캡스톤 디자인 (1)

[Rec.ommend]

12조

20181195 김현석

20151322 이한용

20151353 주석천

방영조 멘토님

박상오 교수님

1. 목차

2. 프로젝트 개요

주위를 둘러보면 자신이 노래를 못 부른다고 생각하는 사람들을 많이 발견할 수 있다. 하지만 이들 대부분은 노래를 진짜로 못하는 것이 아닌 자신에게 맞지 않는 어려운 노래를 부르다 자신감을 잃고 노래와 점점 멀어진 경우이다. 소프라노가 있으면 알토도 있고, 테너가 있으면 베이스도 있듯이 각자의 음역대 그리고 음색이 가지각색인 만큼 각자에게 맞는, 각자가 잘 부르는 노래가 따로 있을 수밖에 없다. 우리 팀은 이런 사람에게 자신에게 맞는 노래를 찾아줄 수 있다면 이들이 자신감을 되찾고 노래 부르는 것을 좋아하게 될 것으로 생각했다.

또한 평소에 노래를 즐겨 듣는 사람도 듣던 노래만 들으면 가끔 지겨울 때가 있다. 하지만 새로운 곡들을 들어보려 해도 취향이 맞지 않아 결국 다시 듣던 노래로 회귀하게 된다. 이런 사람을 위해 사용자가 좋아하는 노래와 톤이나 음색이 유사한 노래를 추천하여 사용자에게 모르는 노래를 발굴하는 재미를 주고 싶었다.

우리 팀은 이런 두 기능을 합쳐서 음성 분석을 기반으로 한, 노래를 추천해주고 검색해주는 서비스를 제공하고자 한다. 사용자에게 자신이나 가수의 육성으로 새로운 노래, 좋은 노래들을 추천받고, 음악을 찾아서 들으면서 소리를 일상생활에 더 가깝게 느낄 수 있도록 구현하고자 한다.

그리고 이런 자신의 목소리를 눈으로 볼 수 있다면 사용자에게 더 좋은 경험을 줄 수 있다고 생각했다. Simple is Best 라는 말처럼 간단한 단색 컬러로 보이지 않는 목소리를 보여주는 것은 사용자 대부분에게 새로울 것이다. 이것을 공유하여 SNS 를 통해 다른 사용자들과 비교하고, 자랑하고, 소통할 수 있도록 하면서 사용자에게 즐거운 경험을 주는 것이 우리 프로젝트의 목표이다.

3. 개발 내용

A. 개요

이 프로젝트를 통해 사용자의 목소리 또는 노래를 입력받아 음색을 추출하여, 그것과 가장 유사한 노래들을 추천해주는 어플리케이션을 개발한다. 상기한 목적을 달성하기 위해 먼저 대중가요의 목록을 수집하여 음색을 추출 후 DB 에 저장한다. 사용자가 앱을 통해 음성 파일을 서버에 업로드 하면 유사도를 계산하여 유사도 상위 10 개의 노래 목록을 사용자에게 전송하여 프론트엔드에서 보여준다. 사용자는 추천받을 노래의 장르와 발매 연도 범위를 선택하여 선호하는 노래 분류 안에서 추천받을 수 있다.

크게 네 부분으로 나누어 협업하였는데, 어플리케이션을 디자인하는 앱 부분,

목소리에 맞는 음색을 추천해 주는 Music Search 부분, 음악과 톤이나 음색이 유사한 노래를 추천하는 Tune Search 부분, 데이터를 연산하여 프론트엔드로 출력해주는 서버 부분으로 나누었다.

B. 음악 DB 수집

i. 노래 리스트 및 메타데이터 수집

노래 리스트는 멜론 홈페이지를 크롤링했다.

https://www.melon.com/chart/search/index.htm 의 월간 차트 및 연간 차트를 크롤링했으며, 각 노래의 제목, 가수, 발매일, 장르 자료를 수집했다. 크롤링은 python 라이브러리인 request 와 BeautifulSoup 를 사용했고, 2000 년부터 2019 년까지의 연간차트 상위 100 곡과 2020 년의 1 월부터 9 월까지의 월간 차트 상위 100 곡을 수집한 결과 총 2093 곡의 노래 리스트가 수집되었다.

수집한 데이터는 python 의 pandas 라이브러리를 사용하여 CSV 파일 형태로 저장했다. 또한 크롤링 환경으로는 Google Colab 를 Google Drive 와 연동하여 사용했다.

ii. 음원 데이터 수집

음원 파일은 구하는 방법이 거의 전무 했으므로, 유튜브를 크롤링하는 방법을 사용했다. 유튜브에 수집한 노래의 "제목 + 가수"를 검색하여 검색 결과 가장 상위에 나오는 동영상의 id 를 수집했다. 이 과정에서는 BeautifulSoup 와 selenium 라이브러리를 사용했다. 동영상의 id 를 수집한 후에는 python 의 pafy 라이브러리를 사용하여 음원을 다운로드했다. 다운로드 된 음원은 후술할 음성 추출 및 음색 추출 후 삭제된다.

iii. 음성 추출

노래의 음성 부분과 사용자의 음성을 비교하기 위한 전처리로 음원의 음성을 추출했다. 음성 추출에는 spleeter 라이브러리를 사용했다.

iv. 음색 추출

음원에서 음성을 추출하고 나면 timbre_models 라이브러리[1]를 사용하여 음색을 추출하게 된다. 추출되는 음색은 'hardness', 'depth', 'brightness',

[1] https://www.audiocommons.org/assets/files/AC-WP5-SURREY-D5.8%20Release%20of%20timbral%20characterisation%20tools%20for%20semantically%20annotating%20non-musical%20content.pdf

'roughness', 'warmth', 'sharpness', 'boominess', 'reverb'의 8 개 속성으로 구성된다. 각각의 음색 속성은 0 또는 1의 값을 가지는 reverb를 제외하면 모두 0 – 100 의 값을 가진다. reverb 속성은 바이너리 값이기 때문에 유사도를 계산하는 데에는 제외했다.

수집된 데이터 베이스의 테이블은 아래와 같은 형태이다.

id	title	singer	hardness	depth	brightness	roughness	warmth	sharpness	boominess	reverb	release	genre
ZEG4RDZyjzE	아시나요	조성모	51.7591152495583	36.7560169979334	66.1978357817665	43.7885578383702	40.943217261794300	51.767675356019500	19.659248006502	1.0	2000.09.01	발라드
Ph5bnQSgQDQ	다 줄거야 (Acoustic Ver.)	조규만	53.1777171280494	41.1677360323686	63.6732176799959	44.8950786019555	42.7425865569673	50.922668870015	25.0726629355085	0.0	2000.01.01	발라드
cENP3aul9IQ	Run To You	DJ DOC	54.6988058911271	28.3460095766620000	64.4428664456754	51.1365409765979	39.8253734397557	46.6723113428821	11.070213546764700	0.0	2000.05.16	댄스, 랩/힙합
hn6AzhRYf1E	거짓말	god	54.2102472064567	40.3390797444506	63.4906248566205	44.1886056009126	44.517562539712	50.4528176320743	24.0025566838045	0.0	2000.11.03	발라드
POu_1kHWNC8	가시나무	조성모	48.61105276360410	48.132941161708700	55.9735140161592	39.90309320078310	46.524989485666610	40.50433239994230	27.893423258823330	0.0	2000.01.27	발라드
v2j9WCHVG8A	혼들린 우정	홍경민	56.04867920263610	36.6572009836171000	65.8796307581128 0	55.8947468727099 0	40.99522588369230	49.08593657275380	17.800429646394800	1.0	2000.06.18	댄스
uVyVAumtsoc	나의 연인(良辰)	임창정	53.318170397327700	32.31046555766170	66.17831525214620	43.210554757452500	37.50698568700690	54.82778170274960	15.168983681374 20	0.0	2000.02.10	발라드
CbxynFLoxro	영원	스카이	54.3930489825229	37.38586405522970	67.0729974834040 0	48.4032655829760 0	39.811162456778500	56.6186560113673 0	17.138386334166850	0.0	1999.11.16	발라드, 록/메탈
N7VBSaZ-jKQ	영	김현정	53.500359583831630	26.486804842091300	66.890634923860 30	50.067467628283 80	36.938958206547 80	51.005135118013700	9.17931829962198	0.0	2000.05.22	댄스
PedYhWdjYGk	오! 가니	컨츄리 꼬꼬	54.1146104119654 0	31.909417531913 00	65.0740104540724 0	49.095984219260100	41.614191785558 80	49.923055438672 40	15.074669389943500	0.0	2000.06.12	댄스
aNQifYGRcU0	기도	정일영	49.481844615789 80	40.336199041914700	61.728959000160000	42.123936479785400	45.838861549665 40	48.738519980486 60	23.17493266604700	1.0	2000.10	발라드, 국내드라마
QI9K6sXLeqo	천설속의 누군가처럼	신승훈	52.3534732774323	34.782312741649900	63.5467071910238	45.683663309873 80	42.751808998701 90	47.917626251840700	15.434456028896200	0.0	2000.03.07	발라드
I6sd5vAxkkg	너를 위해	임재범	53.73503225372710	34.4049861494978 0	65.8567505118899	49.725396862592 30	40.105151036887 50	53.065380112879 40	17.329299594336700	1.0	2000.05.16	발라드
QhsVimBsSZY	바보	박효신	51.269645154368 50	36.177262835372 00	65.663764368201 90	47.883162981672 50	41.540053291299 80	49.768099257404000	17.918518030229 10	1.0	1999.12	R&B/Soul
IWMfoD8cJ-c	그대가 그대를	이승환	51.461709771202800	37.065768081013 30	66.193378992250 80	48.883394859448 40	41.419483092295 10	49.035088342282 90	21.077754623936400	0.0	2003.03.12	발라드
mXTRiFz36d0	매직 카펫 라이드	자우림	56.46052914729170	26.083138270511300	70.9306811532114 0	55.017150954643 00	34.607203326925 90	56.421872974028 20	6.5693029541014900	0.0	2000.07.04	록/메탈
dNRwPslBPLI	중독된 사람	조장혁	52.872814629801900	36.157122879234 60	65.033520018921 80	48.317946987010 70	40.903960118942 00	49.983291190330 10	16.601875117147500	1.0	2000.06	발라드
U82c-HnVE9E	해줄 수 없는 일	박효신	52.149885885343 40	40.145114758446 40	63.536329160290 30	48.793791245497 30	43.419657547161 00	48.449292816853 60	18.764819129179500	0.0	1999.12	R&B/Soul
SzoNVWfUUG0	고백	박혜경	52.78090619516090	30.86323785561430	67.55475949192060	46.194974778671 80	38.61065049566 76	53.918783081619600	9.788822245076520	0.0	1999.12.23	록/메탈

a. 음색 유사도 계산(Music Search)

두 음색 벡터의 유사도는 유클리드 거리의 제곱으로 정의했다.

$$similarity(\mathrm{p},\ \mathrm{q}) = \sum_{i=1}^{n}(p_i - q_i)^2$$
$$= (p.hardness\ - q.hardness)^2 + (p.depth\ - q.depth)^2$$
$$+\ \cdots\ + (p.reverb\ - q.reverb)^2$$

음성 파일이 서버에 수신되면 해당 파일의 음성과 음색을 추출한 뒤 위의 유사도 계산 공식에 대입한다. 이때 유사도 계산은 데이터베이스 상의 모든 노래에 대해 이루어진다.

C. Tune Search

i. 방향

음악의 부분 유사도를 비교하기 위해서는 음악의 특성, 프레임의 특성을 기준으로 비교를 해야한다고 생각하였다. 소리의 특성을 기본적으로 스펙트럼, 신호이기 때문에 일반적으로 소리, 음성의 특징이라고 표현하는 MFCC feature를 사용하여 그 feature들간 similarity를 찾아내고자 하였다. MFCC feature를 만드는 과정은 다음과 같다. 오디오 신호를 프레임별로 나누어 FFT를 적용한 후, mel filter bank를 사용하여 사람이 잘 듣는 가청 주파수대를 필터링하여 mel spectrum의 형태로 변환 시킨다. 그 후 log 변환과 IFFT를 통해 MFCC feature를 추출하게 된다.

ii. 1 차 시도(xcor+fft)

1 차적으로는 xcor 과 fft 를 사용하여 유사도를 확인하는 방식을 고민하였다. 두 시계열 XtXt 와 YtYt 가 존재할 때 교차 상관은 다음과 같이 표현된다.

$$\rho XY(k) = Corr(Xt, Yt + k) = \gamma XY(k)\sigma X\sigma Y = \frac{\gamma XY(k)}{\sqrt{\gamma X(0)\gamma Y(0)}}$$

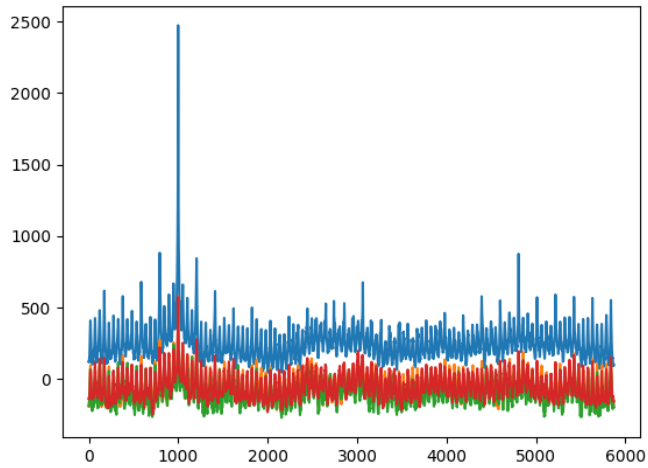
ρXY(k)는 해당 부분에서부터의 두 시계열같의 유사도를 의미한다. 이 방식을 사용한다면 특정 부분에서부터 검색을 하고자하는 노래의 일부가 얼만큼의 유사도를 보이는지 알 수 있을 것이라 생각하였다.

a. ElasticSearch + sequence similarity

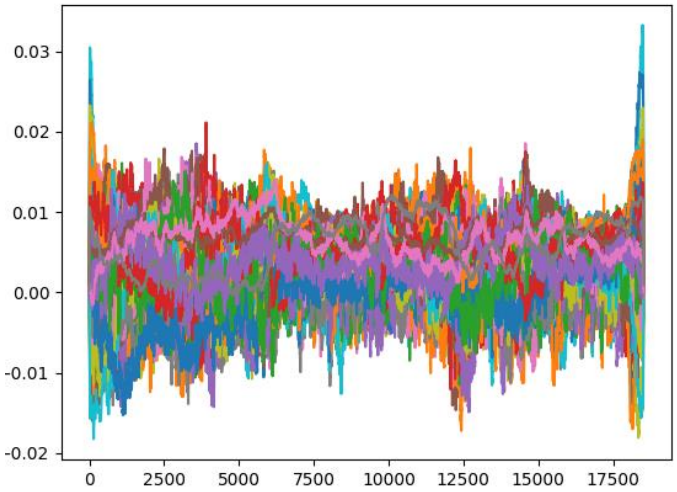
   이를 구현하기 위한 방법으로 ElasticSearch 와 sequence similarity 알고리즘을 직접 구현하여 연산하는 방식을 고민하였다. 하지만 데이터베이스에 있는 모든 곡을 처음부터 끝까지 전부 비교해야했기 때문에 분산 시스템을 사용하더라도 굉장히 time complexity 가 높아지기 때문에 좋은 방식이 아니라고 생각하여 채택하지 않았다.

b. Frame similarity

   xcor + fft 를 사용하는 방식으로 직접 알고리즘을 만들지 않고 신호처리 식을 사용하여 구현하는 방식이었다.



   다음과 같이 만약 같은 곡을 대상으로 부분을 크롭할 경우 다음처럼 일치, 유사부분이 값이 높게 나오는 것을 볼 수 있었다.



하지만 만약 다른 곡을 대상으로 할 경우 다음과 같이 알 수 없는 그래프, 즉 유사도를 정량적으로 측정하기 어려운 특징과 값들이 나오는 것을 볼 수 있었다. Smoothing, convolution 을 통해 나온 결과가 위의 그래프였기에 xcor + fft 의 방식이 적합하지 않다는 결론을 낼 수 있었다.

iii. 2 차 시도(멜로디 유사도)

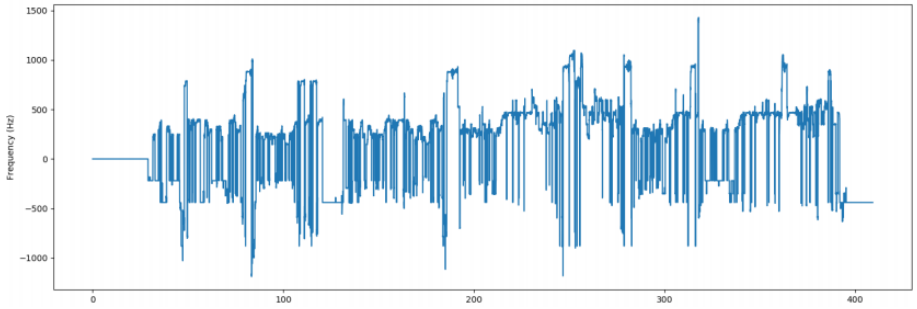
a. 방향

음악간 비슷하다고 느끼는 부분은, 음높이, 음높이가 아닌 패턴, 구조, 기호학적 유사도에 의해 생긴다. 그렇기 때문에 1 차 시도에서 실패한 바탕이 신호 처리적으로만 접근했기 때문이라고 생각했고 새로운 시도에는 멜로디 유사도를 사용하는 방식이 적합하다고 생각하였다.

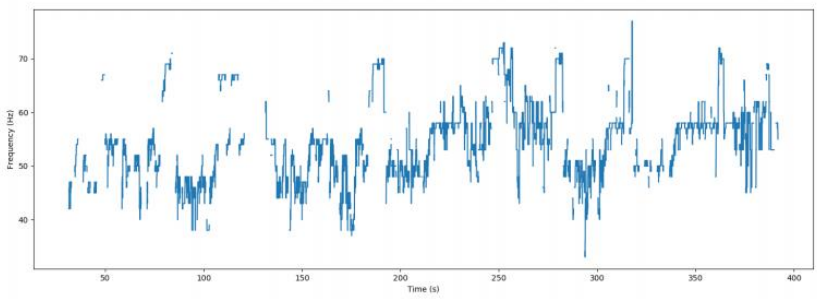
b. 멜로디 유사도 + levenshtein distance

멜로디를 추출 한 후 pitch interval 과 interval ratio 를 사용하여 두 sequence 간 levenshtein 거리를 다이나믹하게 연산하는 알고리즘을 사용하는 방식을 고민하였다.

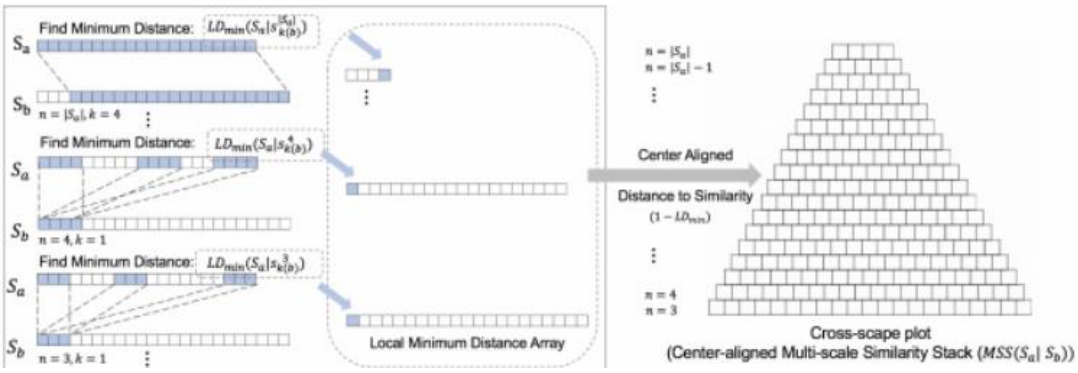
1 차적으로 멜로디를 추출하기 위해 melodia extention 을 사용하여 다음과 같이 멜로디를 추출하였다.

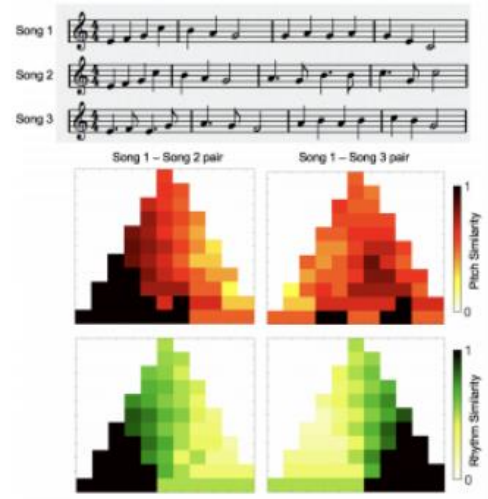


그 후 이 값들에서 필요없는 0 이하 값을 제거하였다.



그 후 h = 12log2(P/C0)의 수식으로 주파수에서 미디를 추출해 내었다.
두 미디간 유사도는 최적의 합이 나오는 순서를 matrix 에서 찾아 그 때의 distance 를 표현한 트리의 형태로 구성하여 연산하였다.[2]



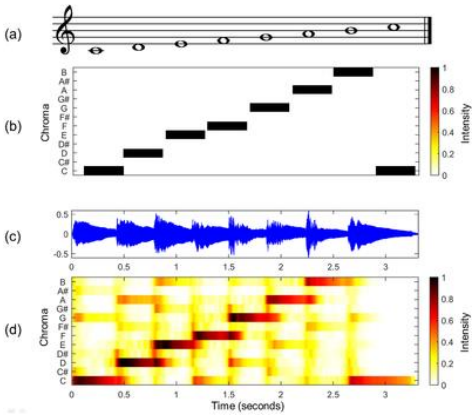


이 논문(알고리즘)은 다음과 같은 결과물을 보여주는 알고리즘을 구현하였으며 곡간의 유사도를 수치적으로 표현할 수 있는 방법이기에 유효하다고 생각하였다.
하지만 멜로디의 추출 자체가 부정확한 문제가 있어 멜로디 추출을 새롭게 제작을 시도하였다.

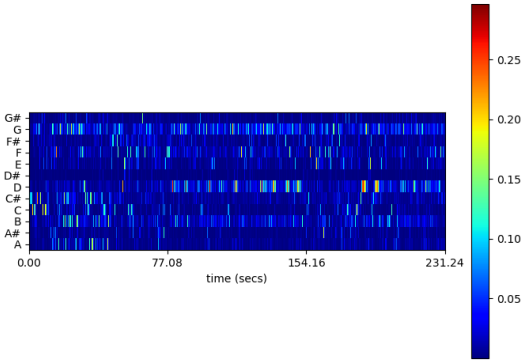
[2] http://archives.ismir.net/ismir2019/paper/000050.pdf

c. 멜로디 알고리즘 제작

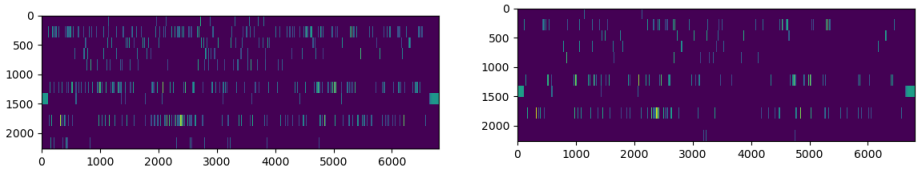
Melodia 는 사전에 제작된 모델을 바탕으로 멜로디를 추출하지만 그 정확도가 낮다는 문제가 있었다.



그래서 다음과 같은 방식으로 크로마그램을 사용하여 곡의 노트를 추출하고자 하였다.



1 차적으로 곡에서 크로마그램을 추출한 결과다, 다양한 악기와 소리가 들어가 있기에 잡음과, 올바른 멜로디가 추출되지 않은 형태를 보여주어 잡음을 처리하였다.



소리가 작은 잡음과 같은 노트를 제거하고 연속되는 노트를 병합하고 잡음을 제거한 결과물이다. 하지만 이 역시 직접 소리를 들었을 경우 노트가 불분명하고 사람의 기준에서 멜로디가 적절하게 들리지 않는 문제가 있었다.

다양한 방식으로 음악간 유사도 기능을 구현하고자 하였지만, 기초 지식, 경험과 짧은 시간으로 만들 수 있는 기능이 아니었다는 것을 알았고 기능 개발에 실패하였다. 프로젝트 후반부, 마지막 주차 발표시에 구글에서 humming

search 라는 기능을 런칭한 것을 확인 할 수 있었고, 해당 기능을 제작하고 개발하는데 상당한 시간이 소요되었다는 것을 볼 수 있었다. 만약 개발블로그, 과정을 프로젝트 시작할 때 미리 보았더라면 조금 더 긍정적인 결과물을 만들 수 있었지 않았을까 하는 아쉬움이 남는다.

D. 서버 개발

서버는 Google Cloud Platform 의 cloud run 상에서 사전에 빌드한 docker 이미지를 바탕으로 서비스를 구동하였다.

다음과 같이 버전별로 이미지를 관리하여 팀원들이 서버 통신을 개발할 때, 앱의 문제인지 서버의 문제인지 버전별로 테스팅을 진행해 쉽게 확인할 수 있는 환경으로 제작하였다.

서버는 flask 의 형태로 개발을 하였으며 사전에 개발해 둔 알고리즘과 csv 를 정제하여 storage 에 저장해 사용하였다. 컨테이너 이미지는 15 분간 deploy 됐다가 instance 가 삭제되는 과정을 거치고 파일 시스템이 명확하지 않은 문제가 있었다. 그렇기 때문에 유저가 전송한 음악, 음석의 경우 temporary file 의 형태로 저장하여 서버에서 처리하는 구조를 두었다.

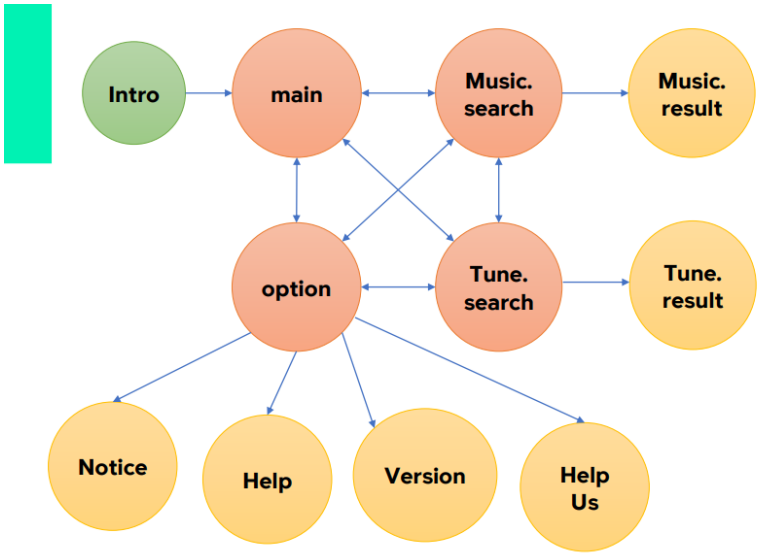
```
@app.route('/voice', methods=['POST'])
def voice():
    reqfile = request.files['voice']
    genre = request.form.getlist('genre')
    start = request.form.get('start','0')
    end = request.form.get('end','9')
    print(request.files['voice'])
    filename, file_extension = os.path.splitext(reqfile.filename)

    temp = tempfile.NamedTemporaryFile(suffix=file_extension, delete=False)
```

E. 어플리케이션 개발

i. 구현기능

a. 초기 앱 UI 구성 및 씬 구성

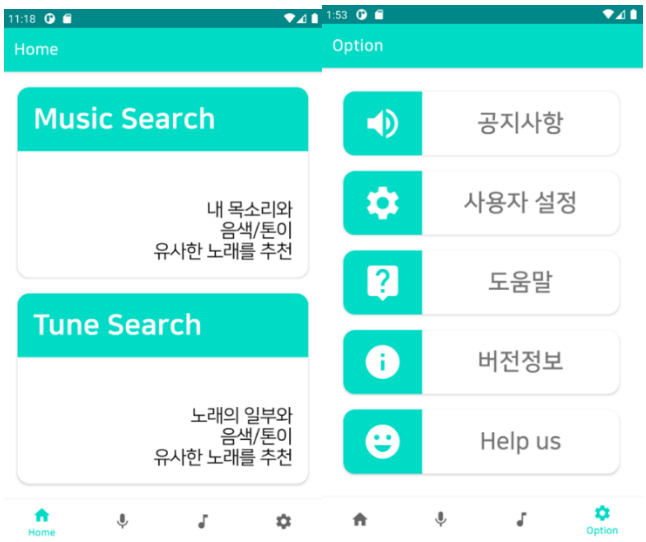


우선 초기 UI에 들어 있던 씬 구성이다. 인트로 화면을 거친 후 메인 씬에 진입하게 된다. 이때 빨간색의 네 화면을 퀵메뉴로 이동해 갈 수 있으며 노란색의 어떤 상태에서도 빨간색으로 전이할 수 있다. 각 서치에는 각 서치의 결과로 넘어가는 씬으로 이동할 수 있고, 옵션 창에서는 공지, 도움말, 버전, 지원의 씬을 구성하여 넘어갈 수 있다. 씬과 씬을 넘어갈 때는 fromAlpha 값과 toAlpha 값을 조정하는 것으로 간단히 fadein, fadeout 애니메이션을 만들 수 있었다.

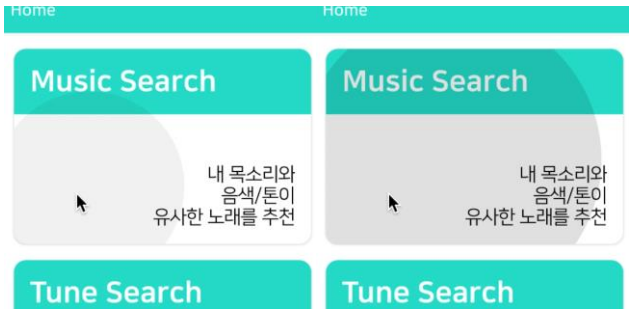
b. UI UX 개선

프로토타입 어플리케이션의 UI가 투박하다는 피드백이 많아 전체적인 UI를 수정함에 더해 객체 지향적인 구조로 어플리케이션을 재구성했다.
안드로이드 스튜디오에서 제공하는 버튼, card view, 하단 내비게이션 바 등의 UI 요소를 적극적으로 활용하여 자연스럽게 material design의 UI 및

UX가 적용되었다. 또한 UI 요소의 색깔을 brand color인 민트색으로 통일시켰다.
버튼과 card view의 코너에 일괄적으로 반지름 15 DP의 둥근 모서리를 적용하여 UI에 통일성을 갖게 했다.



UX의 통일성을 위해 버튼에 기본적으로 적용되는 물결이 퍼지는 듯한 터치 애니메이션을 card view에도 적용했다.



c. 녹음 기능

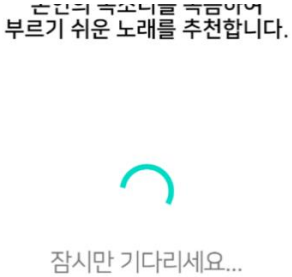
```
if (recorder == null) {
    recorder = new MediaRecorder();
}

recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
recorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
recorder.setOutputFile(RECORDED_FILE);
try {
    recorder.prepare();
    recorder.start();
} catch (Exception ex) {
    Log.e( tag: "SampleAudioRecorder",  msg: "Exception : ", ex);
}

final TextView time1 = (TextView) findViewById(R.id.textView4);
final Timer timer=new Timer();
final TimerTask ti=new TimerTask(){
    int second,minute,seconds=0;
    @Override
    public void run(){
        seconds+=1;
        minute=seconds/60;
        second=seconds%60;
        time1.setText(String.format("%02d : %02d",minute,second));
    }
};
timer.schedule(ti, delay: 0, period: 1000);
```

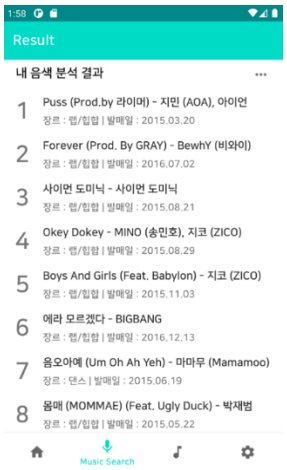
레코더가 없다면 추가하여 MPEG_4 속성으로 녹음을 진행하고

recorded.MP4 형식으로 파일을 녹음한다. 그리고 타이머도 같이 돌아가는데, Timer 를 통해 계속해서 카운트를 늘려가고 이것을 분과 초를 나누어서 녹음되는 시간만큼 초를 증가시켰다. 여기서 stop 버튼을 누르거나 퀵메뉴 4 개 중 어느 것이든 눌렀을 경우 녹음은 종료되고, 해당 경로에 파일이 저장되게 코딩하였다.
또한 녹음 버튼 뒷배경에 녹음되는 소리의 크기에 따라 진폭이 바뀌는 파동 애니메이션을 추가하였다.

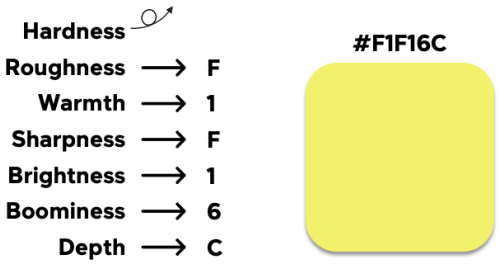


정지 버튼을 누르면 녹음이 끝나고 녹음 파일을 서버에 전송한다. 서버의 응답이 있을 때까지 로딩 바 애니메이션이 돌아간다.

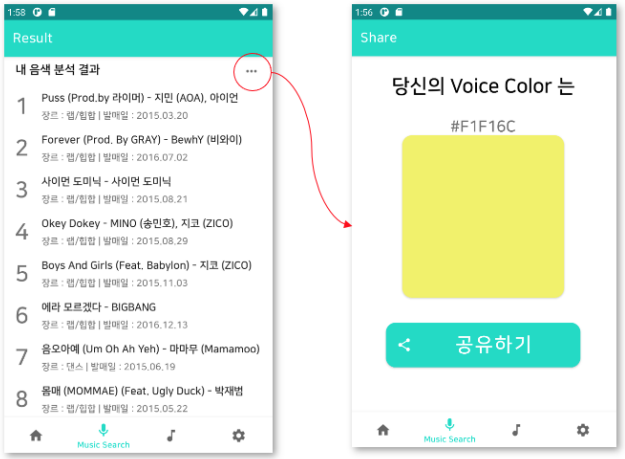
d. 음색 분석 결과 화면
음색 분석 결과는 서버로부터 JSON 형태로 수신된다. JSON 을 적절하게 파싱하여 사용자에게 보여준다. Recycler View 를 이용하여 리스트를 이루는 element 의 레이아웃을 한 번만 정의하고 여러 번 재활용 될 수 있게 했다. 10 개의 곡이 결과로 보이게 되며 화면을 넘어간 노래는 사용자가 스크롤 하여 볼 수 있다.



e. voice color 기능
사용자의 음성 또는 노래의 음색을 추출한 값을 색상에 매핑하여 시각화했다.



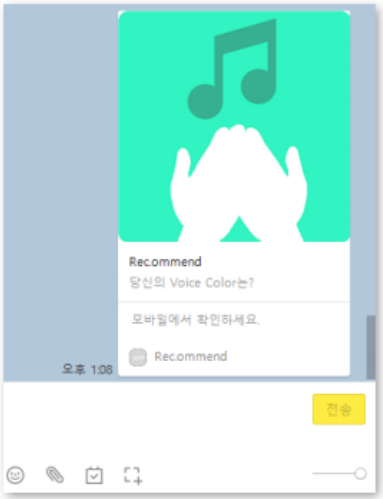
음색의 7 가지 속성 중 음성마다 차이가 적은 Hardness 를 제외하고 나머지 6 가지 속성의 값을 DB 상의 최대, 최솟값을 기준으로 [0, 1] 범위의 값으로 normalize 하였고, 이 값들을 RGB 색상의 12 진수 한 자릿수에 매핑하는 방식으로 voice color 를 생성했다. voice color 는 음색 분석 결과 화면의 우측 상단의 더 보기 버튼을 클릭하여 사용자가 확인할 수 있다.
voice color 는 음색 분석 결과 화면의 우측 상단의 더 보기 버튼을 클릭하여 사용자가 확인할 수 있다.



f. 공유하기 기능

초기에는 카카오톡, 인스타그램, 페이스북, 트위터를 구현하기 위해 각 SNS 의 개발자 페이지를 들어가게 되었다. 먼저 카카오톡의 경우 카카오 링크를 다운받고 개발자 페이지에 앱 등록을 한 후에 사용하게 되었는데, 찾아본 대부분의 코드가 V1 기준으로 작성되어 있어 현재 V1 을 지원하지 않아 방법이 막힌 상황이 되었다. V2 코드들을 찾아보아 코틀린 코드도

사용해보았지만 잘되지 않았고 결국 자바 코드를 찾아 카카오톡 링크를 구현하는 데 성공하였다.



다음으로 페이스북이었는데 다른 찾아본 코드들이 버전이 낮아 호환이 안되어 다른 방법을 고민할 수밖에 없었다. 그러던 중 휴대폰 자체적으로 있는 공유 기능을 사용할 수 있다는 것을 찾아 사용하였다.

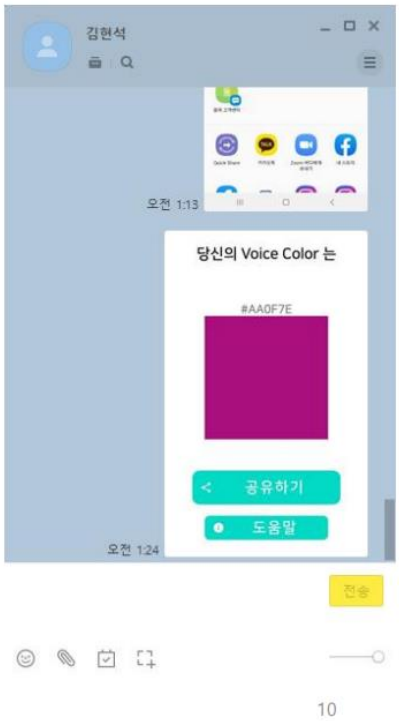
```
private void sharing(){
    LinearLayout con=(LinearLayout)getView().findViewById(R.id.conslayout);
    Bitmap bitmap = Bitmap.createBitmap(con.getWidth(), con.getHeight(), Bitmap.Config.ARGB_8888);
    Canvas canvas = new Canvas(bitmap);
    con.draw(canvas);
    Uri uri=getImageUri(getActivity(),bitmap);


    Intent shareintent=new Intent(Intent.ACTION_SEND);

    shareintent.putExtra(Intent.EXTRA_STREAM,uri);
    shareintent.setType("image/*");
    startActivity(Intent.createChooser(shareintent, title: "공유"));
}
private Uri getImageUri(Context context, Bitmap inImage) {
    ByteArrayOutputStream bytes = new ByteArrayOutputStream();
    inImage.compress(Bitmap.CompressFormat.JPEG,  quality: 100, bytes);
    String path = MediaStore.Images.Media.insertImage(context.getContentResolver(), inImage,  title: "Title",  description: null);
    return Uri.parse(path);
}
```

먼저 해당 화면을 비트맵으로 저장을 해야 하는데 view.getDrawingCache 를 사용하려고 하였으나 deprecated 라는 에러가 나타났다. 현재 버전에서는 사용할 수 없으므로 다른 방법을 사용하여야 했다. 결국 화면 전체를 캔버스에 다시 그린 뒤에 비트맵으로 저장하고 이것을

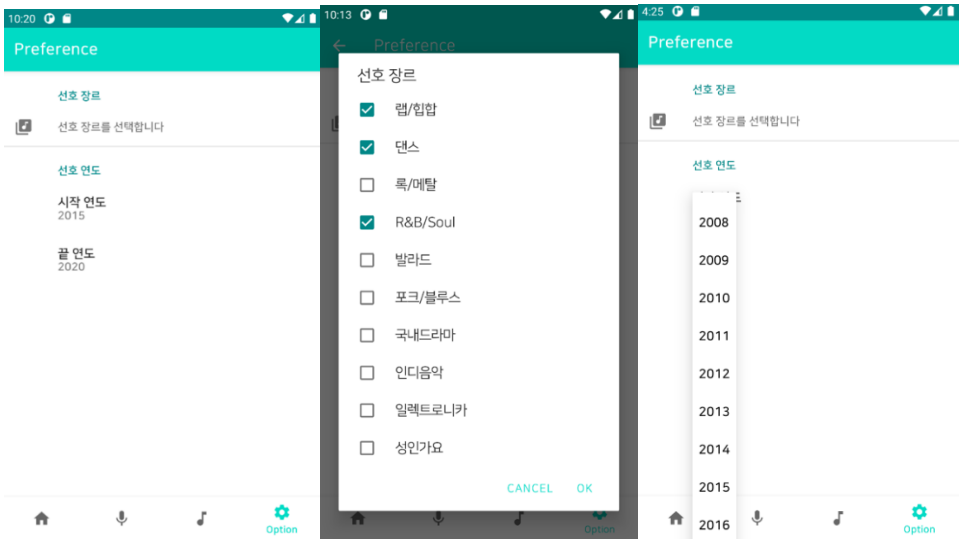
Uri 로 변환하여 사용하여 무사히 구현에 성공하였다.



10

g. 사용자 설정 기능

사용자가 원하는 장르 및 발매 시기의 곡을 추천받을 수 있도록 사용자 설정 메뉴를 구현했다. 사용자는 여러 개의 장르와 발매 시기 범위의 시작 연도, 끝 연도를 설정할 수 있으며, 설정된 내용은 앱에 저장되고 서버에는 녹음 파일을 보낼 때 같이 전송된다.



4. 멘토링

A. 주제 선정 과정

nlp 를 이용한 rpg 게임 npc 대사 자동 생성 프레임워크	nlp 를 이용한 자격시험 문제 자동생성	매장 스탬프쿠폰 관리 서비스	유투버/크리에이터 추천 서비스	음성 기반 노래 추천 분석 서비스
코로나 자동 명부 작성 및 관리 앱 만들기	청각장애인을 위한 유튜브 효과음 자동 자막(의성어), 자동 수화 번역 기능	카메라 이용해서 모션인식으로 일어났는지 안일어났는지 알람	시각장애인을 위한 핸드폰 내비게이션	사진 보정 프로그램
웹사이트 쇼핑 보조	요리 보조	노래를 부르면 본인 음에 맞게 코드 조정	2d 사진을 3d 모델로 변환	현실 코디 추천

주제 선정 과정은 위와 같았다. NLP 의 경우 게임 내의 텍스트가 일관적이라는 문제도 있었고, 다른 주제의 경우 예전 캡스톤 디자인에서 다른 팀이 유사한 주제로 시도하였거나 현재 레벨에서 구현이 힘든 경우도 많았다. 브레인 스토밍을 통해 여러 개의 아이디어를 내보고 여러 기준에 따라 거르는 과정을 반복하였다. 기준들로는 구현할 수 있는가/유사한 주제가 존재하는가/제품을 원하는 고객이 있는가? 의 순서로 잡았는데 확실히 음성 기반으로 한 노래 추천, 분석 서비스가 기준에 들어맞으면서도 재미있고, 눈으로 보이는 결과물이 나와서 선정하게 되었다. 주제를 정하는 과정은 힘들었지만 개발할 때 무엇이 중요한지 깨닫는 계기가 되었다.

B. 프로젝트 방향 설정

성공적인 발표와 프로젝트 진행을 위해 다음과 같은 일이 먼저 필요하단 조언을 받았다.

1. 가요와 매칭하는 파라미터의 수를 늘리면 정확도가 높아질 것이다.

2. 개발 목표를 좀 더 구체화할 필요가 있다.

주제를 일부 수정한 후 추가된 기획으로 프로젝트를 진행하였다.

5. 결론

A. 완성도 평가

작품의 주기능은 잘 구현되었다. 음성 데이터가 녹음되면 서버에서 연산한 후 리스트를 10 개까지 보여주었고, 더 보기 버튼을 누르면 해당 음성 데이터의 Voice Color 와 해당 색의 RGB 코드를 보여준다. 그리고 공유하기 버튼을 누르면 공유 가능한 모든 앱이 나와서 해당 화면을 공유할 수 있고, 도움말을 통해 자신의 음성 데이터의 특징과 왜 그런 색이 나왔는지 또한 알 수 있다. 하지만 다음과 같은 문제점들이 있다.

a. UI 가 민트색 고정이라는 점

b. 긴 처리시간

B. 보완 사항

i. UI

UI 를 본인의 테마에 따라 색을 다르게 하거나 다크 모드를 추가하는 등의 사용자 편의성을 고려하여 구현되어야 할 것이다.

ii. 음악 처리 시간

30 초 이상 걸리는 처리 시간은 긴 편이다. 백그라운드에서 돌아가기 때문에 다른 활동을 할 수 있다고는 해도 근본적인 해결책은 되지 않는다. 따라서 더 나은 알고리즘을 찾아 프로세싱을 개선, 처리 시간을 줄이는 일이 필요할 것이다.

C. 소감

김현석: 안드로이드 스튜디오가 처음이어서 관련 강의를 들으면서 오류를 찾아보고 하나하나 UI 를 만들면서 고생을 하였지만 시행착오를 겪으면서 MVP 의 기능 자체는 구현된 것 같아서 만족합니다. 다만 UI 디자인의 경우 능력도 부족하고 시간도 없어서 더 잘 만들지 못한 것이 아쉽습니다.

이한용: Python 과 Java 를 모두 사용해야 하는 깊이 있는 프로젝트였다. 하지만 Python으로 개발할 때에는 많은 오픈소스 라이브러리의 도움을 받았고, Java로 앱을 개발할 때에는 안드로이드 스튜디오의 자동완성 기능이나 코드 템플릿의 도움을 받아 생각보다 개발을 쉽게 할 수 있었다. 조금 막히는 부분이 있으면 구글링으로 문제를

해결할 수 있었다. 다른 프로젝트를 진행할 때에도 겁내지 않고 개발을 진행할 수 있을 거 같다.

아쉬운 부분도 많았다. 좀 더 다양한 기능을 앱에 추가하지 못한 것 같다. 그리고 우리가 사용한 음색으로 유사도를 계산하는 방법이 두 음성 파일의 유사도를 판별하는 엄밀한 방법인지 약간의 의문이 들어 아쉬움이 남는다.
주석천: 마지막 프로젝트를 진행하면서 아쉬운 부분이 많은 것 같습니다. 한학기라는 시간동안 나름 도전이었던 주제로 프로젝트를 진행하며, 처음으로 기능을 개발하지 못하고 다른 기능으로 대체하는 과정을 거쳤습니다. 개인적으로도 속상한 부분이 있었고 프로젝트의 목표와 조금 달라진 점이 있어 프로젝트의 scale 에 대한 고민을 초기에 조금 더 했어야 했나라는 생각이 드는 것 같습니다.

D. 개발 일정

및 데모	중간 데모 준비														
	중간 데모 발표 준비														
	추가 기능 구현														
	테스트 및 디버깅														
	최종 데모 준비														
	최종 데모 발표 준비														
	최종 보고서 준비														

5. 부록

Github repository: https://github.com/Rec-ommend/app_ui_improving