Study of Recommender System(RS)

2020.07.03 ~ 2020.08.28 during summer break.

숙명여자대학교 컴퓨터과학전공 이혜승, 2hyes.com@gmail.com

What is Recommender System(RS)?

추천 시스템이란, 사용자의 프로필 혹은 과거 피드백을 통해 사용자가 좋아할 만 한 것을 추천해주어 궁극적으로는 products sales를 증가시키고자 하는 시스템이다.

추천시스템은 '**Matrix completion problem**'과 '**Top-k recommendation problem**' 두 가지 문제로 귀결된다. Matrix completion 문제는 user-item 매트릭스의 rating값을 예측하는, top-k 문제는 값 예측보다는 타겟 유저가 관심 가질 만 한 k개의 아이템을 추천해주는 문제이다.

시스템의 기술적인 목적으로는 주로 3가지를 언급한다. 첫째, **Relevance**. 추천에서 가장 중요한 요소이며, 유저에게 가장 연관성 높은 아이템 추천해주어야 한다는 점을 의미한다. 둘째, **Novelty**. popular items만 추천해주는 반복 추천은 판매 다양성을 낮추기때문에 유저가 이전에는 보지않았던 군의 새로운 아이템 추천을 의미한다. 셋째, **Serendipity**. 유저가 예상하지 못했던 아이템을 의미한다. 실제로 사용자가 놀랄 만 한 것을 추천해주는 것이다. 마지막으로, **Increasing recommendation diversity**. 계속 비슷한 아이템만 추천한다면, 해당 아이템들을 모두 싫어할 수도 있다는 위험이 따르며, 사용자가 지루해 할 수 있다. 따라서 다양한 유형의 아이템의 추천이 필요하다.

rating matrix는 implicit 혹은 explicit feedback 값으로 채워진다. Explicit은 정해진 등급(ex. 1~5점)에 맞게 유저가 직접 아이템에 대한 평가를 매기는 것이고, Implicit은 유저의 click 혹은 buying behavior를 주로 이진 값으로 기록한 것이다.

유저에 추천을 해 준 후, 추천이 잘 되었는지 평가가 필요하다. Offline model 평가는 알고리즘 자체의 성능을 평가하는 것이고, online model 평가는 실전 테스트이다. 추천의 순서와 개인화가 중요하다. Offline metric으로는 Mean Average Precision(MAP), Normalizing Discounted Cumulative Gain(NDCG), Entropy Diversity 등, online metric으로는 A/B test, CT 등이 활용된다.

<추천 시스템의 알고리즘>

1. **Collaborative filtering method, CF**
   1) Memory-based methods(neighborhood): item or user-based로 나뉘며, 유사도를 계산하여 가장 비슷한 유저, 아이템을 기반으로 추천.
   2) Model-based methods: 미리 모델을 생성한 후, 트레이닝 단계와 예측 단계를 분리하는 것이다. CF 기법을 classification의 일반화라고 생각하고, 전통적인 머신 러닝 기법을 적용하거나 Latent factor model, SVD와 같은 방법을 추천시스템에 적용.
2. **Content-based recommender, CB**
   : 상품 특성에 기반하여 유저가 평가했던 아이템과 비슷한 아이템을 추천.

Research purpose: debiasing
- fairness exposure

추천 기법으로 자주 활용되는 Collaborative Filtering에서의 가장 큰 단점 중 하나로 long tailed problem이 꼽힌다. 실제 현업에서도 long-tailed items를 판매하는 것이 마진이 더 크기 때문에, 고민하는 문제이기도 하다.

그래프의 x축은 판매/추천 빈도에 해당, y축은 인기도(얼마나 빈번하게 평가되었는지)에 해당한다. 즉, 많이 판매, 추천된 소수의 item들이 인기도 또한 높다는 것을 의미한다. 그래프의 우측에 존재하는 아이템들을 long-tail로 칭한다.

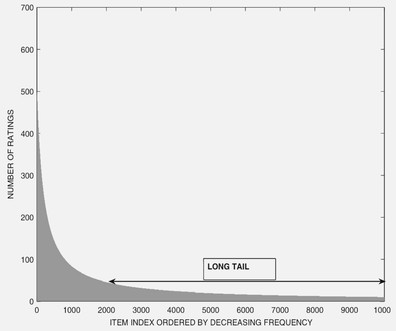
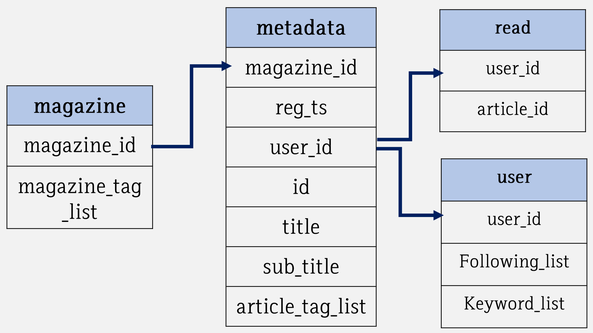


Figure 2.1: The long tail of rating frequencies

이때, Popularity 기반으로 추천을 하게 되면, 인기있는 20%(파레토 법칙에 의하면)의 item만이 계속해서 유저에게 추천되고, 판매된다.

따라서, 연구의 목적은 **과거에 추천 혹은 노출되지 않았던 아이템을 추천**해주는 것에 중점을 두고자 한다.

Recommender database: Kakao brunch



카카오 아레나에서 열린 두번째 competition으로, 브런치에서 제공하는 콘텐츠, 작가/독자, user behavior정보를 활용하여 유저의 취향에 맞는 글을 추천해주는 것이 목표다.

주어진 정보는 위의 표와 같이, 매거진과 글(item), user, user가 읽은 글 정보이다. 평가 대상인 유저의 목록이 제공되고, 그 유저들이 읽을 법한 글을 100개씩 추천해주는 것이 목표이다.

브런치 서비스의 특성상, 구독하는 작가의 글을 읽는 경우가 많고, 또한 2주간 등록된 글들의 소비가 높다. 즉, 구독 작가와, 글의 최신성이 가장 중요하다는 점을 생각하고 추천 시스템을 구성해야 한다.

평가 지표로는 NDCG(추천의 순서에 가중치), MAP, Entropy Diversity(추천의 다양성 평가) 세 가지를 활용한다.

Benchmarking notebooks.

Debiasing 문제 해결 이전에, 추천시스템의 이해와 기본 알고리즘들의 적용을 위해, 리더보드에 올렸던 다른 사람들의 노트북을 벤치마킹해보았다.

1. 전처리 단계
   : 유닉스 시간 변환, 글의 전체/최근 조회수, target 유저가 전체 기간/최근에 본 글, 읽은 구독작가의 글 수, 읽은 매거진의 글 수, 읽은 글의 태그 빈도수, 태그로 파악한 관심 키워드, 글 소비성향 저장
2. 추천 단계
   - Collaborative filtering recommend
   - Popularity-based recommend
   - Following-based recommend
   - Magazine-based recommend
   - Tag-based recommend

코드 링크: https://github.com/2hyes/kakao-arena_brunch

Tag-driven recommender system

벤치마킹 코드에서는 똑같이 작성된 태그만을 유사한 태그로 카운트한다. 실제 의미는 같지만, 다른 형태인 단어들은 같은 경우로 카운트하지 못한다는 단점이 있다. 따라서, Word2vec을 활용하여 tag들을 공간상의 점(벡터으로 표현한다. 하나의 글은 클러스터이고, n개의 점으로 이루어져 있다. 결론적으로 클러스터간 거리를 비교하여, 타겟 유저가 읽었던 글과 비슷한 태그의 글을 추천해주고자 한다. (최소 연결법, 코사인 유사도 활용하고자 함.)

해당 방법에 몇가지 이슈가 존재한다.

1. Word is not in vocab(OOV, Rare words)
   1) 한글 사전을 활용하기 때문에, **영어 단어 태그**는 인식되지않는다. (현재는 삭제한 상태이지만, 추후 단어 번역을 해서 태그의 의미를 살리고자 함)
   2) **추가적 형태소 분석**이 필요한 경우가 있다. '유럽'과 '여행'은 인식되지만, '유럽여행'은 vocab에 없다. 따라서 추가적으로 형태소 분석을 하거나 word2vec 대신 fasttext를 활용하고자 한다. 다만, 형태소 분석을 하면 단어의 의미를 잃는 경우가 생긴다.
   3) '젠트리피케이션'과 같이 한국어로 번역된 단어가 없는 영어 단어들.
   4) 사람, 브랜드 명과 같은 고유명사는 인식되지 않는다.
2. 유사도 기반 글 추천을 했는데, 너무 예전 글일 경우.

Future Work

- Fasttext or konlpy로 추가 형태소 분석 및 skt 수강 이후 리더보드에 업로드
- 추천시스템 알고리즘의 공부 및 이커머스 분야의 domain 지식 습득
- 추천시스템 알고리즘 스터디 자료: https://github.com/myeonghak/rs_study

Collaborative Filtering

1. Memory-based CF(Neighborhood-based CF)

: user / item간 유사도를 기반으로 타겟 사용자에게 item을 추천해준다.

User-based	Item-based
Serendipity ↑ : recommend some novel items	Accuracy ↑ : target user's own ratings are used
Provide a concrete reason X : peer group is a set of anonymous users	Provide a concrete reason : because you watch, buy, ⋯ it.
	More stable : #(users) > #(items) : top-k는 전체에의 영향을 많이 받지 않음

(+) simplicity, intuitive approach / interpretability

(-) offline단계에서 비효율적, Sparsity (neighbor중 아무도 해당 item을 평가하지않았으면, 타겟 유저에게 좋은 추천 X)

Collaborative Filtering

- 코드로 표현할 때에는, python의 pivot함수를 활용해서 matrix를 쉽게 생성할 수 있다.

```
# iu_df: pivot table로 item-user matrix 생성
# row: 글(item)
# column: 사용자(user)
# value: read(읽었는지(1) 아닌지(0))

iu_df = iu.pivot(index='article_id', columns='user_id', values='read')
iu_df = iu_df.fillna(0) # 안읽었으면 0
```

Collaborative Filtering

2. Model-based CF

Neighborhood-based문제 해결시 기본적으로 필요한 ratings matrix(pivot table)을 주로 차원축소해서 활용하거나, 잠재 요인을 추출해서 활용하기 때문에, 공간 효율성이 높다.

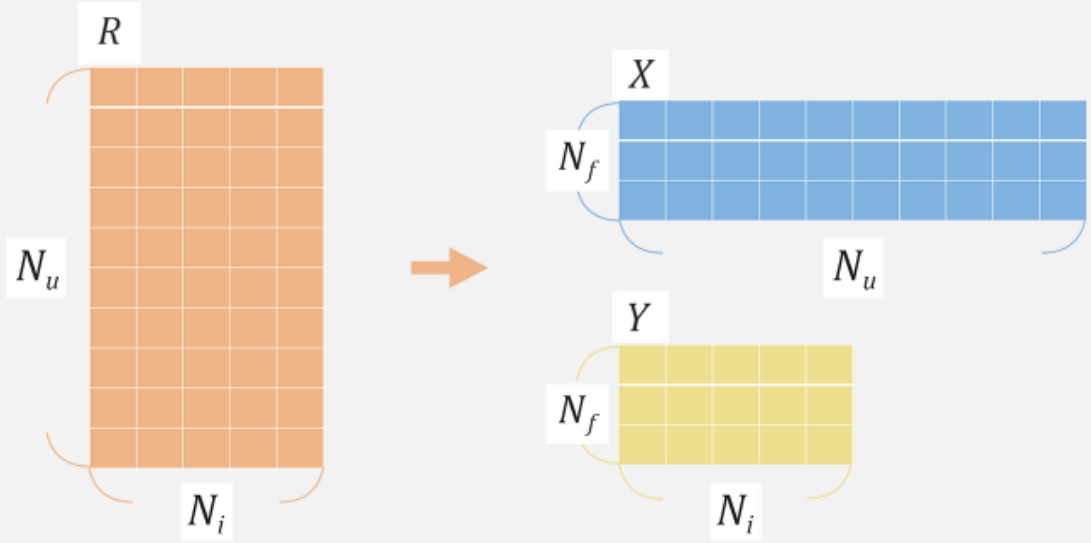
전통적인 ML기법들(decision tree, rule-based, Naïve bayes 등)을 CF에 적용하거나, latent factor model, SVD 등을 활용한다.

<Latent Factor Model>

잠재요인을 추출하는 방법.

X: user latent factor matrix

Y: item latent factor matrix



Word2Vec

- 원-핫 인코딩으로는 단어간 유사성을 표현할 수 없기 때문에, **분산 표현(Distributed Representation)**방법을 통해, 공간상에 비슷한 위치에 등장하는 단어들끼리는 비슷한 의미를 가지도록 나타낸다. 즉, 희소표현처럼 하나의 값만 1, 나머지는 0값을 갖게 하지않고, (희소표현보다) 저차원에 단어의 의미를 **여러 차원에 분산**하여 표현해낸다.
- Word2vec은 문장의 맥락을 통해 단어를 예측하거나,단어로 맥락을 예측해낸다. (전자는 CBOW(Continuous Bag Of Words), 후자는 Skipgram)
- Word2vec은 기본적으로 **단어 기준**으로 유사성을 계산하므로, out of vocabulary(OOV) 혹은 Rare Words에 대해서는 임베딩 벡터 생성 및 유사도 계산이 불가능하다.

  현재 브런치 태그데이터에서도 word2vec으로 임베딩이 불가능한 태그들이 많다. 글 당 태그가 3개정도인데, 1-2개를 삭제하는 것이 태그가 표현하는 글의 의미를 잃게 하지 않을까 하는 생각.

FastText

- FastText는 사전에 훈련된 워드 임베딩(pre-trained word embedding)중 하나로, Facebook에서 294개의 언어에 대해 위키피디아로 학습하여, 사전 훈련된 벡터들을 제공하는 모델이다.
- 가장 큰 특징으로는 하나의 단어를 또 여러 단어들이 존재하는 것으로 간주하는 것이다. 즉, **subword**를 고려하여 학습한다. 따라서 훈련 시 단어의 빈도수가 적더라도(Rare word), n-gram으로 임베딩을 하기때문에 벡터로 변환이 가능하다. 그리고 모르는 단어(Out Of Vocabulary, OOV)에 대해서도, subword를 통해 임베딩벡터를 만들어낼 수 있다.

  # Word2vec – konlpy를 하는 것보다 더 나은 결과이지 않을까?(사실 해봐야 알 듯!)

- 또한 오타나 맞춤법이 틀린 단어에 대해서도 일정 수준의 성능을 보여준다.

Future study

- Centroid간 거리를 계산했을 때에는 정형 tensor였으나, 최단연결법을 활용하기위해 모든 vecto들을 저장해 두어야 함. 다만, OOV문제로 인해 태그 개수가 모두 달라서, embedding시에 벡터 저장에 어려움을 겪고 있음.

→ 해결

- **Model-based**를 어떻게 직접 적용할 수 있을지, 코드로 연습
- Leaderboard에 올리는 방식이 뭔가 다름….
- 이커머스 분야에 대한 도메인 지식이 부족함을 책 읽으면서 느낌. 깊게는 아니어도 대략적인 도메인 지식의 습득은 필요할 것으로 느껴 짐.
- 어떤 알고리즘을 어디에 쓰이는지를 습득하는 시간이 필요함