개인화 추천시스템

Final Project

1조

DA 2022311934 장서호

DA 2022311907 이종우

DA 2022311930 김영채

DA 2022311916 조규원

INDEX

INTRO

Goal

RMSE값과 f1값을 기준으로 rating을 가장 잘 예측하는 최적 모델 찾기

Try1 : 해당 Dataset에 가장 높은 성능을 보이는 기존 모델& 새로운 모델 조합(Hybrid) 탐색 및 적용
Try2 : Metapath2vec을 이용하여 embedding 후 LSTM으로 rating을 예측하는 알고리즘 개발

기존 Rating 데이터

파생변수

	item_id	user_id	rating	year	month	day	season
0	A0955928C2RRWOWZN7UC	B00191WVF6	4.0	2017	2	17	0
1	A0955928C2RRWOWZN7UC	B005WY3TMA	4.0	2015	6	14	2
2	A0955928C2RRWOWZN7UC	B0090XWU8S	4.0	2017	4	22	1
3	A0955928C2RRWOWZN7UC	B00FXYTLIK	4.0	2015	7	10	2
4	A0955928C2RRWOWZN7UC	B00HMZG3YS	4.0	2015	7	10	2
...	...	...	...	...	...	...	...
99737	AZZYW4YOE1B6E	B009AYLDSU	5.0	2013	12	10	0
99738	AZZYW4YOE1B6E	B00E055H5O	4.0	2015	5	25	1
99739	AZZYW4YOE1B6E	B00E8HGWIK	5.0	2013	12	1	0
99740	AZZYW4YOE1B6E	B00M58CMTM	5.0	2014	10	8	3
99741	AZZYW4YOE1B6E	B00VWVZ0V0	4.0	2016	12	27	0

99742 rows × 7 columns

```
for i in range(len(ratings)):
    if ratings['season'][i] in winter:
        ratings['season'][i] = 0

    elif ratings['season'][i] in spring:
        ratings['season'][i] = 1

    elif ratings['season'][i] in summer:
        ratings['season'][i] = 2

    elif ratings['season'][i] in fall:
        ratings['season'][i] = 3
```

user_id / item_id / rating

timestamp 기반
Year / month / day / season

EDA & IDEA

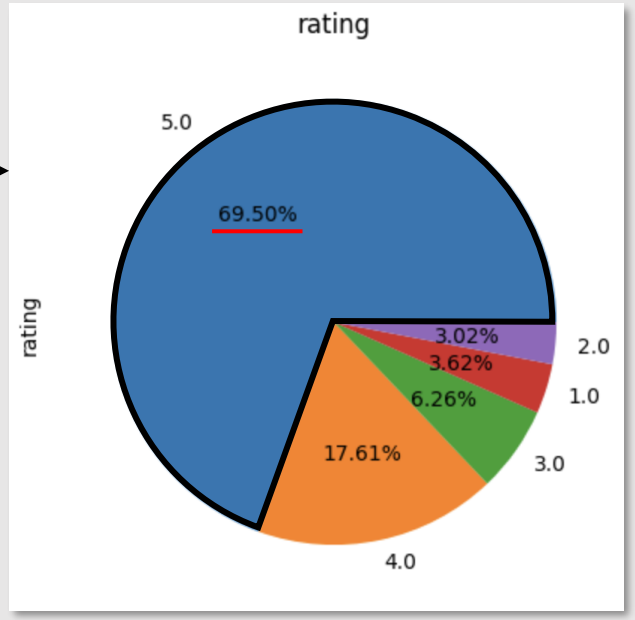
[Rating]

5점이 약 69.50%로 가장 많음, 평균은 4.46

[Timestamp]

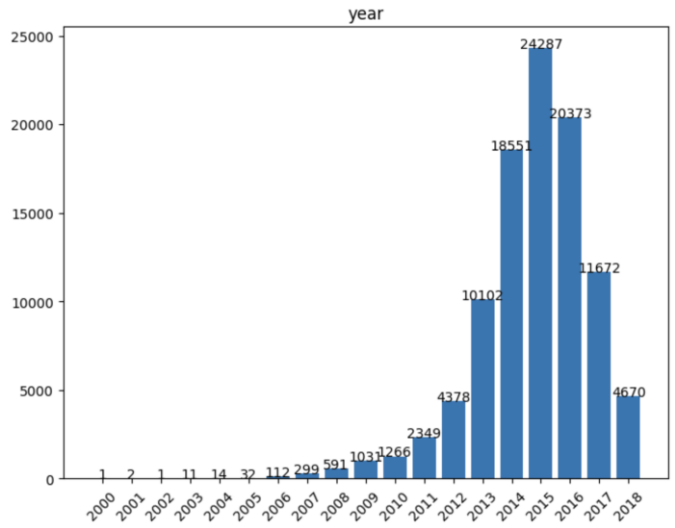
연도, 월, 계절에 따른 구분

[rating]

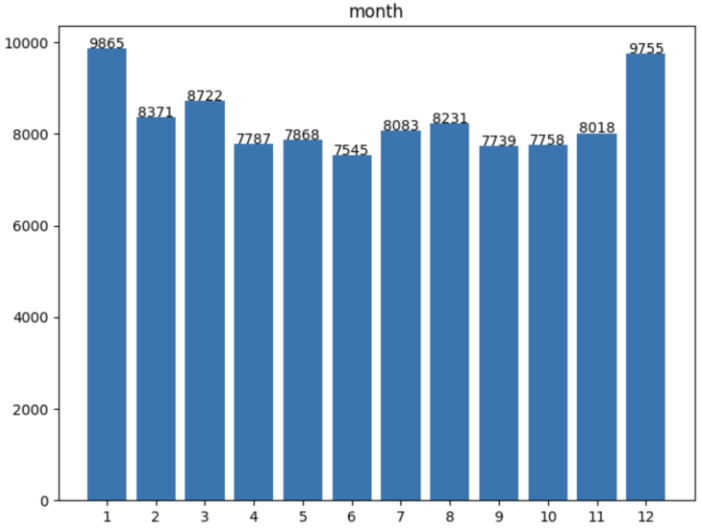



```
count    99742.000000
mean         4.463305
std          0.994834
min          1.000000
25%          4.000000
50%          5.000000
75%          5.000000
max          5.000000
Name: rating, dtype: float64
```

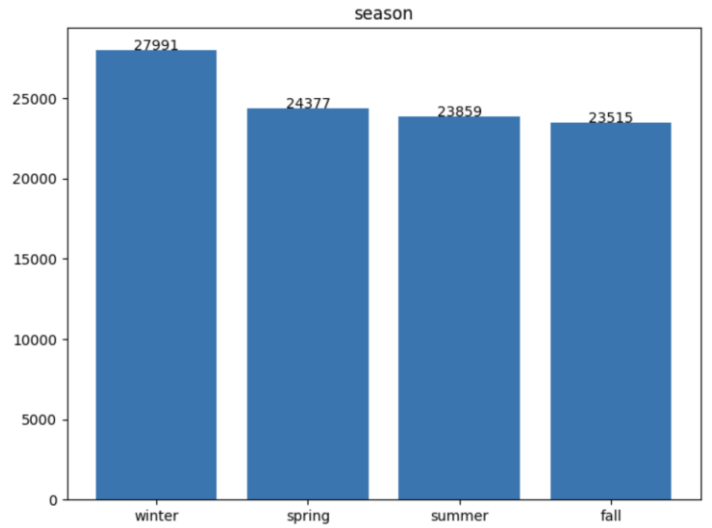
[year]



[month]



[season]



Metapath2vec with LSTM

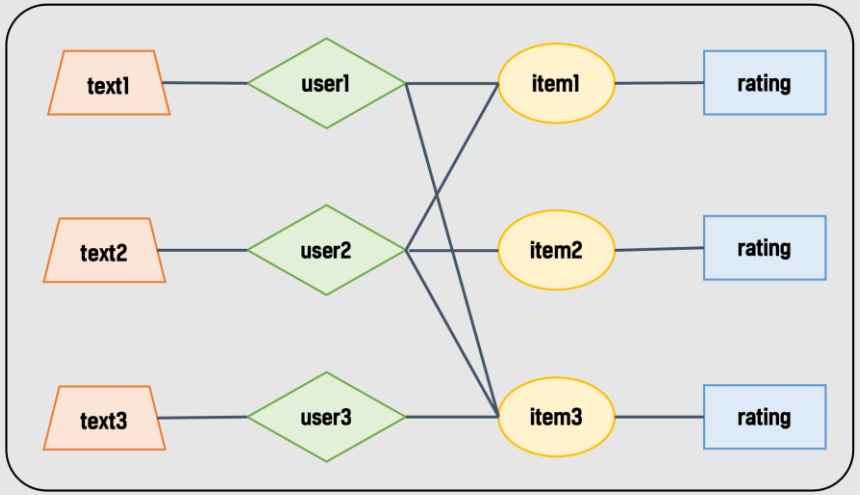
Heterogeneous Graph
Homogeneous Graph와 달리 여러 종류의 node와 edge를 가진 Graph로 우리가 현실에서 쉽게 접하는대부분의 Graph

Metapath2vec
Metapath를 베이스로 Heterogeneous Graph 구조를 저차원의 벡터로 임베딩하는 방법

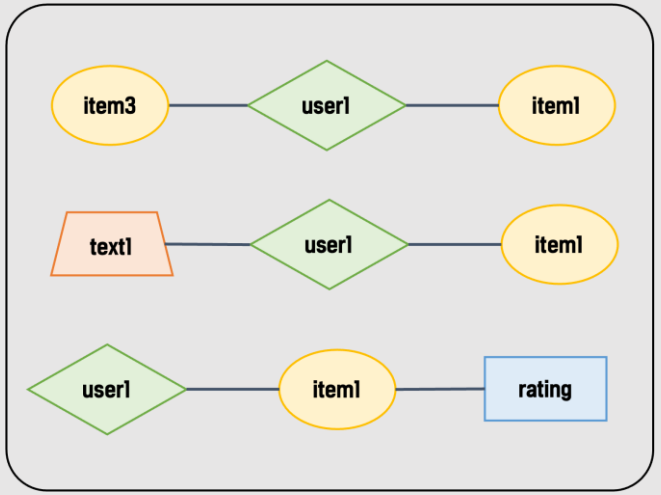
Metapath

$$n_0 \xrightarrow{e_1} n_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{m-1}} n_{m-1} \xrightarrow{e_m} n_m$$

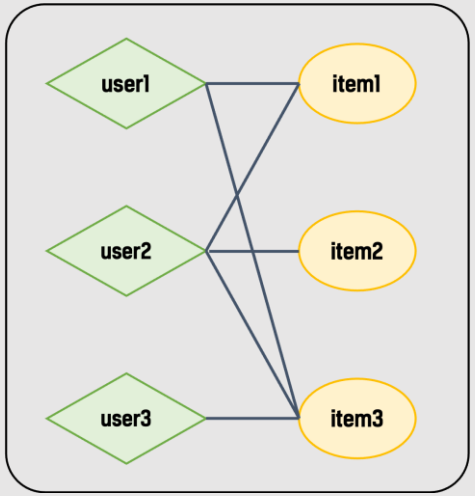
$$o_0 \xrightarrow{l_1} o_1 \xrightarrow{l_2} \cdots \xrightarrow{l_{m-1}} o_{m-1} \xrightarrow{l_m} o_m$$



[Heterogeneous Graph]



[Metapaths]



[Metapath-based Graph]

Metapath2vec with LSTM

1. Metapath2vec을 이용하여 item들을 embedding

```
metapath = [('item','purchased_by','user'),
            ('user','purchase','item')]
```

2. 각 item embedding 뒤에 rating score를 one-hot encoding으로 붙여 user를
특정할 수 있게 만들고, item embedding을 시간 순으로 정렬하여 user의 구매 내역 생성

```
res_dict[group_name].append(cur_embedding+rate_embedding)
```

3. 몇 개 이상의 sequence를 가진 user만 사용할건지 정하기

```
for idx in range(len(set(res_dict.keys())  & set(user_list))):
    _id = user_list[idx]
    if len(res_dict[_id]) > 3:  # 몇개 sequence 이상만 남길거냐 (중요)


def __init__(self, dim_in, dim_h, dim_out, lstm_layer=4, seq_length=4):
```

Metapath2vec with LSTM

4. LSTM에 넣을 때 몇 개의 sequence를 기준으로 학습시킬 것인지 정하기
5. 동일 user의 embedding인지 아닌지 구분할 수 있게 학습

```
if len(pos_idx_list) != 0:
    pos_h = h[pos_idx_list]
    pos_loss = -1 * torch.log(pos_h + 1e-7).mean()
```

```
if len(neg_idx_list) != 0:
    neg_h = h[neg_idx_list]
    neg_loss = -1 * torch.log(1-neg_h + 1e-7).mean()
```

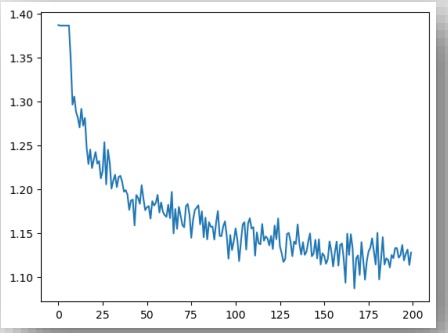
1) 동일 user이면 pos_loss
2) 다른 user이면 neg_loss

6. 학습시킨 LSTM을 이용해 만들어진 user embedding을 이용 / cf_knn으로 user의 rating 예측
7. 아이디어로 embedding된 거리에 반비례하게 기록을 반영하도록 모델 생성

```
dist_f = 1 / (euc_dist_matrix1 + 1e-7)

residual = np.dot(item_ratings, dist_f) / dist_f.sum()

mean_rating = user_mean + residual
```



BaseModel

모델	특이사항	RMSE
CF	(4.0)	1.0819
CF_KNN	K:30, mean_rating=4.0	1.0766
IBCF	mean_rating=4.0	1.1255
IBCF_KNN	K=30, mean_rating=4.0	1.1283
UBCF	prediction = 3.0	1.0578
UBCF	prediction = 4.0	1.0578
UBCF	prediction = 5.0	1.0460
UBCF_KNN	prediction = 3.0, neighbor = 30	1.0472
UBCF_KNN	prediction = 4.0, neighbor = 30	1.0557
FM		0.9229
FM	파생변수	0.9261
MF		0.9232
DL		1.009
AutoEncoder		0.9650
SVD		0.9350
MF + IBCF		F1:0.0135
MF + DL		0.9204
CF+MF+DL		0.9206
metapath2vec	with LSTM	0.9654

이론 상
Hybrid(CF+MF+DL)는
Hybrid(MF+DL)를 포함하는 Model이므로
성능이 무조건 더 잘나와야함.
(search condition 0.00~1.00)

동일 조건에서 재실험
random_state = 12로 Fixed 시키고 재확인

```
최고의 황금비율
MF : '0.90'
ubcf_bias_knn : '0.08'
DL : '0.02'
RMSE = 0.9352549
```

```
Weights - 0.91 : 0.09 ; RMSE = 0.9380283
Weights - 0.92 : 0.08 ; RMSE = 0.9375574
Weights - 0.93 : 0.07 ; RMSE = 0.9371587
Weights - 0.94 : 0.06 ; RMSE = 0.9368324
Weights - 0.95 : 0.05 ; RMSE = 0.9365785
Weights - 0.96 : 0.04 ; RMSE = 0.9363970
Weights - 0.97 : 0.03 ; RMSE = 0.9362881
Weights - 0.98 : 0.02 ; RMSE = 0.9362518
Weights - 0.99 : 0.01 ; RMSE = 0.9362880
```

CF+MF+DL
0.9352

MF + DL
0.9363

→ 동일 조건에서 더 나은 성능 확인

Model

Hybrid(CF+MF+DL)

탐색 범위

CF : min 0.00, max 1.00
MF : min 0.00, max 1.00
DL : min 0.00, max 1.00

하이퍼 파라미터 튜닝

[MF] : tolerance = 1e-7로 줄여서 적용

[ubcf_bias_knn] : 15~45, 3step으로 최적 k값 탐색

[DL] : 기본 코드 사용

1회차	2회차	3회차
Best Ratio	Best Ratio	Best Ratio
MF 0.88	MF 0.88	MF 0.87
CF 0.10	CF 0.09	CF 0.11
DL 0.02	DL 0.03	DL 0.02
RMSE 0.9145	RMSE 0.9233	RMSE 0.9303

DistilHybrid(CF+MF+DL)

탐색 범위

CF : min 0.00, max 0.15 → Search Range 85% 감소
MF : min 0.75, max 1.00 → Search Range 75% 감소
DL : min 0.00, max 0.12 → Search Range 88% 감소

하이퍼 파라미터 튜닝

[MF] : tolerance = 1e-7로 줄여서 적용

[ubcf_bias_knn] : 15~45, 3step으로 최적 k값 탐색

[DL] : 기본 코드 사용

1회차	2회차	3회차
Best Ratio	Best Ratio	Best Ratio
MF 0.91	MF 0.91	MF 0.91
CF 0.07	CF 0.07	CF 0.08
DL 0.02	DL 0.02	DL 0.01
RMSE 0.9252	RMSE 0.9206	RMSE 0.9160

Model

DistilHybrid(CF+MF+DL)

```
## 3중 for문
box_w = [0.75, 0.76, 0.77, 0.78, 0.79, 0.80, 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.90, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.0]
box_x = [0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10, 0.11, 0.12, 0.13, 0.14, 0.15]
box_y = [0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10, 0.11, 0.12]
best_RMSE = 10 #그냥 큰 수치 임의로 지정
for x in box_w: # x는 MF
    for y in box_x: # y는 CF
        for z in box_y: # z는 DL
            if x+y+z == 1:
                try:
                    weight = [x,y,z]
                    predictions = []
                    for i, number in enumerate(result0):
                            predictions.append(result0[i] * weight[0] + result1[i] * weight[1]+ result2[i] * weight[2])
                    #print("Weights - %.2f : %.2f : %.2f ; RMSE = %.7f" % (weight[0], weight[1], weight[2], RMSE2(ratings_test['rating'], predictions)))
                    rmse2 = RMSE2(ratings_test['rating'], predictions)
                    if best_RMSE > rmse2:                       # New best record
                        best_RMSE = rmse2
                        best_ratio = " 최고의 황금비율 \n MF : '%.2f' \n ubcf_bias_knn : '%.2f' \n DL : '%.2f' \n RMSE = %.7f" % (weight[0], weight[1], weight[2],
RMSE2(ratings_test['rating'], predictions))
                except:
                    pass
print(best_ratio)
```

Lessons Learned

①Ratings데이터만 사용

주어진 review 데이터를 활용하지 못함

Review의 text를 활용하여 감성분석을 활용한 파생변수 등 다양한 방법이 존재views 데이터를 활용하지 못함

②새로운 추천시스템 모델을 적용해 보지 못한 것

Embedding한 것을 cf_Knn에만 적용

AFM,MAML 등 여러 추천시스템 모델 적용

③User embedding을 시도했을 때, LSTM sequence의 최적값을 찾지 못한 것

count	3706.000000
mean	18.839450
std	23.478186
min	3.000000
25%	9.000000
50%	12.000000
75%	19.000000
max	301.000000

Thank You