데이터분석캡스톤디자인

12주차 수행보고

Khupid 조

산업경영공학과 김동혁
관광학과 류연주
산업경영공학과 유정수

이번주 to-do list

1. NCF 파라미터 tuning -> O
    a. Early Stopping
    b. epoch, batch size, n_factor
    c. layer size
    d. learning rate

2. contents-based matrix 각자 가져오기 & Data Normalization -> O
    a. 정규화([1] 유저가 클러스터에 남긴 단어 개수/클러스터 단어
    b. [1]들을 모두 더해서 1로 만들고 최종적으로 비율로 나타냄)

3. 추가 데이터 수집 및 전처리 -> O

4. hybrid model 실험 -> O

matrix 정규화의 필요성

각 Group에 속한 word의 갯수가 다름
각 User별 리뷰의 양이 다름

→ Group별, User별 편향된 데이터를 정규화 시킬 필요가 있음

	30	12	2	3	5	5	3	2	2	9	7	3
	group_7	group_0	group_1	group_2	group_3	group_4	group_5	group_6	group_8	group_9	group_10	group_11
p_id												
232	1719	16	6	0	0	7	7	1	0	0	6	3	1
123	899	16	1	3	0	8	7	1	0	1	6	3	0
273	283	16	6	1	1	5	2	0	0	1	4	0	0
326	303	8	7	2	0	9	7	2	0	0	6	1	0
107	1754	15	6	3	1	7	6	0	0	0	5	3	2
183	719	21	5	3	1	9	6	5	0	1	4	5	1
162	1125	20	10	2	1	8	10	2	0	0	2	2	0
326	2086	19	2	1	0	4	8	1	0	0	1	0	1
207	769	12	4	2	1	7	6	1	0	0	2	1	2
183	1288	11	3	0	1	7	5	0	0	0	3	4	0

cluster word 수

User별
사용 word수

Cell(i,j) = Cell(i,j) / (group j의 word 갯수) , 각 셀의 숫자를 해당 cluster가 가지고 있는 word 수로 나눔

↓

Cell(i,j) = Cell(i,j) / Σ Cell(i,j)

위에서 계산한 값을 해당 row의 비율로 환산

NCF 라이브러리 비교

	Microsoft / NCF	Cornac / NCF
Early Stopping	기능 없음	모듈로 구현
cross_validation	기능 없음	모듈로 구현 (k-fold 등)
Hyper-parameter tuning	- batch_size, - learning_rate - layer_size 등 > 기본적인 파라미터 튜닝 가능	- 기본적인 파라미터 튜닝 외 - gridSearch - randomSearch 사용가능

Fine tuning 진행중

	Library 1	Library 2
먹을 곳	F1-score: **0.021004** Recall@10: 0.027271 Precision@10: 0.017080	F1-score: **0.0223** Recall@10: 0.0293 Precision@10: 0.0181
갈 곳	F1-score: **0.311442** Recall@10: 0.438293 Precision@10: 0.241537	F1-score: **0.4033** Recall@10: 0.6282 Precision@10: 0.2670
볼 곳	F1-score: **0.062483** Recall@10: 0.065089 Precision@10: 0.060079	F1-score: **0.0672** Recall@10: 0.0382 Precision@10: 0.3371

item-based model test

cornac.model.UserKNN
cornac.model.ItemKNN

```
(0.0, 3238.0, 4.5),
(0.0, 5144.0, 4.0),
(0.0, 2126.0, 5.0),
(0.0, 6026.0, 3.0),
(0.0, 1158.0, 4.0),
(0.0, 1350.0, 4.0),
(0.0, 472.0, 3.5),
(0.0, 4074.0, 4.0),
(0.0, 3602.0, 3.5),
(0.0, 2219.0, 4.0),
(0.0, 3180.0, 3.5),
(0.0, 1254.0, 4.5),
```

input
user-item-rate

```
TEST:
...
                        |   RMSE | Train (s) | Test (s)
----------------------- + ------ + --------- + --------
UserKNN-Cosine          | 0.6058 |    0.1297 |   1.5848
UserKNN-Pearson         | 0.6370 |    0.1316 |   1.3743
UserKNN-Amplified       | 0.6058 |    0.6493 |   1.2865
UserKNN-IDF             | 0.6062 |    0.1436 |   1.2756
UserKNN-BM25            | 0.6060 |    0.1576 |   1.3454
ItemKNN-Cosine          | 0.7011 |    0.1007 |   1.2257
ItemKNN-Pearson         | 0.7108 |    0.1376 |   1.3304
ItemKNN-AdjustedCosine  | 0.6625 |    0.0848 |   1.2178
```

Cosine similarity

user 1에 대해 각 아이템에 대해 예측한 결과

```
user_knn_idf.score(user_idx=1)
```

```
array([3.770827  , 4.09432958, 3.62831448, 2.66654354, 2.71076991,
       3.89011849, 3.97864726, 3.60349834, 4.18342009, 2.89956895,
       3.58482387, 3.6004619 , 3.58425655, 3.37179816, 3.24161365,
       3.5755642 , 3.60158377, 3.29004869, 3.18406739, 3.80199554,
       3.62148065, 3.57895428, 4.12925059, 4.0508625 , 3.90938915,
       3.57330311, 3.86010928, 3.27580288, 3.39701155, 3.64364341,
```

다음 주 계획

1. **contents-based**
    a. 모델/라이브러리 선정
    b. 직접 구현

2. **Hybrid method 적용 후 최종 결과 도출**

3. **계속해서 NCF 모델 성능 개선**