Movie Rec 랩업 리포트

▼ **목차**

1. 프로젝트 개요

Level2 Movie Recommendation 대회에 참여하여 다양한 엔지니어링 및 머신러닝 방법론으로 실험한 결과와 배운 점을 기록함.

대회 주제

사용자의 영화 평가 이력 데이터를 바탕으로 사용자가 다음에 평가할 영화와 이전에 평가한 영화 중 일부를 예측하는 것.

대회 성능 평가 지표

$$\text{Recall@}K = \frac{1}{|U|}\sum_{u \in U}\frac{|\{i \in I_u | rank_u(i) \leq K\}|}{\min(K, |I_u|)}$$

데이터 명세

컬럼명	설명
user	사용자의 고유번호
Item	영화의 고유번호
time	사용자와 영화의 interaction이 일어난 시간
title	영화의 제목
year	영화의 개봉년도
director	영화별 감독들
genre	영화별 장르들
writer	영화별 작가들

2. 팀 구성 및 역할

팀 한줄 소개

- 서신이삼조

팀 목표

- 개인적 목표 추구 및 원활한 소통을 통한 팀 전체의 성장

팀 문화

- **페어 프로그래밍**: 2명 단위로 태스크를 지정하여 태스크의 완성도를 높이려고 노력함
- **매일 진행 상황 공유**: 어디까지 했고, 어디에서 어려움을 겪고 있는지 투명하게 공유함
- **게더타운에서 코어타임 동안은 상주하기**: 어려움이 있을 때 동료 찬스!
- **제출 규칙**: 매일 인당 1회/11시 이후 자율적으로 제출
- **오프라인 미팅**: 매 주 3명 이상 참여 시 진행

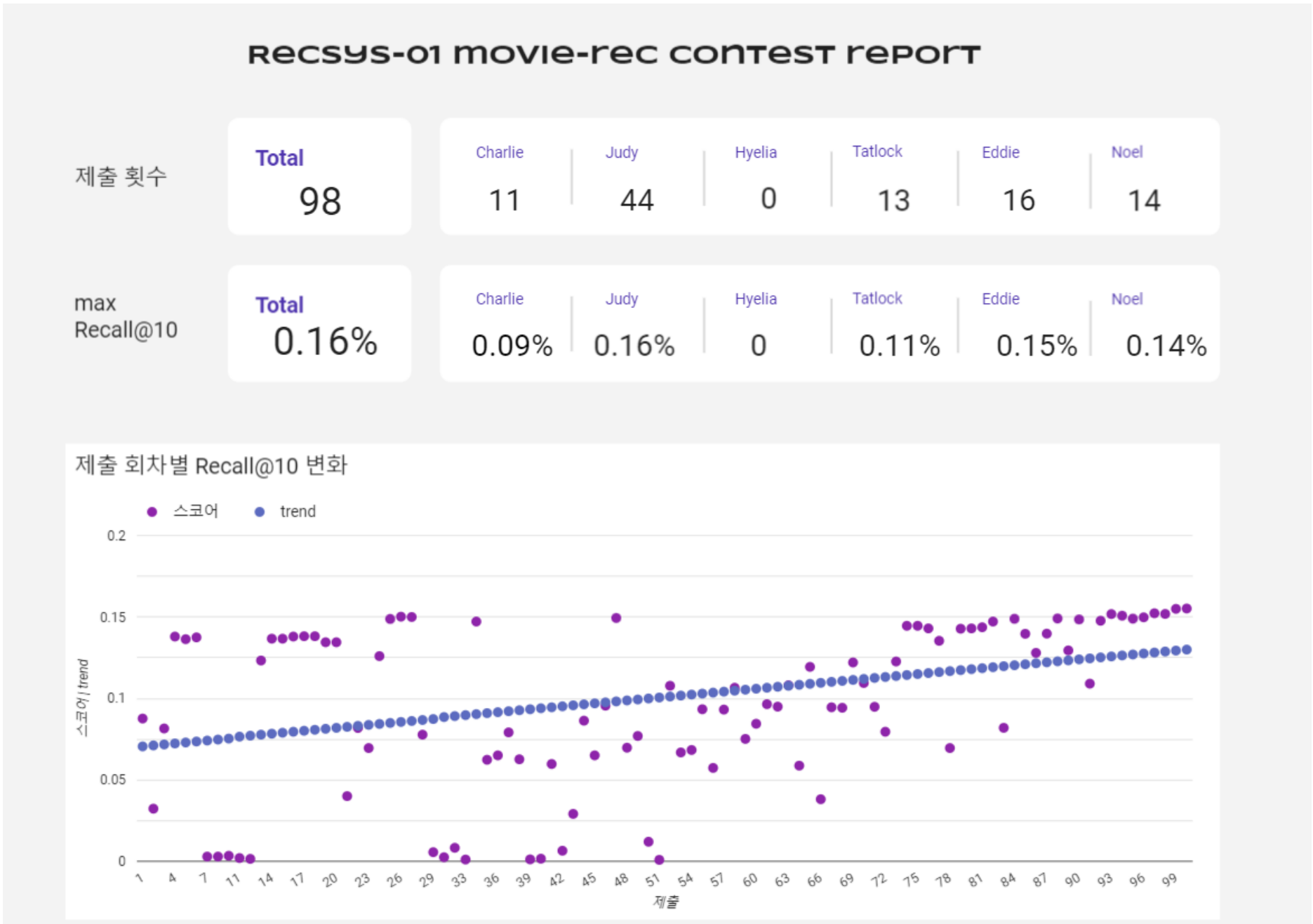
역할

캠퍼	역할
서동은	• EDA, ML modeling • BPR-MF, kNN/ALS, Bert4rec 사용
신상우	• 베이스라인 코드 작성 • context-aware models(DeepFM), AE 기반 모델 구현 및 실험
이주연	• 베이스라인 코드 구성하기 • context-aware models(FM), AE 기반 모델 구현 및 실험
이현규	• EDA, ML modeling • RecVAE, Rule-based, Bert4rec 사용
이현주	• ML modeling, Hyper parameter tuning • GRU4Rec 기반 모델링
조성홍	• 베이스라인 코드 작성 • context-aware models(FM,DeepFM,WDN) 기반 모델 구현 및 실험

3. 프로젝트 수행 절차 및 방법

프로젝트 타임라인

제출기록



4. 사용한 모델들

▼ 1) Rule Based

개요 및 주안점

유저가 본 영화 데이터를 통해 추천에 대한 우선 순위를 정해서 앞으로 볼 영화를 추천을 해준다면 좋은 결과를 나타낼 수 있을 것이라 생각. 유저가 선호하는 장르와 영화의 누적 시청 횟수를 사용하여 추천.

label을 따로 가지고 있지 않은 점에 내가 생각한 방향의 추천 형태는 어떨까? 데이터를 통해 추천에 대한 우선 순위를 정해서 추천을 해준다면 좋은 결과를 나타낼 수 있지 않을까?하고 시도해 본 방법

방법

1. 유저가 아직 안 본 데이터를 모음

	user	item
0	11	[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...
1	14	[2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, ...
2	18	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...
3	25	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...
4	31	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...
...	...	...
31355	138473	[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...
31356	138475	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...
31357	138486	[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...
31358	138492	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...
31359	138493	[3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ...

2. 각 영화의 장르와 누적 시청률이 있는 테이블을 만듬

	item	Crime	Drama	Action	Sci-Fi	Thriller	Comedy	Romance	War	Adventure	Fantasy	Horror	Mystery	Animation	Children	Film-Noir	Musical	Western	Documentary	count
0	1	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	0	0	12217
1	2	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	3364
2	3	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	734
3	4	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	43
4	5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	590
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6802	118700	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54
6803	118900	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60
6804	118997	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	52
6805	119141	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	122
6806	119145	1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	78

3. 유저의 선호하는 장르 Top 3개를 뽑음

	user	genre	genre.1	genre.2
0	11	Action	Adventure	Sci-Fi
1	14	Adventure	Comedy	Drama
2	18	Comedy	Drama	Romance
3	25	Action	Comedy	Drama
4	31	Action	Adventure	Sci-Fi
...	...	...	...	...
31355	138473	Crime	Drama	Thriller
31356	138475	Crime	Drama	Film-Noir
31357	138486	Comedy	Horror	Thriller
31358	138492	Comedy	Drama	Romance
31359	138493	Action	Comedy	Drama

4. 유저에게 TOP3에 속하는 장르의 영화를 아직 안 본 영화 데이터 중에서 누적 시청률이 높은 순위로 10개를 추천함

결과

LB: 0.0084로 하나도 제대로 맞추지 못하는 정도의 결과를 얻음. 각 영화마다 장르를 여러가지 가지고 있는데 장르 비율이 일정하지 않아서 좋은 추천을 하지 못한 것으로 생각함.

▼ 2) Context-aware Model

개요 및 주안점

본 대회는 유저의 explicit feedback이 아닌, implicit feedback을 토대로 새로운 아이템과의 interaction이 발생할지를 판단하고(CTR prediction과 동일), 그 중 가장 확률이 높은 아이템 10개를 추출하는 Task. Sparse한 interaction matrix에 대해서도 유의미한 성능을 기대할 수 있는 모델군이고, 모델 구현과 새로운 feature를 context형태로 만드는 것이 비교적 용이하단 점에서 해당 모델들을 사용함.

모델군의 특징

기본적으로 feature간의 interaction을 다양한 방법으로 학습하도록 구성된 모델들. Context내 각 feature 값들의 조합이 최종 interaction에 영향을 끼친다는 아이디어를 기반으로, interaction에 관여한 feature의 수 기준 low-order 부터 high-order까지 다양한 interaction을 학습할 수 있음.

1) FM(Factorization Machine)

각 feature를 Latent vector로 매핑한 뒤, feature간의 interaction들을 내적을 통해 학습시킴. 모델의 계산 복잡도 문제로 2-order만을 명시적으로 학습시킴.

Exp1: Basic Interaction

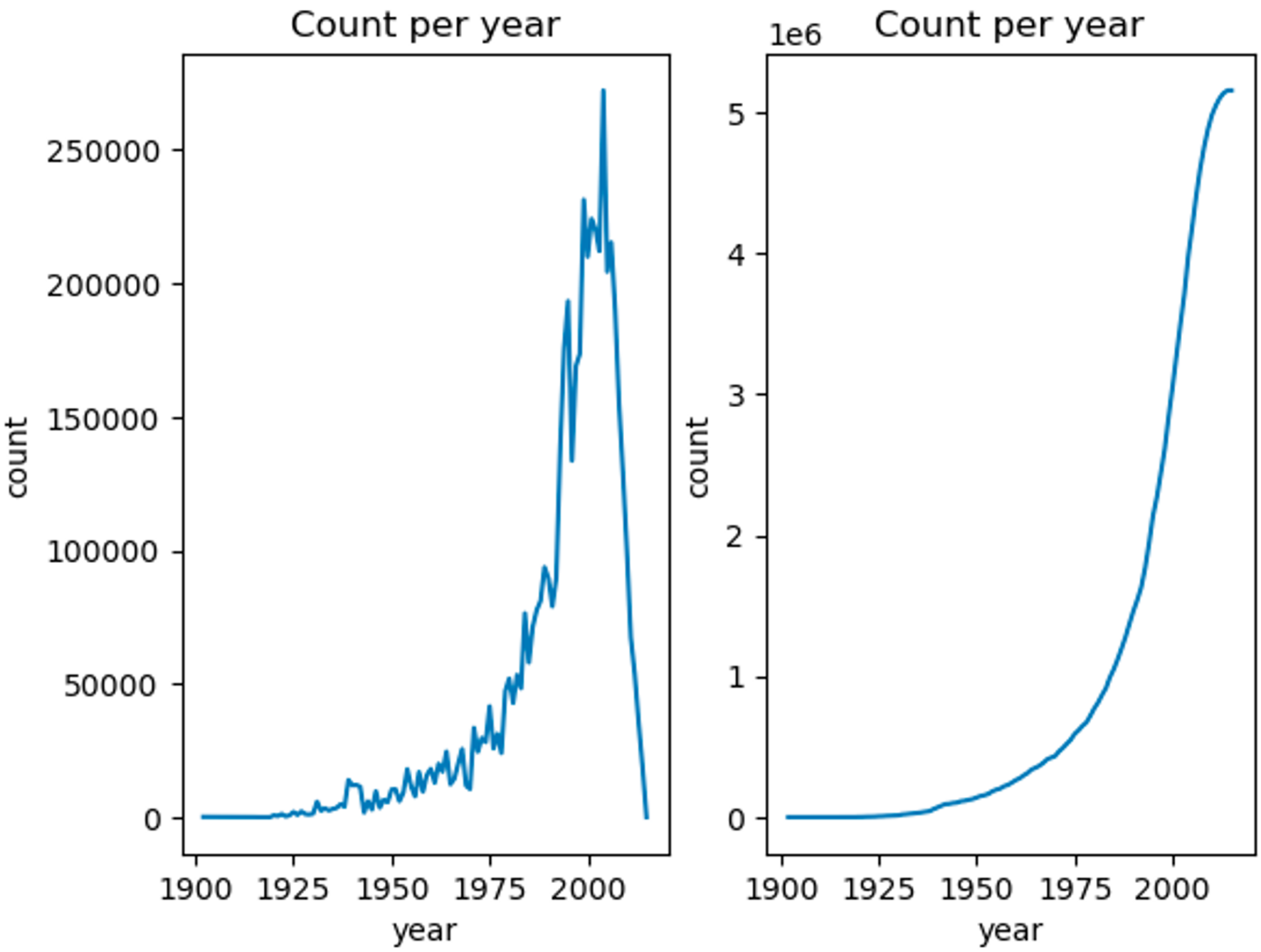
배경 및 기대효과 user와 item으로만 이루어진 interaction 데이터로 학습한 결과를 기준으로 삼기 위해 실험

결과 및 해석 valid Recall 에 비해 public Recall 이 상대적으로 높게 출력됨.

실험명	valid Recall@10	public nDCG@10	public Recall@10
ERA 추가	0.0546	0.2092	**0.0696**

Exp2: Feature 추가 - ERA

배경 및 기대효과 테스트 데이터에 시간순으로 나열한 시퀀스 상 미래의 대상만이 아닌 과거의 대상도 포함된다는 점에서, 당시의 시대적 배경이 사용자가 rating을 남겼을 확률에 높은 영향을 끼쳤을거라 예상함. 예를 들면, 영화에 대한 접근성이 적은 시기보단 접근성이 높은 시기에 개봉한 영화를 접하고 평가했을 확률이 일반적으로 더 높을 것. 또한 해당 데이터가 취합된 플랫폼 사용자가 많아진 시기에 개봉한 영화들의 평가가 더 많지 않을까라고 생각. 실제 데이터 상 1970년대 이후 개봉한 영화들에 대한 interaction 수가 급격하게 늘어나는 것을 확인. 영화별 개봉년도를 아래와 같은 기준으로 category화 한 데이터를 feature로 추가해 학습시켜봄.




```
if x < 1920:
    return 1
elif x >= 1920 and x < 1950:
    return 2
elif x >= 1950 and x < 1970:
    return 3
elif x >= 1970 and x < 1980:
    return 4
elif x >= 1980 and x < 2000:
    return 5
else:
    return 6
```

결과 및 해석 ERA feature를 추가한 뒤 예상과 달리 오히려 성능이 저하됨. 긴 epoch으로 학습시키지 않아 충분히 학습이 되지 않았을 거란 해석과, 해당 feature에 영화 산업의 시대적 배경을 담고 있을거라 생각했던 가정이 틀렸거나, 시대적 배경이 사용자가 리뷰를 남길지 여부와 큰 연관이 없다란 해석이 가능할 것 같음.

실험명	valid Recall@10	public nDCG@10	public Recall@10
ERA 추가	0.0502	0.1940	**0.0624**

Exp3: BPR Loss로 학습시키기

배경 및 기대효과 초기 모델은 학습시 Binary Classification Task를 수행하기 위한 BCE Loss 최적화 형태로 학습시킴. 하지만, 아이템의 계산된 positive 확률을 기반으로 순서를 나열하는 것보다, 아이템간의 상대적인 우위를 직접적으로 최적화하는 형태인 BPR Loss로 학습을 시키는 것이 Top-K Task에서는 더 효과적일거라 기대.

결과 및 해석 BCE Loss 를 사용할 때보다 기본 interaction만 사용했을 땐 차이가 없지만, feature를 추가했을 때 유의미한 성능 향상이 일어남. BPR Loss를 통해 상대적인 우위를 학습시킬 때, feature를 추가하는 것이 둘 간의 차이를 벌리는 데 더 효과적으로 보임.

실험명	valid Recall@10	valid nDCG@10	public Recall@10
BCE Loss	0.0794	0.2199	**0.0696**
BCE + ERA	0.0938	0.2838	**0.0820**

2) Wide and Deep

배경 및 기대효과 이전에 발생했던 interaction의 feature조합을 memorize하는 Wide Component와, 일반적이거나 복잡할 것으로 예상되는 관측되지 않은 interaction의 feature 조합을 학습하는(generalize) Deep Component를 joint-training 형태로 구성한 모델.

다양한 feature를 추가했을때, Deep component를 통해 2-order 보다 더 높은 order의 interaction을 포착할 수 있지 않을까 기대함. FM 모델에서 적은 epoch 으로 학습했을 때 기준 성능 향상을 보였던 feature들을 포함시켜 학습을 시켜봄.

결과 및 해석 WDN 내에선 feature를 추가하는것이 확실히 성능향상에 도움이 되었음. 하지만 FM과 비교해 성능이 비슷하거나 오히려 더 떨어짐. Deep Component의 해석력이 부족하다고 생각해 mlp layer를 더 깊게 쌓거나, 넓게 쌓는 등의 실험을 추가로 진행했으나, 적은 epoch에서 금방 overfitting이 되는 것을 확인함. 사용한 feature set 내 FM에서 충분히 포착가능한 order의 interaction만 존재하고 더 높은 order의 interaction이 존재하지 않기 때문으로 해석함.

실험명	valid Recall@10	valid NDCG@10	public Recall@10
WDN	0.0442	0.1219	**0.0382**
WDN feature 추가	0.0682	0.1811	**0.0588**

▼ 3) Sequence Rec

개요 및 주안점

추천 시스템 대회에서는 “temporal dynamics” 를 고려하는 것이 중요함. 이는 특정 상품에 대한 평가가 시간의 흐름이나, 계절이나 요일, 주, 월 단위의 주기성이 영향을 미칠 수 있기 때문임. 또한 이로 인해 사용자 인터페이스나, 사용자의 취향이나 의견이 변하여 평가에 영향을 줄 수 있음. 따라서 과거의 아이템의 사용 또는 구매의 순서에 따라 바로 다음 상호작용이 어떻게 될 지, 즉, 다음에 평가할 아이템이 무엇인지 예측하는 모델을 사용하여 추천할 수 있음.

(1) GRU4Rec

- **설명** 순환 신경망 (RNN) 중 하나인 Gated Recurrent Unit (GRU) 을 기반으로 한 추천시스템 모델. 사용자가 이전에 상호작용한 아이템들의 순서를 고려하여, 다음에 사용자가 상호작용할 가능성이 높은 아이템을 예측하는 것을 목표로 함.
- **장점** 아이템 시퀀스의 시간적인 순서를 고려하여, 각 세션마다 개별적으로 학습되는 세션 기반 모델임. 따라서 사용자의 특정 세션에 맞춤형 추천이 가능함. 사용자의 상호작용을 세션 단위로 처리하기 때문에, 사용자 프로파일을 만드는 데이터 전처리가 비교적 간단함.
- **단점** 세션 내의 아이템 간의 상호작용에 대한 정보만을 고려하고, 전체적인 사용자 프로파일은 고려하지 않음. 장기적인 의존성을 학습하지 못할 가능성이 존재하여 세션이 길어지면 성능이 떨어짐. Cold start 문제가 있어 새로운 사용자나 세션에 대한 추천 성능이 낮을 수 있음.

(2) GRU4RecBE

- **설명** GRU4Rec 을 확장하여, pretrained BERT 와 Bayesian Personalized Ranking (BPR) loss 를 사용하는 모델. 사용자 간의 차이를 반영하여 학습 과정에서 사용자가 실제로 선호하는 아이템에 대해 더 높은 순위를 주는 방식으로 모델의 성능을 개선함. 이를 통해 학습 과정에서 사용자의 개별적인 특성을 보다 잘 고려하여 더 개인화된 추천이 가능해짐.
- **장점** BPR loss 를 통해 고도화된 개인화 추천 제공하고, 효율적인 학습과 성능 향상을 이룸.
- **단점** BPR loss 추가로 인해 모델의 복잡성이 증가하고, 추가적인 연산으로 비용이 발생함. BPR 은 사용자가 선호하지 않는 아이템 pair 에 대한 학습도 진행을 하기 때문에, 이에 대한 데이터가 부족한 경우 성능이 안 나올 수 있음.

▼ 4) MF method

개요 및 주안점

사용자가 이미 봤던 영화 기록을 바탕으로 원래의 user-item 상호작용 행렬을 가능한 정확하게 근사함

Experiments

Exp1: BPR

배경 및 기대효과

Implicit 데이터와 Top-k 랭킹에 사용하기에 적합한 모델. Rating이 없으므로 EDA와 FE을 통해 user-item간의 rating을 추정하여 confidence로 사용함.

결과 및 해석

confidence 값에 다양한 정보를 반영하였을 때 recall@10이 상승함

model	Recall@10	영화 인기도 반영 Recall@10	영화 인기도 + 장르 선호도 반영 Recall@10	**(영화 인기도 + 장르 선호도) * e^(영화개봉년도-1980) Recall@10**	
BPR	0.770	0.934	0.950	**0.969**	* 1980년도 이후 영화 산업의 급격한 성장을 반영

Exp2: ALS

배경 및 기대효과

ALS는 Alternating Least Squares의 약자로, 주어진 행렬을 두 개의 저차원 행렬로 분해하는데 사용되는 반복적인 최적화 알고리즘. 희소하고 큰 행렬에 효과적일 것으로 기대함

결과 및 해석

confidence 값에 다양한 정보를 반영하였을 때 recall@10이 상승함

model	영화 인기도 + 장르 선호도 반영 Recall@10	
ALS	1.091	factor=64

▼ 5) AEs

개요 및 주안점

전형적인 시퀀스 문제가 아니고, 앞 뒤로 유저가 좋아할 영화를 예측해야 하기 때문에 CF 문제로 정적으로 예측했을 때 더 성능이 좋을 것으로 기대함. 데이터를 압축해 사용자의 영화에 대한 선호 정보를 학습하고 불필요한 정보는 필터링하기를 기대함

모델명	설명
AE	데이터를 압축하고 다시 복원하는 과정에서 중요한 특성을 포착함
DAE	• Input data 에 'random noise'를 추가함 • noise를 추가해서 일반화 성능을 높임
VAE	가지고 있는 데이터의 분포를 정확히 모르기 때문에 정규분포를 이용하여 학습함.
MultiVAE	vae에서 다항분포를 사용함.
RecVAE	vae는 정규분포를 컨트롤하며 모사하지만 recvae의 경우 이전 인코더를 사용해서 현재 받아온 데이터와 이전 인코더의 분포를 적절하게 맞춤

Experiments

Exp1: AE vs. DAE vs. VAE

배경 및 기대효과 추천을 위한 다양한 오토인코더 기반 모델 중 vanilla AE, DAE, VAE 의 성능을 비교함. DAE는 입력에 노이즈를 첨가하여도 입력을 복원하도록 학습하여 더 robust하며, VAE의 경우 정규화항에 의해 오버피팅이 방지되고 확률적 접근 방식을 가져 개인화 추천에 더 성능이 좋은 것으로 알려짐. 따라서 AE < DAE < VAE 순으로 성능이 향상될 것으로 예상함.

결과 및 해석 실험 결과, AE < VAE < DAE 순으로 성능이 향상됨. 일반적인 추천 대규모 데이터셋은 Sparsity가 99% 이상임에 반해 본 데이터셋의 경우 97% 정도로 희소 데이터에 더 높은 성능을 보이는 VAE보다 DAE가 더 적합했던 것으로 보임. MultiVAE (2018)의 Empirical Study에서도 다음이 언급됨

Type	valid Recall@10	valid nDCG@10	public Recall@10
AE	0.2559	0.542	**0.0965**
DAE	0.2342	0.5332	**0.1397**
VAE	0.1617	0.394	**0.1194**

Exp2: Logistic likelihoods vs. Multinomial likelihoods

배경 및 기대효과 implicit feedback 기반 랭킹 추천 오토인코더에서는 기본적으로 데이터 분포를 이항 분포로 가정하는데, 아이템 간 자원의 경쟁이 있는 상황을 반영하여 다항 분포를 가정할수도 있음. MultiVAE (2018)에서는 랭킹 추천에서 Multinomial이 우수한 결과를 냈다고 함

결과 및 해석 실험 결과, DAE, VAE 모두에서 성능이 향상되었으며, 향상의 폭도 유사했음. Implicit 랭킹 추천 태스크에서는 Multinomial 가정이 사용자의 행동 특성을 포착하는 데 더 적합하다고 추측됨

Type	valid Recall@10	valid nDCG@10	public Recall@10
Logistic-DAE	0.2342	0.5332	0.1397
Multi-DAE	0.2272	0.4978	**0.1437**
Logistic-VAE	0.1617	0.394	0.1194
Multi-VAE	0.2066	0.4547	**0.1221**

Exp3: Confidence level

배경 및 기대효과 Implicit feedback 예측을 위한 MF 방법론 중 하나로, 클릭횟수, 시청시간 등을 가중치로 변환하여 loss에 반영함. 더 의미있는 데이터의 loss가 학습에 더 크게 반영되도록 함. 본 대회의 데이터는 기본적으로 영화를 “평가”한 기록에서 평가 점수를 뺀 것으로, 직접 신뢰 수준에 활용할만한 시청 시간, 시청 횟수 등은 없음. 다만 다양한 피처 정보와 평가 시점이 있으므로, 크게 평가 시점, 아이템 인기도, 유저의 장르 선호도를 구분하여 반영하면 더 높은 성능을 낼 것으로 예상함.

결과 및 해석 어떤 confidence를 사용하여도 0.01 이상의 Recall@K 차이를 보이며 성능이 향상됨. 특히 유저의 장르 선호도에 confidence를 주었을 때 더욱 성능이 향상되어, 유저가 많이 “평가”한 장르의 영화에 대해 잘 맞추면 가점을 주고, 잘못 맞추면 패널티를 크게 주는 것이 유저가 평가할 아이템을 높게 랭킹하는 데에 보다 적합한 것으로 예상함

Type	valid Recall@10	valid nDCG@10	public Recall@10
confidence-None	0.2281	0.5229	0.1397
confidence-time	0.2315	0.5236	0.1485
confidence-itempop	0.2318	0.5262	0.1489
confidence-genreprefer	0.2421	0.545	**0.1491**

Exp4: fixed β and KL annealing

배경 및 기대효과 MultiVAE (2018)의 핵심 중 다른 하나는 학습 과정 중 KL divergence의 영향력을 달리 주는 KL Annealing의 도입이었으며, 성능 향상을 기대함

결과 및 해석 결과 이전에 하이퍼파라미터의 선택이 까다로웠음. 몇 스텝만에 최고치에 도달하게 할 것이며, 최고치 beta를 어떻게 설정해야 할 지 감이 안잡힘. 뿐만 아니라 annealing 을 적용하기 전보다 빠르게 overfitting되어 학습이 중단됨. MultiVAE를 발전시킨 RecVAE (2019)에서도 이와 같은 이유로 고정 beta를 사용하였다고 함.

Exp5: RecVAE

배경 및 기대효과

- CF 모델이고 implicit data 기반 Top-k 추천에 최적화가 잘 된 모델
- 팀 내에서 public Recall@10 기준 **0.1502**로 가장 높은 성능을 보임

▼ 6) Ensemble

Voting

개별 모델들의 Top 30 영화를 랭킹한 뒤, 랭킹 합으로 정렬하여 최종 Top 10을 결정함. 한 모델에서 추천하지 않은 영화의 경우 31등으로 간주함.

Experiments

배경 및 기대효과 최고 성능에 달하는 모델은 DAE, RecVAE 이었으나, orthogonal models 간의 앙상블 시 성능이 더욱 향상될 것으로 기대되어, 성능이 비교적 낮았던 kNN, MF 기반 모델을 앙상블에 포함함. 모델 간 예측 확률의 scale이 매우 상이할 것으로 예상되어, 랭크 간 hard voting을 진행함

결과 및 해석 : 예상과 같이, 성능이 높은 모델만을 활용할 때보다 다양한 모델을 활용할 때 앙상블 성능이 더 높았음

Model	final Recall@10	public Recall@10
MultiDae_GenrePreference + MultiDae_Popularity + MultiDae_Time + MultiDae	0.1531	0.1518
MultiDae_GenrePreference + MultiDae_Popularity + MultiDae_Time + MultiDae + MultiVae	0.1536	0.1523
MultiDae_GenrePreference + MultiDae_Popularity + MultiDae_Time + MultiDae + MultiVae + Als	0.1556	0.1550
MultiDae_GenrePreference + MultiDae_Popularity + MultiDae_Time + MultiDae+ MultiVae + Als + kNN	0.1559	0.1552

▼ 7) Custom 베이스라인 코드 작성

배경 및 목적

movie recommendation 대회의 경우 주어진 베이스라인 코드가 S3Rec에 치우쳐 있었음. 지난 프로젝트에서 공유가 원활하지 못했던 이유로, 베이스라인 코드의 활용을 꼽았기에 베이스라인을 활용하기 편한 방향으로 구현하여 이 구조 위에 모델을 구현하기로 함.

작업 중 어려움

- 지난 두 대회의 경우 추천 태스크 중 예측 태스크였으므로, 목표 사용자와 아이템에 대한 추론만 하면 되었으나, 이번 대회의 경우 “TopK 추천” 태스크였기에 모든 사용자와 아이템에 대해 추론/랭킹/TopK 일련의 부가적인 과정이 필요했음
- 또한 “보다 현실적인 시퀀셜 태스크”로 단순히 다음에 볼(평가할) 영화에 대한 랭킹이 아니라, 이전에 본 영화 중 일부도 맞춰야 했음. 따라서 sequential 모델과 더불어 다양한 데이터셋 준비, 학습/평가 매커니즘이 필요했음

작업 내역

- FM 을 베이스 모델로 선 구현해 다른 모델 성능 판단의 기준으로 설정
- 실험 단위 정의: [모델 + negative sample 수 + 사용한 Feature Set]
- 실험 단위별 Data Versioning 기능 구현
  → Train/Valid DataSet 및 Inference용 Total user-item tuple 파일(validation시 모든 유저, 모든 아이템 inference 결과로 성능 측정 위함) 재사용
- Config File 기반 Arguments 관리 기능 구현
- 사용 Feature Set Config 기반 제어 기능 구현
- 저장된 모델 기준 inference 수행 기능 구현
- AE 계열 모델 구현: AE, DAE, VAE, Multi-DAE, Multi-VAE, RecVAE
- Context-aware 모델 구현: FM, DeepFM, WDN
- BPR Loss 기반 학습 구현

데이터 처리 및 모델링 프로젝트 파일 구조

```
.
├── README.md.                  # 프로젝트 설명
├── outputs                     # 학습시 생성되는 데이터 저장 디렉토리
│   ├── datasets                # 학습시 필요한 데이터 셋 저장
│   ├── models                  # 학습시 최고 성능을 기록한 모델 저장
│   └── submissions             # 추론 결과 파일 저장
├── requirements.txt            # 필수 패키지 목록
├── run_inference.py            # 저장 모델로 추론 실행 모듈
├── run_train.py                # 학습 실행 모듈
└── src                         # 핵심 모듈 디렉토리
    ├── __init__.py
    ├── configs                 # 학습 및 추론시 사용 argument 정의 yaml 저장 디렉토리
    ├── data                    # 모델별 필요 데이터 형태로 변환하는 모듈 저장 디렉토리
    ├── ensembles.py            # 앙상블 전략 구현 모듈
    ├── inference               # 추론용 모듈 디렉토리
```

```
    ├── loss.py                # 손실 함수 정의 모듈
    ├── metrics.py             # 성능 지표 함수 정의 모듈
    ├── models                 # 모델 클래스 모듈 디렉토리
    ├── train                  # 모델별 학습용 모듈 디렉토리
    └── utils.py               # 파일 저장 경로 설정 등 유틸 기능 모듈

10 directories, 9 files
```

작업 후 아쉬움

- 예정보다 완성이 미뤄졌고, 막상 완성한 코드를 팀원 모두가 쉽게 흡수할 수 있도록 전달하기 어려워서 작업에 참여한 사람들만 베이스라인 코드를 대회 끝까지 활용하게 된 부분이 아쉬웠음.
- 모델마다 공통으로 묶기 어려운 부분들이 분명히 있는데, 모든 모델에서 공유 가능한 코드로 만들려다보니 시간도 미뤄지고 클린하게 작성하기 어려웠음. 결국 코드 완성 후 data, model, trainer 부분은 모델별로 구현하는 방향으로 변경함
- ML 애플리케이션은 Data, Code, Model 3가지 요소로 구성되는데, Model에 다른 요소가 종속적인 형태라고 생각. Model에 입력되는 데이터 형태에 맞춰 학습 파이프라인을 구성해야 하는데, 이 과정에서 수정되는 Code의 양이 생각보다 많았음. 이 부분을 최소화하는 시도를 못 해본 것이 아쉬움.
- 지난 프로젝트에서 아쉬웠던 Data의 버저닝 및 자동화된 실험 관리 측면에서 어느 정도 소득이 있었음. 같은 Model과 Feature Set, negative sampling count 로 학습시 DataSet을 재사용하도록 구성. 또한, 추가하려는 Feature를 Feature이름으로 함수를 작성하고 config파일에 추가해주면 반영되도록 구현함. 하지만 이전 실험의 재현을 위해선 자동화된 Code 버전 기록도 필요하다고 생각. 현재 구성한 파이프라인에선 이 부분이 해결 안 돼 아쉬움.
- RecBole에 대한 소개가 이뤄진 프로젝트였는데, 직접 구현에만 집중해 RecBole을 다뤄보지 못 한 게 아쉬움.
- 제출 파일 생성을 위해 전체 user-item 조합의 inference가 필요했는데, 사용하는 feature를 늘릴수록 load 되는 메모리의 양이 늘어 OOM 으로 프로세스가 죽음. 이 문제를 해결하지 못 해 많은 feature를 넣어 학습해보지 못 한 것이 아쉬움.

5. 자체 평가 의견

1. 시행착오
    - FM 모델을 평가할 때, valid set에 대해 Recall@10과 nDCG@10을 서로 바꿔 출력함. public 과 valid 의 Recall 이 굉장히 많이 차이나 의아했는데 알고보니 아니었음...
    - AE 모델로 학습 및 평가 시, 학습, 검증 손실과 매트릭 계산 시 데이터셋을 구축하는 데에서 어려움을 겪음. 결론적으로 모든 트랜젝션에 대해 forward processing 한 뒤, 학습/검증 손실을 구할 때는 학습/검증 데이터에만 마스킹을 취하고, 매트릭 계산 시에는 unseen 데이터에 대해 top-k 랭킹을 하도록 적절히 구현함
    - loss.item() 안해서 OOM 터짐
2. 잘한 점
    - 자체 베이스라인 구축
    - Rule-based 등 다양한 방식으로 문제 접근
    - 한 모델에 다양한 방법론을 시도해보며 대조 실험을 진행함
    - 페어 프로그래밍을 통해 보다 완성도 높고, 유연한 코드를 작성함
    - 대회를 통해 다양한 모델들을 문제에 적용하고 다시 공부함
3. 아쉬웠던 점
    - 모든 팀원이 베이스라인을 이해하고 활용할 수 있도록 빠르게 구현하고, 자세히 전달했다면 좋았을 것
    - 기록하자고 정했던 것들이 완전히 이루어지지는 않아서 아쉬움
    - 한 가지 모델을 깊게 파서 하나를 완벽히 이해했으면 더 좋았을 것 같음
    - 팀원들이 다 같이 AE 관련 모델들을 시도해서 앙상블 할 때 시퀀스 모델이 없었음
4. 배운점
    - 실험 목적과 결과에 대해서 자세하게 기록해두는 것이 매우 중요함

- 예측 모델만 구축하는 것과 예측 모델에 기반하여 랭킹 추천을 하는 것은 파이프라인 상 많은 구현의 차이를 야기하는 것을 알았음. 특히 추론에 많은 시간이 걸렸는데, 이 때문에 후보군을 추려내어 랭킹하는 n-stage 추천의 필요성을 깨달음.

서동은

■ 개인 학습 측면

1. Bert4rec의 논문을 이해하고, 다른 사람의 깃헙 코드를 우리 데이터를 읽을 수 있게 수정하고, inference를 추가하는 작업을 공부
2. Benfred의 Implicit 라이브러리를 사용하여 MF의 머신 러닝 알고리즘 사용하는 경험
3. EDA에 근거하여 사용자가 영화에 매긴 rating을 추정하여 사용한 후 성능 비교 실험

■ 공동 학습 측면

1. EDA / Feature Engineering : 팀원과 어떤 피쳐를 왜 만들었고, 어떤 효과가 있었는지 공유하며 EDA, 피쳐 엔지니어링 능력을 기름
2. Inference : 팀원과 top-k 를 추천하는 경우 어떻게 모델이 추론하여야 하는지 토론하고, 코드를 공유하며 공부함
3. 앙상블 : 팀원이 앙상블 할 때 어떤 식의 논리로 코드를 적었는지 설명을 듣고 보팅 앙상블에 대한 이해가 깊어짐

■ 사용한 지식과 기술

○ 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

마지막에 볼 영화를 맞추는 것이 중요한 과제라고 생각해 Bert4rec을 사용. 이를 통해, 전체 파이프라인에 대한 이해와 코드 작성하는 실력을 향상시킴. 또한, 라이브러리를 사용하여 MF 알고리즘을 사용. BPR, ALS, 클러스터링 모델을 기반으로 EDA/FE에 집중함. 이후에는 피쳐 간의 상호작용을 딥러닝을 통해 추출하는 AutoInt라는 논문을 읽고 코드를 사용해 봄. 이를 통해, implicit 데이터의 경우 EDA/FE와 representation learning이 중요하다는 것을 깨달음.

○ 전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

주어진 베이스라인 코드를 사용하지 않고, 깃헙에서 다른 사람의 코드를 베이스라인으로 사용하려고 시도함. 전체 파이프라인 코드를 분석하고, 수정하면서 실력이 성장함. 이전 대회와 달리 이번에는 EDA와 FE를 통해 성능 향상시키는 경험을 함. EDA를 통해 세운 가설이 실제로 맞을지 피쳐로 만들고 성능을 비교하는 실험을 통해 데이터를 다루는 능력이 향상됨. 다양한 추천시스템 라이브러리를 사용하려고 시도하고, 각 라이브러리에 필요한 데이터를 만들고 넣는 공부를 하여 사용 방법을 익힘

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

실력의 한계를 느낌. 깃헙의 파이프라인을 가져와서 수정할 때 잘 작동하지 않아 어려웠음. 특히 추론하는 과정에서 코드를 잘못 작성하여 제출한 결과가 매우 낮은 값이 나와 좋은 결과를 내지 못함. 라이브러리를 사용할 때도 버전이나 필요한 파일 등을 만들때도 어려움이 있었음. 또한, AutoInt를 사용하였으나 모델이 잘 학습되지 않아 결국 제출하지 못한 점도 아쉬움.

○ 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

다시 기본기를 잡는 것이 중요하다는 생각을 함. 왜냐하면, 어설프게 쌓아 올린 지식과 경험들 때문에 위와 같은 한계를 마주 했다고 생각하기 때문. 탄탄하게 실력을 쌓아 올려서 모든 시도가 자연스럽게 원하는 방향으로 흘러갔으면 함. 그리고, 다른 사람의 파이프라인 코드에 너무 많은 시간을 쓰기 보다 이제는 그것을 참고하여 스스로 파이프라인을 만들어야겠다고 생각함.

신상우

학습목표

- 초반에 배운 CF 모델들을 이해하고 적용해보자
- MF, ALS, Autoencoder 등

데이터

현재의 유저, 영화 인터렉션 정보에서 유저가 어떤 영화를 좋아하고 싫어하는지 알 수 없음. 단지 시청함/시청하지 않음의 정보만 있음

“유저가 같은 장르의 영화를 계속 보면 해당 장르의 영화를 좋아한다고 볼 수 있지 않을까?” 라는 생각으로 시작해, 다양한 영화 콘텍스트 정보를 활용하여 유저가 좋아하는 영화를 표현하기 위해 노력함. implicit feedback을 다루는 방법 중 confidence 개념을 적용해 유저가 많이 본 장르에 높은 가중치를 부여함

Multi-DAE & Multi-VAE

실험 배경

- CF 모델을 이용해서 우리 문제를 풀어보고자 함
- 전형적인 시퀀스 문제가 아니고, 앞 뒤로 유저가 좋아할 영화를 예측해야 하기 때문에 CF로 정적으로 예측했을 때 더 성능이 좋을 것으로 기대함.
- 이후 시퀀스 모델과 앙상블을 통한 성능 향상 기대
- 간단하게 구현할 수 있는 AE로 빠르게 실험하고 베이스라인 모델로 삼고자 함
- Autoencoder 기반의 Collaborative Filtering으로 딥러닝 모델의 장점을 CF 방식과 함께 사용할 수 있음
- 데이터를 압축해 사용자의 영화에 대한 선호 정보를 학습하고 불필요한 정보는 필터링하기를 기대함

실험 결과

1) Multi-DAE: public: 0.1380, private: 0.1384

2) Multi-VAE: public: 0.1364, private: 0.1378

베이스 라인 코드 작성

- 팀원들과 공통된 베이스라인 코드 위에서 실험하기 위해 우리만의 베이스라인 코드를 작성함
- DeepFM 모델을 위한 학습, 추론 과정을 작성함

배운점 다음 프로젝트 도전할 점

- RecBole, voting 앙상블
- CF와 MF 등 초반에 배운 추천 시스템의 중요한 모델들을 다시 학습하고 적용해봄
- implicit 피드백을 다루기 왜 어려운지, 어떻게 다뤄야할지 배움
- 지금까지 3번의 대회를 하며 모델링과 데이터 분석에 집중했으니, 최종 프로젝트에서는 다른 부분들도 경험해보고 싶음

이주연

학습 목표

개인 학습 목표 지난 대회를 회고하며 아쉬웠던 점을 추려내어 개인 목표를 설정함. 2개 이상의 **모델을 직접 구현**하는 것, **모델의 구현 목적**을 분명히하는 것을 목표로 하였으며, 특히 모델 구조나 compatibility function, loss를 직접 변경하여 실험하며 체계적으로 실험 관리를 부차적인 목표로 삼음

팀 학습 목표 모든 작업을 깃허브 이슈 및 공용 코드로 관리하고, 개인의 실험을 효율적이고 효과적으로 공유하는 것을 목표로 함

목표 달성을 위한 노력

- **개인 학습 측면** 추천 분야에서 잘 알려진 벤치마크 데이터셋이지만, 대회 목적에 맞게 많이 변형된 만큼, 대회에서 데이터를 어떻게 구성하였는지 제대로 파악하고자 함. 대회 목적에 적합한 모델을 선정한 뒤, 모델을 직접 구현하고, 발전된 접근 방식에 관한 논문을 직접 읽고, 구현해보며 실험하려 노력함
- **팀의 학습 측면** 주어진 베이스라인 코드가 특정 모델에 매우 치우치게 구현되어 있었고, 다른 모델에 대해 동작하도록 변형하기 매우 어려웠음. 모든 팀원이 쉽게 이해하고, 활용할 수 있도록 General Predictor 기반의 베이스라인 모듈 코드를 구성하려 노력함

모델 개선 방법

- **문제 파악** 이번 대회는 다음과 같은 특징이 있었음. 1) 평점 예측이 아닌 Top-N Ranking 문제였음 2) MovieLens Datasets은 영화를 점수로 평가한 데이터이지만, 여기서는 Implicit Feedback으로 변형함 3) 유저별로 맨 마지막 아이템과 시퀀스 내 일부 아이템을 테스트셋으로 설정함. 얼핏보면 시퀀셜 문제로 보이나, 데이터 특성 상 유저의 시청 이력이 아닌 평가 이력이므로 시퀀셜이 아닌, Implicit feedback에 특화된 모델로 푸는 것이 적절하다고 판단함
- **베이스라인 코드 구성** 문제 풀이에 앞서, 팀원 모두가 공유할 수 있는 베이스라인 코드를 작성하기로 함. 다양한 모델 중, side-information을 쉽게 추가하고, collaborative signal을 포착할 수 있는 general predictor인 FM을 기본으로 삼고 코드를 구성.
- **오토인코더 기반 모델 구현 및 실험** FM 모델 구현 뒤 몇 실험을 통해 side-information의 추가 시 사용하는 메모리 대비, 성능의 향상이 적어, 메모리를 효율적으로 활용하면서도 implicit feedback에 효율적인 오토인코더 기반의 모델에 집중하기로 결정함. AutoRec과 VAE, Multi-VAE 논문을 읽고, 각 논문의 수식과 실험을 참고하여 AE, DAE, VAE 모델을 우선적으로 구현함. AE < VAE < DAE 순으로 성능이 좋았는데, 이 대회의 경우, 일반적인 추천 데이터셋 대비 sparsity가 낮고 규모가 작아서 VAE 보다 DAE의 성능이 좋았다고 생각함
- **Confidence Level** 오토인코더에서 side-information을 직접적으로 활용할 수는 없지만, Implicit Feedback을 위한 MF 방법론 중 Confidence Level을 활용하면 side-information을 간접적으로 활용 가능하다고 생각함. 이 중에서도 시퀀셜 특성의 캡처를 기대하며 time 정보와 아이템의 인기도, 유저의 장르 선호도를 각각 적용해봄. 실험 결과, confidence level을 적용하였을 때 모두 성능이 5% 이상 개선되는 것을 확인함.
- **KL annealing** Multi-VAE를 읽고 KL term에 대해 선형적으로 증가하는 beta를 통해 초기 학습 시 정규화항의 영향을 줄이고자 하였는데, 다회의 실험을 통해 하이퍼파라미터의 설정이 까다롭고, beta를 사용하지 않을 때만큼 성능을 내기 어려웠음. MultiVAE를 개선한 RecVAE 논문 리뷰를 통해, Annealing 이 항상 잘 동작하는 것이 아니며, 다른 어떤 방법론 또한 주어진 문제 상황에 적용하여 실증적으로 의사결정을 해야겠다고 깨달음

잘한 점

- 페어프로그래밍을 통해 확장성이 보장된 베이스라인 코드를 작성하기 위해 노력한 점.
- 비교적 단순한 모델이었지만, 데이터 분포의 가정에 변화를 주고, 중요도 높은 데이터를 비중 높게 학습할 수 있도록 가중을 주는 등 Loss에 많은 변형을 주어 실험한 점. 변형을 줄 때 단순히 논문에서 제시한 것을 이식하는 것이 아니라 왜 필요한지에 대해 해석하려 노력한 점
- 실험 관리를 상세하고 보기 쉽게 진행한 점

아쉬운 점과 보완할 점

- 베이스라인 코드의 경우 팀 공유 측면에서 시작하였는데, 작성하고 나니 짧은 기간 내에 팀원 모두의 이해도를 높여 쉽게 사용하고 변형 가능하도록 전달하는 것에서 어려움이 있었음.
- 다양한 모델을 적용하여 비교 실험해보지 못한 점이 아쉬움

이현규

- 학습목표
  - recbole 라이브러리를 사용하여 데이터셋 맞추고 사용하는 방법
  - 비지도 학습 모델 사용 그리고 이해
- 학습목표 달성
  - recbole 사용은 아직 미숙함 추후 다른 데이터를 가지고 해볼 예정
  - 여러 모델 사용과 이해 달성. 하지만 코드 설계 미숙함.
- 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?
  - 데이터를 바라보는 시각도 굉장히 중요하다는 것을 많이 느꼈던 대회였음.
- 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?
  - 아직 많이 부족하다는 것을 많이 느꼈고, 다른 한편으로 슬펐던 것은 여러 모델들 하나씩 돌려서 가장 높은 것 툭 사용한 것이 김빠지게 했다. 이런 기분을 느낀 것이 아마 스스로 현재 가지고 있는 데이터를 바탕으로 어떻게 하면 더 좋은 성능을 나올 수 있도록 모델을 바꿀 수 있을까? 와 같은 고민에 답을 하지 못했기 때문이 아닐까? 생각함.

- 앞으로의 성장
  - 스스로 코드 설계하는 방법도 중요하겠지만 누군가 만들어 놓은 라이브러리를 사용하는 방법을 익히는 것도 중요하다고 생각함.
  - 주피터노트북으로만 만들지 말고 파이썬 파일로 만드는 연습도 해볼 예정.
  - 이번 대회와 같은 데이터를 많이 경험해 볼 것.

이현주

학습 목표

- 이전 레벨에서 부족했던 점을 개선하기 위해, 지속적인 학습을 진행하고자 함.

목표 달성을 위한 노력

- 강의를 들으면서 필기를 하고, 매 강의마다 한 페이지 씩 요약하는 시간을 가짐.
- 관련 논문을 읽고 블로그에 리뷰를 올림.

모델 개선 방법

- 서베이를 통해 Sequential Recommendation 문제에서 비교적 높은 성능을 내고 있는 GRU4Rec 을 선택하여 논문을 읽고 구현함
- GRU4Rec 의 입력 단에서 영화의 줄거리 임베딩을 추가하는 방식으로 개선하여 추천 성능을 향상시킨 GRU4RecBE 의 아이디어를 차용하여, 영화의 제목 임베딩을 추가하는 방식으로 모델을 개선함

잘한 점

- 논문을 그대로 구현한 것이 아니라, 사용하는 데이터와 풀고자 하는 문제에 바꿔서 구현함.
- 추천 시스템의 흐름을 이해하고자 관련 논문을 많이 읽었음.

아쉬운 점과 보완할 점

- 제출 기회를 충분히 쓰지 못한 것이 아쉬움.
- 디버깅에 너무 많은 시간을 사용함.

조성홍

1. 프로젝트 기간 집중한 부분

- 베이스라인 코드 구축: 지난 대회들에서 아쉬웠던 실험 요소들의 자동화된 취합을 해결하기 위해 베이스라인 코드를 직접 구축해보는 것을 목표로 진행.
- 다양한 종류의 모델을 실험해보며 모델 종류별 특징을 비교 정리해보고자 함.

2. 잘했던 부분

- 페어 프로그래밍을 통해 코드를 만들어가는 경험을 해본 것. 혼자 진행할 때는 비교적 추상적인 구현 목표로 구현을 시작해 중간 중간 목표를 수정하는 일이 잦았는데, 페어 프로그래밍을 하는 과정에선 구현 목표를 구체적으로 정하고 시작해 중간 수정 과정이 줄어 효율적인 작업이 가능했음. 구현 목표를 구체화하는 과정에서도 서로 다른 관점에서 의견을 주고 받으며 개념을 되짚어보는 계기로 삼을 수 있었음.

3. 아쉬웠던 부분

- 베이스라인 코드 작성시 목표로 했던 '실험 요소 관리 자동화'에서 부분적인 성과가 있었지만, 처음 기대했던 코드의 수정을 최소화하는 정도로는 구현을 못해 아쉬움.
- 또한 최대한 다양한 종류의 모델을 실험해보고 특징을 비교/정리해보자는 다른 목표는 수행을 못 해 아쉬움. 예상보다 베이스라인 코드 작성에 대부분의 시간을 할애했고, 중간에 방향을 수정해보려 했으나, Task에 맞게 모델과 데이터를 설계하고 구현하는 난이도가 높아 거의 진행하지 못 함.
- RecBole 라이브러리를 활용해보지 못 한 것도 아쉬운 부분.

4. 시도해 보고 싶은 부분

- 이번 베이스 라인 코드 구축 과정에서 설계했던 내용을 정리하면서, 각 구조별로 설계상 이점과 약점을 정리해볼 계획. 이를 토대로 원래 목표로 했던 수준으로 추가적인 디벨롭을 해보고 싶음.
- RecBole 라이브러리에 구현된 모델 중 하나를 정해서, 논문을 읽고 RecBole을 사용해 실험을 진행해보고 싶음. 더 나아가 RecBole과 동일한 결과를 낼 수 있는 코드를 직접 구현해보고 싶음.