Melon Playlist Continuation

권송아(소프트웨어학) , 김소은(응용통계학), 김태윤(소프트웨어학), 김효민(소프트웨어학),
최은서(소프트웨어학), 홍지호(기계공학)

2021 CUAI 중앙대학교 인공지능 학회 동계 컨퍼런스
2021 Chung-Ang University Artificial Intelligence Society's Winter Conference

Abstract

카카오 아레나의 'Melon Playlist Continuation' 대회 참가를 가정하고 진행하였으며, 목표는 플레이리스트에 수록된 곡과 태그의 절반 또는 전부가 숨겨져 있을 때, 주어지지 않은 곡들과 태그를 예측하는 것이다. 숨겨진 곡과 태그를 예측할 수 있는 모델을 만들어 이를 주어진 플레이리스트에 대해 그 플레이리스트와 어울리는 곡을 추천해주는 용도로 사용하고자 한다. 플레이리스트 데이터 EDA를 통해 장르, 곡, 태그에 대한 특징을 파악하고 오토인코더를 사용한 협업 필터링을 기반으로 모델을 설계하여 평가 지표로서 nDCG와 Recall을 사용하였다. 사전 훈련된 모델을 통해 코사인 유사도를 기반으로 플레이리스트 제목과 유사한 태그를 추천하는 시스템을 설계하였다.

Introduction

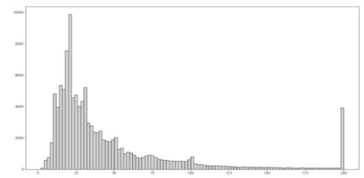
국내 음원 스트리밍 시장에서 독보적인 점유율을 유지하던 '멜론'은 '지니', '애플 뮤직', '스포티파이', '유튜브 뮤직'이 주요 경쟁자로 부상함에 따라 그 지위가 흔들리고 있다. 음원 시장에서의 각 플랫폼의 생존 경쟁도 치열하다. 사용자의 취향이나 분위기에 맞는 곡을 제안하는 추천시스템이 중요하게 부각되고 있다. 음원 스트리밍 플랫폼에는 수천만 개가 넘는 곡을 서비스 하는데, 이 수많은 곡들 중 자신의 취향에 맞는 음악을 사용자가 일일이 찾는 것은 시간이 많이 걸리는 작업이다. 따라서 사용자의 취향에 맞는 곡을 최대한 많이 효율적으로 탐색하는 것을 도와주는 시스템을 구축해야 한다.

Materials and Methods

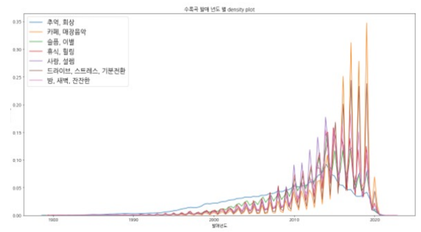
〈 EDA 〉

데이터

장르는 총 254개의 장르코드가 존재하고 대부분의 곡들은 한 개의 대분류 장르와 매핑되어 있다. 플레이리스트의 태그는 1개부터 11개까지 존재하고, 노래는 1개부터 200개까지 존재한다. Train.json을 통해 플레이리스트에는 평균 약 46개의 곡이 수록되어 있고 약 4.1개의 태그가 수록되어 있다. 수록곡의 약 51%, 태그의 약 40.2%는 두개 이상의 플레이리스트에 중복 수록되어 있다.



	min	25%	50%	mean	75%	max	std
songs	1.0	19.0	30.0	45.94	54.0	200.0	43.95

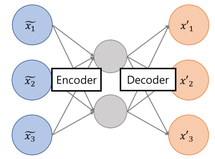


〈 노래 예측 모델 〉

데이터

총 115,071개의 플레이리스트와 707,989개의 노래가 존재하였다. 20,000개의 플레이리스트를 검증 용도, 10,000개 의 플레이리스트를 테스트 용도로 사용하였고, 나머지 데이터는 학습 과정에서 사용하였다. 이 중, 10개 미만의 노래가 수록된 플레이리스트를 학습 과정에서 제외시켰다.

노래 추천 방법



오토인코더 모델 중 하나인 Multinomial Denoising AutoEncoder 모델을 사용하였다. 학습 과정에서는 인코더와 디코더를 학습한다. 인코더는 입력값인 플레이리스트-노래 행렬에 노이즈를 추가해 차원이 200인 잠재공간으로 축소시키고, 디코더는 잠재공간으로부터 입력과 유사한 결과값을 만들어낸다. 이후 테스트 과정에서는 결과값으로 나온 벡터에서 기존의 입력에 존재했던 값을 제외시킨 후 가장 큰 값을 가지는 상위 N개를 추천하게 된다.

결과 및 성능평가: (nDCG@3000 = 0.00521, Recall@3000 = 0.01782)

```
X_t = np.transpose(np.nonzero(X))
song_meta.loc[X_t[:,1]]['song_name']
```

```
17839     사랑, 참...
25260          그래요
46162          복극성
51612    오늘은 가지마
88503          응급실
97412          상록수
109500         오빠는...
117184         그리움
118150    사랑할 것 같아서
```

```
[126] song_meta.iloc[idx_topk_part]['song_name']
```

```
0          Feelings
471988     두근두근
471989     흔연해
471990     Last Dance
471991     Memories
           ...
235997     떠나갈래
235998     Gang Gang
235999     On My Way
235991     좋을텐데
707988     Queen 병로 멜로디
Name: song_name, Length: 707989, dtype: object
```

이별 노래를 담은 플레이리스트를 입력으로 넣었을 때 '떠나갈래', '좋을텐데' 등 이별과 관련한 노래를 예측한 것을 볼 수 있다.

```
Test NDCG@3000=0.00521 (0.00016)    Test NDCG@100=0.00192 (0.00014)
Test Recall@3000=0.01782 (0.00038)  Test Recall@100=0.00275 (0.00016)
```

평가지표로는 관련성 높은 노래를 순서대로 예측했는지 평가하는 nDCG와 실제 추천되어야 하는 값이 얼마나 포함되어 있는지를 나타내는 Recall을 사용하였다. Recall@3000의 의미는 추천된 노래 상위 3000개 중 약 51개가 바르게 추천되었다는 것을 뜻한다. 이 모델을 큰 데이터 세트에서 후보를 생성하는 모델로 사용하고, 후보들의 추천 순서를 재배열하는 모델을 새로 만든다면 더 나은 성능의 추천시스템을 만들 수 있을 것이다.

〈 태그 예측 모델 〉

전처리

우선, 한글이 아닌 영어, 숫자로 작성된 태그를 삭제한다. 이후, FastText의 사전 훈련된 모델로 태그 간 유사도를 분석하고, 비슷한 태그가 많이 매칭된 상위 태그를 기준으로 태그를 다시 라벨링한다. 이어서 결측치를 제거하고, 플레이리스트명의 불용어 (stop words)를 제거 후 토큰화한다.

데이터

전처리를 통해 생성된 111,133개의 플레이리스트와, 16,451개의 태그 중, 2만개를 훈련용으로, 1만개를 테스트용으로 사용하였다.

태그 추천 방법

우선 Word2Vec을 이용해 각 플레이리스트의 벡터값을 구한다. 이는 플레이리스트 이름의 벡터의 평균값으로 계산된다. 이후 코사인 유사도를 이용하여 테스트 세트에 있는 태그를 입력하면 해당 플레이리스트 제목과 유사한 다른 플레이리스트 상위 5개(코사인 유사도가 높은 플레이리스트)를 찾아 태그를 추출한다. 이후 추출된 태그를 빈도순으로 정렬하고 테스트 세트와 중복을 제거한뒤 태그를 추천하게 된다.

결과 및 성능평가: (Recall : 0.055)

위 그림은 추천 결과의 예시를 뽑아온 것이며, 첫번째 추천 결과를 보면 비, 감성, 밤, 새벽 등 유의미한 태그 추천 결과를 볼 수 있다. 평가지표로는 실제 추천되어야 하는 값이 얼마나 포함되어 있는지를 나타내는 지표인 Recall을 사용하였다. Recall은 약 0.055의 값으로, 단순히 태그 출현 횟수를 세어서 추천하기 보다는 제목별 유사도를 태그의 가중치로 둬서 추천하면, 더 나은 성능의 추천시스템을 만들 수 있을 것이다.

Conclusion

본 연구에서는 카카오 아레나에서 제공하는 멜론 플레이리스트 데이터를 이용해 플레이리스트의 숨겨진 곡과 태그를 예측하는 모델을 설계하였다. 그 후, nDCG, Recall 지표를 통해 결과의 정확도를 확인하였다. 이를 통해 주어진 플레이리스트에 대해 비슷한 분위기를 가진 곡을 추천함으로써 음원 스트리밍 서비스 이용자로 하여금 자신의 취향에 맞는 노래를 추천 받을 수 있다.

본 연구에서는 협업 필터링 기반의 모델에만 의존하였다. 그러나 추후 오디오 웨이브 데이터를 이용한 콘텐츠 기반 필터링 기법을 함께 사용한다면 음원 스트리밍 서비스 이용자에게 더 정확한 추천 시스템을 제공할 수 있을 것이라 기대한다.

Reference

[1] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara, "*Variational autoencoders for collaborative filtering*", The Web Conference (aka WWW), 2018.02
[2] fasttext
https://fasttext.cc/docs/en/crawl-vectors.html
[3] 단어 벡터를 이용한 추천시스템
https://wikidocs.net/book/2155