Movie Recommendation Wrap-up Report - Suggestify

1. 프로젝트 소개

1. 1. 프로젝트 개요

- 본 프로젝트는 'Movie Recommendation'으로, 사용자의 영화 시청 이력 데이터를 토대로 사용자가 다음에 시청할 영화 및 좋아할 영화를 예측하는 기존 대회들과는 다르게 사용자의 영화 시청 이력에서 10개를 랜덤으로 제거한 후 제거된 시청 이력이 무엇인지를 맞춰야 하는 대회이다.
- 모든 사용자에게 10개의 영화를 추천하며, 이때 추천 리스트의 정확성(Recall@10)을 평가 기준으로 삼는다.
- 평가를 위한 정답(ground-truth) 데이터는 Sequential Recommendation 시나리오를 바탕으로 사용자의 Time-Ordered Sequence에서 일부 Item이 누락(dropout)된 상황을 상정한다

1.2. 데이터 구조

- 본 대회는 추천시스템 연구 및 학습 용도로 가장 널리 사용되는 MovieLens 데이터를 경진대회 용도로 재구성하여 사용하였다.
- 유저와 아이템의 상호작용 데이터
  - TimeStamp를 포함한 유저(31,360명)와 아이템(6,807개) 간의 Implicit Feedback 5,154,471개로 구성되어있다.
  - Sparsity Ratio는 약 97.58%, 추천시스템 도메인 내에서는 Dense한 편이다.
- 아이템의 메타데이터
  - 장르, 제목, 개봉 연도, 감독, 작가로 구성되어있다.
- 평가 데이터
  - 훈련 데이터에 있는 전체 유저에 대해서 각각 10개씩 추천한다. (총 313,600행)

1.3. 평가지표

- 각 유저에 대해 10개씩 추천된 영화는 $Recall@10$을 사용하여 평가되며, 수식은 다음과 같다.

$$Recall@10 = \frac{1}{|U|}\sum_{u \in U}\frac{|\{i \in I_u | rank_u(i) \leq 10\}|}{k_u}$$

- $rank_u(i)$는 사용자 u를 위한 추천 결과에서 아이템i가 랭크된 위치를 의미한다.

1.4. 활용장비 및 재료

- 개발환경
  - OS: Linux-5.4.0-99-generic-x86_64-with-glibc2.31
  - GPU: Tesla V100-SXM2-32GB * 1

    - CPU cores: 8
- 협업 툴: github, wandb, notion, slack

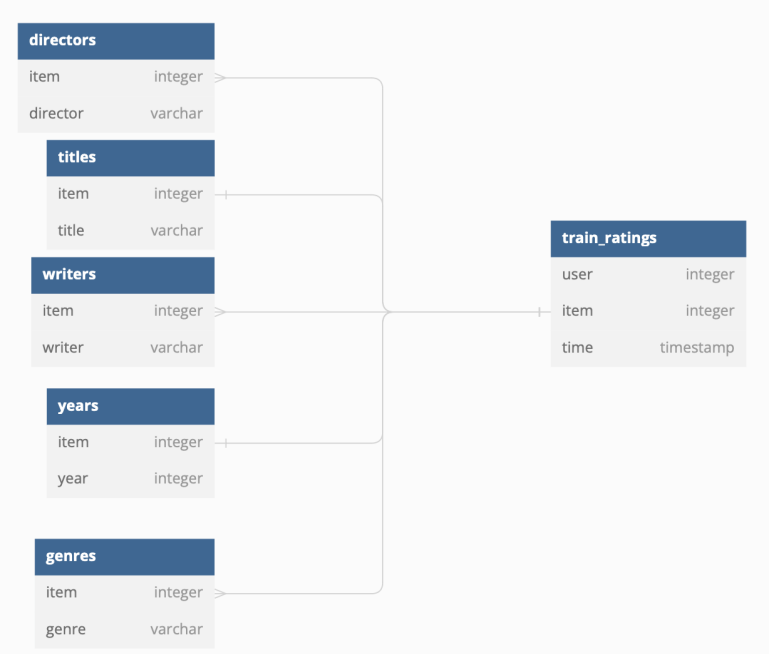
1.5. 프로젝트 구조 및 사용 데이터셋의 구조도(연관도)

- 프로젝트 구조

```
📦level2-movierecommendation-recsys-03-main
 ┗ 📂code
    ┣ 📂EASER
   ┣ 📂EDA
   ┣ 📂bert4rec
   ┣ 📂custom
   ┣ 📂multi
   ┣ 📂recbole
   ┃ ┗ 📂configs - .yaml
   ┃    ┣ ADMMSLIM, DIFFRec, EASE, deepfM
   ┃    ┣ fm, lightgcn, ract, recvae,
   ┃    ┗ seq, slim
   ┣ 📂s3rec
   ┗ 📂voting
```

- 데이터셋의 구조도

```
📦level2-movierecommendation-recsys-03-main
 ┗ 📂train
    ┣ 📜Ml_item2attributes.json
    ┣ 📜directors.tsv
    ┣ 📜genres.tsv
    ┣ 📜titles.tsv
    ┣ 📜train_ratings.csv
    ┣ 📜writers.tsv
    ┗ 📜years.tsv
```



2. 프로젝트 팀 구성 및 역할

2.1 프로젝트 팀 구성

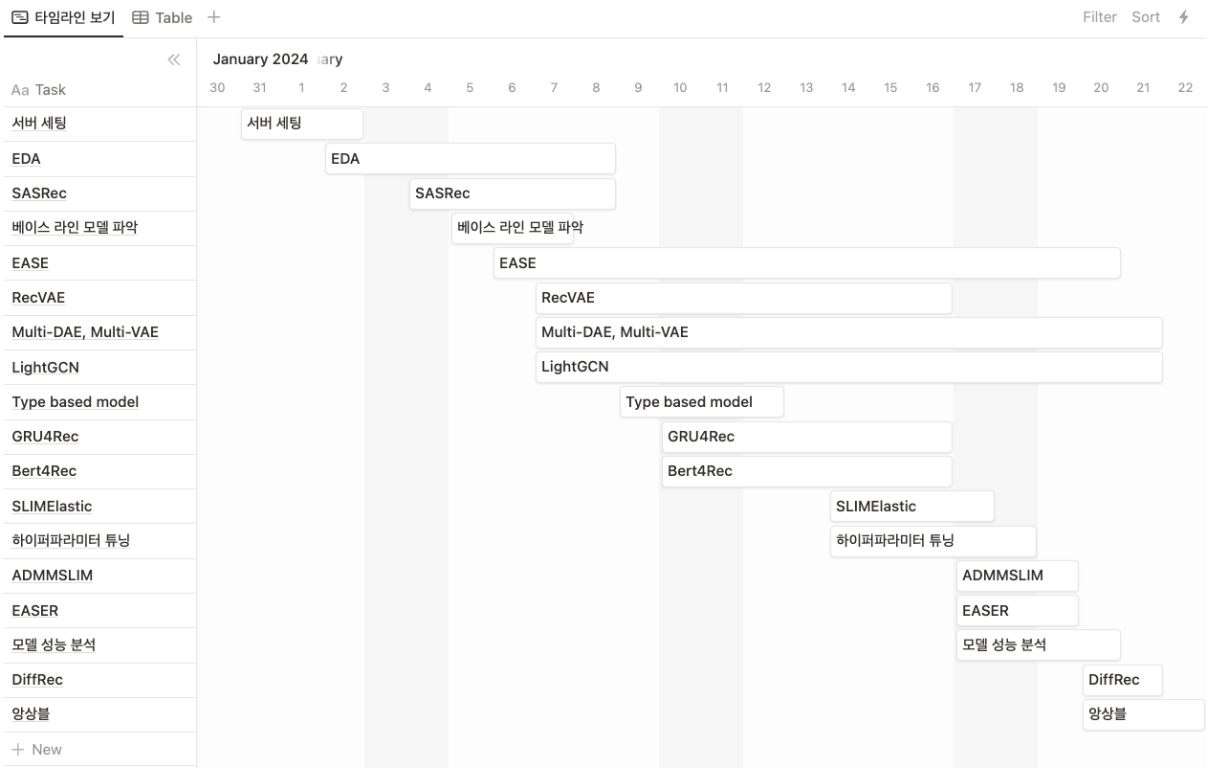
- 팀명 : Suggestify
- 팀 인원 : 6
- 팀 구성 : 김수빈, 박시우, 백승빈, 이재권, 이진민, 장재원

2.2 역할

이름	역할
김수빈	EDA, 모델 선정 및 튜닝, EASER, SASRec, Bert4Rec 실험 수행
박시우	EDA, s3rec baseline 정리, recbole baseline 구축, RecVAE, ADMMSLIM, soft voting 앙상블 구현
백승빈	EDA, EASE, SLIMElastic, DiffRec 모델 실험 및 튜닝, hard voting기반 앙상블 구현
이재권	EDA, LightGCN 모델 실험 및 튜닝
이진민	EDA, Multi-DAE, Multi-VAE 코드 모듈화 및 실험
장재원	EDA, Sequential models, Type based model, 모델들 성능 비교 및 분석

3. 프로젝트 수행 절차 및 방법

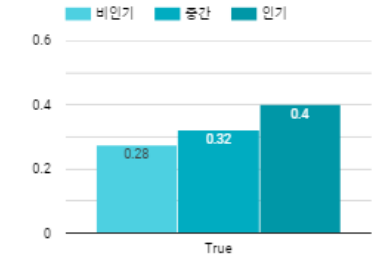
프로젝트 진행 타임라인

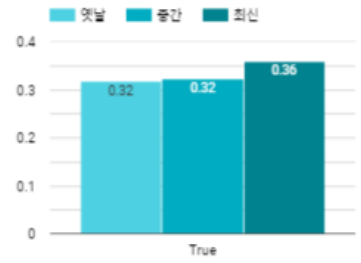


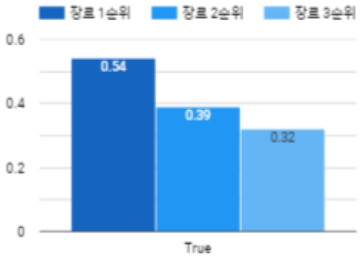
4. 프로젝트 수행 결과

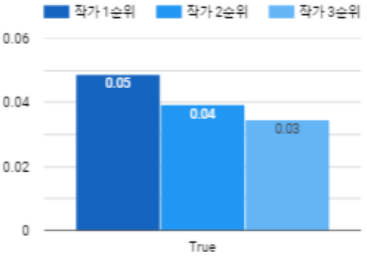
4.1. 탐색적 분석 및 전처리 (EDA & Pre-processing)

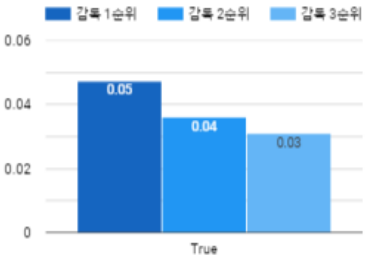
- 각 User별 선호하는 Item을 파악하고 이를 바탕으로 7가지의 type을 형성하였다. (ex: 인기있는 영화 선호, 최신영화 선호, 특정 장르의 영화 선호, 영화시청이력이 많은 그룹) 유저들은 전체적으로 인기있는 영화, 최신영화를 좋아하며, 각자 특정 분류(장르, 작가, 감독)의 영화를 선호하는 것으로 보인다. 특히, 인기있는 영화와 선호하는 장르의 영화는 평균적으로 선호하는 경향이 컸다.











- 시청영화수가 늘어감에 따라 취향이 변하는 현상이 식별되며, 각 type의 성향에 맞는 영화의 추천이 이루어져야 한다. (ex: 영화를 많이 보지 않은 사람에게는 최신 및 유명 영화가 추천되어야 한다.)

	시청영화수	비인기	중간	인기	옛날	중간	최신	선호장르	선호작가	선호감독
All users	164	0.28	0.32	0.4	0.32	0.32	0.36	0.54	0.05	0.05
Type novice	47	0.27	0.29	0.45	0.32	0.32	0.36	0.57	0.08	0.07
Type writer	68	0.24	0.28	0.47	0.33	0.33	0.34	0.56	0.1	0.08
Type director	68	0.23	0.28	0.49	0.36	0.32	0.32	0.56	0.09	0.1
Type popular	73	0.07	0.17	0.76	0.27	0.35	0.38	0.49	0.07	0.07
Type new	86	0.35	0.34	0.3	0.08	0.14	0.78	0.57	0.05	0.04
Type genre	120	0.39	0.31	0.3	0.32	0.31	0.37	0.76	0.05	0.05
Type mania	479	0.42	0.34	0.23	0.35	0.32	0.33	0.52	0.02	0.02

- EDA 결과 링크 : https://lookerstudio.google.com/s/vmuSEZsLsmU

4.2 모델 선정 및 평가

4.2.1 Type based model

- 모델 선정
  - 데이터 탐색 과정에서, 사용자의 특성을 바탕으로 여러 그룹으로 분류했다. 이를 통해, 각 사용자 그룹의 선호에 맞는 영화 추천이 성능 향상에 중요하다는 것을 파악했다. (ex: 영화를 많이 보지 않은 사람에게는 최신 및 유명 영화가 추천되어야 한다) 이러한 발견에 기초하여, 규칙 기반의 모델을 개발하여 각 사용자 유형에 적합한 영화 추천의 효과를 검증했다.
- 결과 분석
  - 인기 있는 아이템을 모든 사용자에게 일률적으로 추천하는 접근법이나, 사용자별 선호 장르에 기반한 추천보다, 사용자의 type을 고려하여 개별적으로 추천하는 방식이 더 나은 결과를 보였다. Type 기반 모델은

인기 영화를 선호하는 type에게는 인기 있는 영화를, 특정 장르를 선호하는 type에게는 해당 장르의 영화를 추천함으로써 더 개인화된 추천을 제공했다.

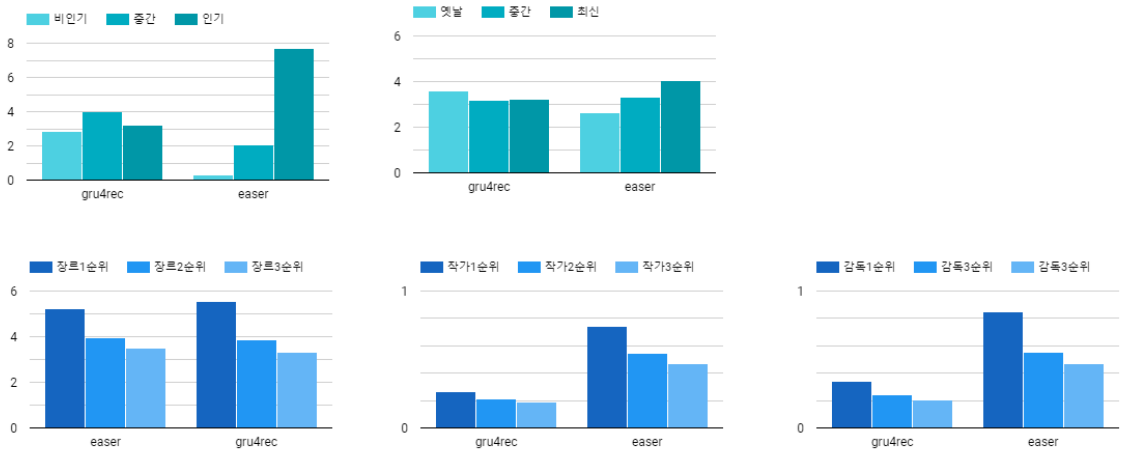
	Public Recall@10	Private Recall@10
Popular item rule based model	0.0673	0.0671
Genre rule based model	0.0619	0.0626
Type based model	0.0687	0.0696

4.2.2 Sequential models

- 모델 선정
  - Ground-truth 데이터는 특정 시점 이후의 데이터를 포함한다. 이는 사용자가 마지막 시점 이후에 시청할 영화를 예측해야 함을 의미합니다. 사용자의 선호는 시간이 지남에 따라 변화하기 때문에, 시간 순서를 고려할 수 있는 Sequential 모델이 더 나은 성능을 보일 것으로 예상했다.
- 결과 분석
  - 주요 Sequential 모델 세 가지를 비교한 결과, GRU4Rec 모델이 Public 데이터셋 기준으로 가장 우수한 성능을 보였다.

	Public Recall@10	Private Recall@10
GRU4Rec	0.0970	0.0809
SASRec	0.0884	0.0833
S3Rec(Pretrained)	0.0829	0.0743
BERT4Rec	0.0687	0.0676

  - 이번 대회의 Ground-truth 데이터는 사용자의 시간 순서에 따른 시퀀스 뿐만 아니라 누락된 아이템도 포함하고 있다. Sequential 모델은 주로 마지막 시청 시간 이후의 영화를 예측하는 데 특화되어 있기 때문에, 이러한 시나리오에서는 성능이 다소 떨어지는 것으로 판단된다.
  - 높은 평가 점수를 받으려면 인기 있는 최신 영화들이 주로 추천되어야 한다. 하지만 Sequential 모델은 다양한 영화를 추천하는 경향이 있어, 이번 대회에서 성능이 다소 낮게 나온 원인으로 판단했다.



4.2.3 LightGCN

- 모델 선정

  주어진 MovieLens 데이터는 시간 정보가 포함된 sequence 데이터이지만 마지막 아이템만 예측하는 것이 아니라 랜덤하게 샘플링한 아이템도 예측해야 하기 때문에 static 데이터로 가정하고 문제를 풀 수 있다. 또한, 충분한 양의 interaction 데이터가 존재하기 때문에 Collaborative Filtering 기반 모델로 좋은 추천이 가능하다고 판단하였다.

  LightGCN은 NGCF에서 feature transformation과 nonlinear activation을 제거하여 학습의 효율성과 성능을 높인 Collaborative Filtering 기반 그래프 모델이다. LightGCN은 사용자와 아이템 간의 복잡한 관계를 그래프 구조를 통해 학습할 수 있기 때문에 좋은 추천 성능을 보인다. 이 실험에서는 RecBole 라이브러리를 기반으로 LightGCN 모델을 구현하고 영화 추천에 대한 실험을 진행하였다.

- Hyperparameter 실험

  LightGCN의 Hyperparameter인 n_layers와 hidden_dim에 대한 최적의 값을 찾기 위해 실험을 진행하였다. n_layers는 그래프에서 정보가 전파되는 층의 개수를 나타내고, hidden_dim은 사용자와 아이템의 임베딩 크기를 나타낸다. Recall@10을 기준으로 모델의 학습 성능을 확인하였고, 10 epoch 동안 Recall@10이 개선되지 않으면 early stopping이 실행되도록 설계하였다.

	Valid Recall@10	Test Recall@10	Public Recall@10	Private Recall@10	Epoch
n_layers=1 hidden_dim=64	0.1894	0.1891	0.1254	0.1277	331
n_layers=2 hidden_dim=64	0.1855	0.1850	0.1236	0.1272	442
n_layers=1 hidden_dim=128	0.1945	0.1937	0.1302	0.1316	268

  n_layers의 경우에는, n_layers를 1에서 2로 증가시켰을때 모델 성능이 하락하였다. 이는 정보 전파 층이 늘어나면서 연관이 적은 간접적인 정보도 포함됨에 따라 직접적인 상호 작용 정보의 영향력이 감소하고 성능이 저하된 것으로 판단된다.

  hidden_dim의 경우에는, hidden_dim을 64에서 128로 증가시켰을 때 모델 성능이 상승하였다. 이는 모델이 학습해야 하는 사용자와 아이템 간의 상호작용에 대한 정보량이 많기 때문에 임베딩 차원을 증가시킴으로써 성능이 향상된 것으로 판단된다.

  따라서, n_layers = 1, hidden_dim = 128인 LightGCN 모델을 채택하였다.

- Test 결과

	Public Recall@10	Private Recall@10
LightGCN	0.1302	0.1316

4.2.4 DiffRec

- 모델 선정

  시도하는 모델들이 대부분 SLIM과 Autoencoder에 치중된 것 같아 CV의 diffusion model을 RecSys에 적용한 DiffRec 모델을 시도해보았다. 데이터의 노이즈를 제거하는데 장점을 가지는 만큼 데이터 내의 불필요한 인터렉션들을 무시하고 좋은 성능을 낼 수 있을 것이라고 기대했다.

- Hyperparameter 실험

  DiffRec 모델은 시간 부족으로 모든 파라미터에 대해 실험을 진행하지 못했다. Timestamp에 대한 중요성이 상대적으로 떨어진다는 판단 하에 시간적 정보를 사용하는 T-DiffRec 대신 noise generating에 관한 파라미

터들 위주로 실험을 진행하였다. 그 결과 noise_min=0, noise_max=0.005, noise_scale=0.01, reweight = True 일 때 더욱 좋은 성능을 내는 것을 알 수 있었다.

- Test 결과

	Public Recall@10	Private Recall@10
DiffRec	0.1413	0.1431

4.2.5 RecVAE

- 모델 선정

  movie lens dataset에서 input과 output의 분리가 어떤 시점 기준이 아니고 "input을 봤다고 할 때 봤거나 볼 가능성이 높은 output은 뭘까?"였기 때문에 시계열 기반의 모델은 어울리지 않을 것이라고 판단했다. 이러한 경우에는 input으로부터의 특징을 뽑아 유사하면서도 새로운 output을 만들어낼 수 있는 생성형 모델의 특징을 가진 autoencoder 기반 모델이 잘 어울릴 것 같았다.

  그 중에서도 Movie lens 20M dataset에 대해 성능 3위에 해당하는 RecVAE 모델을 사용해보고자 했다.

- Hyperparameter 실험

  train 시 batch_size의 경우 2048, positive sample과 negative sample의 경우 그 비율이 1:1인 경우가 가장 안정적이면서도 성능이 좋았다.

  이후 recbole 내에서 모델의 하이퍼 파라미터를 튜닝하기 위해 Hyperopt를 이용했고 튜닝 시간이 너무 길어지지 않도록 하기 위해 search 기법은 'random'을 선택했다.

  오히려 튜닝한 모델의 성능이 기존 모델보다 성능이 좋지 않았는데, 아마 그 이유는 튜닝 실험 횟수를 10번으로 제한하면서도 튜닝할 파라미터 옵션의 개수는 여러 개를 두어 실험 횟수가 부족한 탓이었던 것 같다.

	Private Recall@10	Epoch	Time
original model	0.1349	90	3h 30m
tuned model	0.1344	50	2h 15m

  그렇게 큰 성능 차이는 아니지만, 학습 시간 차이는 꽤 나기 때문에 실제 비즈니스에서 사용할 것이라면 tuning된 모델을 사용할 만 한 것 같다고 판단했다. 그래도 대회에서는 성능을 최대로 끌어내는 것이 목적이기 때문에 기존 모델을 택해서 사용했다.

- 모델 구조 실험

  RecVAE는 인코더는 약 5개의 층, 디코더는 1개의 층으로 이루어져 있었다. 파라미터 수도 많고 학습할 때 이전 epoch에서의 encoder의 파라미터를 load해서 계산한 값으로 학습해 나가기 때문에 모델이 많이 무겁다고 생각했다. 마침 EASE, SLIM 같은 단순한 모델이 학습이 잘되는 걸 보고 인코더의 층을 얇게 구성해봐야 겠다고 생각했다.

  그래서 기존 5개의 층에서 4개, 3개로 줄여보는 실험을 해보았다.

	Valid Recall@10	Valid NDCG@10
5 layer(original)	0.2009	0.1123
4 layer	0.2017	0.1118
3 layer	0.2009	0.1113

  validation 기준 recall@10 값이 4 layer일 때 가장 높았음을 확인할 수 있었다.

- Test 결과

	Public Recall@10	Private Recall@10
RecVAE	0.1349	0.1362

4.2.6 Multi-VAE, Multi-DAE

- 모델 선정

  이 대회의 데이터 셋에서는 timestamp가 주어지긴 했지만 원래의 sequential 데이터가 주어지는 것이 아닌 일부가 중간중간에 빠진 것이므로 sequential model들보다 다른 모델들이 더 성능이 좋을 것이라고 생각했다. 또한 interaction data 크기가 큰 편이라고 생각하여 collaborative filtering 모델을 선정했고 앞서 언급했던 데이터셋의 특징이 autoencoder 계열이 적합하다고 생각해서 Multi-VAE, Multi-DAE 모델을 선정했다.

  논문에서는 Multi-VAE가 대부분의 데이터셋에서 성능이 높으나 active한 유저가 많을 때는 Multi-DAE가 성능이 높다고 하여 주어진 데이터가 비교적 dense해서 Multi-DAE가 더 성능이 좋을 것이라고 예측했다. 그래도 두개의 모델 모두 실험해봤다.

	Public Recall@10	Private Recall@10
Multi-VAE	0.1394	0.1377
Multi-DAE	0.1390	0.1380

- side information 추가

  side information을 활용하지 않는 collaborative filtering model이어서 이 모델에 side information을 추가하면 성능이 더 올라가지 않을까라고 생각했다. 논문에서도 side information을 추가하는 실험을 언급하고 있어 해보게 되었다. side information은 one hot encoding한 후 interaction data에 concatenate 하여 실험해보았다.

  EDA 결과 genre와 director가 영화를 고르는 데에 많은 영향을 준다는 것을 알게 되어 Multi-DAE에 genre와 director 정보를 추가하여 실험해보았다.

	Public Recall@10	Private Recall@10
genre 추가	0.1423	0.1409
director 추가	0.1427	0.1413

- Hyperparameter 실험

  wandb sweep를 이용해서 bayesian optimization을 하고 manual search를 해봤지만 논문에서 사용한 hyperparameter를 그대로 사용하고 epoch를 100으로 늘린 것이 가장 성능이 좋았다.

4.2.7 EASE

- 모델 선정

  대회의 목적이 시퀀스 내에 랜덤한 위치의 아이템을 추천하는 것인 만큼 Sequential 모델보다는 CF계열 모델이 좋은 성능을 보일 것이라 예측했다. EASE는 sparse한 데이터에 강점을 가지고 있어 implicit한 데이터를 사용하는 이번 대회에서 유리할 것이라고 생각하여 선정했다.

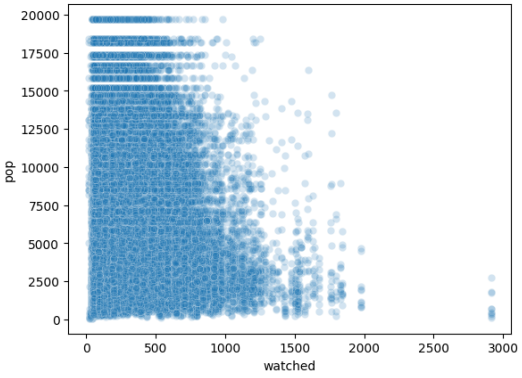
- Hyperparameter 실험

  EASE는 closed form을 가지고 있어 L2 regularization weight를 결정하는 reg_weight에 대해서 hyperparameter tuning을 진행했다. 그 결과 0.2358까지 test recall@10을 높일 수 있었다.

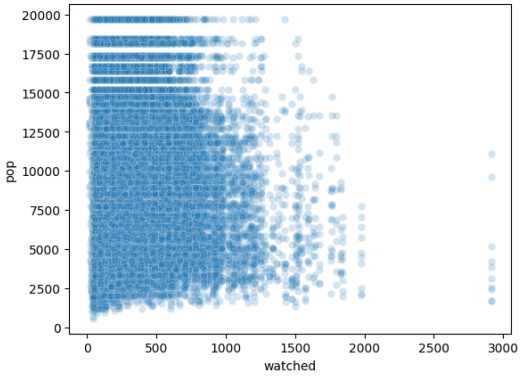
- 모델 실험
  - Mean taste similarity

EASE의 결과를 분석한 결과 관객수가 적은 영화도 활발히 추천되고 있다는 것이 발견했고 아이템의 인기도를 줄이는 시도가 결과가 좋지 않았던 것을 바탕으로 오히려 인기 있는 아이템에 대한 추천을 강화해보고자 했다.유저의 mainstreamness를 설명하는 mean taste similarity를 binary interaction 대신 user feature로 사용했다. 추후 EDA에서 영화 시청 수가 많은 유저일수록 관객수가 적은 영화를 많이 본다는 분석이 있었는데 인기 아이템을 위주로 추천을 하게 되며 그러한 유저에 대한 정확도가 떨어진 것으로 판단된다.

EASE)



Mean taste similarity)



	Recall@10	NDCG@10
Original model	0.2358	0.1378
Mean taste similarity	0.1803	0.1062

- Test 결과

	Public Recall@10	Private Recall@10
EASE	0.1566	0.1565

4.2.8 SLIMElastic

- 모델 선정

  SLIM은 Autoencoder와 유사하게 input 데이터를 복원하는 원리를 가진 linear모델로 EASE의 원형이다. SLIMElastic은 Elastic net로 정규화를 진행하여 모델의 일반화 성능을 높였다.

- Hyperparameter 실험

  Recbole에 제공하는 모델에서는 alpha, l1_ratio, positive_only, hide_item 네 가지의 Hyperparameter가 존재하지만 EASE 논문을 통해 positive weigt에 대한 규제를 없애고 self similarity를 0으로 두는 것이 모델의 성능을 향상시킨다는 것을 확인하여 alpha와 l1_ratio에 대해 greedy search를 진행했다.

	Recall@10	NDCG@10
Original model	0.2057	0.1203
Tuned model	0.2238	0.1314

- Test 결과

	Public Recall@10	Private Recall@10
SLIMElastic	0.1562	0.1562

4.2.9 ADMMSLIM

- 모델 선정

  SLIM(Elastic), EASE와 같은 closed form의 단순한 형식을 통해 0.15 대의 우수한 recall@10을 얻음을 확인한 이후, 유사하면서도 더 발전된 모델을 찾고 있었다.

  그러던 중 기존 SLIM(Elastic)에 비해 훈련 시간을 줄이면서도 ADMM(Alternating Directions Methods of Multipliers)를 이용해 더 나은 최적화를 해낼 수 있다는 모델을 발견해 선정하게 되었다.

- Hyperparameter 실험

  RecVAE와 같은 방식으로 hyperopt 라이브러리와 함께 하이퍼 파라미터 튜닝을 진행했고 마찬가지로 시간 단축을 위해 'random' 방식을 선택했다.

  valid recall@10의 최대값은 0.2314로 그때의 파라미터 값 alpha=0.5, lambda1=3.0, lambda2=500.0이었다. default setting parameter와 lambda2=200.0인 것만 달랐던 걸 보면 아마 recsys 도메인에서 유명한 데이터셋에 해당하는 만큼 그에 어느 정도 값이 맞춰져 있을 거라는 생각이 들었다. 또 public score는 defaut 세팅했을 때와 똑같았기 때문에 큰 효과는 없었음을 알 수 있었다.

- Test 결과

	Public Recall@10	Private Recall@10
ADMMSLIM	0.1524	0.1541

4.2.10 EASER

- 모델 선정

  이번 대회의 데이터들이 SLIM, EASE와 같은 모델에서 높은 성적을 거둔다는 사실을 파악한 이후, 특히 EASE 모델에서 더 나아가보기로 했다. 기존 비선형함수들의 결합으로 깊이를 더해가는 딥러닝 모델과 달리, 선형 전체 순위(Linear Full-Rank) 모델을 확장해 고차원의 인터렉션을 허용하는 방식으로 (Don't go Deeper, Go Higher) 모델의 정확도를 높여나갔다. 그리고 고차 인터렉션의 Non-Negative 제약을 제거, Positive Sample 대비 더 많은 Negative Sample을 실험에 반영할 수 있었다.

- Hyperparameter 실험

  Epoch를 제외한 다른 Hyperparameter(B, C) 에 대해서는 시간관계상 실험을 진행하지 못했다. Epoch에 대해서는 각각 40, 100, 200으로 올리면서 학습을 진행했다. B는 기존 EASE에서도 등장한 아이템과 아이템 간의 Pairwise Relation을 정의한 행렬로, 입력데이터 X와 행렬곱을 수행, 원래 X와의 유사 정도를 비교하면서 B를 학습해간다. C는 유저와 시청한 Item-Pair 간의 Triplet-Relations 을 확인할 수 있는 행렬로, EASER 모델에서 추가된 행렬이다.

- 모델 실험
  - Epoch 40, 100, 200번의 학습횟수를 두고 실험을 진행했다.
- Test 결과

	Public Recall@10	Private Recall@10
40	0.1600	0.1603
100	0.1612	0.1603
200	0.1602	0.1604

전반적으로 높은 Score를 기록하였다. 다만 100 Epoch에서 가장 높은 Public Score를 기록한 것과 달리, Private에서는 200 Epoch 대비 낮은, 상대적으로 큰 Public - Private 차이를 기록했다.

4.3 최종 모델 선정 및 결과

- 다양한 추천 아이템 성향을 가진 모델들 중, EASER 모델이 단일 모델로서 가장 우수한 성능을 보였다. 이 모델은 주로 인기 있는 아이템 위주로 추천하는 모습을 보였다.

	Public Recall@10	Private Recall@10
Popular item rule based model	0.0673	0.0671
Genre rule based model	0.0619	0.0626
Type based model	0.0687	0.0696
GRU4Rec	0.0970	0.0809
SASRec	0.0884	0.0833
S3Rec(Pretrained)	0.0829	0.0743
BERT4Rec	0.0687	0.0676
LightGCN	0.1302	0.1316
DiffRec	0.1413	0.1431
RecVAE	0.1349	0.1362
Multi-VAE	0.1394	0.1377
Multi-DAE (with side information)	0.1427	0.1413
EASE	0.1566	0.1565
ADMMSLIM	0.1524	0.1541
SLIMElastic	0.1562	0.1562
EASER	0.1612	0.1603

- 효과적인 추천 시스템은 인기도와 장르를 종합적으로 고려하여 각 type에 맞는 Item을 추천한다. EASE 모델은 전반적으로 인기 있는 Item을 높은 비율로 추천하며, 특히 type_popular 유형의 사용자에게는 이러한 인기 아이템의 추천 비율이 매우 높다. 반면, RecVAE 모델은 사용자의 선호 장르를 반영하여 추천하는 데 강점을 보이며, type_genre 유형의 사용자에게 특히 선호하는 장르의 아이템을 추천한다.

- 앙상블을 구성할 때는 서로 다른 결과를 제공하는 모델들을 조합하는 것이 중요하다. 이러한 다양성은 추천 시스템의 성능을 크게 향상시킬 수 있다. 따라서, 서로 다른 성향의 추천을 제공하는 모델들(EASE, ADMMSLIM, RecVAE)을 앙상블 전략에 통합하여 사용함으로써, 보다 다양한 추천을 가능하게 하여 성능을 올렸다.

EASE 10개의 제출 중 각 유저별로 조건을 만족하는 영화의 수

type	비인기	중간	인기	옛날	중간	최신	선호장르	선호감독	선호작가
1... popular	0.16	1.86	7.98	2.61	3.32	4.07	5.01	0.43	0.35
2... genre	0.54	2.69	6.77	2.48	3.13	4.39	7.14	0.34	0.34
총 합계	**0.19**	**1.92**	**7.9**	**2.6**	**3.31**	**4.09**	**5.17**	**0.42**	**0.35**

ADMMSLIM

type	비인기	중간	인기	옛날	중간	최신	선호장르	선호감독	선호작가
1... popular	0.32	1.83	7.85	2.7	3.33	3.97	4.82	0.45	0.39
2... genre	0.93	2.73	6.34	2.7	3.14	4.17	6.83	0.38	0.41
총 합계	**0.37**	**1.89**	**7.74**	**2.7**	**3.32**	**3.98**	**4.97**	**0.44**	**0.39**

RecVAE

type	비인기	중간	인기	옛날	중간	최신	선호장르	선호감독	선호작가
1... popular	0.79	2.47	6.74	2.72	3.22	4.07	5.52	0.32	0.23
2... genre	2.31	3.2	4.49	2.59	3.01	4.4	8.11	0.22	0.21
총 합계	**0.9**	**2.52**	**6.57**	**2.71**	**3.2**	**4.09**	**5.71**	**0.31**	**0.23**

Ensemble(EASE, ADMMSLIM, RecVAE)

type	비인기	중간	인기	옛날	중간	최신	선호장르	선호감독	선호작가
1... popular	0.4	2.05	7.55	2.64	3.34	4.03	5.32	0.41	0.3
2... genre	1.33	2.99	5.68	2.53	3.05	4.42	7.71	0.3	0.3
총 합계	**0.47**	**2.12**	**7.41**	**2.63**	**3.31**	**4.05**	**5.5**	**0.4**	**0.3**

- 4.3.1 Type based model의 실험결과 우리는 Type에 맞는 추천이 성능을 향상시킨다는 사실을 알게되었다. 단일 모델 중에서는 EASER가 가장 우수한 성능을 보였으며, 주로 인기 있는 상품을 추천하는 경향이 있었다. 반면, 앙상블 모델은 더 다양한 상품을 추천하는 방식을 채택했다.
- 이러한 특성을 바탕으로, EASER 모델은 인기 있는 상품을 선호하는 type popular에게, 앙상블 모델은 특정 장르를 선호하는 type genre에게 각각 추천을 적용하여 만든 개인화된 추천 모델(EASER, 앙상블 모델*에 type별 추천 적용)은 좋은 성능을 보여주었다.
- 최종적으로는 가장 높은 public score 값에 해당하는 두 결과 값을 제출하였다.

	Private Recall@10	Public Recall@10
EASER, 앙상블 모델*에 type별 추천 적용	0.1614	0.1605
앙상블 모델*(EASE, ADMMSLIM, RecVAE)	0.1613	0.1611

5. 자체 평가 의견

5.1 잘한 점들

- kanban board, 주말 데일리 스크럼 공유 등을 추가해서 서로의 상태 공유를 더 적극적으로 하려고 노력했다.

5.2 아쉬웠던 점들

- recbole에 대한 의존성이 높았다.
- 팀원 간 코드 리뷰가 소극적이었다.
- 상태 공유가 제대로 되지 못한 것 같다.
- 앙상블에 대한 논의를 대회가 끝나갈 때 쯤에 논의를 시작해서 다양한 앙상블 전략을 활용하진 못했던 것 같다.

5.3 프로젝트를 통해 배운 점 또는 시사점

- 가독성, 효율성 등의 방면에서 좋은 코드를 작성하기 위해서는 팀원들의 의견이 필요한데, 각자 코드를 깃에 올리는 시점도 다르고 올렸다고 하더라도 활발한 리뷰보다는 기능을 하는지 안하는지 여부에만 중점을 두고 리뷰가 이루어진 것 같다. 앞으로는 좀 더 활발하게 다른 사람이 코드를 올렸을 때 반응해준다면 좋을 것 같다.
- 현재 상태를 팀원끼리 잘 공유하는 것이 중요하다고 느꼈다. kanban board의 상태 업데이트를 미루지 말고 바로바로 하고, 주말 데일리 스크럼도 기입 마감 시간을 정해서 하는 건 어떨까 싶다.

---

6. 개인 회고

6.1 김수빈

이번 프로젝트를 진행하며 시도한 것

이번 프로젝트는 Book-Recommendation, DKT에 비해서는 실질적으로 투자한 시간이 적어졌다. Level 3 때 있을 Final Project를 기획하는데 많은 시간을 들였고, 상대적으로 Movie Recommendation Project에 대해서는 비중이 낮아진 것이다.

초기에는 한 가지 목표를 잡고 시작했다. Transformer 기반의 모델들을 이해하는 것. 3가지 모델을 실험했는데, 실질적으로 가장 많은 시간을 들인 모델은 Transformer 기반의 Bert4Rec과 SASRec이었다. 이를 직접 구현해보면서 이해도를 높이고자 했다. 그리고 EASE 모델을 개량한 EASER를 구현하고, 이에 대해서 분석하는 작업을 수행했다.

이번 Movie Recommendation의 경우에는 기존 대회와는 달리, Data의 Train/Test가 따로 분리되어 있지 않은 유저-아이템 인터렉션 데이터였기에 이를 어떻게 사용할지에 대해서 결정해야 했고, 또한 NDCG와 Recall을 지표로 사용하다보니 이에 대한 구현 역시 필요했다.

이번 프로젝트를 진행하며 배운 것

과거 추천시스템 기초에서 배운 여러가지 내용들을 실습하는 느낌이 가장 강하게 든 프로젝트였다. 실제 현업에서도 가장 많이 사용하는 데이터셋이기도 하고, Top-K를 추출해 추천해주는 방식, 평가 지표(Recall, NDCG) 등등이 실전적인 느낌을 강하게 주었다고 생각한다.

그리고 모델이 깊어지면 깊어짐에 따라서 성능이 오를 것이라는 기대와 달리, 얕은 AutoEncoder 기반 모델들이 높은 성적을 거둔 것이 굉장히 기억에 남았다. 전반적으로 Sequential 모델에 대한 성능이 낮았다. 데이터는 기존의 시청이력이 모두 주어져있고, 맨 마지막 결과를 예측하는 기존 MovieLens 데이터와 달리, 기존의 시청 이력에 중간중간 구멍이 나있는 형태였다. 마스터님의 말씀에 따르면, 다음 아이템을 예측하는 것보다 중간 예측이 더 쉽기에, 쉬운 Task의 모델들이 더 잘 통했을 것이라고 했다.

“절대적으로 좋은 모델은 없다.” 이것이 내 결론이었다. 모든 모델은 각자만의 특징이 있고, 장점과 단점이 있으며, 내가 쓸 데이터셋과 맞는지 안맞는지도 해보지 않고서는 알기 어렵다는 것이었다. 이를 위해서는 모델 선정 이전부터 과연 어떤 문제가 있는지를 생각해보는 문제 정의 단계가 필요할 것이라고 생각했다.

이번 프로젝트를 진행하며 아쉬웠던 것

1. EASER 모델을 구현하는 것에는 성공했으나, 모델의 아키텍처와 특징에 대해서 정확히 이해하지는 못한 것 같다.
2. 깃허브 및 칸반보드 활용, 실험 결과 정리를 제대로 진행하지 않아서, 적지 않은 실험을 했음에도 어떤 결과가 있었는지, 이 실험을 통해 무엇을 알 수 있었는지에 대한 자료들이 많이 소실되었다. 일정에 치이다보니 메모를 제대로 안하고 그냥그냥 빠르게 넘기는 습관이 좋지 않다고 느꼈다.
3. BERT4Rec, EASER에서 NDCG@K, RECALL@K 등의 Metric 및 Inference 코드 구현 과정에서 어려움을 많이 겪었고, 일부는 실제로 구현하지 못했었다. 무언가 기본기가 흔들린 것 같아서, 이 부분에 대한 보완이 필요하다고 생각했다.

다음 프로젝트에서 시도할 것

실질적으로 우리의 최종 프로젝트와 Movie Recommendation의 프로젝트가 연결되는 지점이 많다고 생각했다. 지금의 내용들을 잘 정리하고, 우리의 모델이 실제로 어떻게 서비스로 구현될 수 있을지에 대해서 고민하는 시간을 가져야겠다고 생각했다.

6.2 박시우

이번 프로젝트를 진행하면서 시도한 점

이번 프로젝트를 시작하면서 처음 세웠던 목표는 학습, 성과 면에서 두 가지로 나뉘었었다.

- 학습: autoencoder 완벽 이해 후 from scratch 구현, recbole 라이브러리 익숙해지기(+raytune.yaml 만들어서 튜닝), 팀원 코드 리뷰, 모델 구조 변형
- 성과: 대회 2위 이내 (성능 0.20 이상 목표)

이 중 시도한 건 아래와 같았다.

autoencoder 이해: autoencoder에 대한 이론적 이해가 부족해서 초반에 논문 내용 숙지에 어려움이 있었다. 논문만 보는 것이 아니라 정리해둔 블로그 글도 참고하면서 수식을 직접 정리하는 시간을 가지면서 autoencoder의 기본 모델과 변형애 대해서 잘 이해할 수 있게 되었다.

recbole 라이브러리: 추천 시스템 프레임워크인 recbole이 어떤 방식으로 구동되고 어느 정도의 자유도가 있는지, 장단점이 무엇인지 직접 체감해보고 싶어 사용해보았다. 라이브러리 내에서 튜닝도 시도해보았다.

팀원 코드 리뷰: 팀원이 git으로 공유한 코드를 보고 처음 사용하는 입장에서 개선했으면 하는 점을 리뷰했다.

모델 구조 변형: 가설을 기반으로 RecVAE 모델 구조에서 오토인코더의 layer 수를 줄여보았다.

이 밖에도 모델의 예측 아이템 n개를 hard voting 하는 것보다 예측 값을 soft voting 하는 것이 더 정교하게 추천 아이템을 조정해서 좋은 성능을 낼 것이라고 판단해서 soft voting ensemble 코드를 구현했다.

이번 프로젝트를 진행하면서 배운 점과 아쉬운 점

recbole 라이브러리 내 코드가 업데이트가 잘 되고 있지 않다는 게 치명적인 단점이라는 것을 알게 되었다. 일례로 tuning을 할 때 오류가 발생했고 직접 소스 코드를 수정해야 했다. 라이브러리 내에서 사용하는 다른 라이브러리가 업데이트한 내용을 반영하고 있지 않는 부분이 상당 부분이었기 때문이었다.

또 recbole 내에서 모델 구조를 바꾸는 게 어려울 것이라고 생각했는데 예상과는 다르게 모델, 데이터 등 필요한 부분을 custom해서 쓸 수 있게 해둔 것을 보면 나름의 자유도를 보장해 주고 있다는 것을 알 수 있었다.

recbole 베이스라인을 만들고 나니, 직접 모델 파이프라인을 개발할 필요성을 못 느껴서 의존성이 높아진 것 같다. 학습 목적의 대회였던 만큼 이 부분이 아쉬운 것 같다.

EASE 모델을 발전시킨 NEASE와 깊은 인코더, 디코더로 구성된 FLVAE를 결합한 VASP 모델이 Movie lens 20M dataset에서 좋은 성능을 보였어서 우리 프로젝트에도 써보려고 했었다. 마지막 주에 시도를 한 것이어서 시간 부족 이슈로 최종적인 테스트를 해보지는 못해서 아쉬웠다.

다음 프로젝트에서 개선할 점 및 시도할 것

모델 파이프라인을 직접 구현해보면서 팀원 간 코드 리뷰를 활발히 진행하고 싶다.

6.3 백승빈

이번 프로젝트를 진행하며 시도한 것

- EASE:

  지난 프로젝트에서는 단순히 다른 대회 우승 솔루션을 재연하기만 했던 것이 아쉬워서 논문들을 읽고 내용을 적용해보려 시도했었다.

  EASE가 좋은 성능을 내는 것을 확인한 후 CF계열 모델의 mainstream에 대한 편향을 줄이고자 SLIM의 extension에 대한 논문에서 제시된 아이템의 popularity의 영향을 줄이는 rescaling을 적용시켜보았다. 하지만 0.02정도로 성능이 무척 떨어지는 것을 발견하였다. 이후 EASE의 결과를 분석하며 관객수가 적은 영화도 생각보다 많이 추천되고 있다는 것이 발견했고 차라리 mainstream에 대한 추천을 강화시켜보면 어떨까하는 생각이 들어 다른 논문에서 제시되었던 유저의 mainstreamness를 설명하는 mean taste similarity를 binary interaction 대신 user feature로 사용해봤다.

- DiffRec:

  시도하는 모델들이 대부분 SLIM과 Autoencoder에 치중된 것 같다는 생각이 들어 CV의 diffusion model을 RecSys에 적용한 DiffRec 모델을 시도해보았다. 성능은 준수했지만 타 모델들과의 추천 경향성이 비슷했고 모델의 다양성을 늘리려는 시도에는 적합하지 않았던 것 같다. Diffusion model이 AutoEncoder와 유사하게 이미지 생성에 사용되기 때문인 것으로 추측된다.

- Voting 구현:
  recall@k를 사용하는 모델들의 output을 앙상블하기위해 Voting 기능을 구현했다. 처음에는 단순히 hard voting 방식을 시도했지만 같은 등장 횟수를 가지는 아이템들이 많고 그 사이의 우선순위를 처리하기 어려움이 있어 아이템의 ranking에 따라 점수를 부여하고 합산하여 최종 추천 아이템을 결정했다. 또한 모델 별로 비율을 다르게 적용할 수 있도록 했다. 이번 대회에서는 사용한 모델들이 Autoencoder와 Slim에 치우쳐진 면이 있었고 또한 여러 조합을 시도해볼 시간이 부족하여 큰 효과는 보지 못했다.

다음 프로젝트 때 개선할 점

- 어떻게 하면 실험과 변형에서 유의미한 결과를 얻을 수 있을지 고민이 필요한 것 같다. 이번 대회에서 EASE에 진행했던 실험들이 모두 큰 효과를 보지 못했는데 아직 EDA보다는 배경지식으로부터의 인사이트로 움직이는

것이 큰 것 같아 실험 전에 데이터를 통해 1차적으로 가설에 대한 검증을 우선 진행을 해봐야 할 것 같다. 또한 소수의 논문에 의존하지 말고 다양한 논문을 읽으며 시야를 넓히는 것 또한 필요할 것 같다.

- 팀원들에게 좀 더 의지해도 좋을 것 같다. EASE모델에 rescaling을 적용한 후 예상과는 너무나도 다른 결과에 당황하여 팀원들에게 상담하지 않고 좀 더 살펴보다가 넘겼는데 다른 사람들과 결과를 공유하고 같이 고민해봤으면 결과에 대한 보다 명쾌한 설명이나 과정에서의 오류를 찾아냈을 수도 있었을 것이란 생각이 들어 아쉽다.

6.4 이재권

이번 프로젝트를 진행하며 시도한 것

이번 프로젝트에서는 RecBole, PyTorch Geometric과 같이 추천 시스템에 사용 가능한 라이브러리들을 활용해 보고자 하였다. RecBole은 추천 시스템을 간단하게 구현할 수 있지만 구현의 자유도가 낮은 라이브러리이다. PyTorch Geometric은 추천에 필요한 함수들을 직접 구현해야 하지만 다양한 그래프 모델들을 활용할 수 있고, 구현의 자유도가 높은 라이브러리이다. 이 라이브러리들을 활용하여 그래프 기반 추천 시스템을 구현하고자 하였다.

이번 프로젝트를 진행하며 아쉬웠던 것

PyTorch Geometric 라이브러리를 활용하여 GNN 기반 모델들을 구현하기 위해 시도하였다. 하지만 PyTorch Geometric 라이브러리에 익숙하지 않아 시간 상의 한계로 구현을 끝마치지 못하여 RecBole 라이브러리만 활용하여 실험을 진행하였다.

다음 프로젝트에서 개선할 점

LightGCN 실험에서 시간 상의 이점을 위해 batch size를 크게 설정하여 실험을 진행하였다. 하지만 이는 batch size를 크게 설정하면 오버피팅의 위험이 있다는 사실을 관과한 실험이었다. batch size를 크게 설정하면, 모델은 비슷한 데이터 세트를 반복적으로 학습하게 되어 각 batch 내 데이터의 다양성이 감소하고, 같은 batch 내 데이터 간의 상관성이 증가하여 모델의 일반화 능력을 감소시킬 수 있다. 실험 결과에서 validation score와 public/private score의 차이가 큰 것을 확인할 수 있는데, batch size의 크기로 인한 오버피팅이 원인으로 판단된다. 추후에는 인공지능 관련 지식들을 근거로 신중하게 파라미터 튜닝을 진행하겠다.

다음 프로젝트에서 시도할 것

HAN, HGT와 같이 Heterogeneous Graph를 학습할 수 있는 모델을 통해 GNN에 side information을 활용해 보고, Collaborative Filtering 기반 그래프 모델과 비교하여 성능이 향상되는지 확인해보고 싶다. 또한, PyTorch Geometric 공식 문서를 통해 라이브러리를 더 공부하여 Recommendation, Link Prediction, Graph Classification 등 그래프 학습의 여러 분야를 직접 구현해보고 싶다.

6.5 이진민

이번 프로젝트를 진행하며 시도한 것

- github의 칸반보드를 사용하면서 진행했다
- 모델을 실험하기 전에 논문을 읽어보고 모델에 대한 이해를 한 후 실험을 진행했다.
- Multi-DAE에 side information을 추가해봤다.

이번 프로젝트를 진행하며 배운 것

- 기존 Multi-DAE가 collaborative filtering 모델이어서 side information을 추가해도 될까라는 생각을 했지만 성능에 향상에 도움이 되었다.

이번 프로젝트를 진행하며 노력한 것

- 원래에 모델에 새로운 시도를 해보고 EDA를 하며 근거를 갖고 실험을 해보자고 노력했었다.

이번 프로젝트를 진행하며 아쉬웠던 것

- 다른 것과 함께 진행하고 코드를 모듈화하는 리팩토링에 많은 시간을 썼어서 보다 다양한 시도를 하지 못했던 것 같다.
- 코드리뷰 또한 피어세션에 함께 진행하자고 했지만 최종프로젝트 회의 및 진행하고 있는 상황 공유 때문에 거의 하지 못했었다.
- Feature engineering에는 거의 시간을 쏟지 못한 것이 조금 아쉽다.
- 오버피팅이 발생하여 valid score로 성능을 확신할 수 없어 실험을 제대로 수행하진 못했던 것 같다.

다음 프로젝트에서 개선할 점

- 코드리뷰가 활발히 되지 못해서 이 부분을 개선해야될 것 같다.
- wandb에 올릴 때 어떤 것을 시도한 것인지 정리를 해야겠다. 나중에 확인하려고 보니까 어떤 것을 시도한 것인지 찾기가 힘들었다.

다음 프로젝트에서 시도할 것

- 다음 프로젝트에서는 피어세션때 pull request에 있는 코드들을 팀원들과 함께 리뷰를 하면서 진행할 것

마주한 한계 및 아쉬웠던 점

- genre와 director 정보를 기본 one-hot encoding하여 사용했는데 genre, title 정보를 word2vec을 이용해서 실험해보려고 했으나 시간이 얼마 남지 않아 끝까지 구현을 하지 못했었다.
- 앙상블을 미리 생각하지 않고 있어 거의 recbole 모델만 앙상블에 사용하게 되었다.
- 유저의 취향변화를 반영할 수 있는 강화학습 기반의 모델(RaCT)을 실험해보려고 했으나 시간이 없어 기본 모델을 돌려보는 것밖에 하지 못했다.

6.6 장재원

이번 프로젝트를 진행하며 시도한 것

- **EDA(탐색적 데이터 분석) 확장**
  - 다양한 기법을 활용해 탐색적 데이터 분석을 수행했다. 특히, 대시보드를 통해 데이터의 특성을 시각적으로 파악하고, 추천 시스템의 성능 분석에서 각 모델의 독특한 특성을 발견했다.
- **가설 검증:**
  - 사용자들이 인기 있는 영화를 선호한다는 가설과 각 사용자별로 취향이 다양할 것이라는 가설을 EDA를 통해 검증했다. 이 과정에서 사용자 유형과 선호도의 관계를 깊이 이해할 수 있었다.
- **모델 성능 분석:**
  - EDA 결과를 바탕으로 각 모델의 추천 결과를 분석하였다. 이를 통해 모델별로 특정한 성능을 보이는 원인을 파악하였고, 이러한 인사이트는 추후 모델 선택과 앙상블 전략을 수립하는 데 큰 도움이 되었다.

- **앙상블 전략 개발:**
    - 서로 다른 결과를 제공하는 모델들(EASE, ADMMSLIM, RecVAE)을 조합하여, 다양한 성향을 반영할 수 있는 효과적인 앙상블 전략을 개발하였다. 이 전략은 추천 시스템의 전반적인 성능 향상에 기여하였다.

이번 프로젝트를 진행하며 아쉬웠던 것

- **Context-aware Recommender Systems의 활용 실패**
    - 영화의 인기도와 장르가 추천 시스템에 중요한 영향을 미친다는 EDA 결과를 바탕으로, Context-aware Recommender Systems에서 side information을 활용해보았다. 장르와 같은 부가 정보를 모델에 통합해 성능 향상을 기대했고, 다른 모델들에서는 이러한 정보 활용이 모델의 성능을 개선시켰다. 그러나 Context-aware Recommender Systems은 side information를 추가했음에도 불구하고 기대했던 성능 향상을 보이지 못했다. Context-aware Recommender Systems의 효과적인 적용을 위해서는 단순히 맥락 정보를 추가하는 것 이상의 깊은 이해와 분석이 필요하다는 것을 깨닳았다