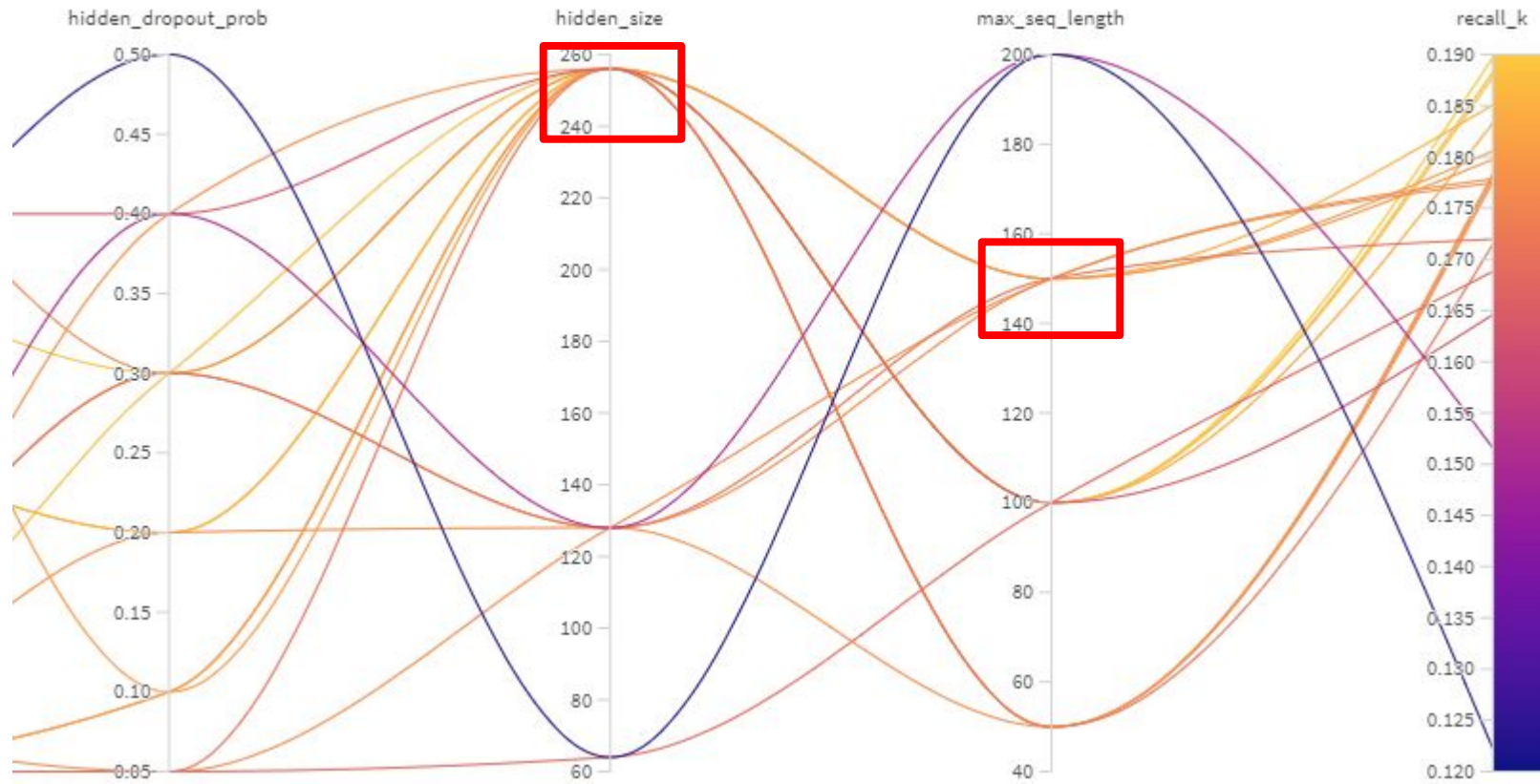
변수 중요도

Correlation

Recall과의 상관관계

중요 하이퍼파라미터를 찾거나 **Recall**을 크게 하기 위해
각 하이퍼파라미터들을 증가 또는 감소해야겠다는 직관을 얻게됩니다.

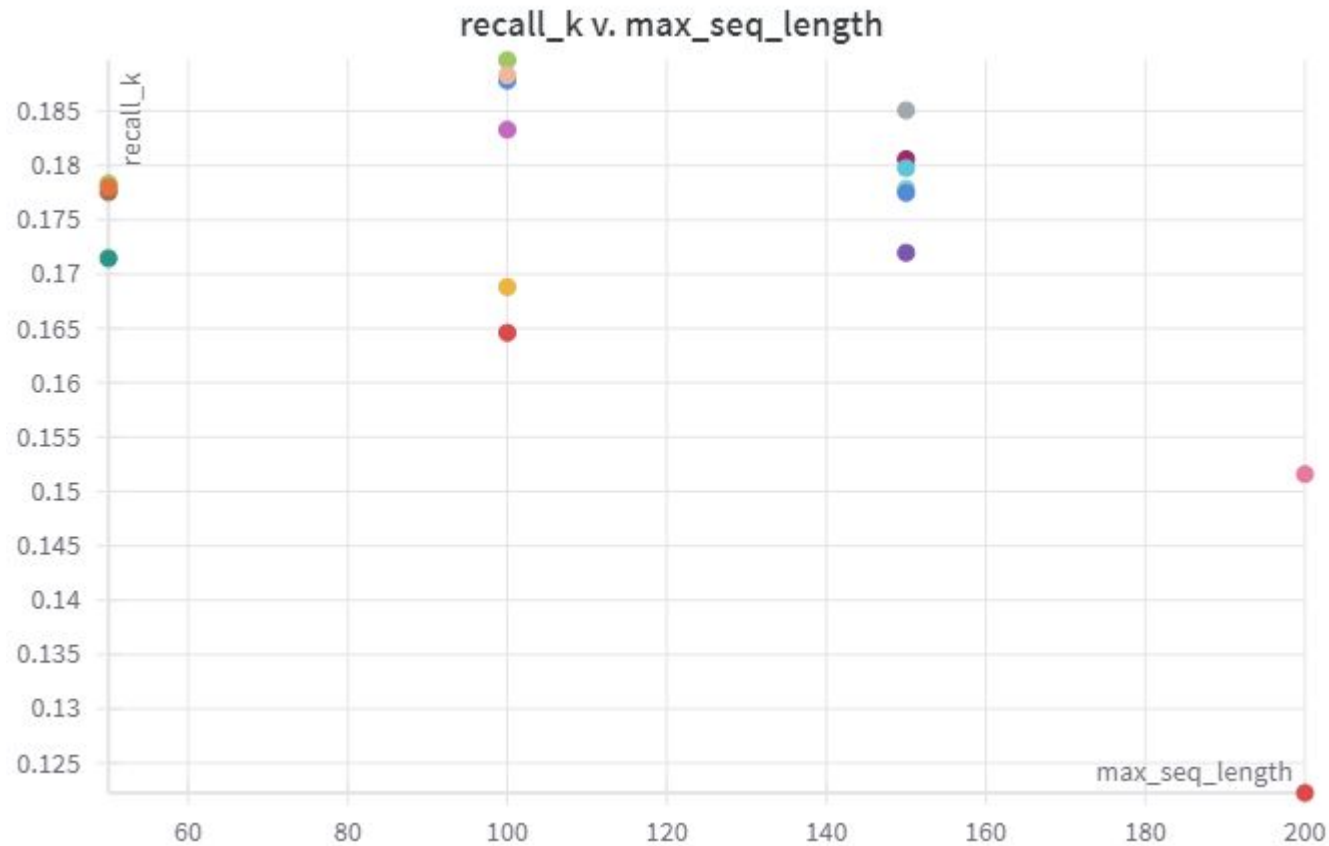
Part 3 Wandb 사용법



- Recall이 큰 모델
- Recall이 작은 모델

모델의 전반적인 경향성을 알 수 있는 그림입니다.

Part 3 Wandbox 사용법



X축

하이퍼파라미터 (max_len)

Y축

Recall@K

산점도를 그려 훨씬 직관적으로 해당 하이퍼파라미터의 영향도를 살펴볼 수 있습니다.

Part 3 최종 주요 하이퍼파라미터 변경 값

항목	기존 값	수정한 값
Adam_Beta1	0.9	0.7
Adam_Beta2	0.999	0.9999
Hidden_DropOut	0.5	0.3
Attension_DropOut	0.5	0.2
Max_seq_len	50	300
Hidden_size	64	256

Part 3 PreTrain을 사용하지 않은 이유

1, Wandb를 이용한 하이퍼파라미터 튜닝이 까다롭고 시간소모도 심합니다.

2, cold-start 유저가 없기 때문에 장르라는 Context 기반 추천을 할 이유가 떨어집니다.

3, 여러 번 실험한 결과 pre_train을 진행한 경우 에폭 초반 loss는 확실히 작지만 에폭이 진행될 수록 pre_train을 진행하지 않는 모델에 따라잡히는 모습입니다.

4

Ensemble

Part 4 SASRec 이외의 시도했던 모델들

MF

LightFM

DeepFM

Normal
Auto
Encoder

Multi-DAE

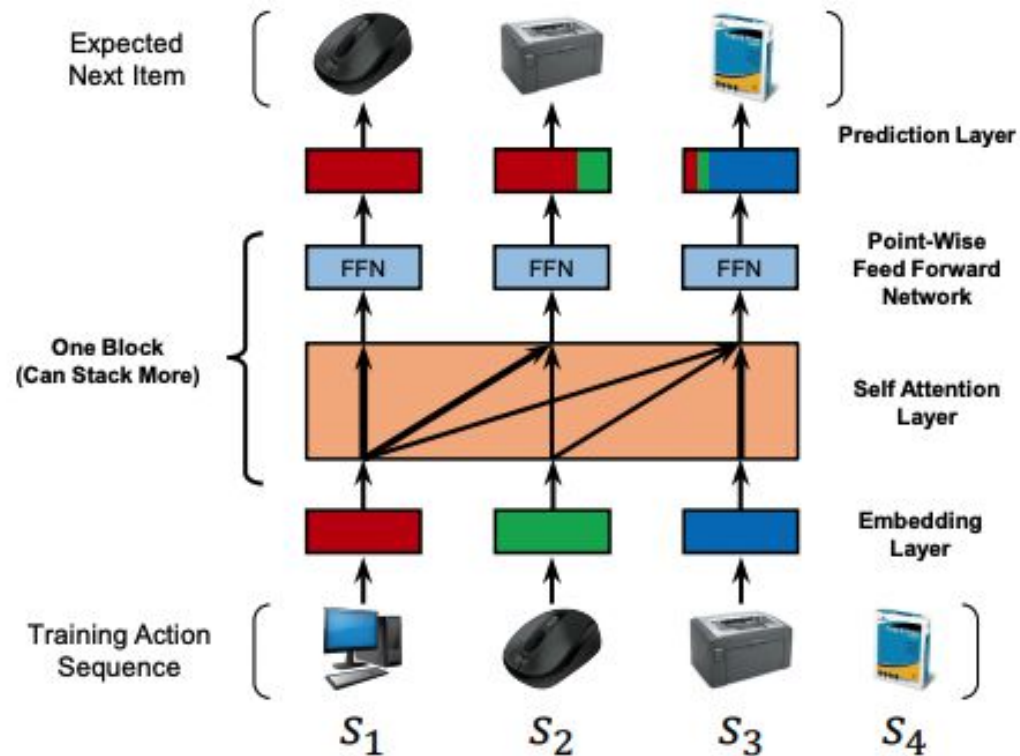
Multi-VAE

EASE

- 그러나 **EASE**를 제외하곤 단순 인기도 기반 **Rule-Base** 모델보다 좋지 못했습니다.
- **EASE** 모델 사용시 **Public** 기준 0.1599로 압도적인 성능을 기록했습니다.

Part 4 Ensemble

- SAS Rec

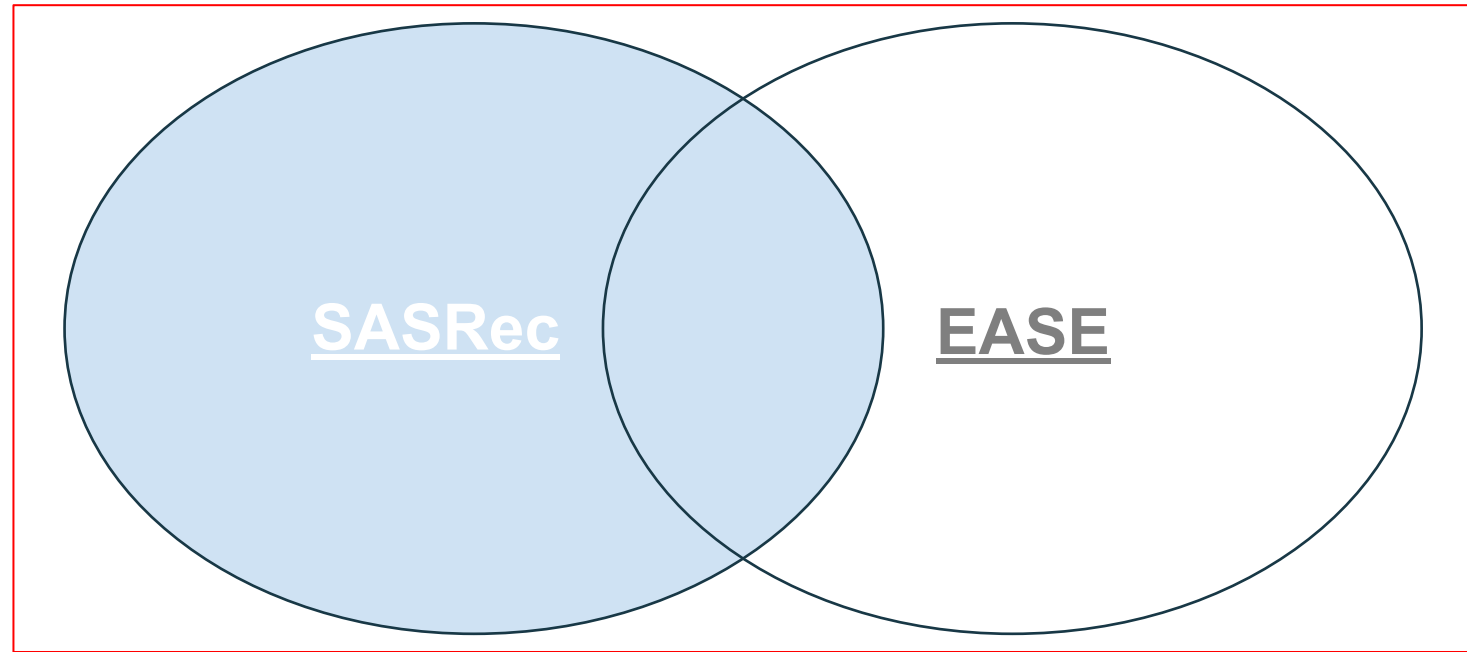
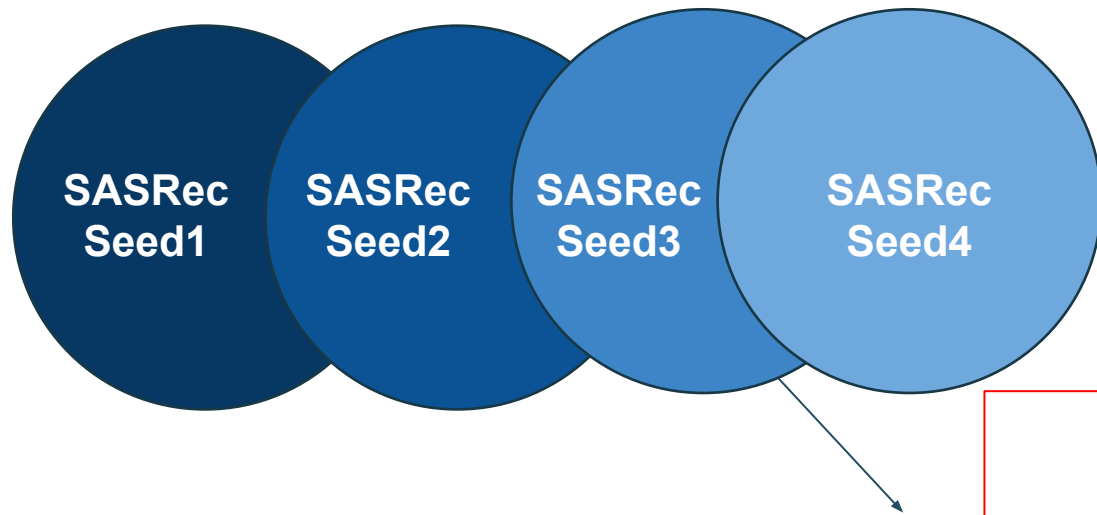


-EASE



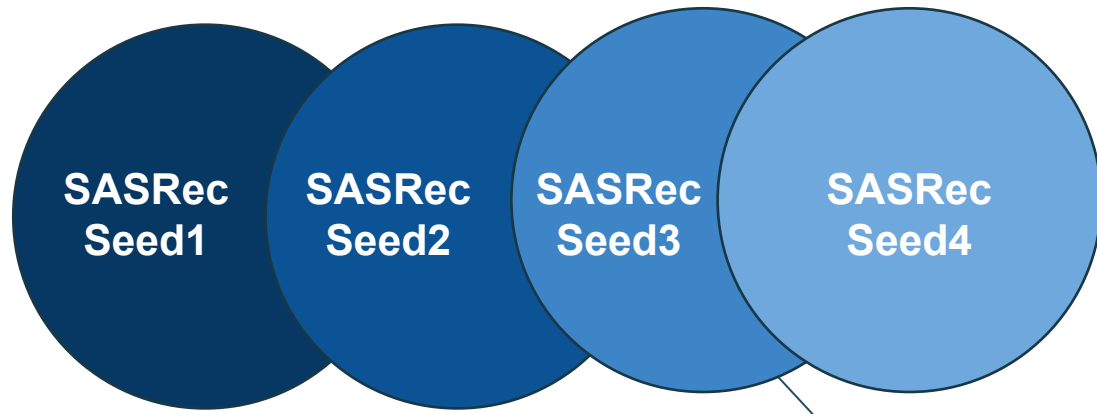
$$(\text{SASRec } 0.1324) * (0.5) + (\text{EASE } 0.1599) * (0.5) \Rightarrow 0.1816$$

Part 4 Ensemble

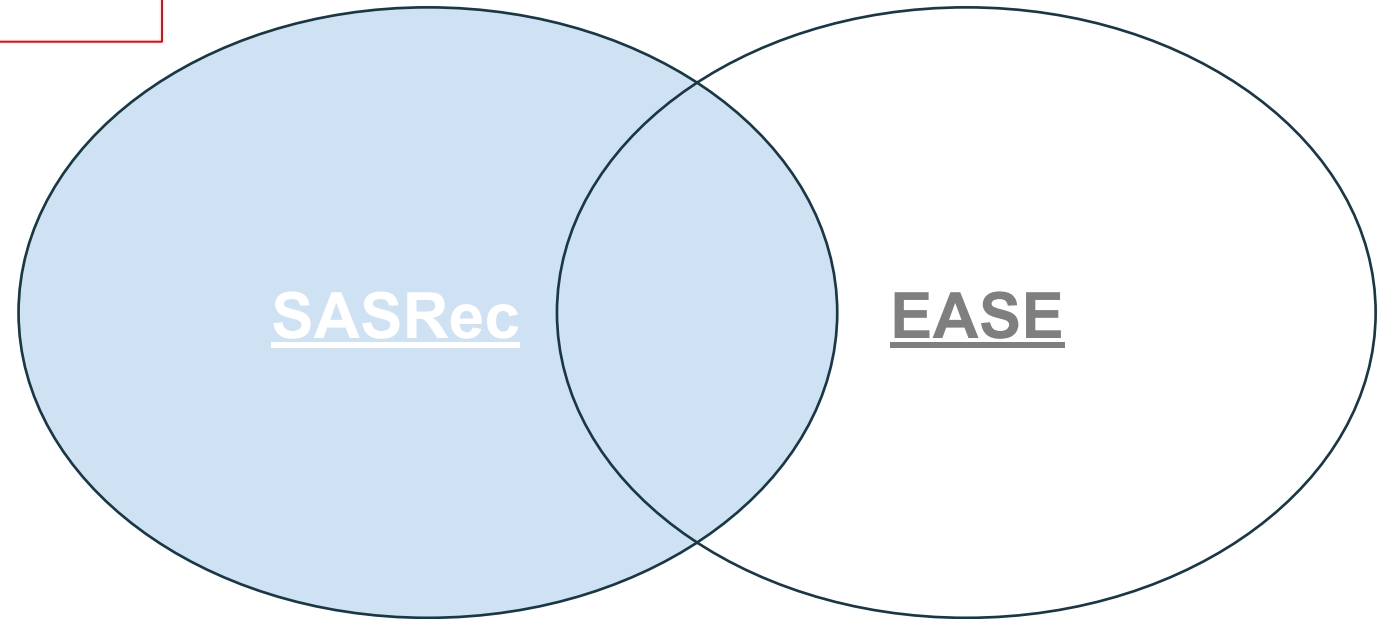


- 겹치는 항목 -> 강력 추천 항목
- 안겹치는 항목 -> EASE 1순위, SASRec 2순위

Part 4 Ensemble



0.1816 => 0.1869



- 시드 값을 바꾼 여러 개의 아웃풋을 앙상블하여 **SASRec** 모델 아웃풋을 만들었습니다.



감사합니다
