# K-최근접 이웃 알고리즘을 활용한 Melon Playlist Continuation

#### 개요

- Efficient K-NN for Playlist Continuation (RecSys'18 Challenge) 논문 기반
- K개의 유사한 플레이리스트를 구하고 그 플레이리스트에 수록된 곡에 점수를 부여
- 가장 점수가 높은 100개의 곡을 추천

#### 코사인 유사도

• 내적공간의 두 벡터간 각도의 **코사인**값을 이용하여 측정된 벡터간의 유사한 정도를 의미한다.

$$similarity = cos(\Theta) = \frac{A \cdot B}{||A|| ||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

• -1 ~ 1의 값을 가지며, 1에 가까울 수록 유사하다고 판단.

## Finding similar playlists 1

$$s_{uv} = \sum_{i \in I} \frac{r_{ui}r_{vi}}{\|R_u\|_2 \|R_v\|_2},$$

•  $S_{uv}$ : 두 플레이리스트(u,v)간의 코사인 유사도

## Finding similar playlists 2

• 두 플레이리스트의 등장하는 곡의 빈도수를 이용해 구한 weight을 곱한다.

$$s_{uv} = \sum_{i \in I} ((f_i - 1)^{\rho} + 1)^{-1} \frac{r_{ui}r_{vi}}{\|R_u\|_2 \|R_v\|_2}$$

- $f_i$ : Number of playlists containing track i
- p: optimal value (0.44)

## Finding similar playlists 3

- Amplification
  - $s_{uv}^a$ : to amplify the importance of more similar playlists. (a=2)
- Normalization

$$\tilde{S}_{uv} = \frac{S_{uv} - minS_u}{maxS_u - minS_u}$$

• 최종적으로  $\tilde{s}_{uv}^a$  를 사용한다.

#### Prediction

- 최종적으로  $S_{uv}^a$ 가 제일 높은 1500개의 플레이리스트를 구하고 각 플레이리스트에 등 장하는 곡들에게 점수( $S_{uv}^a$ )를 부여한다.
- 가장 점수가 높은 곡들 100개를 추천한다.

#### 최종결과

니더보드							제출내역	제출하기
공개	리더보드							
#	∆1d	팀명	멤버	스코어	곡 nDCG	태그 nDCG	제출 횟수	마지막 제출
1	-	js (My team)		0.289624	0.269720 (1)	0.402414 (7)	3	8일 전
2	-	둥꿍둥꿍	2	0.284174	0.251201 (2)	0.471018 (2)	8	13일 전
3	-	세진이와아이들	2	0.28394	0.250919 (3)	0.471058 (1)	2	22일 전
4	-	hh		0.276863	0.244249 (4)	0.461677 (3)	1	한 달 전
5	-	sverige		0.268129	0.235251 (5)	0.454436 (4)	3	2일 전
6	-	귀여운 회긔		0.267297	0.234293 (6)	0.454323 (5)	4	한 달 전
7	new	rena		0.241663	0.220567 (7)	0.361209 (8)	3	6시간 전
8	▼1	멜룬파이브	2 2 2	0.225228	0.186099 (8)	0.446957 (6)	36	2달 전
9	▼1	MTG		0.028538	0.005149 (10)	0.161075 (10)	7	한 달 전
10	▼1	음잘알		0.025581	0.000129 (11)	0.169805 (9)	2	22일 전
11	▼1	TASP		0.010762	0.008083 (9)	0.025944 (11)	8	한 달 전

• 플레이그라운드 리더보드 기준 1위, 실제 대회 리더보드 기준 21위