랩업리포트

Wrap-Up Report, RecSys04 Team RECCAR

▼ 목차

▼ Part 1. 프로젝트 Wrap Up

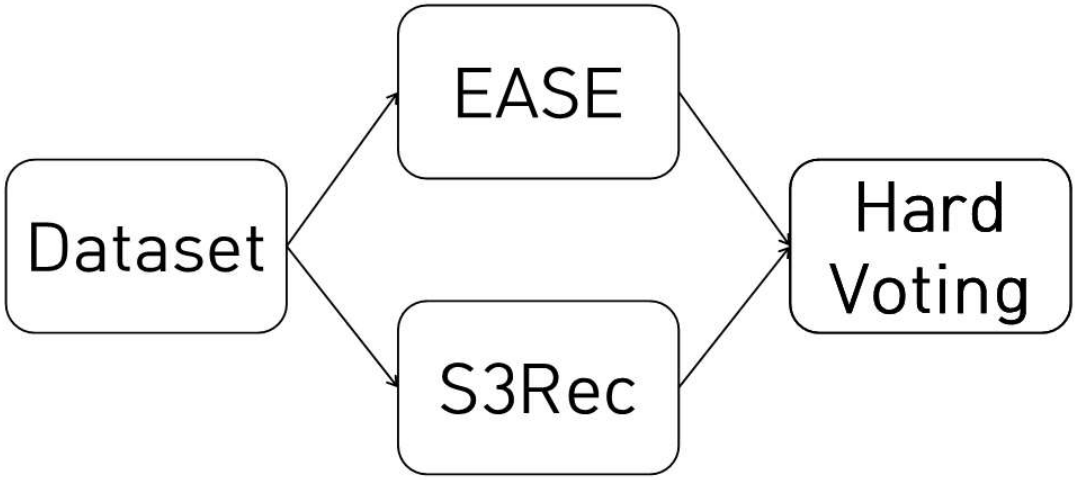
▼ 1-1. 프로젝트 개요

1-1-1. 프로젝트 목표

사용자의 sequential 영화 시청 데이터(user, item, timestamp)와 영화 메타 데이터를 바탕으로 사용자가 다음에 시청할 영화 및 좋아할 영화를 예측하는 것이 최종 목표입니다.

마스크와 시퀀셜이 섞여있어 모델을 앙상블하려고 했습니다.

1-1-2. 프로젝트 구조



sequential한 데이터에서 좋은 성적을 달성한 S3Rec 모델과 전체 데이터를 바탕으로 좋은 성적을 달성한 EASE 모델을 Hard Voting 방식으로 ensemble했습니다.

1-1-3. 협업 도구

Slack

• 파일 공유, 논의해야할 사항을 Slack을 이용해 공유하였습니다.

Github

• Git Flow 브랜치 전략

  master, 실험 branch를 나누어서 진행했습니다. 실험의 결과에 따라서 master branch에 merge하였습니다.

• Github Issues 기반 작업 진행

  팀원들이 진행하고 있는 실험의 세부 내용 공유를 위해 사용했습니다.

• Github Projects의 칸반 보드를 통한 일정 관리

  팀원들이 현재 진행하고 있는 실험을 공유했습니다. 이를 통해 실시간 실험 관리에 유용했습니다.

Notion

실험결과 Parameter와 Recall@10을 쉽고 명확하게 공유하기 위해 노션을 활용했습니다.

1-1-4. 개발환경(AI Stage Sever)

- OS: Ubuntu 18.04.5 LTS
- GPU: Tesla V100-SXM2-32GB

▼ 1-2. 프로젝트 팀 구성 및 역할

- 김성연_T4040(팀장)

⇒ EDA, 전반적인 팀 프로젝트 타임라인 잡기, 베이스라인 주석 달기, Rule base 모델 만들기, 베이스라인 개선 후 적용

- 배성재_T4097(팀원)

⇒ AutoEncoder 적용, glocal-K 모델 적용, EASE 적용 및 파라미터 튜닝, 베이스라인 주석 달기

- 양승훈_T4122(팀원)

⇒ EDA, 베이스라인 주석 달기, 베이스라인 개선 후 적용, MF모델 적용

- 조수연_T4208(팀원)

⇒ EASE 적용, lightFM 적용

- 홍재형_T4233(팀원)

⇒ AutoEncoder 적용, 베이스라인 주석 달기, 미션 제출코드 작성

- 황선태_T4236(팀원)

⇒ 미션 주석 달기, 베이스라인 주석 달기,

▼ 1-3. 프로젝트 수행 절차 소개

1-3-1. EDA

- 같은 유저가 같은 영화를 시청한 기록은 단 한 개만 존재합니다.

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

```
0
```

유저가 본 영화는 제외하고 추천해야 합니다.

- 가장 적은 시청 기록을 가진 영화는 16명이 시청했고 가장 적은 영화 시청 기록을 가진 유저는 16개의 영화를 시청했습니다.

```
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
```

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

✓ 0.1s

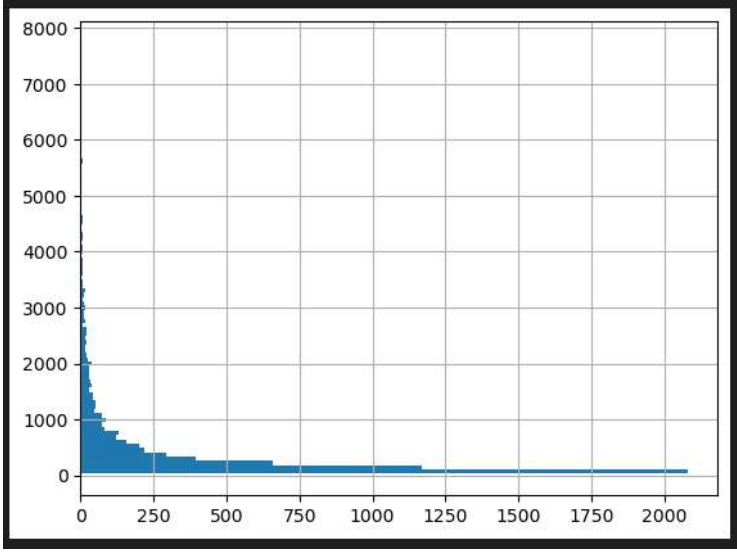
```
2571      19699
2959      18437
296       18202
318       18168
356       17339
          ...
117881       38
126          36
4241         34
2555         34
51372        27
```

극단적인 Cold Start 문제는 발생하지 않습니다. Context 기반 추천을 굳이 하지 않아도 될 것 같습니다.

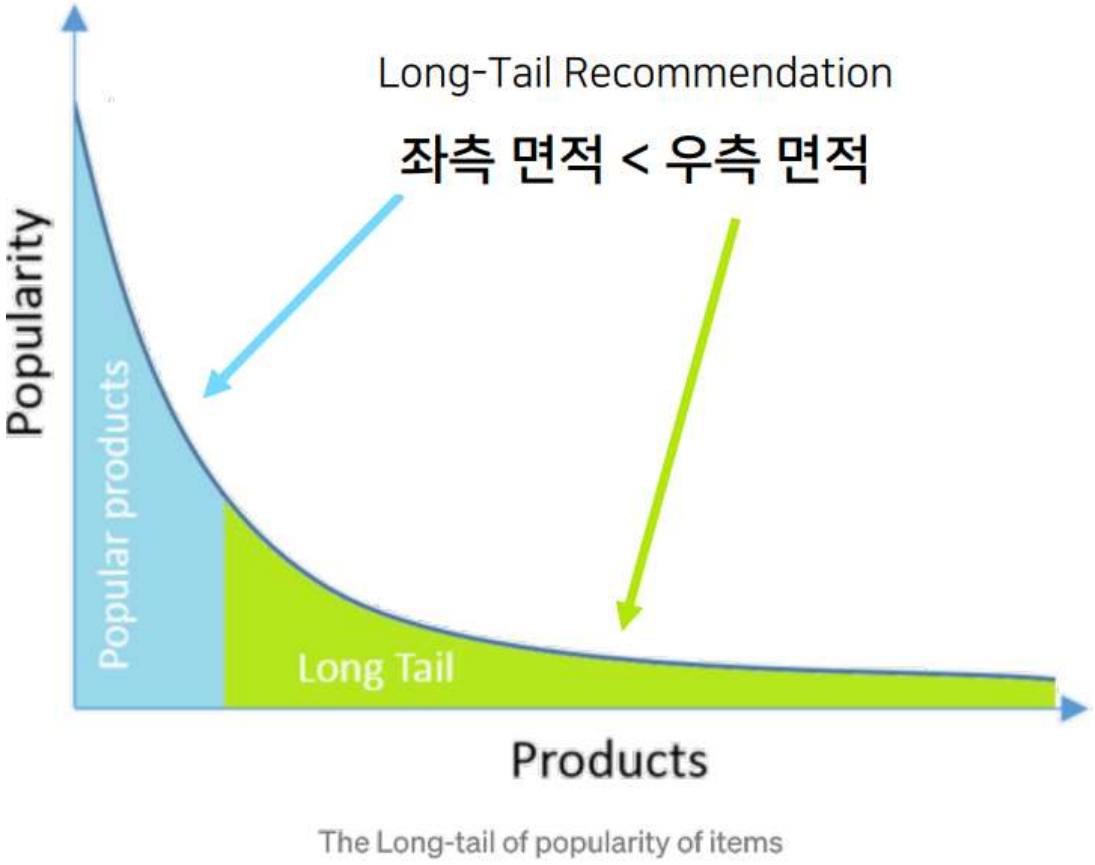
- 인기 있는 영화의 소비가 치중되어 있는 전형적인 Long-Tail 구조 입니다

```
train['item'].value_counts()[100:].hist(orientation='horizontal', bins=100)
```

```
<AxesSubplot: >
```



아래는 일반적인 Long-Tail 그래프 입니다.



인기 없는 영화를 인위적으로 잘라낸 데이터 셋으로 추측됩니다. Long-Tail 문제가 꽤 심각한데 인기 있는 영화 소비가 생각보다 큽니다.

1-3-2. Rule-Base 모델로 기준점 잡기

모든 사람에게 가장 인기가 많은 영화 Top10을 추천하는 Rule-Base 모델입니다.

이 때 유저 단위로 이미 본 영화가 존재한다면 다음 순위 영화를 추천합니다.

```
for user in tqdm(user_lst):
    cnt = 0 # 반복문마다 새는 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
```

리더보드 기준 Recall@10 값이 0.0673을 기록했습니다.

이 값을 기준점으로 잡고 이 값보다 낮은 모델은 개인화 추천으로써의 기능을 하지 못하는 모델이라고 판단하였습니다.

1-3-3. SASRec 모델 중심의 프로젝트 진행과정

SASRec 모델 데이터 인코딩 방식 변경

```
train['item'].nunique(), train['item'].min(), train['item'].max()
✓ 0.8s
(6807, 1, 119145)
```

6807개 영화 번호는 1번부터 119145번까지 존재합니다.

```
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)
```

안 본 영화를 랜덤하게 뽑는 neg_sample은 random.randint 함수를 사용해 1~119145 사이 숫자를 뽑는데요. 실제 영화 번호가 아닌 값도 neg_sample을 통해 뽑힐 수 있습니다.

이 문제를 개선하기 위해 영화 번호를 1부터 6807번까지 인덱싱하였습니다.

리더보드 기준 Recall@10 값이 0.0884에서 0.1014까지 눈에 띄는 성능 개선이 있었습니다.

max_len 변수에 대해서

max_len 변수는 다음 영화가 무엇일지 예측하는데 사용하는 영화 시청 기록 길이입니다.

값을 늘린다면 더 많은 영화 기록 정보를 이용해 다음 영화를 예측하게 된다는 장점이 있으나 너무 많은 기록이 영화 추천에 부정적일 수도 있다는 의견도 있었습니다.

베이스라인 defalut 값 50에서 20으로 줄였을 때 리더보드 기준 Recall@10 값이 0.1014에서 0.0982로 줄었습니다.
베이스라인 defalut 값 50에서 100으로 늘렸을 때 리더보드 기준 Recall@10 값이 0.1014에서 0.1064로 올랐습니다.
많은 영화 기록 정보가 다음 영화를 예측하는데 중요하다는 것을 추후 하이퍼파라미터 튜닝에서 고려하였습니다.

데이터 증강

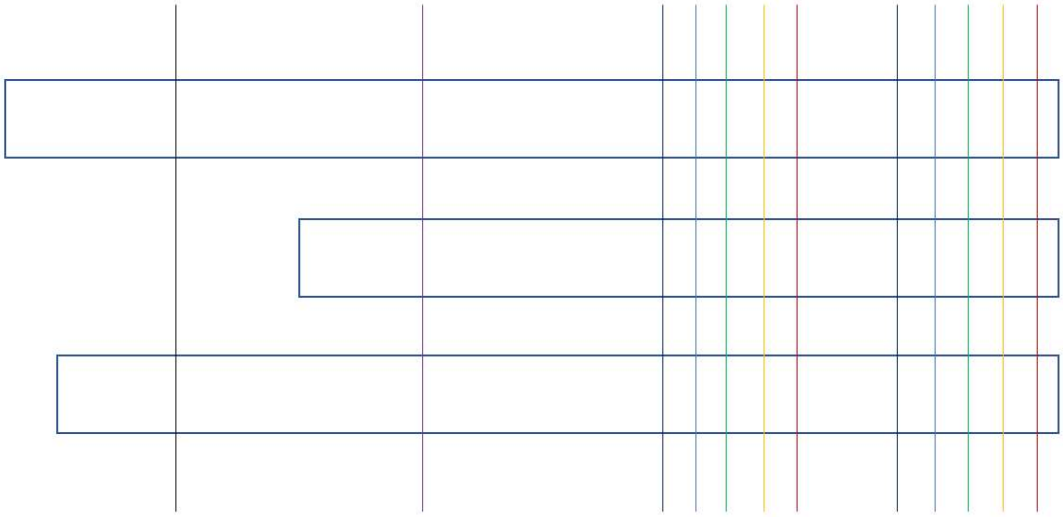
기존 베이스라인 모델에서는 max_len 이하 영화 시청 기록은 모두 자릅니다.

max_len 보다 긴 영화 시청 기록이 있는 데이터는 손실이 날 수 있다고 생각했고 이전 DKT 프로젝트에서 진행했던 데이터 증강을 떠올리게 되었습니다.

단순히 겹치지 않는 방식으로 증강을 수행한 결과 리더보드 기준 Recall@10 값이 0.1064에서 0.1067로 성능이 유의미하게 증가했다고 보기 어려웠습니다.

이 원인으로 다음 영화라는 최신 데이터를 맞춰야 하는 모델에서 과거 데이터가 모델 학습에 노이즈가 될 수도 있다고 판단했습니다.

그래서 현재 데이터에 가중치를 조금 더 주는 다음과 같은 방식을 시도했습니다.



(같은 색깔이 하나의 데이터 셋을 의미합니다.)

최신 데이터에 대해서 5번까지 슬라이싱-윈도우을 사용하여 데이터 증강을 합니다. 이후 단순히 겹치지 않는 방식으로 데이터를 증강하였습니다.

위 방식을 사용하면 epoch당 최신 데이터를 더 많은 비중으로 접하게 됩니다.

리더보드 기준 Recall@10 값이 0.1064에서 0.1209로 성능이 유의미하게 증가했습니다.

시퀀셜 모델이 아닌 모델들

MF, LightFM, DeepFM, AutoEncoder, Multi-DAE, Multi-VAE, EASE 여러 모델을 시도했습니다.

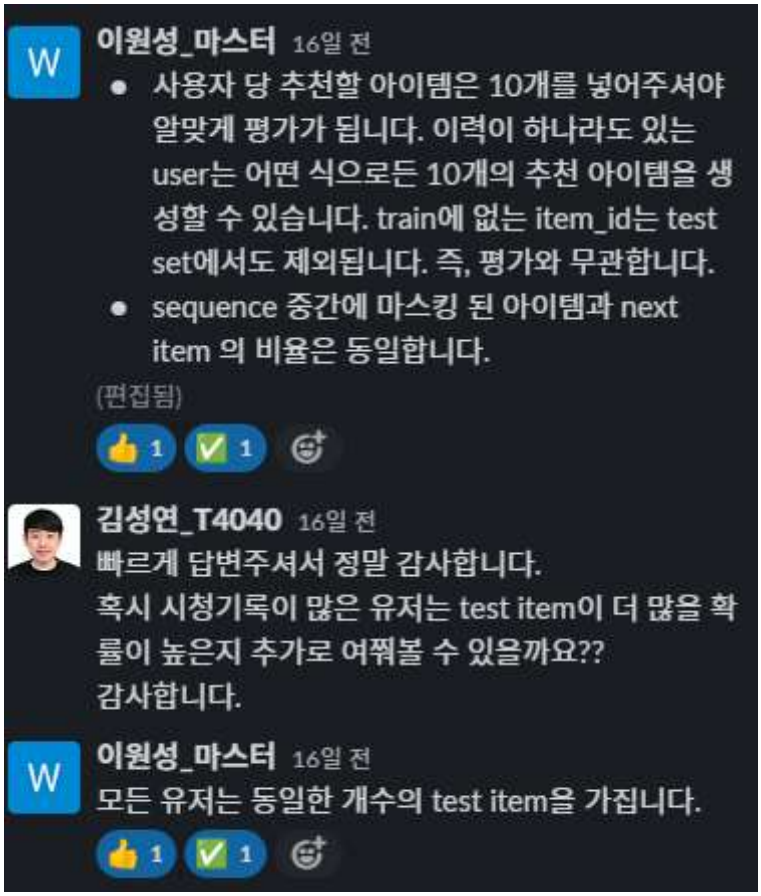
그러나 EASE를 제외하고는 인기도 기반 Rule_base 모델 0.0673을 넘지 못했습니다.

후술하겠지만 딥러닝 모델은 하이퍼파라미터에 상당히 민감하기 때문에 튜닝을 시도했다면 조금 더 개선할 여지가 있었을 것 같습니다.

간단한 구조인 EASE 모델 사용 시 리더보드 기준 Recall@10 값 0.1599로 압도적인 성능을 기록했습니다.

valid set 구성

SASRec 모델을 튜닝하기 전 test set와 유사한 좋은 valid set을 구축하는 것이 먼저입니다.



test set에 대해 마스터님께 질의한 결과 모든 유저는 동일한 개수의 test_item을 가지고 마스킹 된 아이템과 next_item의 비율은 동일합니다.

위 구성에 맞게 추천을 하기 위해선 마스킹된 영화를 맞추는 모델과 다음 영화를 맞추는 모델을 결합한 앙상블을 해야합니다.

우리 상황에서는 단일모델 성능이 어느정도 보장된 모델로 시퀀셜한 모델인 SASRec와 시퀀셜하지 않은 모델 EASE을 앙상블 하는 것이 바람직합니다.

그렇다면 SASRec 모델이 다음 영화를 맞추는데 최적화하는 것이 앙상블 했을 때 큰 효과가 난다고 판단됩니다.

이러한 이유로 유저 기준 마지막 영화만을 SASRec 모델 valid set으로 구성했습니다.

추가로 유저 단위로 본 영화는 네거티브 샘플링에서 제외되는데 valid set에 있는 영화 또한 제외됩니다.

하지만 test set은 네거티브 샘플링에서 제외되지 않기 때문에 에폭이 늘어날 수록 valid recall은 증가하나 test recall은 감소하게 됩니다.

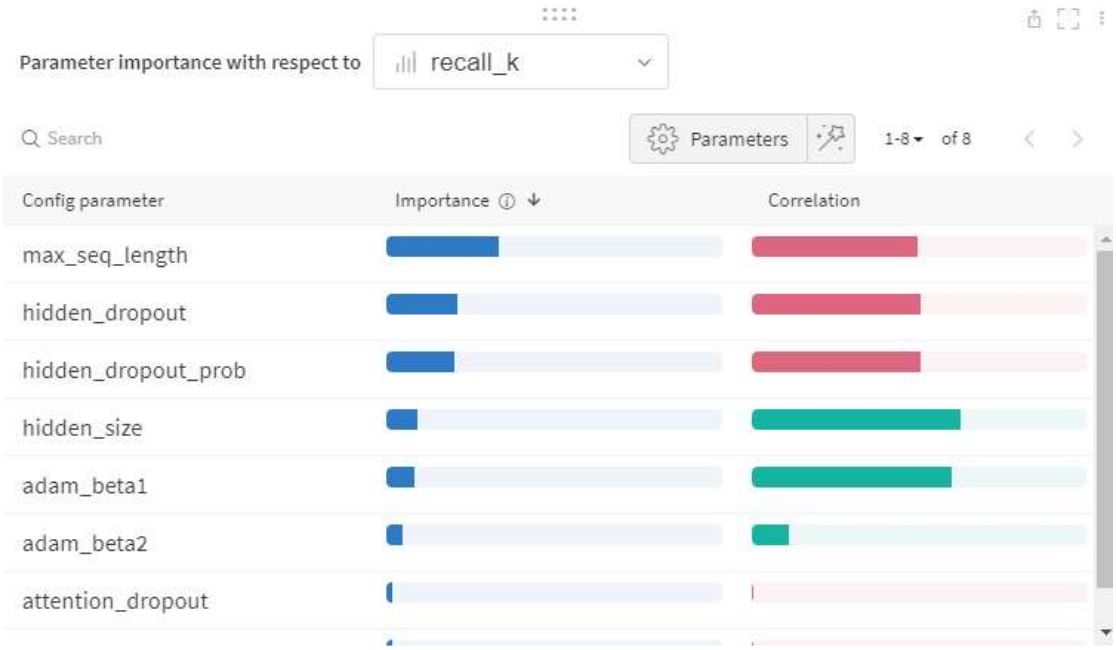
```
user_set = set(user_item_list[:-1]) # valid에 있는 데이터는 네거티브에 안걸림.
# input_ids 길이만큼 target_neg에 negative samples 생성
for _ in input_ids:
    target_neg.append(neg_sample(user_set, self.args.item_size))
```

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

위 방식이 실제로 앙상블 효과를 극대화하였습니다.

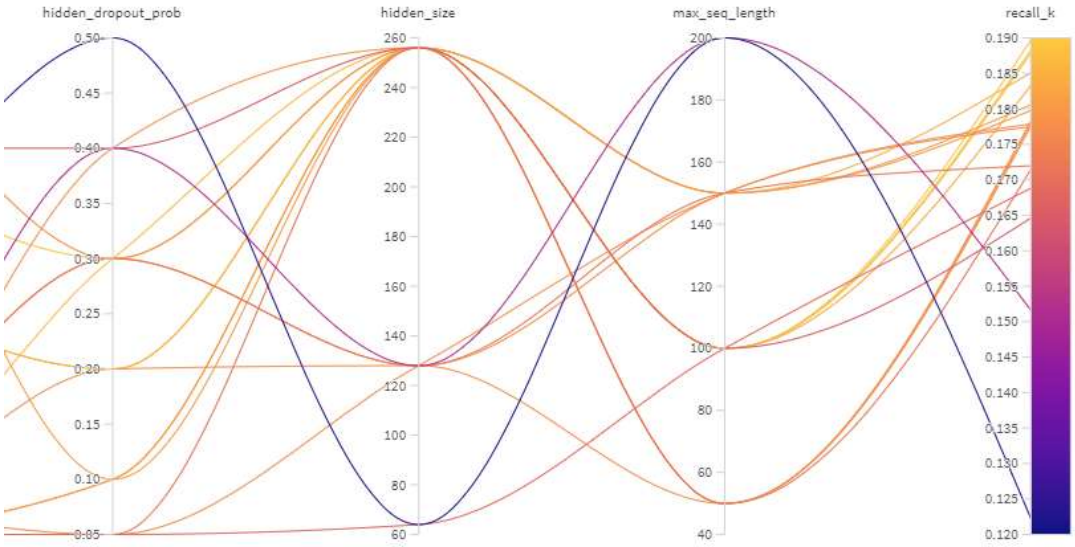
하이퍼파라미터 튜닝을 통한 모델 고도화

Wandb sweep을 이용하여 하이퍼 파라미터 튜닝을 진행하였습니다.



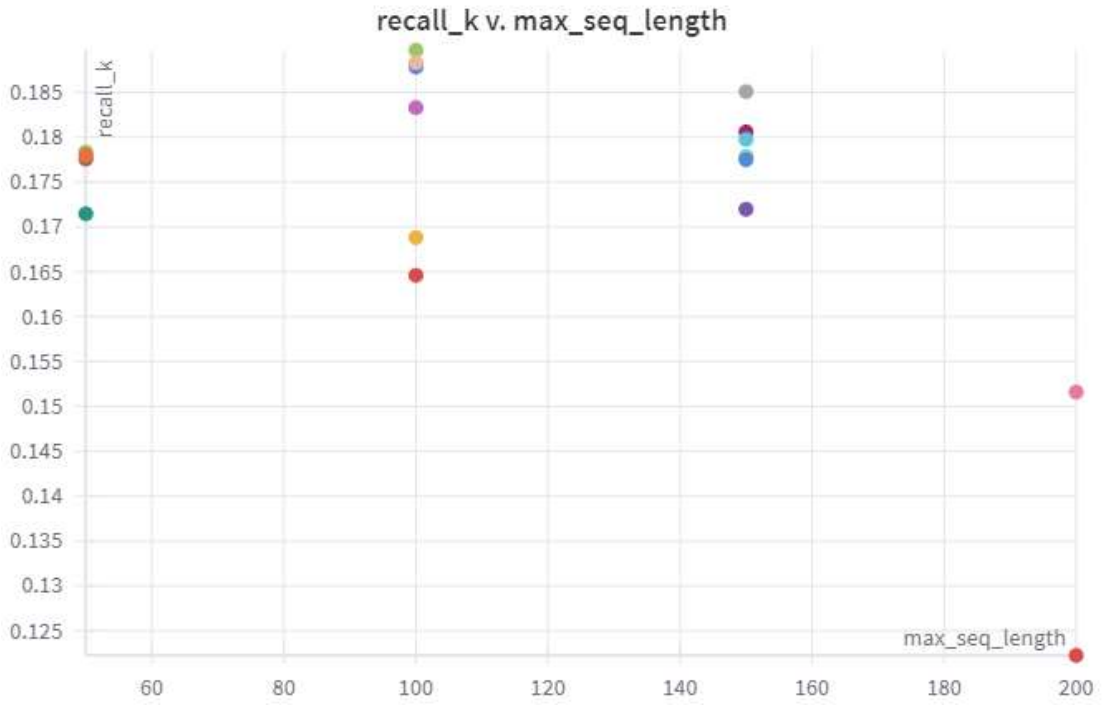
Importance은 변수 중요도를 나타내고 Correlation은 값이 양수라면 해당 변수가 커진다면 Recall@k가 커짐을 의미합니다.

이를 이용하여 중요 하이퍼파라미터를 찾고 Recall@k를 크게 하기 위해 각 하이퍼파라미터들을 증가 혹은 감소해야겠다는 직관을 얻게됩니다.



전반적인 경향성을 알 수 있는 표 입니다. Recall_k가 큰 값은 금색, 작은 값은 남색에 가깝습니다.

위 자료를 보면 hidden_size 256에 금색 선이 몰려있고 max_seq_length 150에 금색 선이 몰려있는 것이 눈으로 확인되네요.



위와 같이 한 하이퍼파라미터와 Recall@k 간 산점도를 그려 훨씬 직관적으로 해당 하이퍼파라미터의 영향도를 살펴볼 수 도 있습니다.

주요 하이퍼파라미터 변경점은 다음과 같습니다.

- 옵티마이저 adam 내 betas defalut 값이 (0.9, 0.999)을 (0.7, 0.9999)로 변화시켰습니다.
- drop_out 비율을 (0.5, 0.5)에서 (0.2, 0.3)로 변화시켰습니다.
- max_len 값을 50에서 300으로, hidden_size을 64에서 256으로 늘렸습니다.

여러 시도를 통해 valid recall은 다음 영화만 고려하기 때문에 리더보드 점수와 조금 차이가 있지만 **경향성은 일치**하는 것을 확인했습니다.

추가로 다음과 같은 이유로 pre_train은 사용하지 않았습니다.

- Wandb를 이용한 하이퍼파라미터 튜닝이 까다롭고 시간소요도 심합니다.
- cold-start 유저가 없기 때문에 장르라는 Context 기반 추천을 할 이유가 떨어집니다.
- 여러 번 실험한 결과 pre_train을 진행한 경우 에폭 초반 loss는 확실히 작지만 에폭이 진행될 수록 pre_train을 진행하지 않는 모델에 따라잡히는 모습입니다.

단일 모델 리더보드 기준 Recall@10 값이 0.1209에서 0.1324로 큰 성능 개선을 보였습니다.

SASRec(0.1324)+ EASE(0.1599) 5:5 앙상블시 public기준 0.1816을 기록했습니다.

앙상블

앞서 언급한대로 다음 시청 영화를 예측 한 SASRec 모델과 마스킹 된 영화를 예측한 EASE모델을 앙상블하였습니다.

이 때 두 모델에서 중복으로 추천한 영화는 강력히 추천해야하는 영화로 판단하고 우선순위를 가장 위로 두었습니다.

다음으로 단일 모델 성능이 EASE가 우수했기 때문에 EASE 모델 1순위 영화를 먼저 추천하고 SASRec 모델 1순위 영화를 다음으로 추천하는 방식으로 진행했습니다.

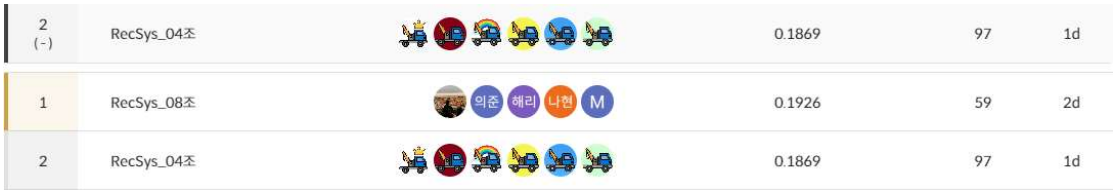
또 시드 값을 바꿔서 SASRec 모델을 돌리면 네거티브 샘플링이 다르게 되어 Top@10 추천 리스트가 계속 바뀌게 됩니다.

시드 값을 바꿔서 나온 여러 개의 아웃풋을 앙상블 하면 훨씬 더 로버스트한 모델이 나옵니다. 6개의 추천 리스트를 앙상블 한 결과 리더보드 기준 Recall@10 값이 0.1816에서 0.1869까지 증가하였습니다.

어떤 영화가 네거티브 샘플링이 됬는지가 모델에 영향력이 큰 점이 앙상블을 했을 때 로버스트한 결과를 가져오는 이유가 되는 것 같습니다.

대회 결과

- public 2등



- private 2등

▼ 1-4. 자체 평가 의견

잘했던 점

- 단순히 대회에서 순위를 올리는 것보단 더 깊이 이해를 하면서 실력을 증진하는 부분에 집중했습니다.
- 베이스라인을 개선하거나, 논문을 읽고 모델을 구현해보는 등 기본에 충실한 방법으로 성과를 냈습니다.
- 페어 코딩으로 서로 자세히 피드백을 나누며 코드를 이해하고 개선하는 방식을 이용해보았습니다.
- 이전 대회에서의 경험을 바탕으로 보다 수월하게 Git Branch 전략을 사용했습니다.

시도 했으나 잘 되지 않았던 것들

- 이번 대회에서는 강의와 미션을 깊이 이해하고 적용해보는 것이 목표였는데, 그것들로 성능 개선을 하지는 못했습니다.
- 페어 코딩 등의 활동에 더 몰입하려다 보니 모든 팀원들 간에 개인 상황 공유가 원활하게 되지 않았습니다.
- 기본에 충실하는 쪽으로 목표를 기울이다 보니, Movie Recommendation은 외부에 참고할 만한 자료가 매우 많은데도 불구하고 여러가지를 탐색하지 못했습니다.

아쉬웠던 점들

- 팀원 간의 개인 상황 공유는 지난 대회에서도 아쉬운 점으로 삼았던 부분인데, 이번 대회에서도 크게 개선하지 못했습니다.
- 대회가 반복되다 보니 체력적인 문제로 깊게 몰입하지 못했습니다.

▼ Part 2. 개인 회고

▼ 김성연

- **이번 프로젝트에서 나의 목표는 무엇이었는가?**

리더보드 점수에 감정소모하지 않으나 완성도 있고 어느정도 설명력 있는 딥러닝 모델을 만들고 싶었습니다. 그동안 딥러닝 모델을 이용해 성과를 내지 못했기 때문입니다.

팀적으로는 진행한 작업들을 모두가 이해하는 프로젝트를 하고 싶었습니다. 소통하려는 노력과 읽기 쉬운 코드를 작성하려는 노력을 했습니다.