Melon Music Recommendation

추천시스템 3팀

4기 김동욱 5기 조신형
6기 김서현 6기 황다연

CONTENTS

1 추천 시스템이란?

- Content-based, Collaborative filtering

2 멜론 음악 추천

- Word Tag, Track in Playlist

3 코드 시연

- Code

DSL
Yonsei

1 추천 시스템이란?

Content-based, Collaborative filtering

추천 시스템의 로직을
말씀드리겠습니다.

Recommendation System

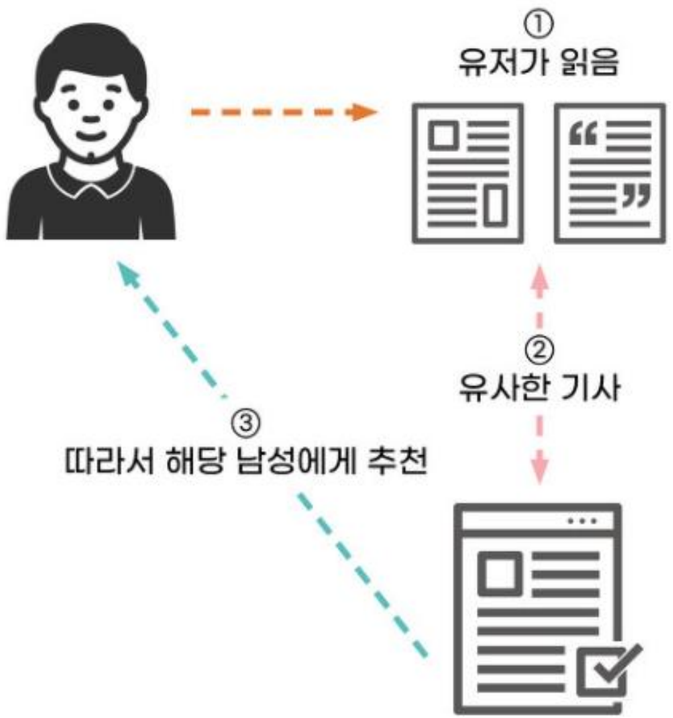
추천시스템

사용자-아이템 간 데이터를 활용해 취향을 고려한 아이템을 추천하는 시스템

- **추천에 사용되는 데이터**
  - Explicit Data: 상품에 대한 직접적인 만족/선호도
    ex) 별점, up-down 투표
  - Implicit Data: 선호도를 간접적으로 유추할 수 있는 행동들
    ex) 영상 시청 시간, 팔로잉 목록, 링크 클릭

- **추천 방식**
  - Personalized
    - Content-based: 아이템의 속성을 바탕으로 추천
    - Collaborative filtering: 여러 사용자들로부터 얻은 기호 정보를 바탕으로 추천
  - Non-personalized

Content-based (내용 기반)

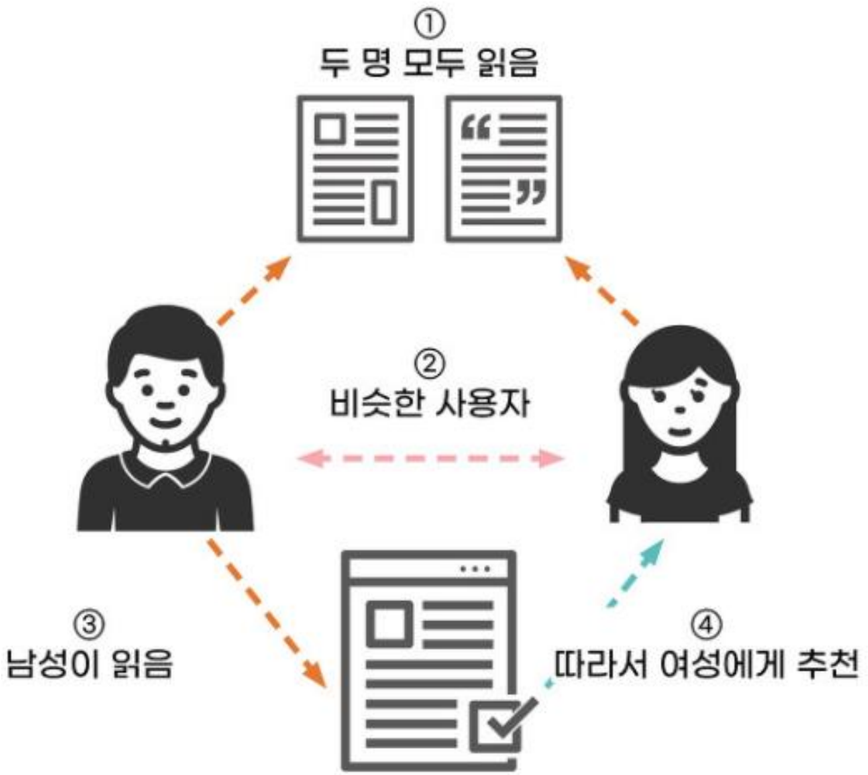


Product-Associated Recommendation

- Text, audio, image 등 다양한 형식의 아이템의 특성을 바탕으로 추천
- 독특한 취향의 유저에게나 유명하지 않은 아이템들에 대해서도 추천 가능
- 추천된 아이템의 특성에 대한 설명 가능

Recommendation System

Collaborative Filtering (협업 필터링)



User behavior-based Recommendation

- 여러 사용자들로부터 얻은 선호도에 대한 데이터를 바탕으로 추천
- 사용자가 많을수록 계산이 오래 걸리는 반면, 사용자가 일정 수준 이상이어야 정확한 결과를 낼 수 있다는 딜레마가 있음
- 예시: '이 상품을 구매한 사용자가 구매한 다른 상품들'

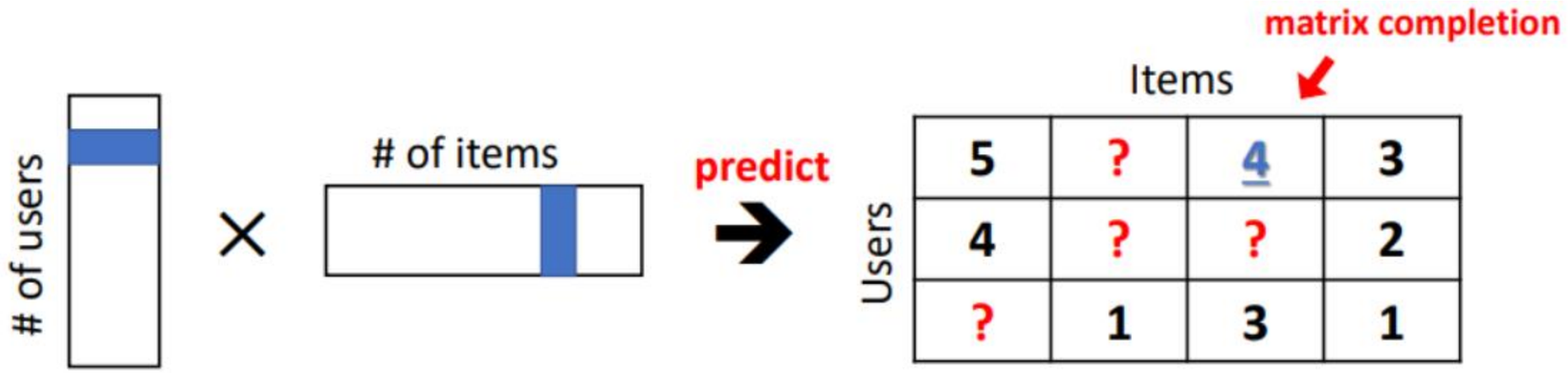
https://blog.daehong.com/237

Collaborative Filtering (협업 필터링)

유저들의 선호도에 대한 희소행렬을 바탕으로 missing value imputation 진행

- **Memory-based**
  (Nearest neighbor)
  - User-User : 성향이 비슷한 유저들의 정보를 참고 (row-wise)
  - Item-Item : 특정 유저가 평가한 아이템을 바탕으로 다른 아이템을 평가 (column-wise)

- **Model-based** : 행렬분해(matrix factorization)를 바탕으로 희소행렬을 차원 축소



Melon Playlist Continuation

각 추천 방식의 특징 비교

Content-based

- 항목 자체의 속성을 바탕으로 추천: Cold Start로부터 자유로움
- 콘텐츠의 형식 (ex. 음악, 영상)에 따라 추출할 수 있는 정보가 다름: 다양한 형식의 항목을 추천하기 어려움

Collaborative Filtering

- Cold Start: 신규 사용자 등 데이터가 없는 상태에서는 추천 불가
- Long tail: 소수의 인기 있는 항목에만 관심을 보여 관심이 저조한 항목은 추천되지 못함

2 멜론 음악 추천

Word Tag, Track in Playlist

Melon 음악 추천 대회에서의 모델링을
말씀드리겠습니다.

EQ

대회 목적 : Playlist에 어울리는 음악 추천

데이터 : Melon users의 playlist의 메타데이터

플레이리스트의 **수록 곡, 태그**, 제목, 좋아요 수, 최종 수정 시각

: 곡의 메타데이터

곡 제목, 앨범 제목, 아티스트명, 장르, 발매일

Melon Playlist Continuation

: 대회 소개

Content Based

User의 태그와
유사한 태그의 Playlist 선별

가을 이별 노래

산책할 때 좋은 노래

Collaborative

선별된 Playlist의
곡을 기반으로 곡 추천

규현_광화문에서

바이브_가을타나봐

Melon Playlist Continuation

: 추천 모델 개요

Content-based: Word2Vec 모델 이용

하이퍼파라미터 정의

```
num_features = 64       # 임베딩 벡터 차원 수
min_word_count = 30     # 특정 단어의 최소 빈도수 (빈도가 적은 단어는 학습하지 않습니다!)
context = 5             # 컨텍스트 윈도우 크기 (양 옆으로 몇개의 단어를 참고할지)
downsampling = 1e-3     # 다운샘플링 비율
```

```
from gensim.models import word2vec

# 모델 학습용 데이터 정의
texts = []
for ele in tag_list_prepro:
  texts.append(ele.split())


print("Training model...")
model = word2vec.Word2Vec(texts,
                          size=num_features,
                          min_count=min_word_count,
                          window=context,
                          sample=downsampling)
```

모델 학습

사용자가 입력한 태그값과 유사한 태그값을 가진 플레이리스트를 선별하기 위해 Word2Vec 모델 이용

Melon Playlist Continuation

태그값과 인풋값의 Embedding Vector 생성하기

```python
def get_embedding_vector(words, model=word2vec_model, num_features=64):

  # 출력 벡터 초기화
  feature_vector = np.zeros(num_features, dtype=np.float32)

  num_words = 0
  index2word_set = set(model.wv.index2word)    # 어휘사전

  for w in words:
    if w in index2word_set:
      num_words += 1
      feature_vector = np.add(feature_vector, model[w])

  feature_vector = np.divide(feature_vector, num_words)
  return feature_vector.reshape(1, -1)


# 플레이리스트 데이터에서 태그들의 임베딩벡터 계산 후 추가
plylst["embedding_vector"] = plylst.tags.apply(get_embedding_vector)
```

어휘사전에 포함된 단어 벡터들의 평균값으로 embedding vector 생성

각 플레이리스트의 태그값으로 embedding vector 생성

Embedding Vector간 코사인 유사도 계산

```
from tqdm import tqdm
from sklearn.metrics.pairwise import cosine_similarity

word1 = ["클래식", "차이코프스키", "교향곡"]
vec1 = get_embedding_vector(word1)

cs = []
subset_plylst = plylst.sample(frac=0.1)
for vec2 in tqdm(subset_plylst["embedding_vector"]):
    if pd.Series(vec2.sum()).isnull()[0]:    # 벡터 값이 없는 경우
        cs.append(0)
    else:
        cs.append(cosine_similarity(vec1, vec2)[0][0])

subset_plylst["cosine_similarity"] = cs
```

사용자로부터 입력값을 받은 후, embedding vector 생성

입력값 및 태그값으로부터 추출한 embedding vector간의 코사인 유사도 계산

Melon Playlist Continuation

플레이리스트 선정 및 곡 추천 강도 계산

코사인 유사도 기준으로 정렬 후 플레이리스트(pid) 선택

곡의 id를 BM25 가중치로 대체

```
p = np.zeros((n_songs, 1))
p[plylst_use.loc[pid,'songs_id']] = np.array(plylst_use.loc[pid,'bm25_song']).reshape(-1,1)

val = train_songs_A.dot(p).reshape(-1)

songs_already = plylst_use.loc[pid, "songs_id"]

cand_song = train_songs_A_T.dot(val)
```

기존 플레이리스트와 해당 플레이리스트간의 유사성 계산

기존 플레이리스트의 곡들과 유사성 가중치를 곱하여
곡의 추천 강도 계산

Melon Playlist Continuation

추천 강도를 기준으로 곡 추천

곡의 추천 강도를 내림차순 정렬 후 상위 100개 추출

```
cand_song_idx = cand_song.reshape(-1).argsort()[-100:][::-1]


cand_song_idx = pd.Series(cand_song_idx[np.isin(cand_song_idx, songs_already) == False]).sample(10)
rec_song_idx = [song_sid_id[i] for i in cand_song_idx]

# 해당 플레이리스트에 대한 추천곡 정보
get_songs(rec_song_idx)
```

해당 플레이리스트에 이미 있는 곡은 제외하고 10개의 곡 추천

Collaborative Filtering

	Song_1	Song_2	Song_3 …	… Song_M
Playlist_1	1	0	0	1
Playlist_2	0	1	0	1
Playlist_3	1	1	0	0
…				
			…	
…				
Playlist_N	0	1	1	1

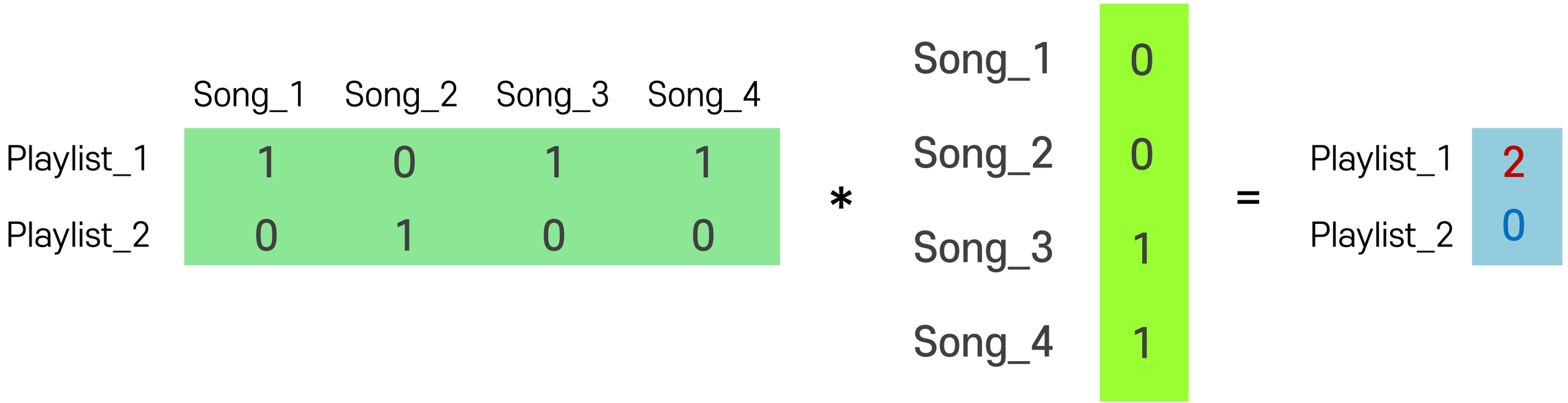
Melon Playlist Data로 희소행렬 생성

행: Playlist ID
열: Song ID

1: 해당 playlist에 해당 곡이 있음
0: 해당 playlist에 해당 곡이 없음

Melon Playlist Continuation

Collaborative Filtering



희소행렬 User vector 유사도

Melon Playlist Continuation

Collaborative Filtering

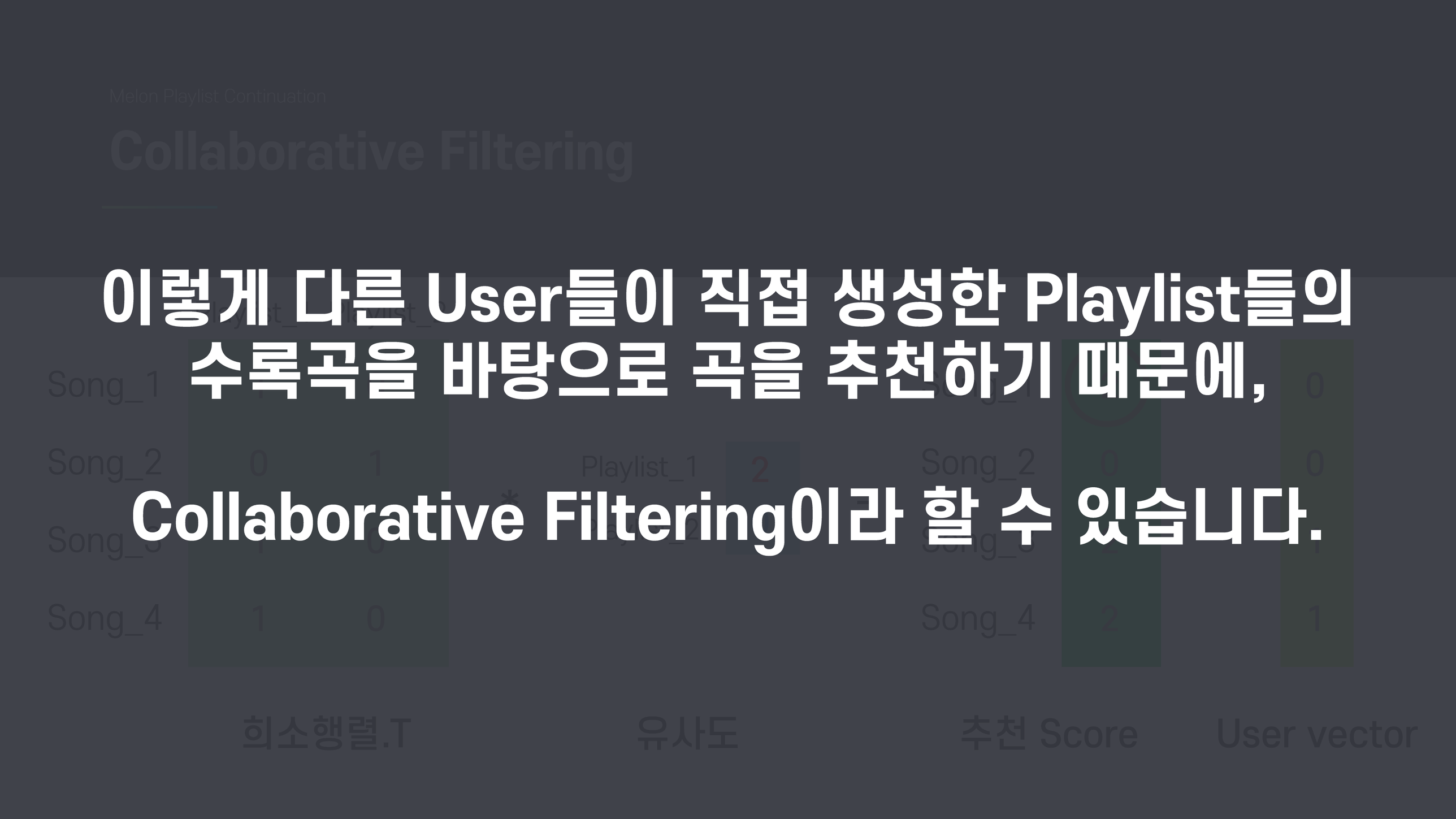
Melon Playlist Continuation

Collaborative Filtering

Collaborative Filtering

이렇게 다른 User들이 직접 생성한 Playlist들의 수록곡을 바탕으로 곡을 추천하기 때문에,

Collaborative Filtering이라 할 수 있습니다.



Melon Playlist Continuation

Collaborative Filtering

	Song_1	Song_2	Song_3 ···
Playlist_1	1	0	0
Playlist_2	0	1	0
Playlist_3	1	1	0
···			

Value: Playlist에서의 곡의 유무

BM25 가중치로 대체

D: playlist
q: song

IDF

$$\mathrm{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\mathrm{avgdl}}\right)}$$

Playlist의 길이

Playlist에서 곡의 등장 빈도 (1)

Playlist들의 평균 길이

$b,\ k_1$는 가중치 상수
$(b = 0.75, k_1 = 1.2)$

Melon Playlist Continuation

Collaborative Filtering

Bm25 적용 후 ->

Bm25 적용 전 ->

곡 nDCG	태그 nDCG	Submission		Uploader
0.158814	0.339545	results.zip	다운로드	jo.shin
0.169471	0.344403	results.zip	다운로드	jo.shin
0.215733	0.367519	results.zip	다운로드	jo.shin
0.168486	0.343424	results.zip	다운로드	jo.shin
0.215733	0.350189	results.zip	다운로드	jo.shin
0.159643	0.340179	results.zip	다운로드	jo.shin

3 코드 시연

Code

앞서 소개한 코드를 시연
해보겠습니다.

Thank You