RS

20.08.14

컴퓨터과학전공 1715237 이혜승

Collaborative Filtering

- 여러 사용자들이 제공하는 ratings 정보 활용
- Main challenge: sparsity.(user와 item의 수가 계속 증가하면서, unobserved ratings가 너무 많음)
- 1) MEMORY BASED methods(Nearest neighbor-based)

  : 귀납적. 체험적. 설명력good.

  : user가 item에 매긴 점수(ratings) 혹은 구매이력(0 또는 1) 등의 user behavior를 기반으로 추천해주는 것.

  : user-item matrix를 생성한 후, 사용자가 아직 평가하지 않은 item을 예측하는 문제가 됨 (pivot table로 지원됨)

  row- user, column –item

  (+) 아이템 특성에 의존하지않으므로, 아이템 추천이 다양하게 이루어질 수 있음

  (-) sparse ratings matrices에는 잘 작동하지 않음.

  (-) 새로운 아이템 / 새로운 유저 추천 불가 → 사용자가 프로파일을 미리 쌓아두도록 유도(가입을 하고 선호도를 입력)

  - 아이템 기반: 비슷한 item (**비슷한 rating 분포**를 갖고 있는) 을 소비한 고객들이 구매한 상품들 추천
  - 사용자 기반: 비슷한 고객(**평가했던 item에의 ratings가 비슷**)이 구매한 상품을 추천

Collaborative Filtering

- 여러 사용자들이 제공하는 ratings 정보 활용
- Main challenge: sparsity.(user와 item의 수가 계속 증가하면서, unobserved ratings가 너무 많음)
- 1) MEMORY BASED methods(Nearest neighbor-based)

	ITEM1	ITEM2	ITEM3	ITEM4	ITEM5
USER1	1	1	1	0	0
USER2	0	0	0	1	1
USER3	1	1	0	0	0
USER4	0	0	0	0	1

Target User

User-based CF

User-item matri의 row기준으로, 유사성을 비교.(Pearson correlation / cosine similarity)

타겟 유저가 user3이면, user3와의 코사인 유사도를 각각 계산한다.
차례로 0.81, 0, 1, 0값 → user3와 가장 유사한 user는 **user1**

결과적으로, user1은 봤지만 user3는 보지않은 item3를 추천해준다.

Collaborative Filtering

- 여러 사용자들이 제공하는 ratings 정보 활용
- Main challenge: sparsity.(user와 item의 수가 계속 증가하면서, unobserved ratings가 너무 많음)
- 1) MEMORY BASED methods(Nearest neighbor-based)

	ITEM1	ITEM2	ITEM3	ITEM4	ITEM5
USER1	1	1	1	0	0
USER2	0	0	0	1	1
USER3	1	1	0	0	0
USER4	0	0	0	0	1

Target User: user4

Item-based CF
User-item matrix의 column기준으로, 유사성을 비교 .(Pearson correlation / cosine similarity)

타겟 유저가 user4이면, user가 본 *item5와 비슷한 item*을 추천하고자 한다.
따라서, col기준으로 item의 코사인 유사도를 계산하면, 차례로 0, 0, 0, 1, 0.71
→ item5와 가장 비슷한 item은 **item4**

결과적으로, user4가 본 item과 가장 비슷한 item인 item4를 추천해준다.

Collaborative Filtering

- 여러 사용자들이 제공하는 ratings 정보 활용
- Main challenge: sparsity.(user와 item의 수가 계속 증가하면서, unobserved ratings가 너무 많음)
- 2) MODEL BASED methods
  : 머신러닝을 활용해서 평점을 예측하는 모델을 구축하는 방식

  과거 사용자의 평점 데이터를 이용하여 모델링 → 평점 정보가 없어도 item에 대한 user의 rating을 예측할 수 있음.

  - 행렬분해기법(Matrix Factorizaion)
  - 특이값 분해(SVD), PCA
  - 연관규칙분석(Association Rule Mining)
  - Probabilistic Models: 클러스터링, 베이지안 네트워크
  - Regression, Deep Learning, SVM . . .etc. . ..

Contents-based Filtering

- 아이템의 feature에 기반하여 비슷한 아이템 추천.
- 타겟 사용자가 특정 아이템을 선호 → 해당 아이템과 유사한 아이템을 추천
- Only 아이템 특성만을 이용!
  (+) 다른 사용자 영향 x
  (+) 새로운 아이템 추천 가능
  (-) feature를 만들어 줘야하는 것이 어려울 수 있음(metadata)
  (-) 계속 비슷한 아이템만 추천 (다양성x)

Item-based CF vs. CB

- 아이템 기반 collaborative filtering과 Contents-based filtering의 차이점을
- CF기법은 기본적으로 **user와 item의 interaction matrix**를 활용한다.
- 반면 CB는 user가 어떤 점수를 매겼는지가 아닌, item의 특성 자체를 기반으로 추정하는 것이다.

	ITEM1	ITEM2	ITEM3	ITEM4	ITEM5
USER1	1	1	1	0	0
USER2	0	0	0	1	1
USER3	1	1	0	0	0
USER4	0	0	0	0	1

[Item-based Collaborative Filtering]

USER4가 본 ITEM과 가장 비슷한 ITEM인 ITEM4를 추천해준다.

	feature1	feature2	feature3	feature4
ITEM1	0	0	1	1
ITEM2	0	1	0	1
ITEM3	1	1	0	1
ITEM4	0	0	0	0
ITEM5	0	0	0	1

[Contents-based Filtering]

EX)ITEM2와 ITEM3가 비슷하므로,
ITEM2를 본 USER에게 ITEM3를 추천해줄 수 있다.

Collaborative Filtering

- CF python으로 구현 1) item-user matrix 생성

```
"""
item-based CF를 위한 item-user matrix 생성
: cold start에 취약한 CF기법. → 어느정도 정보가있는 user와 item에 대해서만 matrix생성
"""
def item_user_df_generator(metadata, read, from_dt, to_dt):
    # users_read: 일정 기간 동안 users가 읽은 article의 수 (중복 제거)
    partial_read = read[(read.date >= from_dt) & (read.date <= to_dt)]
    users_read = partial_read[['user_id','article_id']].drop_duplicates().groupby('user_id').count()

    # not_cold_start_users(ncsu): 일정 기간 동안 읽은 article의 수가 평균보다 높은 users(list형태)
    not_cold_start_users = users_read[users_read['article_id'] > users_read['article_id'].mean()].index.tolist()
    # not_long_tail_items(nlti): 최근 조회수가 상위 5%인 article(item)
    not_long_tail_items = metadata[metadata['recent_view'] > metadata['recent_view'].quantile(0.95)]['id'].tolist()

    # incsu와 nlti에 대해서 item-user matrix 생성
    iu = read[read['user_id'].isin(not_cold_start_users) & read['article_id'].isin(not_long_tail_items)]
    iu = iu[['user_id','article_id']].drop_duplicates()
    iu['read'] = 1 # iu 테이블에 있는 user-item은 전부 사용자가 해당 글 읽은 것이니까, 1 로 넣어둠.(implicit rating?)

    # iu_df: pivot table로 item-user matrix 생성
    # 행: 글 / 열: 사용자 / 값: read(읽었는지(1) 아닌지(null))
    iu_df = iu.pivot(index='article_id', columns='user_id', values='read')
    iu_df = iu_df.fillna(0) # 안읽었으면 0

    return iu_df
```

Collaborative Filtering

- CF python으로 구현 2) item-based CF구현

```
"""
item-based CF 추천
: 유사도 구해서, 가장 비슷한 N개 아이템의 가중합 -> 예측.
"""
def collaborative_filtering(idx, metadata, read, r_list, recommended, adjusted, from_dt, to_dt):
    # Step 1 : item-user matrix 생성
    iu_df = item_user_df_generator(metadata, read, from_dt, to_dt)

    # Step 2: item-user matrix에서 item에 대해 cosine similarity 구하기
    cosine_array = cosine_similarity(iu_df, iu_df)
```

Collaborative Filtering

- CF python으로 구현 2) item-based CF구현

```
    # Step 3: 가장 비슷한 100개의 item의 weighted mean을 이용해 predict
    predicted_array = np.zeros(shape=(len(iu_df.index),len(iu_df.columns))) # 사이즈: (선정한 user수 31645, 선정한 item수 18652)

    for i in range(len(cosine_array)):# 유사도 행렬 행: 31645, 열: 31645
        # 뒤에서 101번째부터 끝까지
        # 처음부터 끝까지 역순으로 1칸간격
        # 즉, 유사도 가장 높은 1번 ~ 100번순서대로 index출력
        top_100 = cosine_array[i].argsort()[-101:][::-1]
        top_100 = np.delete(top_100, 0) # 0번째 삭제(자기자신)

        weighted_sum = np.array([0])
        for top_idx in top_100:
            weighted_sum += (cosine_array[i][top_idx] * iu_df.values[top_idx])
        predicted = weighted_sum / len(top_100)
        predicted_array[i] = predicted

    iu_predicted = iu_df.values*(-99999) + predicted_array
```

Collaborative Filtering

- CF python으로 구현 2) item-based CF구현

```
# Step 4: 각 user에 대해 weighted mean이 높은 상위 100개 article을 저장
    cf_dic = {}
    for i in range(len(iu_predicted.T)):
        cf_dic[iu_df.columns[i]] = iu_df.index[iu_predicted.T[i].argsort()[-100:][::-1]].tolist() # 뒤에서 100번
째 역순으로 1칸씩 값만 뽑아내기


    cf_based_recommend_list = []

    if idx in cf_dic.keys():
    #key값 즉, 사용자 아이디에 idx가 있다면 --> 추천 항목에 넣음.
        already = r_list + recommended
        n_rec = int((100-len(recommended)) * adjusted)
        cf_based_recommend_list = pd.Series(cf_dic[idx])[pd.Series(cf_dic[idx]).isin(already)==False].tolist()[:n_
rec]

    return cf_based_recommend_list
```

Brunch data

- 추천모델

1. collaborative_filtering
   : item-based CF 추천 - 유사도 구해서, 가장 비슷한 N개 아이템의 가중합 -> 예측.
2. popularity based recommend
   : 일정 기간동안 조회수가 높은 인기글 기반 추천
3. popularity based recommend2
   : 전체 기간동안 조회수가 높은 인기글 기반 추천
4. following based recommend
   : 구독 작가 기반 추천(target user가 최근 / 전체 기간 동안에 읽은 글 중에서 구독작가 글의 비율을 고려하여 추천)
5. following based recommend2
   : 구독 작가 기반 추천(target user가 구독하는 작가의 글을 추천)
6. magazine based recommend
   : 매거진 기반 추천(target user가 최근 / 전체 기간 동안에 읽은 글 중에서 매거진 글의 비율을 고려하여 추천)
7. tag based recommend
   : 글의 태그 기반 추천 (target user가 최근 또는 전체 기간 동안 읽은 글들에서 자주 나오는 태그를 고려하여 추천, 즉 유저의 interest를 태그로 파악)

Brunch data

- 추천 순서

f1: metadata_hot에 대해 following_based_recommend을 이용하여 'recent_view(최근 조회수)' 순으로 추천

f2: metadata_reg에 대해 following_based_recommend을 이용하여 'reg_ts(발행 시간)' 순으로 추천

f3: metadata_hot에 대해 following_based_recommend2을 이용하여 'recent_view(최근 조회수)' 순으로 추천

f4: metadata_reg에 대해 following_based_recommend2을 이용하여 'reg_ts(발행 시간)' 순으로 추천

p1: metadata_pop에 대해 popularity_based_recommend을 이용하여 추천

cf: metadata(원본)에 대해 collaborative filtering을 이용하여 추천

m1: metadata_hot에 대해 magazine_based_recommend을 이용하여 'recent_view(최근 조회수)' 순으로 추천

m2: metadata_all에 대해 magazine_based_recommend을 이용하여 'reg_ts(발행 시간)' 순으로 추천

t: metadata_hot에 대해 tag_based_recommend을 이용하여 'recent_view(최근 조회수)' 순으로 추천

p2: metadata(원본)에 대해 popularity_based_recommend2을 이용하여 추천 (100개가 되지 않았을 경우)

구독작가에 대한 추천을 가장 중요하게 -> 인기 글 -> CF(비슷한 아이템 글) -> 매거진 -> 비슷한 태그 글 -> 인기글

Future work

- 저널: 추천시스템 기법 연구동향 분석 - 손지은(고려대학교), 김성범(고려대학교), 김현중(서울대학교), 조성준(서울대학교), 2015.04

(링크: http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE06270308)

- 해당 저널에서 알고리즘 재숙지
- 어떤 알고리즘에는 어떠한 문제가 있고, 그 문제를 해결하기 위해 등장한 방안에 대한 연구들이 함께 제시됨

- P190 **신규 논문** - 논문의 키워드 활용. 나이브 베이즈 모델 활용해서 추천

논문추천 시스템에서 평가정보나 이용정보가 부족한 신규논문을 사용자에게 추천하기 위해서 논문 이용 정보 대신 논문의 키워드를 활용하는 방식을 제안하였다. 신규논문이 입수되면 신규논문의 키워드와 사용자들이 기존에 이용한 논문의 키워드를 비교하여 각 사용자들에게 추천될 확률을 계산한 뒤, 가장 추천 될 확률이 높은 N명의 사용자들에게 신규논문을 추천하게 된다. 이때 사용된 알고리즘은 문서 분류에 있어 대표적으로 사용되는 나이브 베이즈 모델이다.

→ 추천 시기에 새로 작성된 글의 경우, 태그 정보 및 글 내용을 활용해서 추천해줄 수 있을 것 같다.

Future work

- 저널: 추천시스템 기법 연구동향 분석 - 손지은(고려대학교), 김성범(고려대학교), 김현중(서울대학교), 조성준(서울대학교), 2015.04

(링크: http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE06270308)

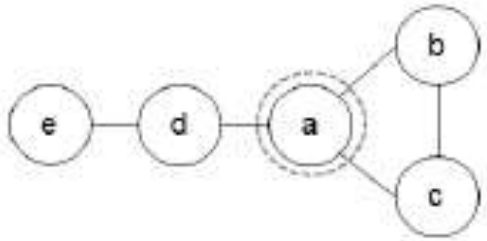
- 해당 저널에서 알고리즘 재숙지
- 어떤 알고리즘에는 어떠한 문제가 있고, 그 문제를 해결하기 위해 등장한 방안에 대한 연구들이 함께 제시됨

- **P190 새로운 고객 – 취향을 파악할 만 한 구매 이력정보가 존재x → CF기법과 사회연결망 기법의 중심성을 결합한 분석을 시도.**

User-user 유사도행렬 → 네트워크로 표현하여 각 노드의 중심성 계산 -> 중심성이 가장 높은 사용자를 cold start user에 추천

	사용자 a	사용자 b	사용자 c	사용자 d	사용자 e
사용자 a	1	0.7	0.8	0.8	0
사용자 b	0.7	1	0.6	0	0
사용자 c	0.4	0.6	1	0	0
사용자 d	0.8	0	0	1	0.9
사용자 e	0	0	0	0.9	1

(a) Similarity matrix



(b) User network

Figure 4. Example of recommendation for new customers using collaborative filtering and social network analysis

→현재는 그냥 가장 popular한 글을 추천해주고 있지만, 중심성이 높은 유저의 취향이 가장 대중적이고 신뢰성이 높다는 가정 하에 해당 유저의 item목록을 새로운 고객 or history가 적은 고객(CF적용 불가한)에게 추천할 수 있을 것 같다.

Future work

목표

1. 타겟 사용자가 좋아할 만 한 글 추천 2. 그동안 잘 추천되지 않았던 / 잘 읽히지않았던 글 추천

① 브런치 사용을 잘 하는 user 및 조회수가 높은 item

: 그대로 item-based CF적용

② Long-tailed item: 잘 읽히지 않았던 글 (즉, 조회수가 낮은 글)

: tag정보 → word2vec → 유사도 계산 → 비슷한 글 추천

: contents-based filtering(작가, 매거진 등의 정보를 feature로 활용해서 글 자체의 특성 비교) → 유사도 계산해서 타겟 사용자가 그동안 봤던 글과 가장 유사한 글을 추천해준다.

③ Cold-start item: 추천기간에 새로 작성된 글

: 브런치 특성상 구독하는 작가 글 읽음 → 구독작가 기반 추천

: 자주 읽었던 매거진 글 읽음 → 매거진 기반 추천

: 관심있는 글(그동안 읽었던 글들과 유사한 글이 관심있을 것이라고 가정.) – 참고한 코드의 경우에는 똑같은 태그가 2개이상일 때, 해당 태그를 관심사로 판단했음. 만약에 태그가 '유럽', '여행' / '유럽여행'이라면 비슷한 글임에도 불구하고 유사한 글로 분류해주질 못한다. 따라서 word2vec을 활용해서 유사도를 비교하여 가장 비슷한 태그를 가진 글을 추천해주고자 함.

④ Cold-start user / 최근or전체 이용이 적은 user

: 앞에서 소개한 CF기법과 사회연결망 기법의 중심성을 결합한 분석

reference

- Kakao Arena 2nd Competition[Arena: Brunch Article Recommedations]
- Kakao arena 깃헙[github:kakao arena]: 베이스라인 제공
- 참여팀 깃헙2[github: jihoo-kim]
- 참여팀 깃헙3[github: yeonmin]
- 저널: 추천시스템 기법 연구동향 분석 - 손지은(고려대학교), 김성범(고려대학교), 김현중(서울대학교), 조성준(서울대학교), 2015.04