A black and white photograph of a woman with long hair, wearing over-ear headphones and looking down with a thoughtful expression.

# MELON Music Recomendation



**Team Standard**  
Presentation

# 01

## Introduction

### Team Standard



## 1.1 Introduction



# Welcome Message

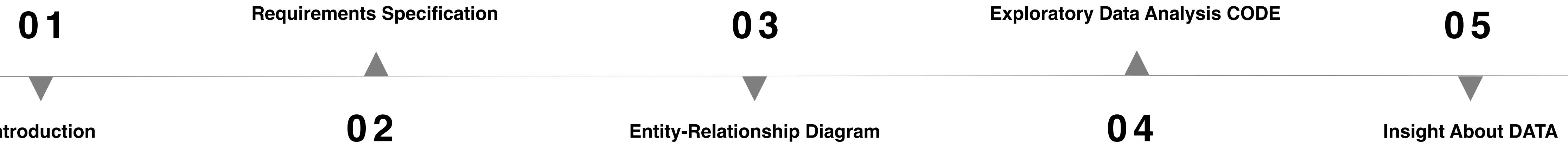
훌륭한 코드 한 줄은 한명이 작성할 수 있지만, 완성도 높은 소프트웨어는 훌륭한 팀웍을 통해 만들어진다고 믿습니다.

저희 팀은 개인의 지성보다는 집단지성을 통해 훌륭한 가치를 만들어내는 팀입니다.

편안한 분위기 속에서의 잡담은 팀원간의 신뢰를 만들어 냅니다. 이러한 잡담을 통해 낮아진 커뮤니케이션의 벽으로 한 층 더 자유롭게 아이디어를 나누며 보다 창의적인 결과물을 만듭니다.

빠르게 변화하는 시장에서 많은 양의 데이터를 다루기 위해선 그에 맞는 전문가적인 지식이 필요합니다. 저희는 새 시대의 흐름을 주도하는 분석가가 되기 위해서 새로운 기술과 역량을 확보하기 위해 노력하고 있습니다.

# 0.1 Index



## 0.2 Index

06

워드 임베딩

워드 임베딩(Word Embedding)은 단어를 벡터로 표현하는 방법으로, 단어를 밀집 표현으로 변환합니다. 이번 챕터에서는 희소 표현, 밀집 표현, 그리고 워드 임베딩에 대한 개념을 이해합니다.

협업 필터링

협업 필터링(collaborative filtering)은 많은 사용자들로부터 얻은 기호정보(taste information)에 따라 사용자들의 관심사들을 자동적으로 예측하게 해주는 방법이다.

08

컨텐츠 기반 모델

콘텐츠 기반 필터링은 말 그대로 콘텐츠에 대한 분석을 기반으로 추천을 구현하는 방법이다. 콘텐츠를 분석한 프로파일(item profile)과 사용자의 선호도(user profile)를 추출하고 유사성 분석을 통해 추천을 수행한다.

07

09

딥러닝 모델

심층 학습(深層學習) 또는 딥 러닝(Deep structured learning, deep learning 또는 hierarchical learning)은 여러 비선형 변환기법의 조합

10

개선 사항

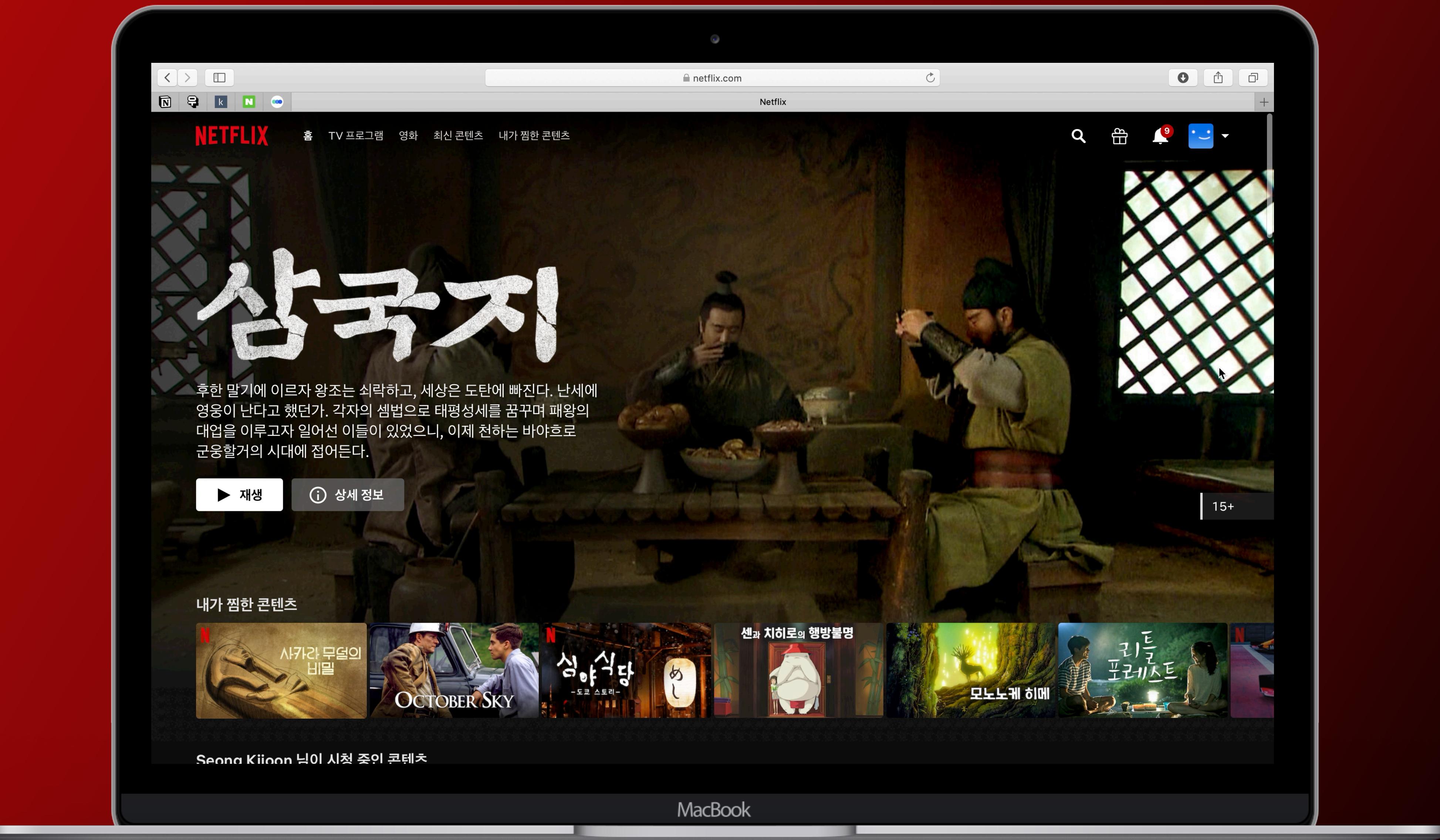


# 02

## 요구사항 명세서

Requirements Specification

## 2.1 Netflix



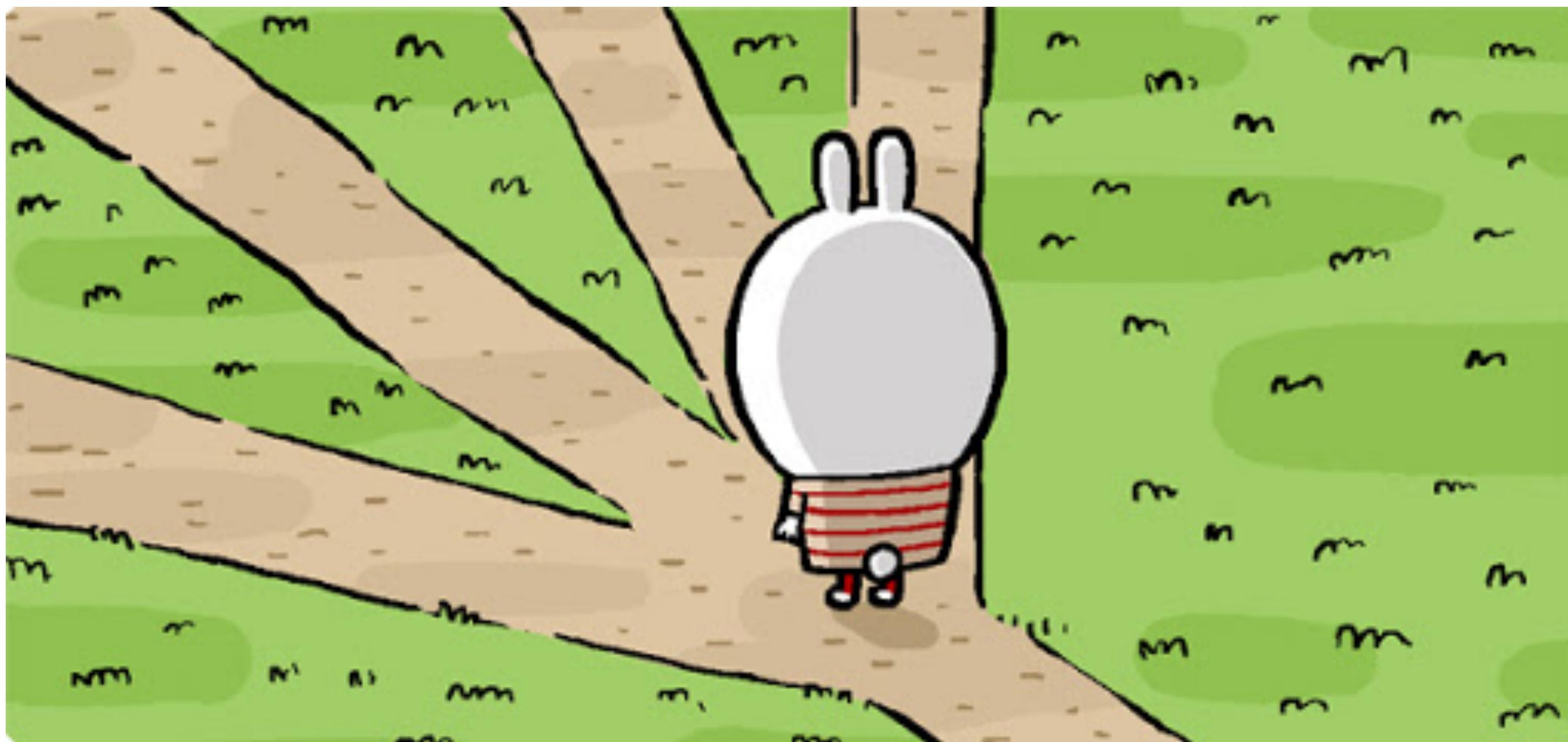
## 2.2 NETFLIX Competition

총 상금! \$ 1,000,000(11억)



## 2.3 Why Recommendation Algorithm?

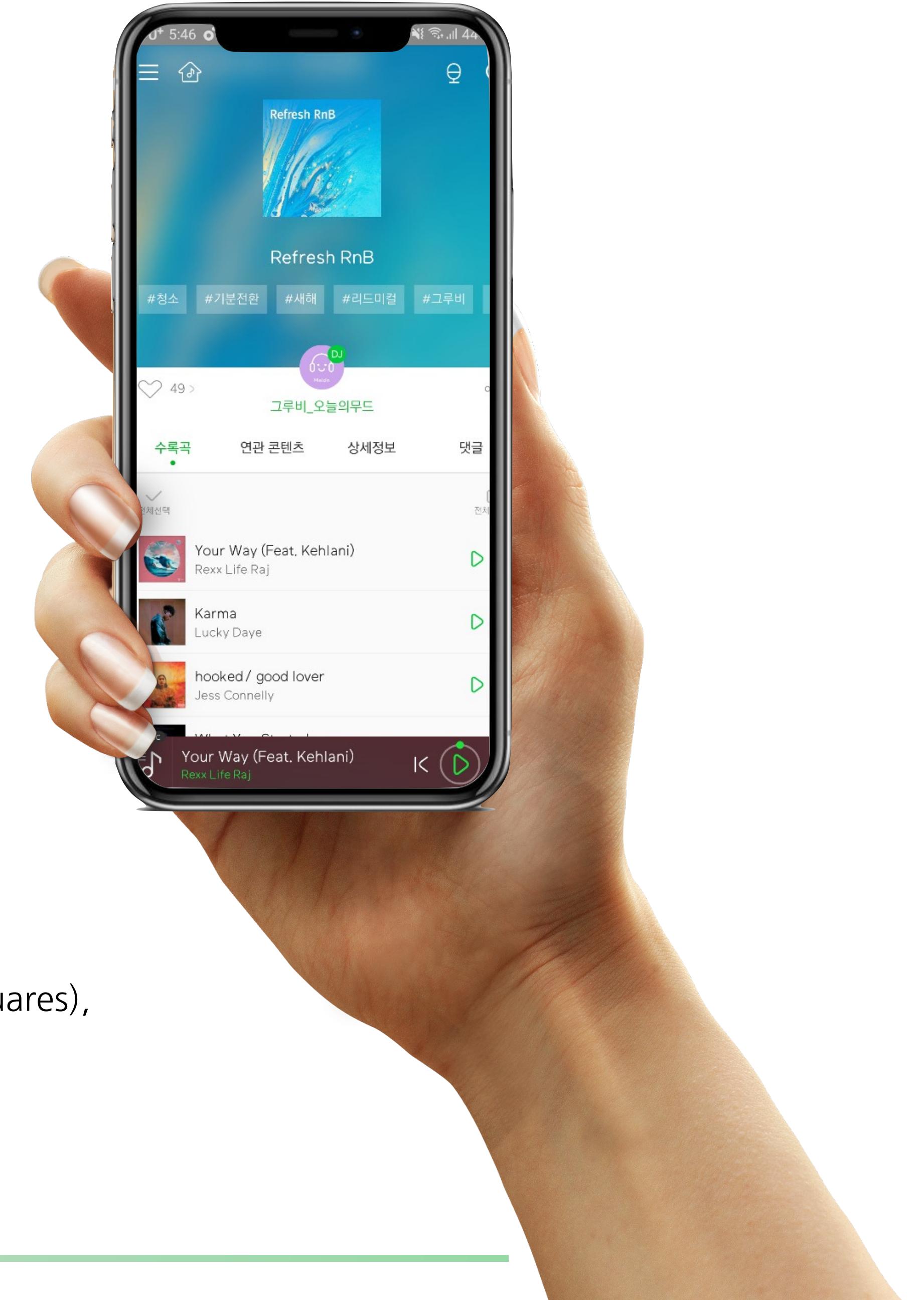
# Why Recommendation Algorithm?



## 2.4 Melon Playlist

# Melon Playlist Recommendation Algorithm?

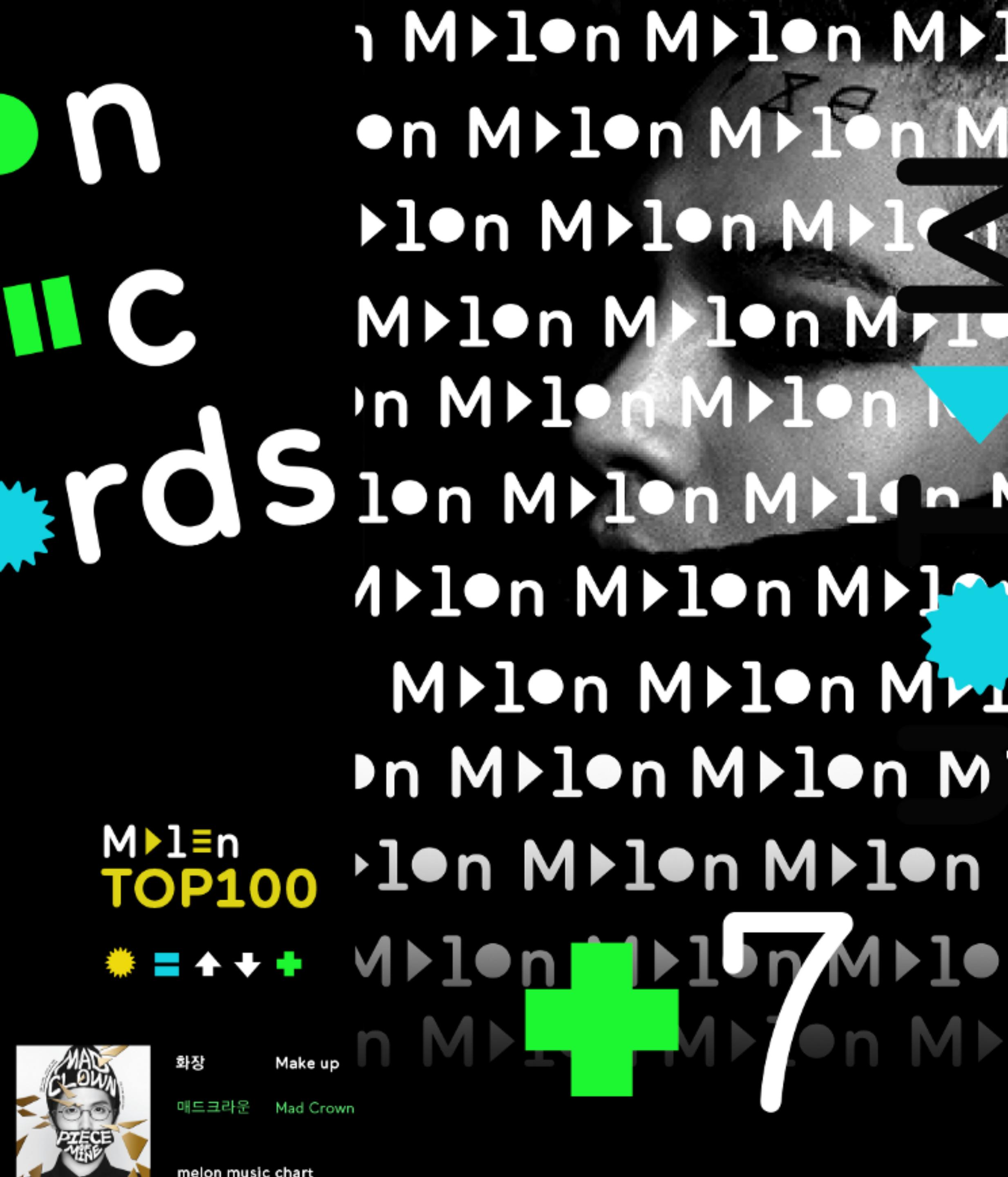
- 목표** : 각 플레이리스트 별 원래 플레이리스트에 있었을 것이라 예상되는 곡 100개,  
태그 10개 예측 및 추천
- 개발 환경** : colab(GPU), python 3.x 등
- 추천 알고리즘** : 대표적인 추천 알고리즘 적용  
콘텐츠 기반, 협업 기반(행렬분해 기반인 ALS(Alternating Least Squares),  
최근접 이웃 기반인 cosine, 자연어처리 모델인 word2Vec, 딥러닝)



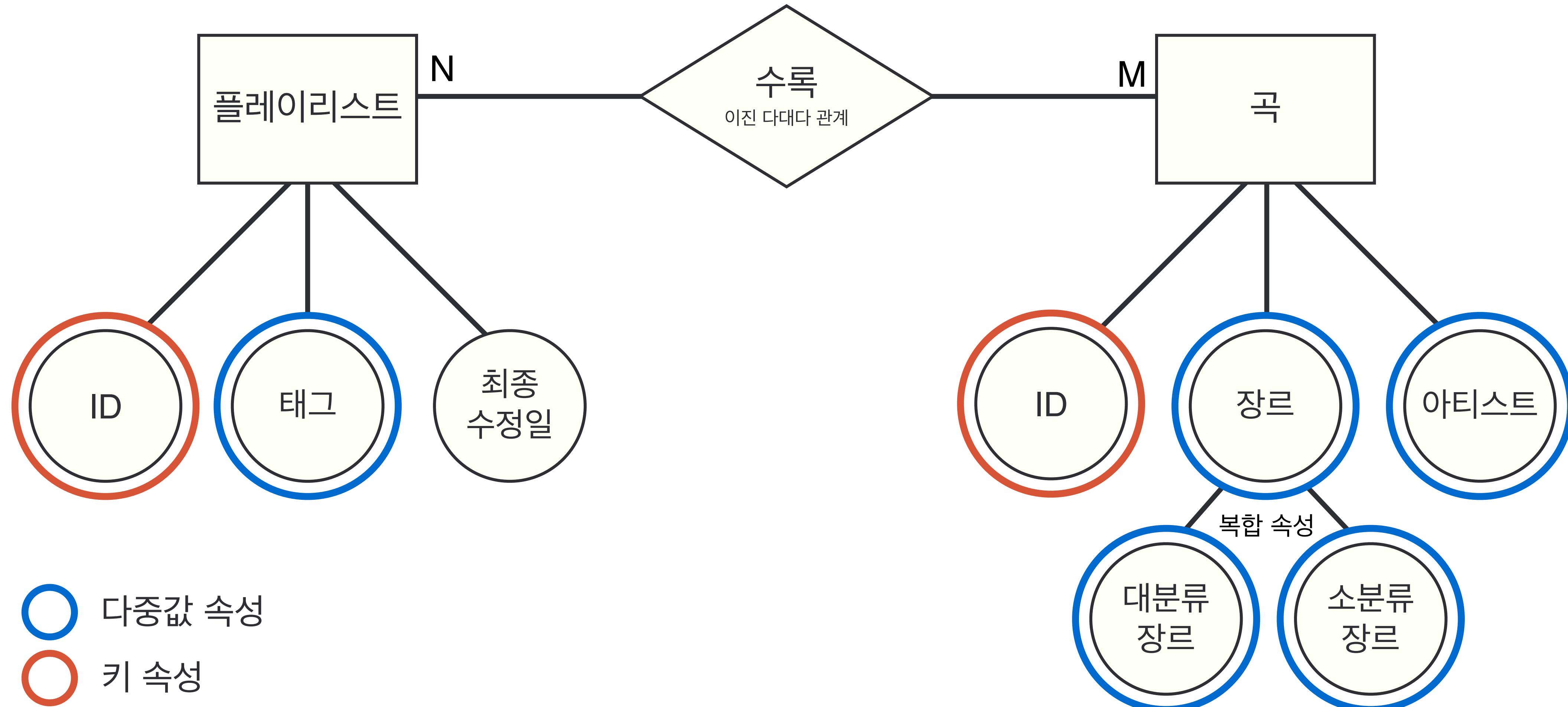
# 03

## ERD

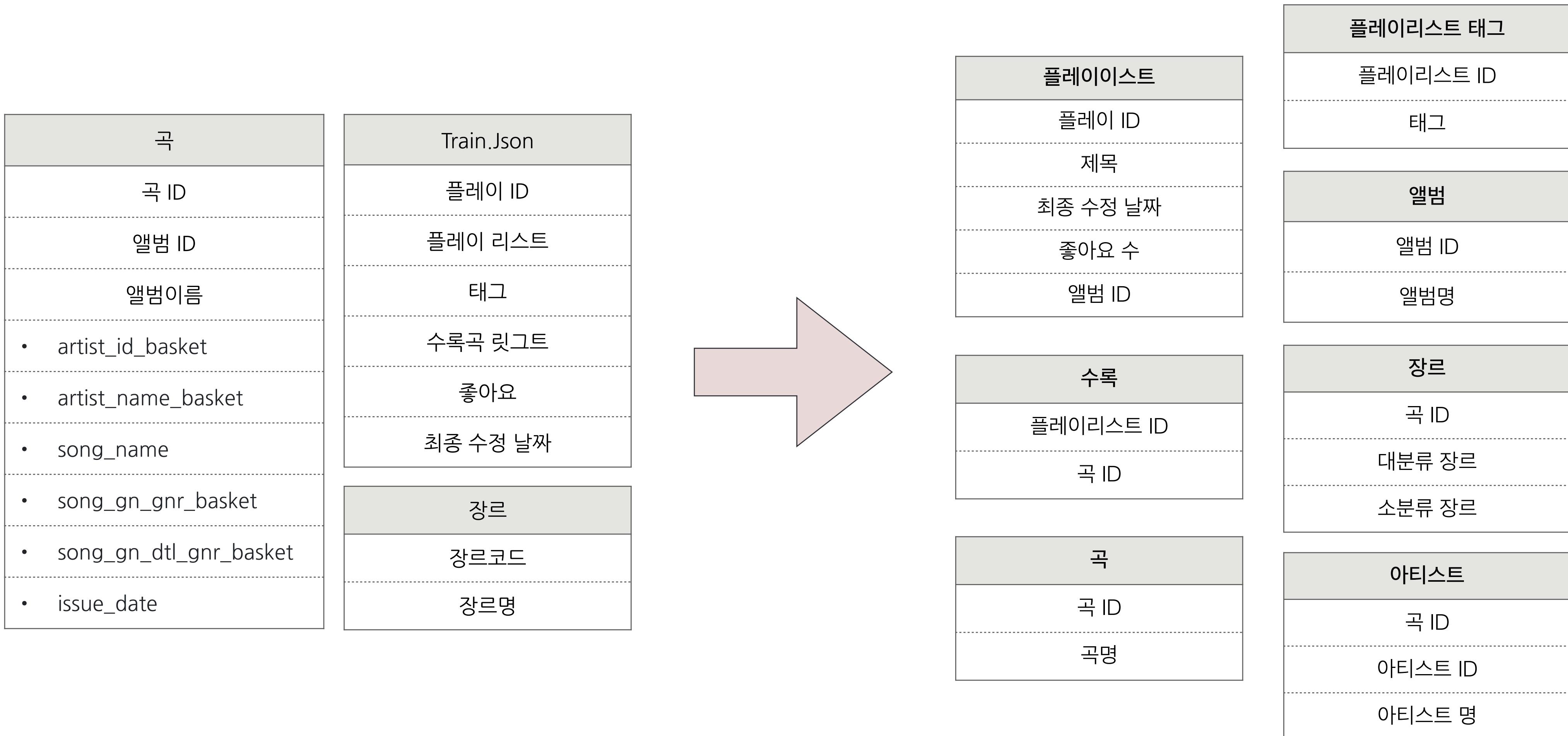
Entity-Relationship Diagram



### 3.1 Conceptual Data Modeling



## 3.2 Final Data





# 04

## EDA CODE

Exploratory Data Analysis CODE

## 4.1 Data Analysis

### “ 전체 곡 정보 데이터에서 대분류 장르의 분포 ”



#### CODE SUMMARY

- 장르 트리와 정보 테이블을 이용해서 곡별 장르 테이블을 만들고, 장르로 그룹핑을 해서 장르별 곡 수 테이블을 만들어 bar plot으로 시각화



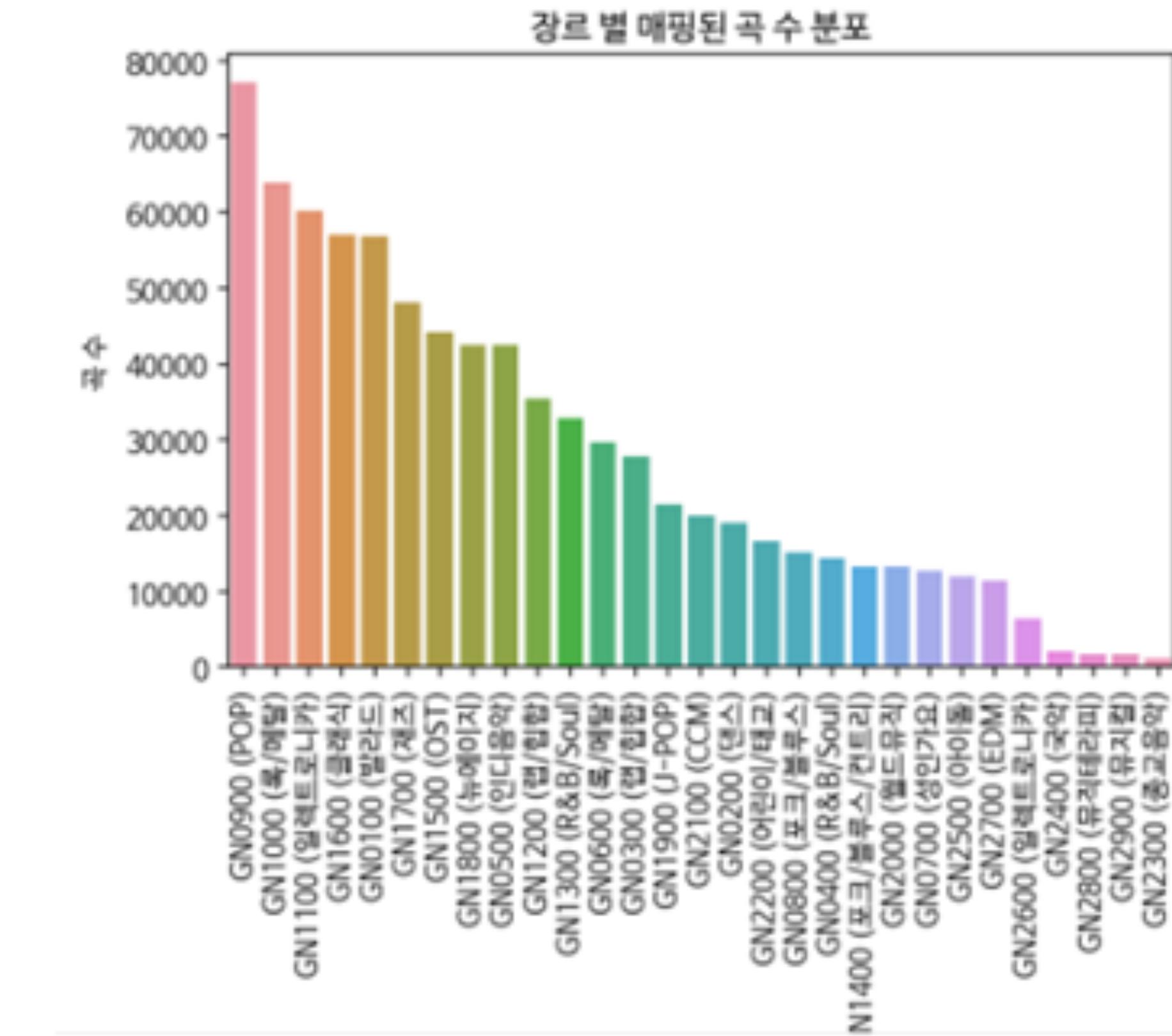
#### INSIGHT 1

- 그 결과, 대부분의 곡들은 한 개의 대분류 장르와 매핑되어 있고, 전체 곡의 약 13%는 2개 이상의 대분류 장르를 가진다는 것을 알 수 있었습니다.



#### INSIGHT 2

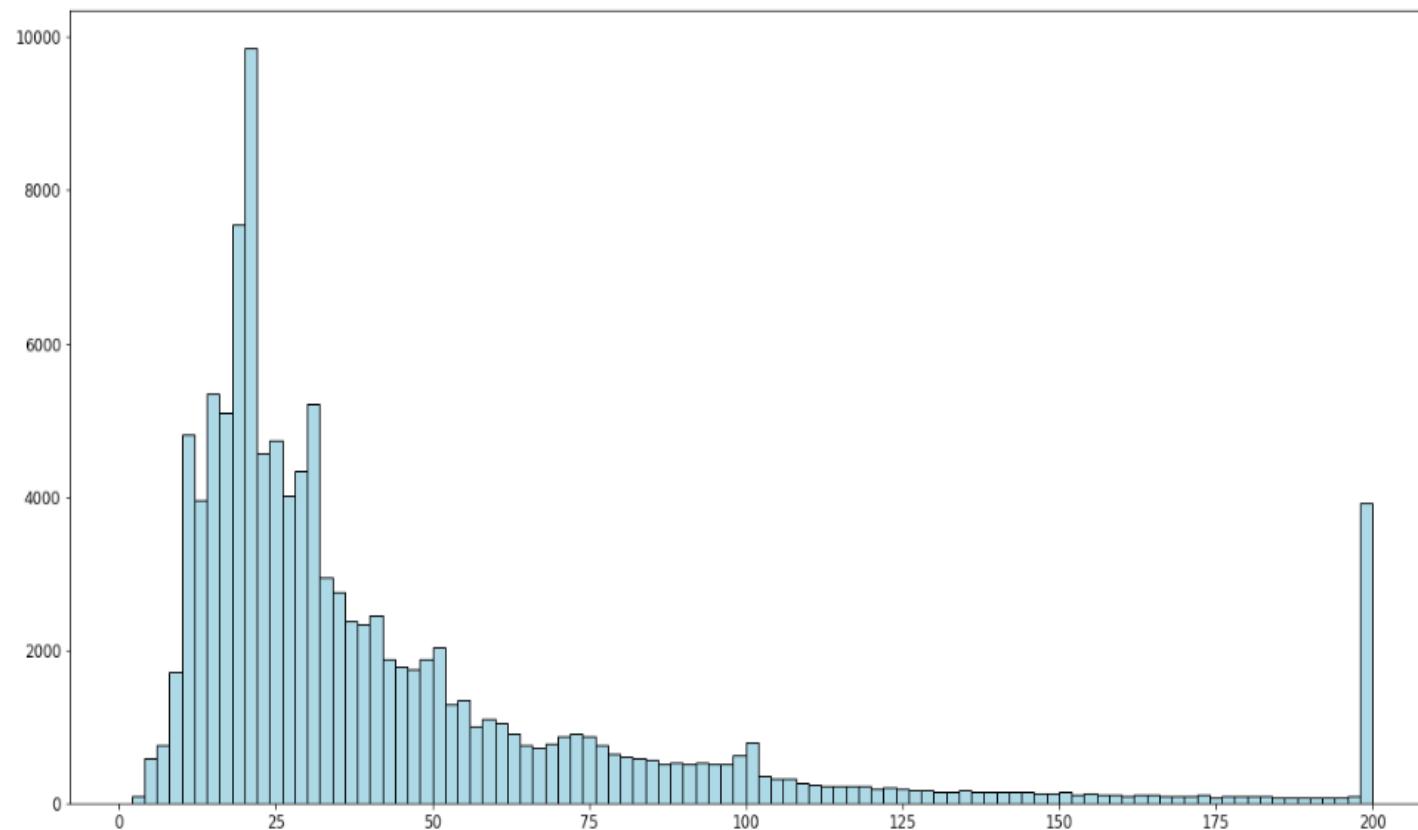
- 곡의 가장 많은 대분류 장르는 pop과 록/메탈, 일렉트로니카, 클래식, 발라드 순이었고. 국악, 뮤지컬, 뮤직테라피, 종교음악 장르는 1% 미만의 비중을 보입니다.



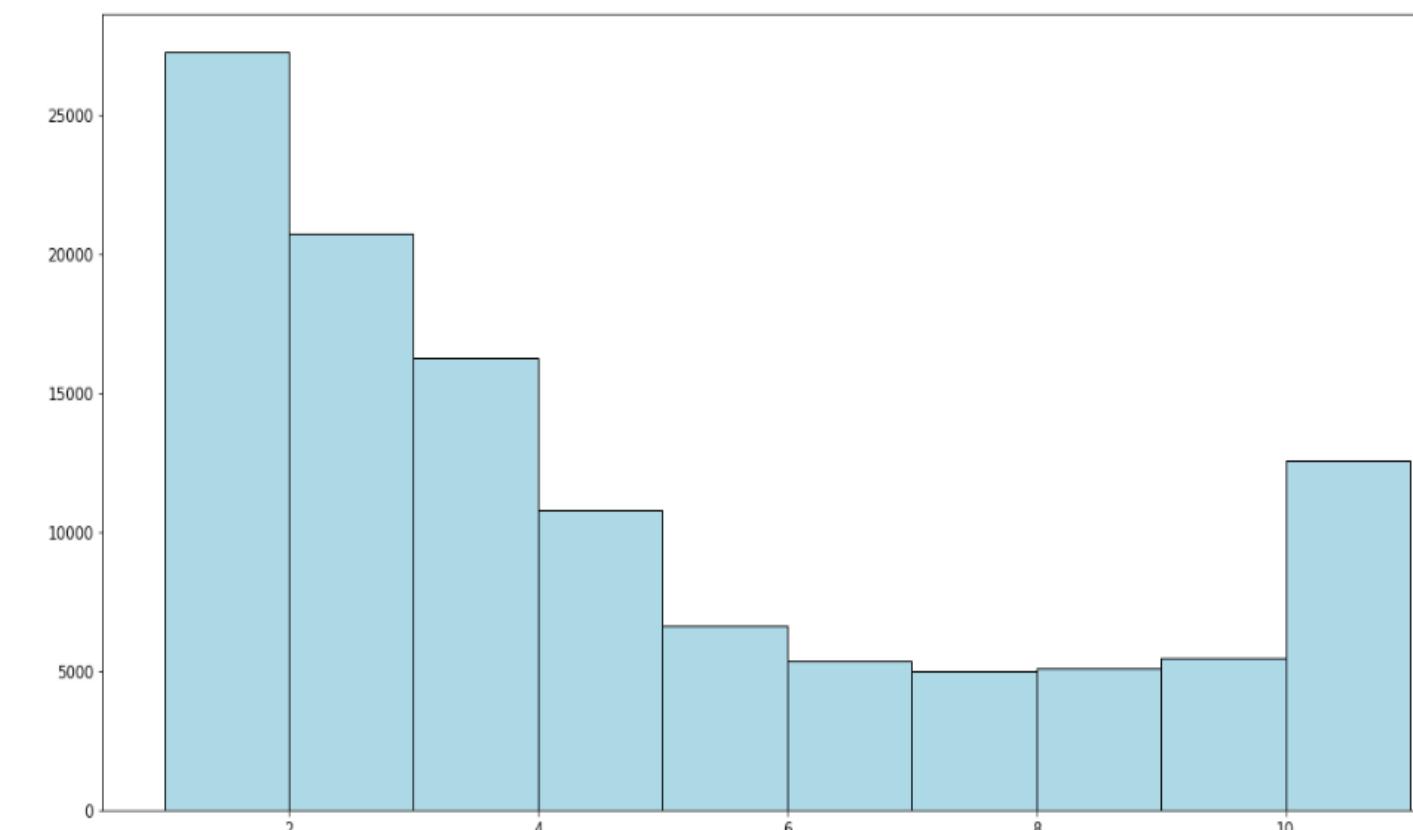
## 4.2 Data Analysis

### “ 플레이리스트 별 수록곡/태그/장르 수 분포 ”

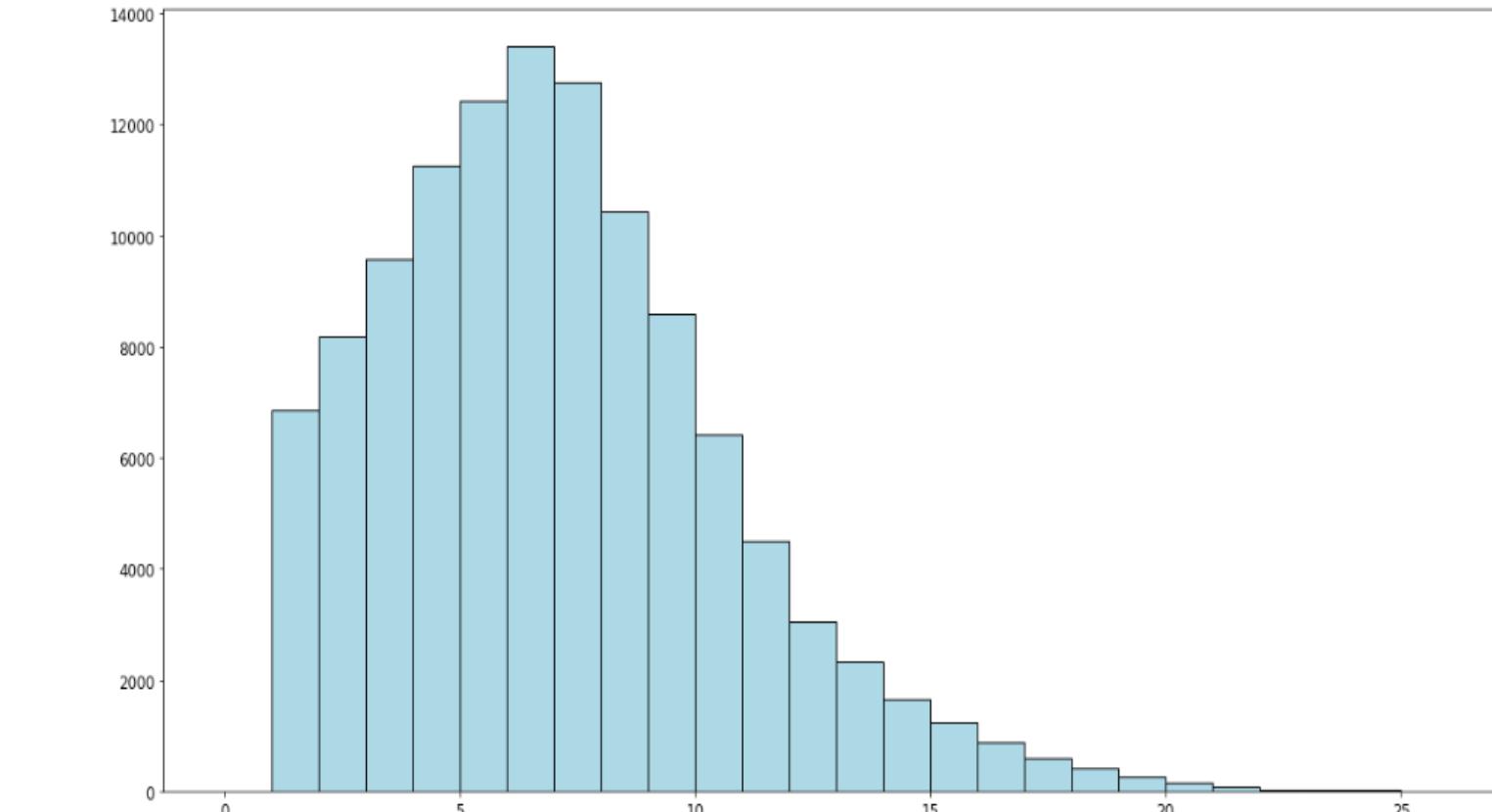
- 플레이리스트 별 수록된 곡 수의 분포를 확인해본 결과 **평균 약 46개(중앙값 30개)**의 곡이 수록되어 있으며 최대 200곡의 수록곡을 포함한 플레이리스트가 존재
- 플레이리스트 당 태그는 **평균 약 4.1개(중앙값 3개)**가 포함되어 있으며 가장 많은 태그 수는 11개
- 플레이리스트에 수록된 곡중에서 대분류 장르 수 분포를 확인해본 결과 플레이리스트 당 **평균 6.6개(중앙값 6개)**의 장르를 포함



플레이리스트 수록곡



플레이리스트 태그



플레이리스트 장르

## 4.3 Data Analysis

# “ 많은 번도를 보이는 태그 ”



# CODE SUMMARY

- 플레이리스트별 태그 테이블을 만들어 매핑 빈도 수가 1000회 이상인 태그만 뽑아서 빈도 기준으로 워드 클라우드로 시각화



# INSIGHT 1

- 기분전환 태그명이 가장 높은 비중을 차지하고 있으며 여름/가을  
겨울 등 계절이나 드라이브, 카페, 매장음악 등 특정 상황과 어울  
리는 태그 등이 상위권에 분포



## **INSIGHT 2**

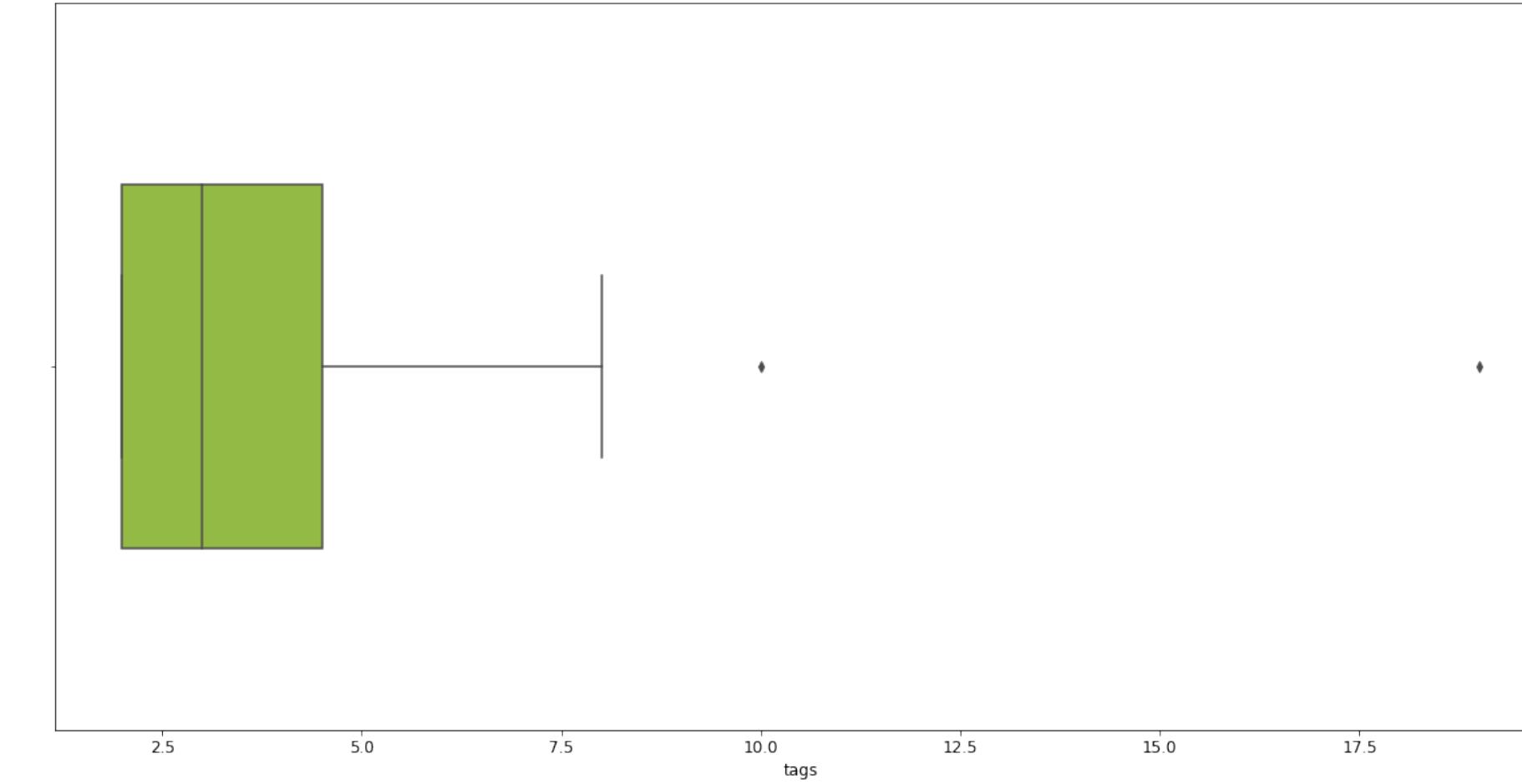
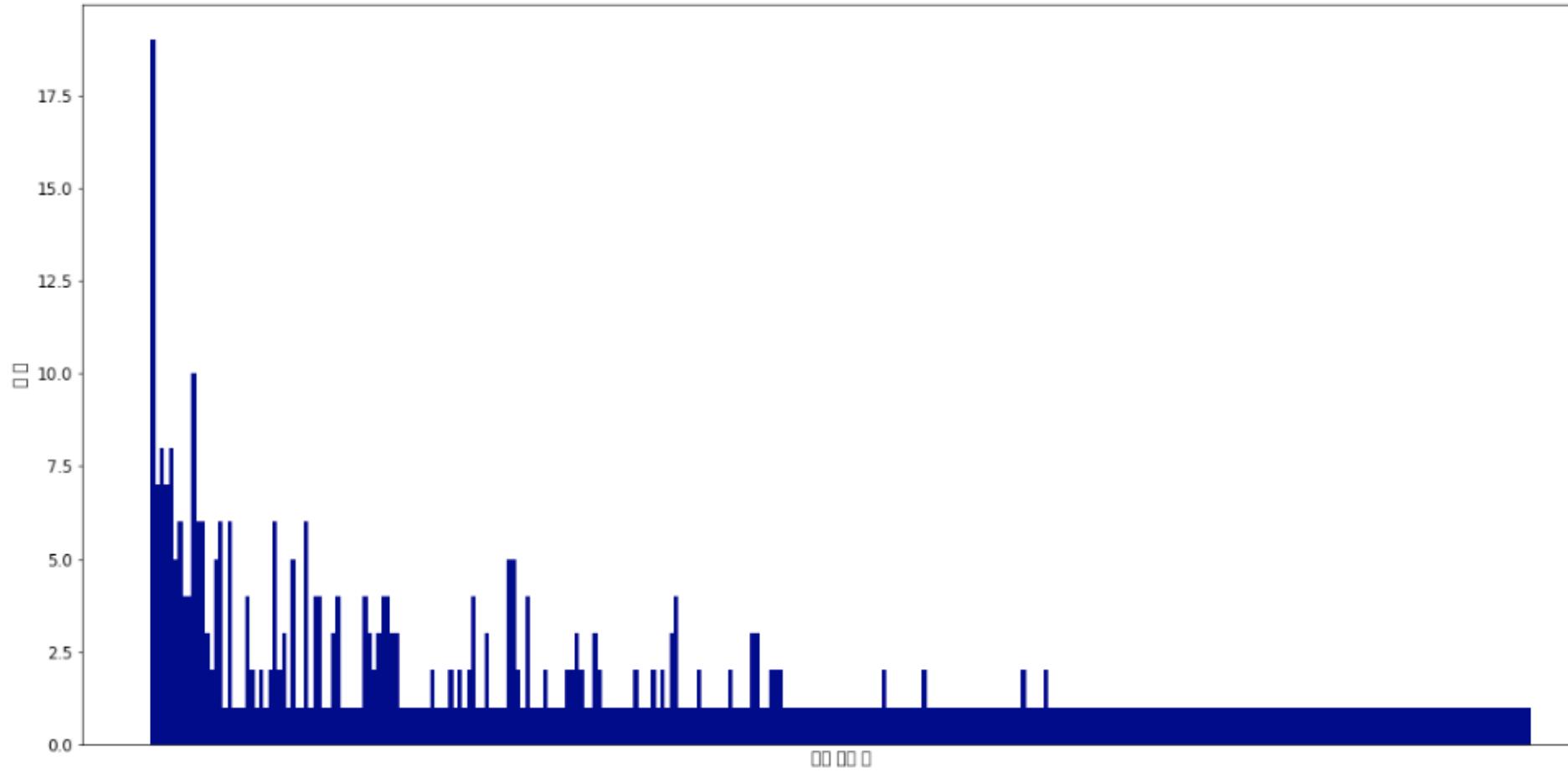
- 발라드, 힙합, 팝, 인디, 락, 댄스, 뉴에이지 등 장르와 관련된 태그도 일부 상위권에 속함



## 4.4 Data Analysis

### “ 역으로 곡 별 태그를 매핑해본다면? ”

- 각 플레이리스트에 한 개 이상의 태그들이 매핑되어 있고, 한 곡 이상의 곡들이 수록되어 있음
- 이 부분을 고려하여 곡이 자신이 속한 플레이리스트들의 태그를 모두 가져오는 방식으로 곡의 메타 정보를 하나 더 파생시킬 수 있음
- 곡별 매핑되는 태그의 수는 평균 18개(중앙값 7개)
- 29160개의 태그 중에서 한 플레이리스트에 306개의 태그가 매핑되었습니다. 그 중 71개의 태그가 중복 매핑되며, 일부 몇개의 태그가 특히 많이 매핑됨. 따라서 플레이리스트와 곡의 특성으로 태그가 중요하다는 것을 알 수 있음



- 대부분의 태그가 곡에 한번씩만 매핑된 것을 확인
- 중복 매핑된 태그들만 모아서 매핑 수를 box plot으로 시각화를 진행하는 계획을 세웠다

- 중복 매핑된 71개의 태그 중 가장 많이 매핑된 태그는 19개로 '락'이 가장 많았고 다음으로 '기분전환', '까페', '팝', '밤' 순
- 일부 태그가 특히 많이 매핑된 것을 알 수 있다.

# 05

## Insight

Insight About DATA



## 5.1 Insight

# Insight

## 플레이리스트와 곡의 속성으로 태그가 중요

- 29160개의 태그중에서 한 플레이리스트에 306개의 태그가 매핑
- 그 중 71개의 태그가 중복매핑됐으며, 일부 몇개의 태그가 특히 많이 매핑됐음

## 플레이리스트와 곡의 속성으로 장르가 중요

- 총 30개의 대분류 장르 중 한 플레이리스트에 평균적으로 약 6개의 장르만이 포함됨

# Review

- 리스트 타입으로 다중값 속성이 많아 그것들을 분리하는 작업이 필요
- ERD 작업으로 나눈 테이블 형태로 DB에 적재해 사용할 수 있었으면 훨씬 메모리 부담이 적고 시간을 절약할 수 있었을 것



06

# 워드 임베드

# Word embeddings

워드 임베딩(Word Embedding)은 단어를 벡터로 표현하는 방법으로, 단어를 밀집 표현으로 변환합니다. 이번 챕터에서는 희소 표현, 밀집 표현, 그리고 워드 임베딩에 대한 개념을 이해합니다.

## 6.1 자연어처리 모델인 Word2Vec

태그: 플레이리스트에 대해 유저가 연상 및 관련 있다고 생각한 단어



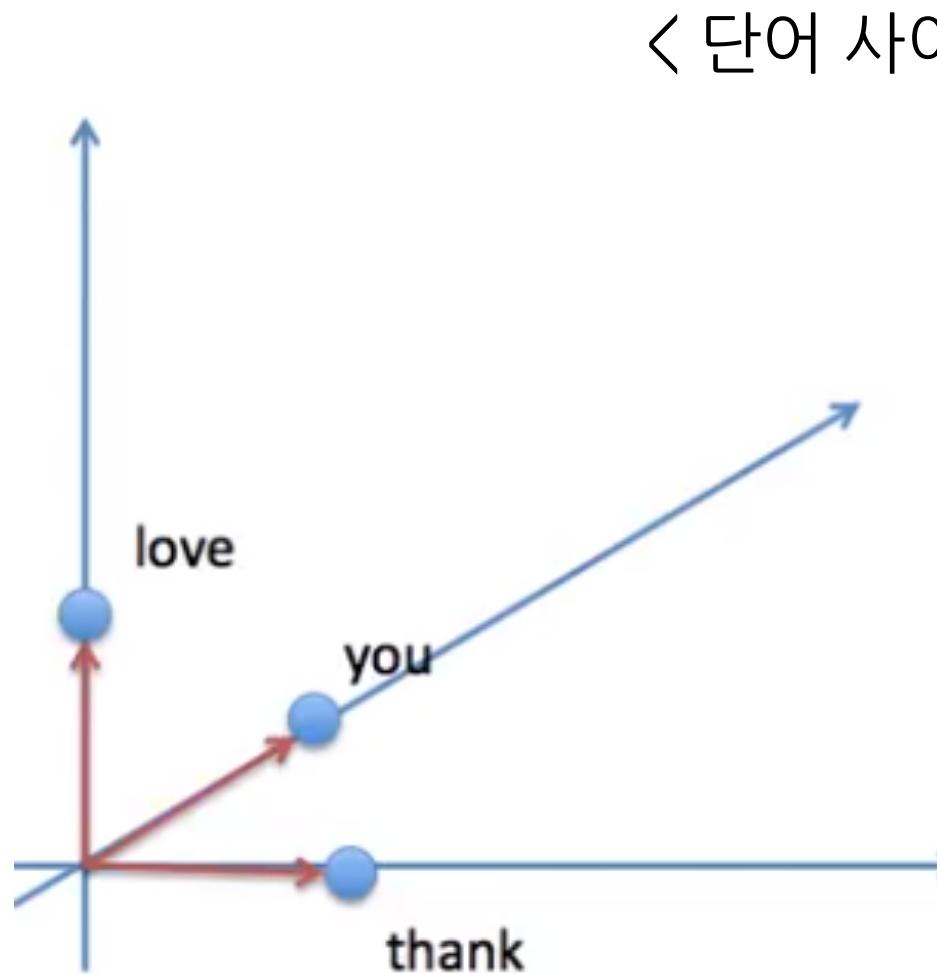
- ‘문자’로 구성된 태그를 추천하기 위한 자연어 처리 모델
- 대표적인 모델이 word2vec이며, 중심 단어를 통해 주변 단어를 예측하는 skip-gram을 사용

## 6.2 원핫인코딩 vs 단어 임베딩 중 무엇이 좋을까?

“ 기계가 학습을 하기 위해서는 문자열이 숫자로 변환되는 과정이 필요 ”

### one-hot-encoding

대표적인 인코딩 방법으로 원핫인코딩이 있지만 우리가 예측하고자 하는 태그는 맥락상 비슷한 태그이므로 관계성을 보여주지 못하는 원핫인코딩은 적합하지 않음

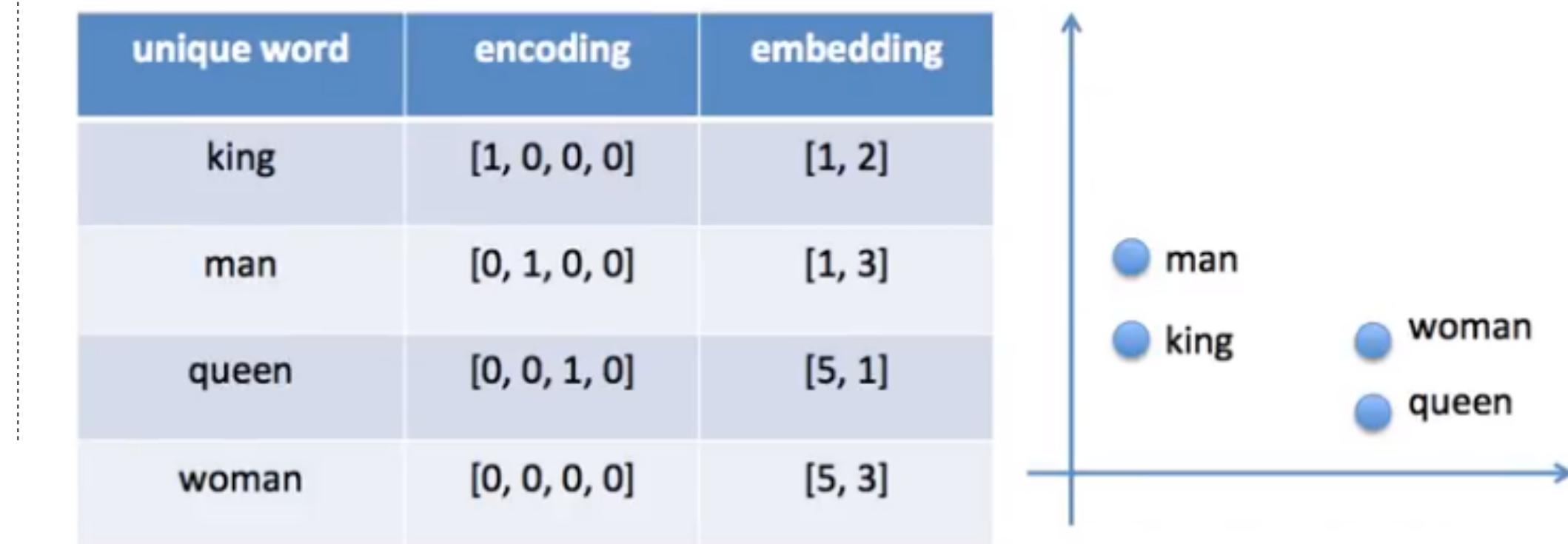


unique word	encoding
thank	[1, 0, 0]
you	[0, 1, 0]
love	[0, 0, 1]

### word embedding

단어 임베딩은 한 단어가 중심단어일 때, 주변에 나타난 단어들의 빈도 등을 통해서 지정한 차원 만큼의 벡터를 만들고, 유사한 맥락에서 등장하는 단어들을 판단할 수 있게됨  
==> 단어들의 관계성을 보여줄 수 있으므로 적합

〈 맥락 상 비슷하거나 반대인 단어 등 관계성 o 〉



## 6.3 모델 학습 및 추천

### 전처리

플레이리스트 수록곡 ID들, 태그들, 플레이리스트 제목단어를 하나의 배열로 변환

Before			After
<code>id</code>	<code>tags</code>	<code>playlist_title</code>	
92056	98906 [좋은노래, 퇴근길] 듣기좋은, 좋아서 듣는 노래		['좋은노래', '퇴근길', '듣기좋은', '좋아서', '듣는', '노래']

### Gensim의 Word2Vec을 사용하여 학습

```
sg = 1 # skip-gram 사용 / 0이면 CBOW 사용  
topn=150  
min_count = 1  
size = 150  
windows = 100
```

## 6.4 모델 학습 및 추천

### 가중된 벡터 값을 생성

학습된 모델을 통해 생성된 벡터에서 [ (각 곡/태그/제목별 벡터값) \* (곡/태그/제목단어 가중치) / (곡/태그/제목수) ]를 계산하여 벡터 값을 생성

```
tmp_vec += song_weight * w2v_model.wv.get_vector(song) / len(song_dict[q['id']])
tmp_vec += tag_weight * w2v_model.wv.get_vector(tag) / len(tag_dict[q['id']])
tmp_vec += title_weight * w2v_model.wv.get_vector(title_word) / len(preprocess_string(q['playlist_title'], custom_filters))
```

### Gensim의 WordEmbeddingsKeyedVectors을 사용

각 플레이리스트 별 (3)번 벡터값을 input함

```
2 p2v_model.get_vector(str(98906))
3
array([
  0.79248989, -4.79923201,  3.13499331,  2.2403059 , -0.71109766,
 -0.24234186,  0.81300825,  1.16993129, -0.87729043, -1.36111498,
  0.14313479,  0.6634022 ,  0.01262588, -0.86097521, -1.33925653,
 -0.69053715, -0.14812325,  0.29269749,  3.81254673, -0.99004757,
  0.54842728, -2.30717754, -2.00491118,  1.66831827, -1.56661594,
  3.29112601, -0.80598533, -2.79047704,  1.32879865,  1.42013025,
 -0.20750301, -2.3087647 , -0.5689798 ,  1.55505204, -2.35712123,
 -0.97213835,  1.98404896,  1.50481546, -0.40067652, -1.48690569,
  0.07291625, -1.60640442, -2.27486134,  0.7363804 ,  4.07068634,
  2.67006302, -1.64201844,  0.86746323, -0.73713374,  3.60533094,
 -0.84034514, -1.46857631, -1.27185941,  0.36380965, -1.39114141,
 -0.86746466, -1.08718157, -2.44982386,  0.6354633 ,  2.44016862,
 -1.52987623,  1.05234873, -3.64600825, -3.05636692,  2.08586621,
  2.61181617, -1.59590673, -0.00830192, -2.70062995,  1.31939232,
```

```
-1.2294203 , -1.1799736 , -1.09805918,  5.23267746,  0.57583559,
-2.81491876,  2.44514465,  0.29398185, -4.55340195, -0.98100442,
 1.63972509, -1.79609263, -0.26529837,  1.42314756, -0.18204841,
 0.92877179, -1.45639729, -0.58407485,  0.0921841 , -1.23727298,
-0.98167998,  0.65646964, -0.64840609, -1.40987468, -1.37957191,
 1.10513401, -0.5505507 ,  0.42068154,  2.76477408, -0.06672392,
 1.64982152, -1.04663527, -1.0795393 , -1.62093675, -0.31274199,
 0.85382217,  0.98225427, -0.1983825 , -0.8590517 ,  0.71487761,
 1.64279819, -1.77736557, -0.61382443,  1.0919981 , -1.48907793,
 0.90190452,  1.49406672, -0.10676186,  1.00459504,  3.18575883,
 0.72510922,  1.38968813,  1.05266333,  1.31776857, -2.90409184,
 1.54255652, -3.69286346,  3.15855622,  2.2000041 ,  0.26495039,
 0.91480744, -1.28472233,  1.47366059, -1.86640656,  2.86081696,
 -0.67372102,  1.92020929,  0.71912932, -0.2544294 ,  2.51824808,
 -0.67447793,  0.17868312,  2.83666134,  0.55705285, -0.68288732,
 3.3847332 , -0.23893154, -1.36193514, -3.42485976, -1.52461648])
```

## 6.5 모델 학습 및 추천

### 추천

플레이리스트 별 벡터 값이 유사한 플레이리스트 150개를 추천받아 그 중 빈도가 높은 태그 10개를 추천받음

#### [ Input ]

tags	id	playlist_title
[좋은노래, 퇴근길]	98906	듣기좋은, 좋아서 듣는 노래

#### [ nDCG ]

Tag nDCG: 0.162665

#### [ 정답 ]

tags	id	playlist_title
[출근길, 나만알고있는노래, 운동]	98906	듣기좋은, 좋아서 듣는 노래

#### [ 예상 ]

id	songs	tags
98906	[374156, 418935, 118049, 144663, 659496, 68352...]	[버스, 지하철, 퇴근, 드라이브, 기분전환, 저녁, 밤, 감성, 출근길, 휴식]

## 6.6 모델 평가 및 연구과제

### Result

- 해당 값은 카카오 아레나 멜론 공개리더보드 100위의 값과 동일
- 하이퍼파라미터를 다양하게 조정하였으나 큰 차이는 없었음
- 추후 사용한 ALS 모델과 비교하였을 때, tag\_nDCG가 매우 높게 나옴

[ nDCG ]

Tag nDCG: 0.162665

### 추후 연구 과제 및 기대사항

- 논문을 참고하여 하이퍼 파라미터 조정 시 성능 향상 기대
- 하이퍼 파라미터 조정후에도 충분한 성능이 나오지 않는다면, 전처리 또는 모델 자체에 대한 수정 필요
- 태그 예측에 있어서 양상블로 다른 모델과 결합 시 성능 향상에 기여할 수 있을 것으로 추후 연구에 활용 가능

# 07

## 협업 필터링

### Collaborative filtering

협업 필터링(collaborative filtering)은 많은 사용자들로부터 얻은 기호정보(taste information)에 따라 사용자들의 관심사들을 자동적으로 예측하게 해주는 방법이다. 협력 필터링 접근법의 근본적인 가정은 사용자들의 과거의 경향이 미래에서도 그대로 유지 될 것이라는 전제에 있다. 예를 들어, 음악에 관한 협력 필터링 혹은 추천시스템(recommendation system)은 사용자들의 기호(좋음, 싫음)에 대한 부분적인 목록(partial list)을 이용하여 그 사용자의 음악에 대한 기호를 예측하게 된다.

이 시스템은 특정 사용자의 정보에만 국한 된 것이 아니라 많은 사용자들로부터 수집한 정보를 사용한다는 것이 특징이다. 이것이 단순히 투표를 한 수를 기반으로 각 아이템의 관심사에 대한 평균적인 평가로 처리하는 방법과 차별화 된 것이다. 즉 고객들의 선호도와 관심 표현을 바탕으로 선호도, 관심에서 비슷한 패턴을 가진 고객들을 식별해 내는 기법이다. 비슷한 취향을 가진 고객들에게 서로 아직 구매하지 않은 상품들은 교차 추천하거나 분류된 고객의 취향이나 생활 형태에 따라 관련 상품을 추천하는 형태의 서비스를 제공하기 위해 사용된다.



# MACHINE LEARNING

## 7.1 협업 필터링 모델 사용 이유

“ 협업필터링은 다른 유저와의 관계를 통해 곡을 추천하는 알고리즘 ”

### 곡은 태그와 다른 성질

‘비발디’의 ‘사계’, ‘태연’의 ‘사계’, ‘MC THE MAX’의 ‘사계’는 텍스트는 동일하나 완전 다른 속성의 것

### 유저에게 곡을 추천하는 방법

1. 곡 자체의 성질 파악
2. 다른 유저와의 관계 파악

## 7.2 라이언's Favorite Movie

라이언에게 영화를 추천한다면?

	토르	어벤저스	아이언맨	어바웃타임	Her	라라랜드
토르	안봄	0	안봄	5	안봄	4
어벤저스	5	5	안봄	1	0	안봄
아이언맨	안봄	4	5	1	0	1
어바웃타임	1	1	안봄	안봄	5	5
Her	안봄	3	3	안봄	안봄	3
라라랜드						

### 7.3 잠재요인이란?

	액션 선호도	로맨스 선호도
액션수치	5	95
로맨스 수치	100	0
	85	20
	23	92
	50	50

	토르	어벤져스	아이언맨	어바웃타임	HER	라라랜드
액션수치	100	90	85	30	10	20
로맨스 수치	10	20	15	90	85	95

## 7.4 선호도 예측

	토르	어벤저스	아이언맨	어바웃타임	Her	라라랜드
곰	0.0	0	0.1	5	4.5	4
고양이	5	5	4.2	1	0	0.1
강아지	4.5	4	5	1	0	1
딸기	1	1	0.0	4.0	5	5
도마뱀	2.9	3	3	3.0	3.2	3

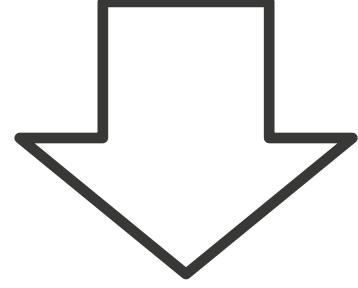
## 7.5 우리 프로젝트에 이를 대입한다면?

유저는 플레이리스트가 됨, 아이템은 곡이 됨, 점수는 1 or 0이 됨

최종적으로 근사 되는 값은, 플레이리스트에 어울리는 정도의 값이 된다

	노래1	노래2	노래3	노래4	노래5	노래6
새벽 2 AM	0.9	1	1	0	0	0
Yello Mix Tape	1	1	0.8	0	0	0.1
맥주와 함께하는	0.0	0	0.1	1	1	1
느낌적인	0	0	0.0	0.9	1	1
비오는날	0.5	0	1	0.5	0.5	1

## 7.6 데이터 전처리

플레이리스트			곡들			
느낌적인 날			[Song 1, Song 2 ..]			
그루브 있는 리스트			[Song 6, Song 3..]			
Yellow Mix Tape			[Song 4, Song 7 ..]			
비오는날			[Song 8, Song 9 ..]			
			 ‘기존 데이터를 희소행렬화’			
	song1	song2	song3	...	song997	
P1	NaN	1	NaN		NaN	1
P2	NaN	NaN	NaN		NaN	NaN
P3	NaN	NaN	NaN		NaN	1
P4	1	1	NaN		NaN	NaN
P5	NaN	NaN	NaN		1	NaN

## 7.7 AlternatingLeastSquares 모델 투닝

메인 하이퍼 파라미터는 ‘ factors ’

```
[23]: # make final_rec json
arena_util.write_json(final_rec_lst_50, 'rec50.json')
arena_util.write_json(final_rec_lst_80, 'rec80.json')
arena_util.write_json(final_rec_lst_100, 'rec100.json')
```

```
[24]: evaluator = evaluate.ArenaEvaluator()
evaluator.evaluate('./arena_data/answers/val.json', './arena_data/rec50.json')

Music nDCG: 0.0796247
Tag nDCG: 0.0321261
Score: 0.0724999
```

```
[25]: evaluator = evaluate.ArenaEvaluator()
evaluator.evaluate('./arena_data/answers/val.json', './arena_data/rec80.json')

Music nDCG: 0.0852301
Tag nDCG: 0.0267591
Score: 0.0764594
```

```
[26]: evaluator = evaluate.ArenaEvaluator()
evaluator.evaluate('./arena_data/answers/val.json', './arena_data/rec100.json')

Music nDCG: 0.0867067
Tag nDCG: 0.0221192
Score: 0.0770186
```

factors가 높아질 수록 Music은 높아지고, Tag는 낮아지는 것 확인

```
[38]: # song_rec_600
als_model_600 = AlternatingLeastSquares(factors = 600)
als_model_600.fit(ratings.T)
with open('./als_model_600.pkl', 'wb') as f:
    pickle.dump(als_model_600, f)
```

```
song_rec_600 = {}
for i in val['id']:
    plst_rec_song = als_model_600.recommend(i, ratings, 100)
    song_rec_600[i] = plst_rec_song
```

```
# song_rec = 600 / tag_rec = 50
final_rec_lst_com = []

for key in song_rec_600.keys():
    final_rec = {}
    final_rec['id'] = key
    final_rec['songs'] = [int(i) for i, s in song_rec_600[key]]
    final_rec['tags'] = [id2tag[int(i)] for i, s in tag_rec_50[key]]
    final_rec_lst_com.append(final_rec)
```

```
arena_util.write_json(final_rec_lst_com, 'rec_600_50.json')
```

```
evaluator = evaluate.ArenaEvaluator()
evaluator.evaluate('./arena_data/answers/val.json', './arena_data/rec_600_50.json')

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=15.0), HTML(value='')))
```

```
Music nDCG: 0.0959749
Tag nDCG: 0.0321261
Score: 0.0863976
```

## 7.8 Result

```
[63]: val[val['id'] == 136555]
```

tags	id	plylst_title	songs	like_cnt	updtdate
4393 [벚꽃놀이, 벚꽃, 봄캐롤, 봄]	136555	개화시기 맞춰 준비하는 벚꽃 놀이 플레이리스트	[447803, 581789, 601517, 585621, 696278, 32906...]	19	2019-05-02 18:19:18.000

```
[56]: for song in val[val['id'] == 136555]['songs'].values[0] :  
    print(song_meta.loc[song, 'song_name'], end=', ')
```

부끄럼, 봄이 좋냐??, Picnic, 백허그, Untitled01, 노래, 예뻤어, 다만 널 사랑하고 있어, Sunshine (With 권순관 of 노리플라이), 벚꽃이 떨어질 때, Desert, 너라는 세상, 바라봄 (Feat. 재희 of 마인드유), 왜 그러긴 (Feat. 그리풀리), 조금씩, 천천히, 너에게, 좋다, 이게 사랑일까 (Feat. 하은), 벚꽃 엔딩, Everyday, Madeleine Love,

```
[57]: for song in rec[rec['id'] == 136555]['songs'].values[0] :  
    print(song_meta.loc[song, 'song_name'], end=', ')
```

꽃송미가, 봄봄봄, Happy Things, 봄인가 봄 (Spring Love), 봄이 와, 오늘밤은 어둠이 무서워요, 꽃길만 걷게 해줄게, 농담 반 진담 반, 폰서트, Beautiful Day, 벚꽃이 지면, 소란했던 시절에, 지구가 태양을  
네 번, 좋아해 (bye), 바래진 기억에, 어떻게 생각해, 공원여행, 봄날의 기억, 선물, 캠퍼스 커플 (With 옥상달빛), 내꺼라면, Bye bye my blue, 봄바람 (Feat. 나얼), 너를 보네 (Feat. 권정열 Of 10cm), 너를  
공부해, 봄날, 말하고 싶은 게 있어, 봄 사랑 벚꽃 말고, 봄처녀, Love Blossom (러브블러썸), 나만, 봄, 나랑 아니면, Sunshine, 처음엔 사랑이란게, 200%, Get (Feat. Beenzino), 선을 그어 주던가, 로맨틱하  
게, 너에게만 반응해 (Feat. 이소은), 심술, 동화, Love Fiction, 우리 오늘 만날까, 남이 될 수 있을까, Love Love Love, 나 요즘, 노래방에서, 모닝콜, No Make Up, 살빼지 마요, 그게 뭐라고, 좋아하는 마음,  
오늘, 봄날에 눈이 부신, 그대와 함께, 무지개 (Feat. 문닭 Of 오브로젝트), 잊어버리지마 (Feat. 태연), 너만 생각하면, 봄 타나봐 (BOMTANABA), 완벽한 봄날 (Feat. 소심한 오빠들), Good Night, 잘됐으면 좋  
겠다, 썸 탈꺼야, 우주를 줄게, 안아줘, Perhaps Love (사랑인가요) (Prod.By 박근태), 너라면 괜찮아, 프리지아, 고마워 내 사랑, 내가 되었으면, 조별과제, 그만 좀 예뻐, 너를 그리다, 팔레트 (Feat. G-DRAGO  
N), 여자친구가 생겼으면 좋겠다 (Feat. 쇼코), 알았더라면, 왜 또 봄이야, 너란 봄 (Feat. 하림), 한여름밤의 꿀, It Was You, 바람이 부네요, Lay Back, 기억을 걷는 시간, It's You (Feat. ZICO), Nostalgi  
a, 좋은 날, 커플, Psychic (사이릭), Talk, 꿈에, 남자 없이 잘 살아, 미쳤나봐 (With 권정열 Of 10cm), 너를 그리다, 장마, 사랑이 그리워, Lonely (Feat. 태연), 스토퍼, 봄바람, 위임위임, 너를 좋아하니까)

```
[58]: for song in answer[answer['id'] == 136555]['songs'].values[0] :  
    print(song_meta.loc[song, 'song_name'], end=', ')
```

스쿨버스, 겨울에서 봄, 봄트랙, Timid Youth, Sunshine, 꿀차, Our love is great, 청혼, 좋은 날엔 언제나, 바야흐로 사랑의 계절, 바라만 봐도 좋은데, Love Shine, 봄날, 벚꽃 그리고 너, 꽃송미가, 행운을  
빌어요, 봄봄봄, 혹시 자리 비었나요?, 오월, 봄날에 눈이 부신, 설레고 있죠,

## 7.9 Result

### Result

최종곡nDCGScore는 0.0958점으로 리더보드의 82등과  
유사한 점수까지 낼 수 있었음.

#	△1d	팀명	멤버	스코어	곡nDCG	태그nDCG	제출 횟수	마지막제출
82	-	중간고사대체과제 ✓		0.139085	0.102915 (83)	0.344047 (75)	18	3달 전
83	-	드림카카오99.9%		0.130958	0.092798 (87)	0.347197 (74)	3	6달 전

## 7.10 Result

```
[65]: val[val['id'] == 17672]
```

```
[65]:   tags      id          plylst_title  songs  like_cnt      updtd_date
116 [봄] 17672 봄꽃이 만발한 길을 걸으며 듣고싶은 잔잔한 봄 연주곡      []         9 2017-04-05 09:12:45.000
```

```
[66]: for song in val[val['id'] == 17672]['songs'].values[0] :
    print(song_meta.loc[song, 'song_name'], end=', ')
```

```
[67]: for song in rec[rec['id'] == 17672]['songs'].values[0] :
    print(song_meta.loc[song, 'song_name'], end=', ')
```

내 평생에 가는 길, 오! 사랑, I Understand, HATE MY GUTS, Hummel : Variations On The March From Nicolo Isouard's `Cendrillon` Op.40a - Theme And Var. I, II (훔멜 : 이수아르의 `상드리옹` 행진에의 변주곡 작품번호 40a - 주제와 변주 1, 2), Vivaldi: Concerto For Violin And Strings In E Major, Op.8, No.1, RV 269 &#34;La Primavera&#34; - 3. Allegro (Danza pastorale), What Shall We Do With The Drunken Sailor, 삼팔선의 봄, Chopin : Waltz No.7 In C Sharp Minor Op.64-2 (쇼팽 : 왈츠 7번 올림 다단조 작품번호 64-2), Santorini, Breath (Intro), Have Yourself A Merry Little Christmas, 벌꿀오소리 (Feat. Changstarr\*, viceversa, BahN), 손잡아요 (Feat. 노재정), 독종, Boss, So Lady (Non-LP Version), Big Bang Baby, Floatation (Single Edit), Ring Ring (Feat. Vory), Historia De Un Amor, Jah Army (Feat. Damian `Jr. Gong` Marley), Hollyhood - A COLORS SHOW, Who`s Gonna Shoe Your Pretty Little Feet?, 戰士の休息 / Senshino Kyuusoku (전사의 휴식), Man In The Mirror (Acoustic), 잘지내요 청춘, Boom, Shelter in Place, Hinatani Saku Hana (양지에 피는 꽃), 그녀에게, Demonic Science, What Bpm Is Love?, Like Spinning Plates (Live), Autumn Leaves, 러브스토리 (Feat. 크러쉬), Comfortable, Strange Fruit, Schubert : Rosamunde D.797 - Overture (슈베르트 : 로자문데 - 서곡), Saturday Night Movies, On My Way, Gang Gang, 떠나갈래, Water Lily, 헨리 클레이미 워크 : 할아버지의 시계 (Henry Clay Work : Grandfather's Clock), Start A Fire (Feat. Christina Milian), 우리는 하나님의 성전, Tango (Feat. Bonny) (Radio), 좋을텐데, If They Knew, Schumann : Kinderszenen Op.15 - I. Von Fremden Landern Und Menschen (슈만 : 어린이의 정경 작품번호 15 - 1번. 미지의 나라들) (엄마 심장소리 자장가 태교), Genesis, Lagrima, 기억시옷 (Feat. Paloalto) (Prod. BOYCOLD), Swing Of Mind, Walk In My Shoes, 낮은 이의 노래 (MR), Real Joy, 43, Schumann : 12 Klavierstücke Op.85 - XII. Abendlied (슈만 : 12개의 피아노 소품 작품번호 85 - 12번. 저녁의 노래), Dead Wrong (Feat. Ty Dolla \$ign), Love Explosion, 이 밀음 더욱 굳세라, 봄날의 구름이 피어나 (Feat. 빌리어코스티), Amadeu, Learnin&#39; The Blues, Only You, 서두르지 말아요 (Remix), 빠빠빠 출라, How Deep Is Your Love, 그대여, 메이데이 (Mayday), Piano Man, Burgmuller : Etudes Op.109 - Spinning Song (부르크뮐러 : 연습곡 작품번호 109 - 물레의 노래), 짤레꽃 (엄마), Wake Up, 사티: 짐노페디 1번 (3 Gymnopodies No.1), Cowboys, Under Your Skin, 심청전 2부, 사랑하는 계절, Rhythm Of The Night, 새벽빛 만개, Psychiatric (Original Mix), The Last Love Song, Blues In The Dungeon (With Stuff Smith), 둘지 (Remix), A Moonlight Song, 없니, Tunel Do Tempo, 비를 내려줘요, 그대라면 믿을게요 (Piano & String Ver.), Bojangles, Las Vegas Turnaround (The Stewardess Song), White Christmas, A Nuh Me Seh So (Ragga Hip-Hop Mix), I Decide (Feat. Leejee Violet), 이별 후 잠들기 5분 전 (With 현정), Tell Me Why, Queen 명곡 멜로디,

```
[68]: for song in answer[answer['id'] == 17672]['songs'].values[0] :
    print(song_meta.loc[song, 'song_name'], end=', ')
```

벚꽃 그늘에 쉬었다 가자, 봄 사랑 벚꽃 말고, 첫, 그 설렘, 별이 든다 (Solo Piano Ver.), 봄이 주는 선물, Bom Bom, 슬픔의 피에스타, A Song For Benny, 봄의 정원 `물방울`, 春よ,&#26469; / Haruyokoi (봄이여 오라) (Ver. 2), Rainy Day (비오는 날), 花園 / Hanazono (꽃의 정원), Sakura (벚꽃), 봄을 그리다, 1997 Spring, 꽃, 봄날의 이별 `안녕`, 봄타령,

## 7.11 Result

### 🔧 개선 사항

#### ① First

전체 데이터의 값과 수에 따라 최적의 하이퍼 파라미터가 다른 것을 알 수 있었음

#### ② Second

들어오는 데이터에 따른 튜닝 자동화

#### ③ Third

콜드 스타팅의 문제 해결 불가

A complex network graph with numerous small, glowing teal and yellow nodes connected by thin, translucent lines, forming a dense web-like structure.

# 08

## 컨텐츠 기반 모델

### Contents Based Model

콘텐츠 기반 필터링은 말 그대로 콘텐츠에 대한 분석을 기반으로 추천을 구현하는 방법이다. 콘텐츠를 분석한 프로파일(item profile)과 사용자의 선호도(user profile)를 추출하고 유사성 분석을 통해 추천을 수행한다. 가령 슈퍼맨이라는 영화에 대한 사전 분석을 통해 SF, 영웅 등의 특징을 기록하고 슈퍼맨을 본 사람에게 배트맨이라는 비슷한 종류의 영화를 추천해주는 방식이다.

### 사용자가 특정한 아이템을 선호하면?

그 아이템과 비슷한 콘텐츠를 가진 다른 아이템을 추천

### 사용자가 특정 곡을 좋아한다면?

그 곡의 장르, 아티스트, 앨범 등 유사한 다른 곡을 추천하는 방식

#### [ Workflow ]

1. 곡의 특성을 추출하여 하나의 열로 통합(장르id, 아티스트id, 앨범id)
2. countvectorizer로 곡-특성 희소행렬 만들
3. countvectorizer의 코사인 유사도 도출
4. 유사도가 높은 순으로 정렬하여 곡의 id 추출

## 8.2 Data Analysis

전처리

```
df_song_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 707989 entries, 0 to 707988
```

```
Data columns (total 11 columns):
```

#	Column	Non-Null Count	Dtype
0	id	707989 non-null	int64
1	song_name	707989 non-null	object
2	artist_id_basket	707989 non-null	object
3	artist_name_basket	707989 non-null	object
4	album_id	707989 non-null	int64
5	album_name	707985 non-null	object
6	song_gn_gnr_basket	707989 non-null	object
7	song_gn_dtl_gnr_basket	707989 non-null	object
8	genre_literal	707989 non-null	object
9	artist_literal	707989 non-null	object
10	album_literal	707989 non-null	object

```
dtypes: int64(2), object(9)
memory usage: 59.4+ MB
```

장르, 아티스트, 앨범 정보를 모두 한 열에  
포함(유사도 측정)

```
df_song_info['genre_literal'] = #
    df_song_info.song_gn_dtl_gnr_basket.apply(lambda x : (' ').join(x))
df_song_info['artist_literal'] = #
    df_song_info.artist_id_basket.apply(lambda x : str(x[0]))
df_song_info['album_literal'] = #
    df_song_info['album_id'].apply(lambda x : str(x))

df_song_info['all'] = df_song_info['genre_literal'] + #
    " " + df_song_info['artist_literal'] + #
    " " + df_song_info['album_literal']
```

```
df_song_info['all']
```

```
0      GN0901 2727 2255639
1      GN1601 GN1606 29966 376431
2      GN0901 3361 4698747
3      GN1102 GN1101 838543 2644882
4      GN1802 GN1801 560160 2008470
...
707984      GN2001 166499 65254
707985      GN0901 11837 44141
707986      GN0105 GN0101 437 2662866
707987      GN1807 GN1801 729868 2221722
707988      GN0601 GN0604 895 34663
Name: all, Length: 707989, dtype: object
```

곡의 주요 특성: 장르, 아티스트, 앨범

### 8.3 콘텐츠 기반 모델

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

count_vect = CountVectorizer(min_df = 0, ngram_range = (1,2))

song_mat_1 = count_vect.fit_transform(df_song_info_1['all'])
song_sim_1 = cosine_similarity(song_mat_1, song_mat_1)
song_sim_sorted_ind_1 = song_sim_1.argsort()[:, ::-1][:, :10]
```

1. CountVectorizer 희소행렬 생성
2. 희소행렬로 코사인 유사도 분석
3. 코사인 유사도 상위 10개곡 추출

## 8.3 콘텐츠 기반 모델 결과 및 개선점

### 추천결과

		id	song_name	artist_id_basket	artist_name_basket	album_id	album_name	song_gn_gnr_basket	song_gn_dtl_basket
1	1	Bach : Partita No. 4 In D Major, BWV 828 - II....	[29966]	[Murray Perahia]	376431	Bach : Partitas Nos. 2, 3 & 4	[GN1600]	[GN1600]	
1078	1078	Schumann : Sonata For Piano No.1 In F- Sharp M...	[29966]	[Murray Perahia]	48317	Schumann : Piano Sonata, Op.11 & Kreisleriana,...	[GN1600]	[GN1600]	
6688	6688	J.S. Bach: French Suite No.1 in D Minor, BWV 8...	[29966]	[Murray Perahia]	10004046	Johann Sebastian Bach : The French Suites	[GN1600]	[GN1600]	
29430	29430	Chopin : Barcarolle In F-Sharp Major, Op.60	[29966]	[Murray Perahia]	47306	Chopin : Impromptus	[GN1600]	[GN1600]	
16884	16884	Bach : English Suite No.6 In D Minor, BWV811 -...	[29966]	[Murray Perahia]	49708	Bach : English Suites Nos.1, 3 & 6	[GN1600]	[GN1600]	
48111	48111	Brahms : Handel Variations Op.24 (브람스 : 헨델 변주곡...)	[29966]	[Murray Perahia]	1072776	Brahms : Handel Variations	[GN1600]	[GN1600]	

### 추후 연구 과제 및 기대사항

- 데이터의 양이 방대한 경우 희소행렬을 사용한 코사인 유사도를 구할때 용량 문제 발생(데이터 70만개)
- 코사인 유사도를 구하는 방법을 논문 및 다양한 자료를 바탕으로한 연구가 필요

## 8.4 Classic Boy Lian

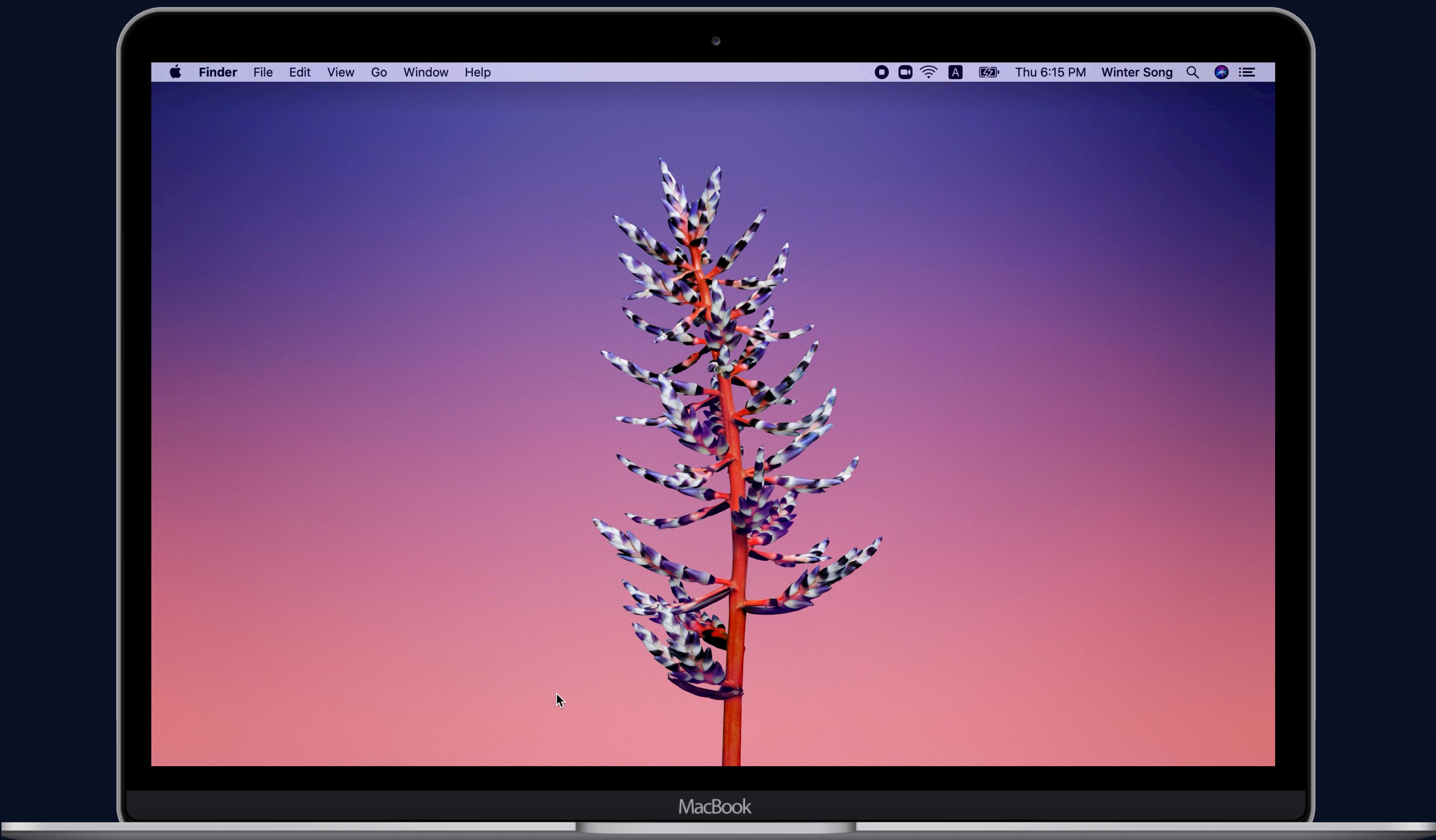
### 클래식을 좋아하는 개발자 라이언

라이언이 듣는 노래는 무엇일까?

- 라이언은 클래식 노래를 들으면서 일하기 좋아하는 개발자다.
- 클래식 중 바흐의 노래를 특히 좋아한다.
- 바흐의 노래와 비슷한 10 곡을 추천을 원함



## 8.5 Home Boy Lian



09

# 딥러닝 모델

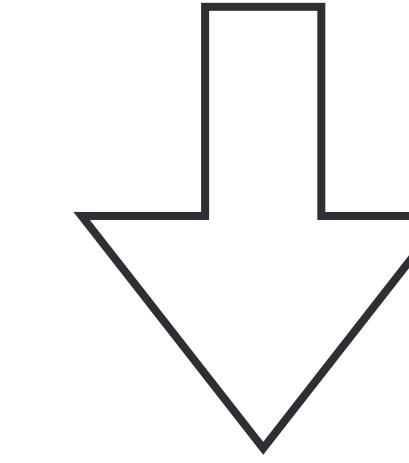
# Deep Learning

심층 학습(深層學習) 또는 딥 러닝( Deep structured learning, deep learning 또는 hierarchical learning)은 여러 비선형 변환기법의 조합을 통해 높은 수준의 추상화(abstractions, 다양한 데이터나 복잡한 자료들 속에서 핵심적인 내용 또는 기능을 요약하는 작업)를 시도하는 기계 학습 알고리즘의 집합<sup>[1]</sup>으로 정의되며, 큰 틀에서 사람의 사고방식을 컴퓨터에게 가르치는 기계학습의 한 분야라고 이야기할 수 있다.



## 9.1 How about Neural based

신경망을 사용해 태그를 추천할 순 없을까?

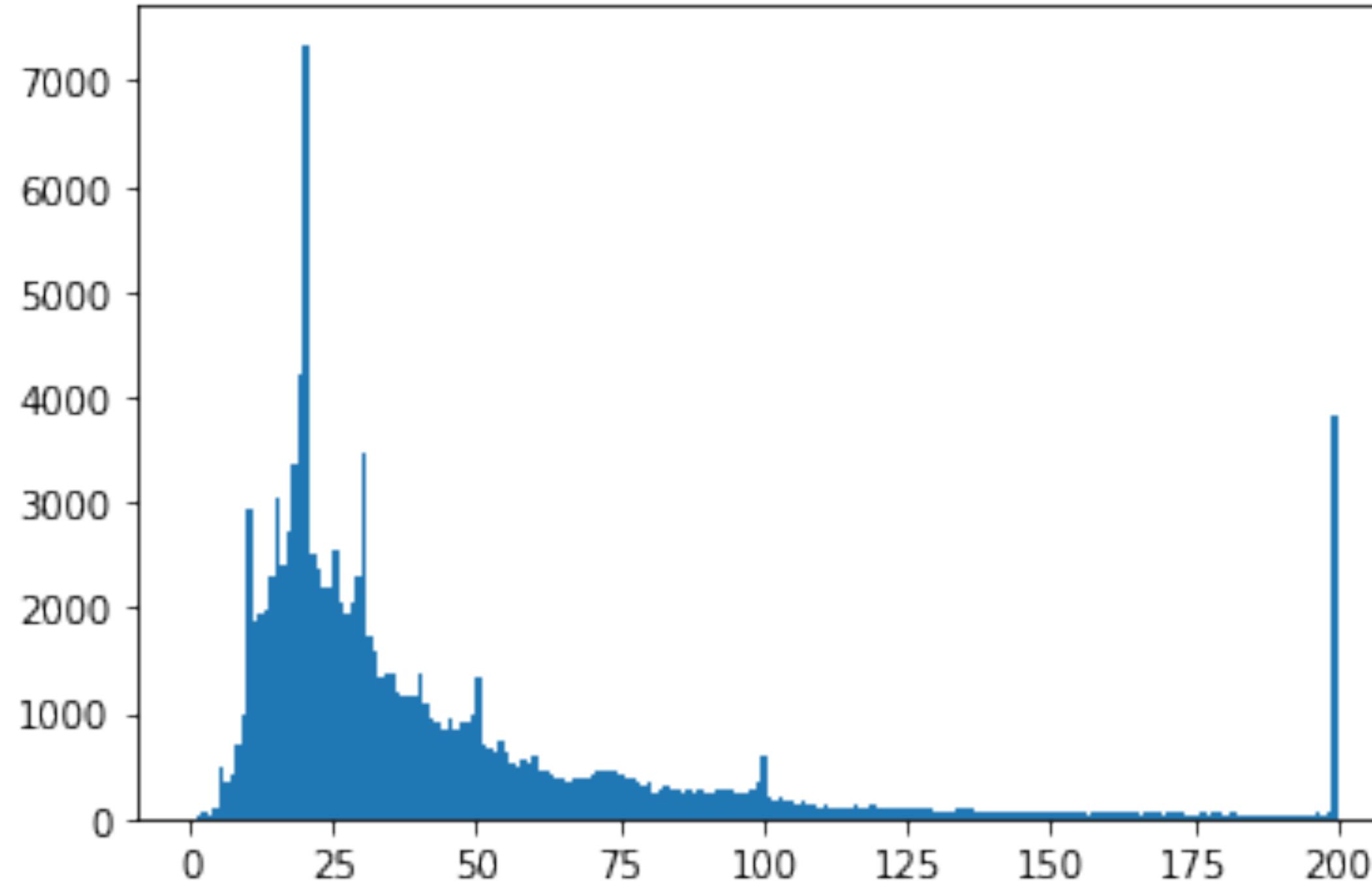


플레이리스트의 곡을 입력하면  
어울리는 태그를 추천할 수 있는 신경망을 만들어보자!

## 9.2 플레이리스트별 곡의 개수 빈도 계산

### 플레이리스트별 곡의 개수 빈도 계산

입력층 Neural 결정하기



Min	1.000000
25%	19.000000
50%	30.000000
75%	54.000000
Max	200.000000



- 플레이리스트의 75%는 19곡 이상이 있다!
- 입력층의 Neural은 19개로 결정
- 곡의 대분류 장르를 입력
- 플레이리스트의 곡이 19개 이하일 경우 0으로 채움

## 9.3 등장한 태그의 빈도를 계산

〈출력층 Neural 결정하기〉

```
array([['기분전환', 16465],  
       ['감성', 11417],  
       ['휴식', 11215],  
       ...,  
       ['다같이흔들어보세', 1],  
       ['계절변화', 1],  
       ['', 1]], dtype=object)
```

### 등장한 태그의 빈도를 계산

ex) “기분전환”태그는 16465회 등장

✓ 1,000회 이상 등장한 태그 64개를 출력층으로 결정

☰ 플레이리스트별 태그 결정

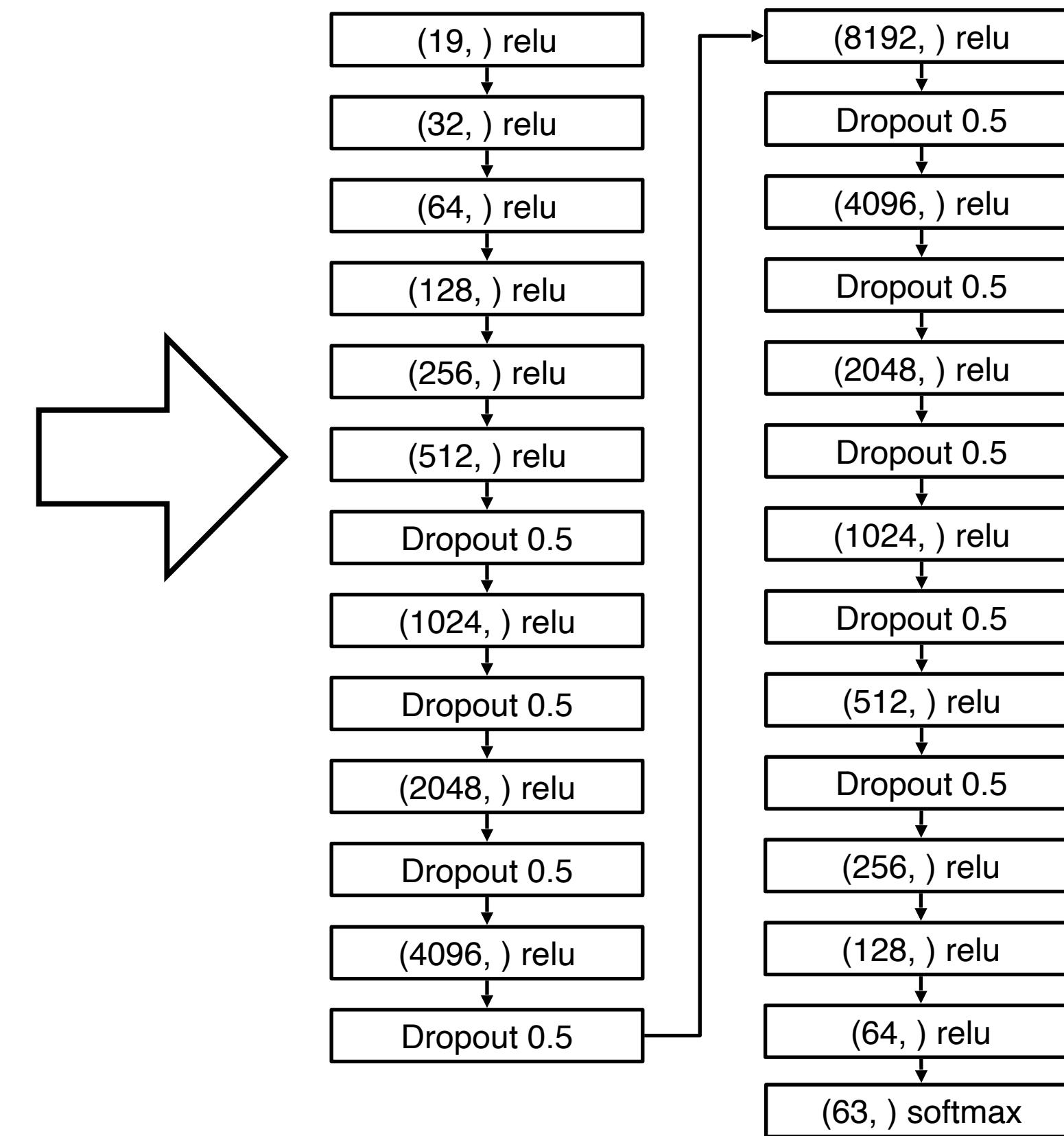
- 1) 태그 64개를 하나씩 word2vec에 입력하여 유사한 단어 300개를 추출
- 2) 태그 A를 유사한 단어 300개중 하나라도 플레이리스트가 가지면 해당 플레이리스트의 태그는 A

## 9.4 Data I/O

입력 데이터  
11907개, 검증데이터 5670개

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	14	9	10	9	9	9	10	9	13	14	10	19	9	9	14	9	10	9	11
1	1	6	1	6	1	6	1	5	5	5	5	1	5	5	10	10	17	17	1
2	4	4	5	4	1	4	3	5	5	3	8	5	4	5	4	5	5	5	5
3	15	1	9	13	13	13	17	1	5	1	1	17	15	15	17	18	17	1	
4	1	1	25	2	25	2	25	25	2	2	2	25	2	2	2	2	25	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
18895	1	25	15	9	9	13	4	9	4	15	15	9	17	13	18	4	1	13	
18896	15	4	5	5	5	1	1	1	1	1	4	1	1	5	1	1	1	1	
18897	10	10	10	15	9	9	10	15	10	9	10	14	9	9	10	14	15	15	
18898	5	5	5	1	1	1	1	1	1	1	1	5	1	5	1	1	1	1	
18899	1	17	1	10	14	9	14	1	19	19	14	5	5	19	17	19	5	21	

신경망  
18layer, 400epoch

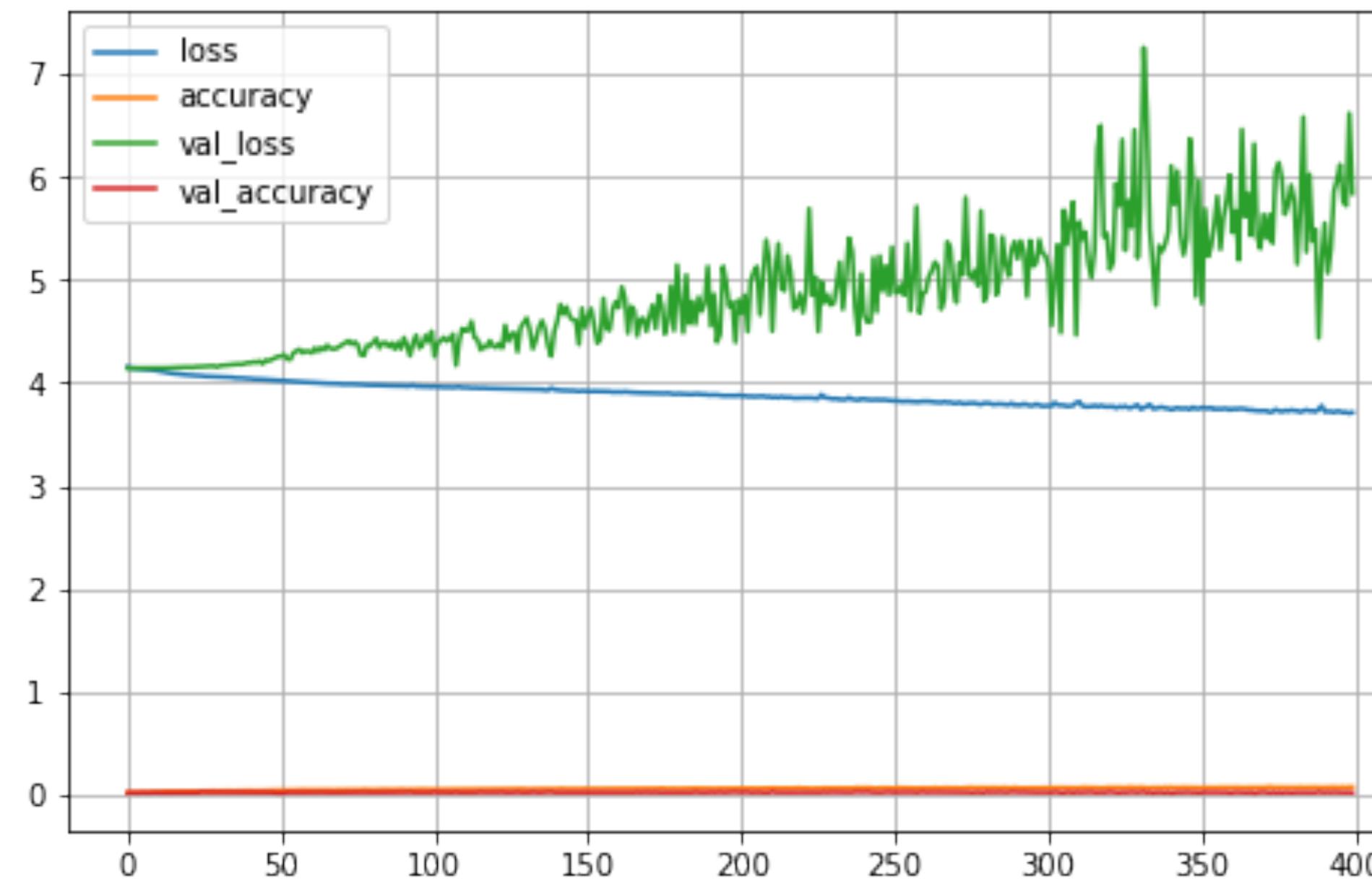


출력 데이터

tag_index	
0	22
1	11
2	1
3	5
4	16
...	...
47352	62
47420	62
47494	62
47509	62
47580	62

## 9.4 Result

< 학습결과 >



< 400epoch 수행 결과 >

약 0.068의 accuracy를 갖는 신경망이 만들어졌다. Test 데이터로 태그 추정 결과 약 0.015의 accuracy를 가진다.

< 결론 >

- 곡의 장르와 태그사이에 연관성을 찾을 수 없었다.
- 사용자가 입력한 태그와 장르명 사이의 연관성을 이용한다면 조금 더 높은 accuracy를 가질 수도 있을것 같다.

```
Epoch 396/400
373/373 [=====] - 84s 225ms/step - loss: 3.7218 - accuracy: 0.0636 - val_loss: 6.1235 - val_accuracy: 0.0174
Epoch 397/400
373/373 [=====] - 84s 225ms/step - loss: 3.7127 - accuracy: 0.0691 - val_loss: 5.7665 - val_accuracy: 0.0159
Epoch 398/400
373/373 [=====] - 84s 224ms/step - loss: 3.7153 - accuracy: 0.0626 - val_loss: 5.7254 - val_accuracy: 0.0159
Epoch 399/400
373/373 [=====] - 84s 225ms/step - loss: 3.7045 - accuracy: 0.0696 - val_loss: 6.6235 - val_accuracy: 0.0181
Epoch 400/400
373/373 [=====] - 84s 224ms/step - loss: 3.7128 - accuracy: 0.0682 - val_loss: 5.8426 - val_accuracy: 0.0144
time: 9.0 h 14.622377872467041 m ( 33674.62237787247 s )
```



# 10

## 개선사항 및 결언

### About the project

Investment generally results in acquiring an asset, also called an investment. If the asset is available at a price worth investing, it is normally expected either to generate income, or to appreciate in value, so that it can be sold at a higher price. Investment generally results.

income, or to appreciate in value, so that it can be sold at a higher price. An investor may bear a risk of loss of some or all of their capital invested.

### 개선사항

1. 배치학습으로 단 한번만 학습하여 기존의 데이터만 사용할 수 있음. 온라인으로 변화하는 데이터에 대한 학습 필요
2. 모델의 파라미터 조절이 필요
3. 안정적인 결과 도출을 위해 여러 모델에 가중치를 주어 voting하는 방식 필요

### 결언

실제 데이터를 이용하여 일상에 사용하는 시스템을 구현해냈다는 점에 의미가 있었다.