Project Nowsic
완료 보고서

INDEX

4. 문제점 및 해결 내용

4-1 사진 분석

4-2 음악 분석

4-3 추천

4-4 웹페이지

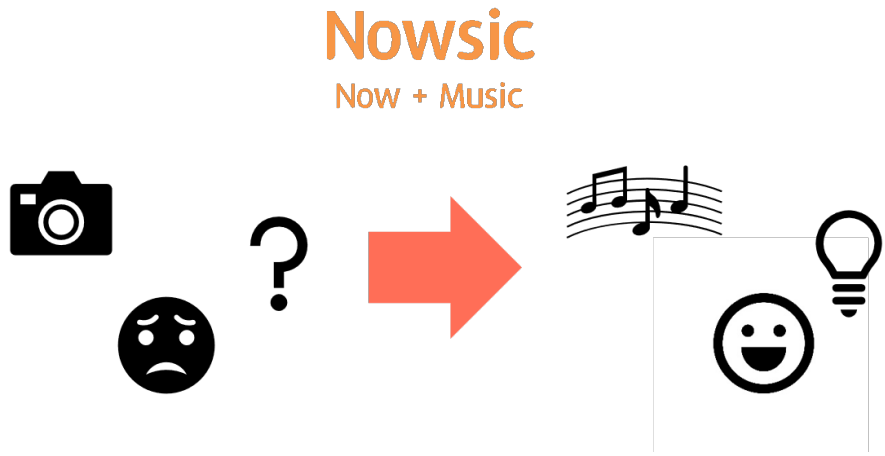
5. 계획대비 자체 평가

5-1 성공

5-2 실패

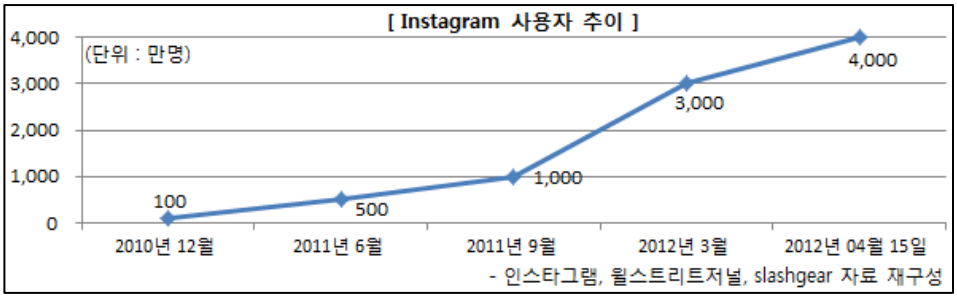
1. Nowsic Team 소개

2. Nowsic 목표 및 기대효과



Nowsic은 지금을 뜻하는 Now와 음악을 뜻하는 Music을 합쳐 만든 단어로, '지금에 어울리는 음악을 추천한다.'라는 의미를 담아 작명했다.

Nowsic의 최우선 목표는 사진으로부터 음악을 추천하는 서비스를 구현하는 것이다. 이를 바탕으로 음악으로부터 사진을 추천하는 기능을 추가하며, 최종적으로는 사진에서 음악으로, 음악에서 사진으로 넘나드는 재미를 제공하는 서비스를 목표로 한다.



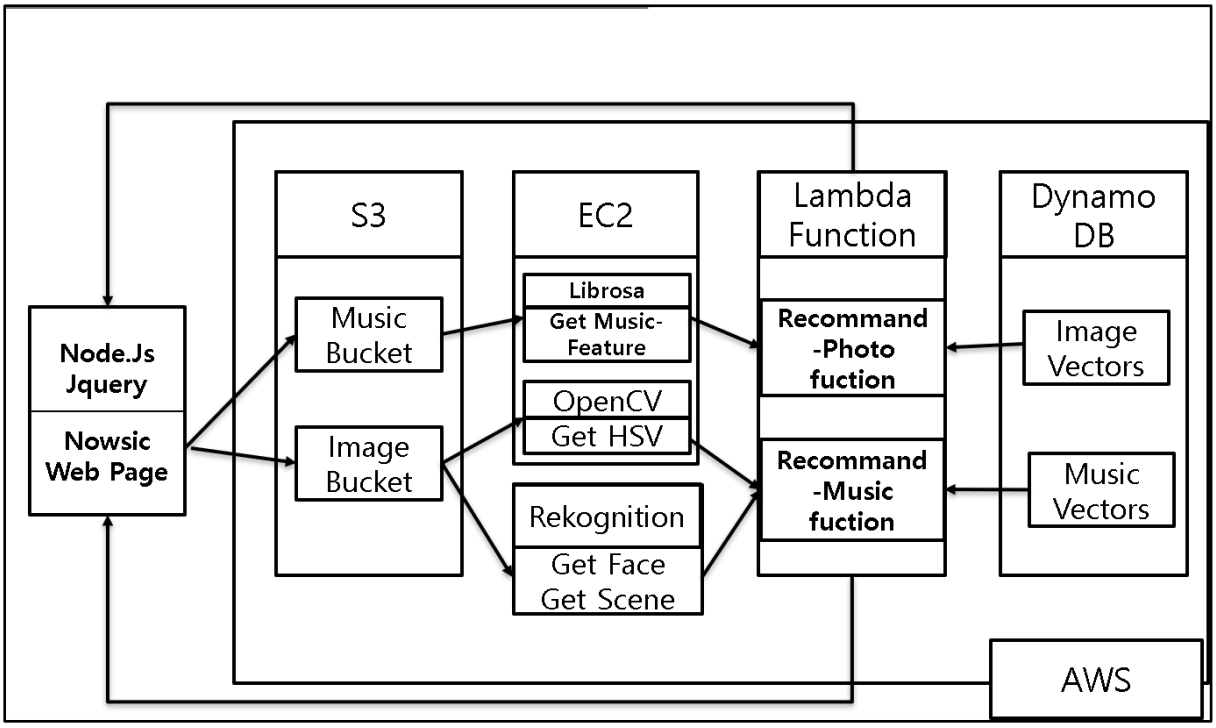
이러한 서비스는 사진에 관심이 많은 사용자들에게 어필할 수 있을 것이며, 사진과 강력한 연관성을 가지는 SNS인 인스타그램이 강력한 성장세를 보이고 있다는 점에서 충분한 수요를 가질 것이라 예상된다.

또, 온라인으로 제공되는 서비스들 중 전 세계적으로 인기를 끌고 있는 것으로 유튜브와 페이스북의 예를 들 수 있는데, 이들의 성공요인 중 하나는 높은 연관성을 보이는 컨텐츠를 연속해서 제공하는 것이다. Nowsic 또한 사진과 음악에 대한 분석을 기반으로 높은 연관성을 보이는 사진

과 음악을 연속해서 추천할 수 있으므로, 즉각적으로 연관성이 있는 컨텐츠를 즐기는 데에 익숙해진 사용자의 흥미를 끌 수 있을 것이다.

3. 상세 개발 내역

3-1 Frame Work

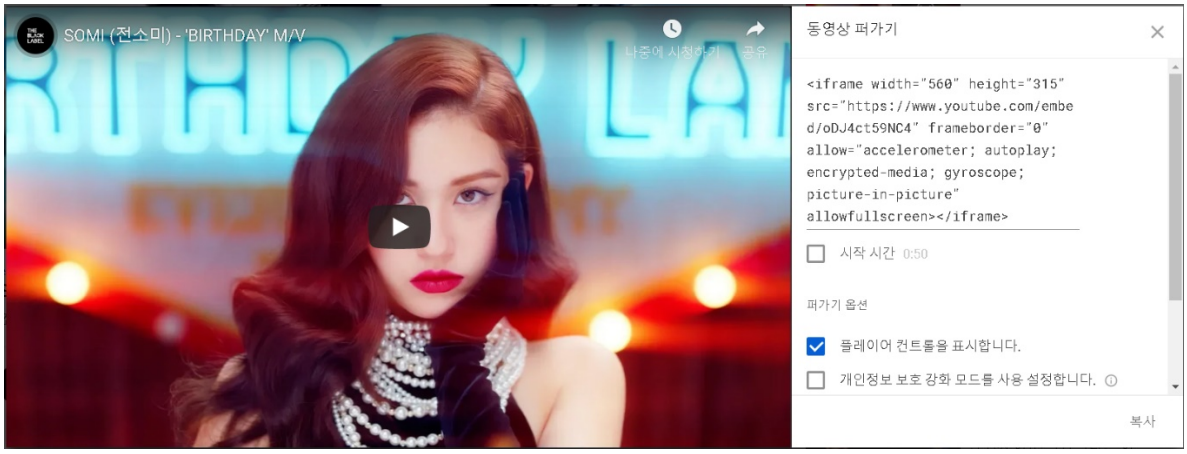


간략하게 표현된 Nowsic 실행 과정

Nowsic의 기능은 주로 AWS를 통해 구현되었다. 데이터를 저장하기 위해서 S3와 DynamoDB를, 사진에 포함된 감정 및 객체 정보를 파악하기 위해 Rekognition을, 분석 및 추천 작업을 자동적으로 이루어지도록 하기 위해 SNS와 SQS, Lambda를 사용하였고, Lambda에서 실행하기 어려운 작업을 수행하기 위해 EC2를 사용하였다.

AWS 이외에도 음악을 분석하기 위해서 Librosa를, 사진의 색채정보를 얻기 위해서는 OpenCV를 사용하였고, 사용자와 서비스를 연결하는 웹페이지 구현에는 Node.js와 JQuery Ajax를 사용하였다.

3-2 음악 분석



화면의 오른쪽 코드를 복사하면, 왼쪽과 같이 유튜브 음악을 들을 수 있는 링크가 제공됩니다.

대부분의 음악에는 저작권이 있기 때문에, 이번 프로젝트에서는 사용자에게 추천하는 음악을 웹페이지에서 youtube 소스코드를 통해 제공하고, DB에 저장할 음악의 Key값으로는 각 youtube 페이지의 11자리 고유 주소를 사용했다.

3-2-1 특징값

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		0	1	2	3	4	5	6	7	8	9	10	11	12	13
2	0	#BlackFrid	YC8cvSoH	-87.1754	53.83419	10.85873	23.97271	6.853712	11.72947	8.216396	12.3309	-3.64402	5.802326	2.114673	5.511439
3	1	#SELFIE P/	0zVK4BpW	-78.3111	70.6615	20.1504	26.25571	16.3365	23.16427	5.24333	11.7316	-0.47327	7.435034	5.145523	11.62483
4	2	2CELLOS -	DcFpvolR	-169.214	151.468	-23.4338	26.45985	7.96073	-3.87414	-1.18921	-1.99122	-9.14637	-1.55139	-11.7643	-9.46663
5	3	2CELLOS -	Xj3gU3jAC	-137.004	140.1576	-5.71738	18.00534	6.75519	0.876657	7.239064	10.16959	-6.14667	1.774349	3.214742	1.979143
6	4	2CELLOS -	uT3SBzmE	-262.014	122.1173	-28.919	19.76325	-8.32614	-2.18029	-12.8863	-0.34435	-14.4653	-16.0587	-9.21176	-0.80321

각 음악에 대한 특징 값을 저장.

일반적으로 음악의 느낌을 나타내는 요소로는 음악의 장르, 조(key), 소리 크기, 속도, 박자, 멜로디 등이 있다. 하지만 이 중 장르와 조는 음원 파일을 통해 알아내기 어렵고, 예외가 다수 존재하기 때문에 일괄적으로 적용하기 어렵다. 따라서, 이를 대신하여 Librosa를 이용해 음원파일을 구성하는 프레임의 수학적인 특징값을 추출하기로 했다.

효율적인 처리를 위해 음악의 30초-60초 구간을 추출할 구간으로 선택했으며, 음악 분석에 사용한 특징값은 MFCCs, BPM, RMSE, Chromagram 총 네 종류를 사용하였다.

각 특징값의 영향력을 고려하여 MFCCs에서 30개(Mel-fiter bank에 의해 30개로 나뉜 구간이 30초동안 나타내는 MFCCs 평균) , BPM에서 30개(30초 길이 구간 중 각 1초 동안의 BPM 평균), RMSE에서 30개(30초 길이 구간에서 각 1초 동안의 RMSE 평균), Chromagram에서 12개(30초 동안 나타난 각 12음계 비중의 평균) 총 102개의 값을 추출해 사용하였다.

① MFCCs(Mel-Frequency Cepstrum Coefficients)

음성인식을 포함해 오디오 분석에 널리 사용되는 특징 값으로, 인간의 청각적 특성에 맞춰 음파의 고유한 특징을 포착하기 위해 사용되는 특징 값이다. 디지털 신호로 저장된 음파를 FFT(고속 푸리에 변환)를 통해 프레임 단위의 주파수 형태로 변환한 다음, 각 프레임의 주파수 대역을 달팽이관에서의 주파수 특성을 고려해 설계된 Mel-filter bank로 나누고, 각 bank에서의 에너지를 구한 다음 각각의 에너지에 로그를 취한 뒤 DCT(이산 코사인 변환)를 적용해 구해진다. 분석에서는 음악을 30개 구간의 Mel-filter bank로 나누고 각각의 bank에서 얻어지는 MFCC가 30초동안 보이는 평균을 사용했다.

② BPM

BPM은 Beat Per Minute의 약자로, 분당 몇 박자가 울리는 속도인지를 나타낸다. 분석에서는 총 30초 구간 중 각 1초 동안의 평균 BPM을 이용했다.

③ RMSE

Root Mean Square Energy. 각 프레임에서 나타나는 에너지의 평균 제곱근 오차로, 음악의 객관적인 시끄러움을 판단하는 요소이다. 분석에서는 30초 구간 중 각 1초 구간의 평균 RMSE를 사용했다.

④ Chromagram

각 구간에서 12개의 평균율이 어느 비율로 혼합되어 있는지 추정하는 특징값이다. 분석에서는 12개의 음정이 30초동안 보인 비율의 평균을 사용했다.

3-2-2 K-Means Clustering

Cluster	Title(URL)
0	Two Steps From Hell – Victory (https://www.youtube.com/watch?v=hKRUPYrAQoE)
	여자친구 GFRIEND - 해야 (Sunrise) M/V (https://www.youtube.com/watch?v=TbPHPX3hSPA)
1	Taylor Swift – Delicate (https://www.youtube.com/watch?v=tCXGJQYZ9JA)
	Mungla Mungla Dj Remix 2019 | Mungda song mix | Djsstudio (https://www.youtube.com/watch?v=djQNHHXH45c)
2	Vivaldi Four Seasons (Spring) - Y. Wang, E. Ax, N. Goerner, J. Quentin (https://www.youtube.com/watch?v=oHg5hR8ojqE)
	Behzod Abduraimov - La Campanella - Liszt (https://www.youtube.com/watch?v=_ph6DkOyB8w)
3	오반 (OVAN), 빈첸 (VINXEN) - 눈송이 Snowflake [Music Video] (https://www.youtube.com/watch?v=zmiaMxbrnno)
	[MV] 10cm _ however(그러나) (https://www.youtube.com/watch?v=hNnoi32CyrA)
4 (Close 0, Close 2)	[MV] KYT(코요태) _ FACT(팩트) (https://www.youtube.com/watch?v=lx8_5Go4LKQ)
	Gustavo Dudamel - Hungarian Dance No. 5 - Brahms (https://www.youtube.com/watch?v=ynCEvFaJCZg)

각 클러스터를 대표하는 음악들. 클러스터 중앙에 가까우면서, 인접 클러스터와는 충분히 거리가 있는 음악을 선택하였다. 4번 클러스터는 특성을 나타내기 위해 각각 소속된 음악 중 0번 클러스터와 가장 가까운 음악과 2번 클러스터에 가장 가까운 음악을 선택하였다.

앞서 추출한 특징값들에서 유의미한 정보를 얻어내기 위해, 각각 102개의 값을 담고 있는 408개의 음악을 표본으로 사용했다. 특징값 사이의 영향력을 균등화하기 위해 표본 내에서 특징값끼리 Normalization 한 다음, 데이터들 간의 유클리드 거리의 분산을 최소화 하는 분류 방법인 K-Means Clustering을 사용해 음악들을 분류했다. 408개의 음악을 5개의 클러스터로 나눈 결과는 다음과 같다.

① 0 Cluster : 격렬하고 강렬한, 시끄러운 음악들이 주로 나타남

② 1 Cluster : 0 Cluster에 비해 덜 격렬하며, 파티 분위기의 음악들이 주로 나타남

③ 2 Cluster : 다수의 차분한 클래식 음악이 나타남

④ 3 Cluster : 하이라이트가 분명하고, 부드러운 선율의 음악들이 나타남

⑤ 4 Cluster : 이 클러스터에 속하는 음악들이 공통적인 경향을 보이지는 않지만, 각 음악은 다른 클러스터와 가까운 만큼 가까운 클러스터의 특징을 보인다.

fromC \ toC	C0	C1	C2	C3	C4
C0	100	74.6	56.4	61.9	71.6
C1	71.7	100	36.8	40.7	48.5
C2	44.1	27.3	100	74.6	64.8
C3	53.4	34.9	64.9	100	62.1
C4	70.5	52.0	71.4	67.6	100

위의 표는 각 클러스터의 중심으로부터 다른 클러스터의 원소들과의 거리의 평균을 이용하여 클러스터 간의 수치적 유사도를 표현한 것이다. (각 클러스터 내부의 분포형태가 다르므로, 정확히 대칭되는 형태는 이루지 않았다.)

수치 분석 결과, 0번 클러스터와 1번 클러스터가 인접하며, 2번 클러스터와 3번 클러스터가 인접한다. 4번 클러스터는 1번 클러스터와 살짝 약하게, 나머지 클러스터와는 비슷한 거리를 유지한다.

앞서 파악한 각 Cluster에 소속된 음악들의 경향과 수치적 분석을 통해, 각 Cluster의 성질을 아래와 같이 정의하였다.

0 Cluster : 2 Cluster와 가장 멀고 1 Cluster와 가까운 'Loud' Cluster, 격렬한 음악.

1 Cluster : 3 Cluster와 가장 멀고 0 Cluster와 가까운 'Happy' Cluster, 행복한 음악.

2 Cluster : 0 Cluster와 가장 멀고 3 Cluster와 가까운 'Calm' Cluster, 차분한 음악.

3 Cluster : 1 Cluster와 가장 멀고 2 Cluster와 가까운 'Sad' Cluster, 슬픈 음악.

4 Cluster : 위의 Cluster에 소속되지 못한 음악들이 모인 'Else' Cluster. 애매모호한 음악.

3-2-3 Music Vector

	0	1	2	3	4	5	6	7
0	#BlackFrid	YC8cvSoH	4	0.545057	0.540461	0.467866	0.403419	0.595615
1	#SELFIE P/	0zVK4BpW	1	0.582492	0.656222	0.348937	0.367867	0.478252
2	2CELLOS -	DcFpvolR	2	0.469244	0.334719	0.721744	0.688022	0.575116
3	2CELLOS -	Xj3gU3jAC	4	0.435071	0.355989	0.522677	0.517067	0.501086
4	2CELLOS -	uT3SBzmD	2	0.255183	0.125671	0.671269	0.564984	0.42056
5	3 Doors D	kPBzTxZQ	2	0.412825	0.286532	0.703478	0.631251	0.600462
6	3 Doors D	xPU8OAjjS	4	0.52902	0.392513	0.619837	0.600209	0.681994
7	4 Levels C	dOxlEwX9l	2	0.132862	0.097268	0.464579	0.315729	0.236267
8	4 Non Blo	6NXnxTNI	4	0.529466	0.423559	0.62705	0.53658	0.718679
9	5 Seconds	Jqs5EaAau	0	0.526677	0.52292	0.371313	0.388064	0.458807
10	5짧 SINFC	ukKW4SR-	2	0.380548	0.247442	0.76973	0.611442	0.581941
11	A R I Z O	SXhtzMDT	0	0.603405	0.593419	0.43132	0.45462	0.543177
12	a-ha - Tak	djV11Xbc9	4	0.574669	0.439949	0.664143	0.593847	0.76082
13	ABIR - Tar	Nd-kL7Tx	0	0.527557	0.47798	0.465881	0.478072	0.47752
14	Above & E	ztLdBiFb6	1	0.506409	0.59596	0.331549	0.398705	0.379353
15	AC_DC - T	v2AC41dg	4	0.50762	0.48462	0.449781	0.404459	0.586613
16	Adele - Sc	hLQl3WQ	3	0.313655	0.220024	0.541494	0.574148	0.382413

각 음악이 속한 Cluster와 음악 벡터

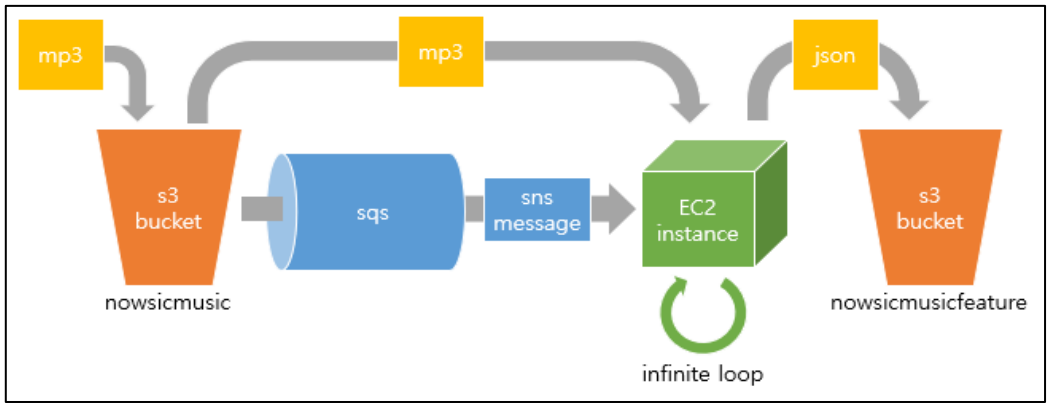
K-means Clustering을 통해 음악을 5개의 Cluster로 분류했지만, 이 정보만 이용하여 사진과 음악을 연결 짓기에는 정확성이 떨어진다.

각각의 음악의 특성을 좀 더 구체적으로 정의하기 위해, 모든 음악에 대해 각각 n번 Cluster 중심과의 거리를 구하고(dn), 이 중 가장 먼 거리를 기준으로 삼아(Dn) 나누어서 1에서 뺀 값 (1-dn/Dn)을 각 음악의 n번 클러스터와의 적합도로 표현한다. 클러스터는 0번부터 4번까지 총 5개가 존재하므로, 5개의 적합도를 모아 5차원 벡터를 만들 수 있다. 이 벡터를 이용하여 각 음악의 특성을 표현한다.

$$\textbf{Music Vector} = (1-\frac{d0}{D0}, 1-\frac{d1}{D1}, 1-\frac{d2}{D2}, 1-\frac{d3}{D3}, 1-\frac{d4}{D4})$$

dn은 n번 Cluster 중심과 각 음악 특징 값 사이의 유클리드 거리

Dn은 모든 음악 중 n번 Cluster의 중심과 가장 멀리 있는 음악의 특징값이 가지는 유클리드 거리



Music Vector 생성의 1~4번 과정 도식화

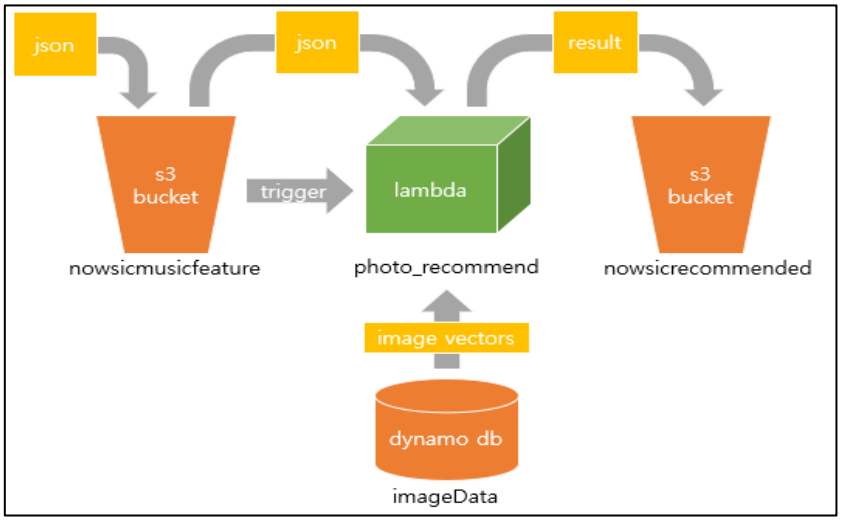
앞선 특징값 추출 및 K-Means Clustering을 통해 얻은 분류 결과를 기반으로, AWS에서 새로 업로드 된 음악의 특성을 표현하는 Music Vector를 생성하는 과정은 다음과 같다.

① 웹페이지를 통해 nowsicmusic 버킷에 음악 파일을 업로드하면 SNS로 메시지를 생성해 SQS에 전달

② EC2는 무한루프를 돌면서 SQS의 메세지를 체크, 새로운 SNS 메시지가 있을 경우 분석 해야 할 음악파일을 nowsicmusic 버킷에서 찾아 다운로드

③ 다운로드 받은 음악파일에서 지정한 구간의 지정한 특징값을 추출

④ 해당 특징값을 담은 json 파일을 만들어 nowsicmusicfeature 버킷에 전달



Music Vector 생성의 5~6번 과정 도식화

⑤ nowsicmusicfeature에 json파일이 업로드 되면 이를 트리거로 photo_recommend 함수 작동

⑥ 해당 특징값을 408개의 음악을 분류할 때 사용한 Norm을 사용해 유사 Normalization을 수행한 다음(표본의 수가 충분히 크기 때문에 오차가 적다고 가정), 미리 photo_recommend 함수에

기록해둔 각 Cluster의 중심 좌표를 이용해 앞서 정의한 방법대로 Music Vector 생성

(⑦ imageVector DB에 저장된 Image Vector들과 비교하여 가장 적절한 Vector값을 가지는 사진을 찾아 선택하고, 추천의 근거로 사용한 정보를 json파일에 담아 결과 버킷(nowsicresult)에 업로드)

3-3 사진 분석

ImageKey	Emotions	Labels	SV
NOWSIC00000002.jpg	face0 0 HAPPY 0 SAD 0 CALM 0 Others	94.0553283691 outdoors 0 kid	78.3291983852 85.3117935409
NOWSIC00000008.jpg	face0 0 HAPPY 0 SAD 0 CALM 0 Others	95.2871627808 outdoors 1 kid	102.133189158 123.442595156
NOWSIC00000012.jpg	face0 0 HAPPY 0 SAD 0 CALM 0 Others	0 outdoors 0 kid	120.287125087 174.528435248
NOWSIC00000020.jpg	face3 39.6581834157308 HAPPY 18.98179805278778 SAD 16.0021507...	0 outdoors 0 kid	71.5570876586 176.641908881
NOWSIC00000024.jpg	face2 39.34269046783447 HAPPY 33.80987548828125 SAD 8.0802659...	82.8473510742 outdoors 0 kid	143.619700115 128.042912341
NOWSIC00000028.jpg	face0 0 HAPPY 0 SAD 0 CALM 0 Others	98.7058944702 outdoors 0 kid	115.61955594 158.221309112

얼굴 수, 감정, 야외 여부, 아이를 의심한 모습. Image Data를 저장한 모습. 얼굴이 없으면 감정정보 또한 0이다.

AWS에서는 사진에 포함된 인물의 표정으로부터 감정을 분석하고, 사진에 포함된 객체를 감지하는 Rekognition 서비스를 제공한다. 이를 이용하여 사진으로부터 핵심적인 정보를 추출하고, 추가적으로 OpenCV를 이용해 사진의 채도와 명도를 파악하여 음악 추천의 근거로 사용한다.

3-3-1 사진 분석 요소

① 음악과 사진을 연결 짓기 위한 가장 강력한 근거로는 사진 속 인물의 감정을 사용한다. Rekognition을 통해 사진 속 인물이 현재 느끼고 있을 감정을 신뢰도로 얻을 수 있다. 파악할 수 있는 감정들 중 Surprised, Disgusted, Angry, Confused는 공통적으로 격렬한 음악이 어울리는 감정들이므로 이들 중 가장 큰 값 하나만을 사용해 격렬한 음악과 연결 지을 척도로 사용한다.

② 어린이가 등장하는 사진에는 일반적인 대중가요나 클래식보다 동요가 어울리므로 사진 속에 어린이가 등장하는지 여부를 파악한다.

③ 야외의 경우, 일반적으로 좀 더 시끄러운 음악이 적합하므로 사진 속 장소가 야외인지 여부를 파악한다. 사진에 사람이 등장하지 않을 경우 비중이 높은 척도로 작용한다.

④ 앞의 세가지 정보는 AWS Rekognition을 통해 얻고, OpenCV를 이용하여 사진의 채도와 명도를 파악한다. 밝고 투명한 색채일수록 좀 더 시끄럽고 빠른 템포의 음악을 추천한다. 사진에 사람이 등장하지 않을 경우 비중이 높은 척도로 작용한다.

3-3-2 Weight Table

	Happy	Sad	Calm	Others	Outdoor	Saturation	Brightness
Loud_C =	4.0	1.0	0.0	5.0	0.35	0.04	0.03
Happy_C=	5.0	0.0	1.0	4.0	0.35	0.03	0.04
Calm_C =	1.0	3.0	5.0	0.0	0.15	0.06	0.07
Sad_C =	0.0	5.0	3.0	1.0	0.15	0.07	0.06
Else_C =	2.0	3.0	3.0	2.5	0.25	0.05	0.05
	Max100	Max100	Max100	Max100	Max100	Max255	Max255

사진에서 나타나는 감정, 장소, 채도, 명도에 음악의 Cluster마다 어울리는 정도에 따라 가중치를 부여해 표를 만들고(Weight Table), 이를 이용하여 사진에 대응하는 Image Vector를 생성한다.

예를 들면, 웃는 사람이 등장하는 사진은 'Happy' Cluster에 어울릴 것이고, 우는 사람이 등장하는 사진은 'Sad' Cluster에 어울릴 것이다. 앞서 분석한 Cluster간의 수치적인 관계 및 각 Cluster에 포함된 음악들의 경향을 근거로 초기 가중치 값을 위의 표와 같이 부여하였다.

예를 들어, Happy **80**%, Sad **10**%, Calm **0**%, Others **30**%, Outdoor **70%**, Saturation **100**, Value(Brightness) **100**인 사진이 있다면 이 사진에 대응되는 **Image Vector**는

Image Vector = **(** (4.0*80) + (1.0*10) + (0.0*0) + (5.0*30) + (0.35*70) + (0.04*100) + (0.03*100),

(5.0*80) + (0.0*10) + (1.0*0) + (4.0*30) + (0.35*70) + (0.03*100) + (0.04*100),

(1.0*80) + (3.0*10) + (5.0*0) + (0.0*30) + (0.15*70) + (0.06*100) + (0.07*100),

(0.0*80) + (5.0*10) + (3.0*0) + (1.0*30) + (0.15*70) + (0.07*100) + (0.06*100),

(2.0*80) + (3.0*10) + (3.0*0) + (2.5*30) + (0.25*70) + (0.05*100) + (0.05*100) **)**

= **(** 511.5 , 551.5 , 133.5 , 103.5 , 292.5 **)**

와 같이 구해진다.

(Others는 Surprised, Angry, Disgusting, Confused 중 가장 큰 값)

이는 순서대로 'Loud' Cluster, 'Happy' Cluster, 'Calm' Cluster, 'Sad' Cluster, 'Else' Cluster와의 적합도를 나타낸다.

3-3-3 Image Vector

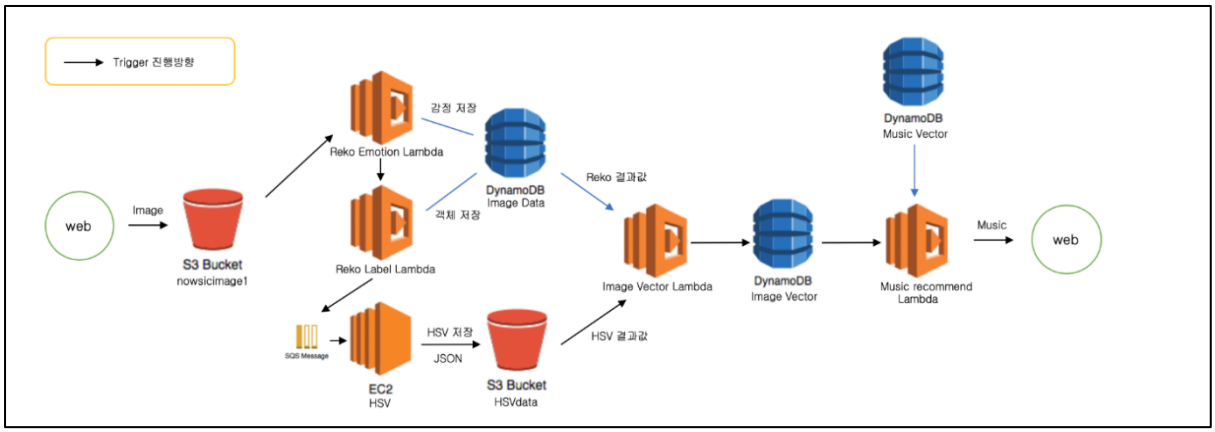


Image Vector를 생성하는 과정은 다음 순서로 이루어진다.

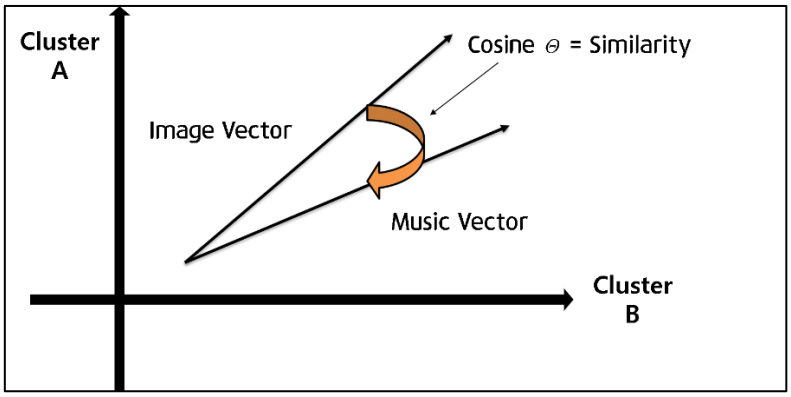
① 웹페이지를 통해 nowsicimage1 버킷에 사진이 업로드 되면 이를 트리거로 Rekognition을 통해 사진 속 인물의 감정을 파악하는 imageRekoF 함수가 작동하고, 결과를 imageData DB에 저장한 다음 imageRekoL 로 람다 트리거를 보낸다.

② imageRekoL 람다에서 Rekognition을 통해 사진 속의 등장인물이 어린이인지, 또 장소가 야외인지 아닌지를 파악하는 파악하도록 하는 Lambda 함수인 imageRekoL의 트리거가 되며, 결과를 imageData DB에 저장한 다음 nowsic.hsv1 버킷에 사진의 key값을 이름으로 하는 파일을 저장한고 EC2에 SQS를 통해 메시지를 전달한다

③ 메시지를 전달받은 EC2 instance는 nowic.hsv1에 저장된 key값에 해당되는 사진을 nowsicimgae1 버킷에서 찾아 다운로드 받은 다음, OpenCV를 통해 평균 채도와 평균 명도를 계산하고 결과를 nowsic.hsv2 버킷에 json파일로 저장한다.

④ nowsic.hsv2 버킷에 업로드 된 json 파일을 트리거로 Image Vector를 생성하는 Lambda 함수인 imageVector가 json파일에 저장된 채도 값과 명도 값을 imageData DB에 저장한 다음, 최종적으로 결정된 imageData와 함수에 저장된 Weight Table을 바탕으로 Image Vector를 생성하여 이를 imageVector DB에 저장한다.

3-4 추천



2차원에 표현한 코사인 유사도

코사인 유사도는 벡터로 표현된 두 데이터 사이의 거리를 내적 결과로 표현하는 방식이다.

앞서 음악을 분석을 통해 Music Vector를 생성하고, 사진을 분석을 통해 Image Vector를 생성하였다. 두 벡터의 각 성분은 Cluster와의 적합도를 나타내므로, 각 벡터의 크기를 1로 resize한 다음 내적하면 코사인 유사도를 통해 둘 사이의 유사도를 측정할 수 있으며, 높은 유사도는 추천의 근거가 된다. 만약 사용자가 사진으로부터 음악을 추천 받길 원한다면 업로드 된 사진으로부터 Image Vector를 생성하고, DB에 미리 저장해둔 음악의 Music Vector들과 비교하여 내적 결과가 1에 가까운 Music Vector를 가진 음악들을 우선적으로 추천한다. 음악에서 사진을 추천할 때도 같은 방식을 사용한다.

ImageKey	cluster	Kid	Vector
NOWSIC00000002.jpg	1	0	0.534354891907 0.535347403825 0.343448683395 0.342456171478 0....
NOWSIC00000008.jpg	1	1	0.515604445298 0.518453979301 0.368738028776 0.365888494773 0....
NOWSIC00000012.jpg	2	0	0.294369877392 0.310261656612 0.5693894569 0.55349767768 0.431...
NOWSIC00000020.jpg	1	0	0.510858037391 0.548097263225 0.35993200433 0.32823193668 0.44...
NOWSIC00000024.jpg	1	0	0.477863684278 0.489418645162 0.381243281187 0.416012958488 0....
NOWSIC00000028.jpg	1	0	0.507254047527 0.512172219293 0.379222162503 0.374303994592 0....

최종적으로 1로 resize 된 Image Vector

url	cluster	music_vector	nc_coef	not_choiced
-8cYArQVm8Q	0.0	0.4996933273312166 0.486283993718992 0.3941940238804983 0.373...	1.0	0.0
-OfOkiVFmhM	1.0	0.5017374223065733 0.6671847288402922 0.2849237120165467 0.31...	1.0	0.0
-nk5ig_1BIA	1.0	0.5217813814564258 0.5918545499354652 0.31928176921684664 0.3...	1.0	0.0
1cQh1ccqu8M	4.0	0.4843061042191308 0.41686494303964017 0.40045777808436445 0....	1.0	0.0
1prweT95Mo0	2.0	0.30809159353558413 0.2013414699665543 0.6234483055675905 0.5...	1.0	0.0
1s8nNp-C89k	0.0	0.5242023590487126 0.45724301467454465 0.37711653604381007 0....	1.0	0.0

크기 1로 resize 된 Music Vector

사용자에게 음악과 사진을 추천하기 위해, imageVector DB에 400여개, music_cluster DB에 400여 여 개의 Key값과 그에 대응되는 벡터를 저장해 두었다. Image Vector에 대응되는 Key값은 nowsicimage1 버킷에 저장된 해당 사진의 이름이며, Music Vector에 대응되는 Key값은 해당 음악을 들을 수 있는 youtube의 고유 주소 11자리이다.

3-5 피드백

K-Means Clustering은 단순히 분산을 최소화하는 경계를 구할 뿐이므로, 각각의 Cluster에는 전체적인 Cluster의 분위기와 맞지 않는 음악이 섞여 있을 수 있다. 이러한 음악을 추천하는 것은 적절치 않으므로, 이를 추천치 않도록 해야 한다.

또, Weight Table을 구성하는 가중치는 Clustering 분석 결과를 이용해 Nowsic Team에서 추정한 값이므로, 엄밀한 정확도를 보장하지 못한다.

이러한 문제들을 해결하기 위해, 사진-음악 추천 시 사진에 가장 적합한 Cluster로부터 유사도가 높은 음악 중 4개의 음악을 추천하고(Major Cluster), 그 외의 Cluster 중에서 유사도가 높은 음악 중 2개의 추천한 다음(Minor Cluster), 웹페이지에서 사용자의 선택을 수집하도록 했다.

사용자의 선택은 nowsicfeedback 버킷에 json파일로 저장되며, 이를 트리거로 getUserFeedback Lambda 함수가 작동하여 Music Vector를 저장해 놓은 music_cluster 테이블과 Weight Table의 가중치를 수정하는 근거로 사용할 ClusterFeedback 테이블에 json 파일에 담긴 사용자 선택 결과를 누적한다.

① 부적합 음악 탐지

url	cluster	music_vector	nc_coef	not_choiced
-8cYArQVm8Q	0.0	0.4996933273312166 0.486283993718992 0.3941940238804983 0.373...	1.0	0.0
-OfOkiVFmhM	1.0	0.5017374223065733 0.6671847288402922 0.2849237120165467 0.31...	1.0	0.0
-nk5ig_1BIA	1.0	0.5217813814564258 0.5918545499354652 0.31928176921684664 0.3...	1.0	0.0
1cQh1ccqu8M	4.0	0.4843061042191308 0.41686494303964017 0.40045777808436445 0....	1.0	0.0
1prweT95Mo0	2.0	0.30809159353558413 0.2013414699665543 0.6234483055675905 0.5...	1.0	0.0
1s8nNp-C89k	0.0	0.5242023590487126 0.45724301467454465 0.37711653604381007 0....	1.0	0.0

빨간 네모 상자 안이 nc_coef와 not_choiced cound야

만약 Cluster에 지나치게 어울리지 않는 음악이 섞여 있다면, 그 음악은 계속해서 사용자에게 선택 받지 못할 것이다. Music Vector를 저장해 놓은 테이블인 music_cluster의 Attribute로 not_choiced와 nc_coef를 추가한 다음, 초기값으로 0과 1.0을 설정한다.

만약 어떤 음악이 Major Cluster로써 등장할 때 마다 계속해서 다른 Major Cluster의 음악만 사용자의 선택을 받는다면, 이는 음악을 추천하기 위해 적절치 못한 Cluster를 선택한 게 아니라 해당 음악이 Cluster에 어울리지 않는 것이라고 확신할 수 있다. 사용자가 다른 Major Cluster의 음

악에 만족 평가를 내릴 때마다 같이 등장한 다른 Major Cluster의 음악들의 not_choiced 항목을 1씩 증가시키되, 한번이라도 사용자 만족을 받는다면 not_choiced의 카운트를 초기화 시킨다. 만약 초기화 되기 전에 10번의 카운트가 쌓인다면, 이 카운트를 초기화 하고 nc_coef에 0.99를 곱한다.

nc_coef는 사진에서 음악을 추천하는 근거인 Image Vector와의 내적 결과, 즉 코사인 유사도를 계산할 때 항상 뒤에 덧붙여 마지막에 곱해지도록 한다. 어떤 비율의 사진에 대해서도 불리한 결과가 되기 때문에, nc_coef가 낮아질 수록 점점 추천 목록에 등장할 수 없게 된다.

② 가중치 수정

MajorCluster	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

가중치 재조정을 위한 데이터가 모르는 건.

사진과 가장 적합도가 높다고 분석한 Major Cluster의 음악이 아닌 Minor Cluster의 음악에 만족하는 비율이 높다면, 이는 Image Vector를 계산할 때 사진 특성에 할당된 가중치를 수정해야 한다는 뜻이다.

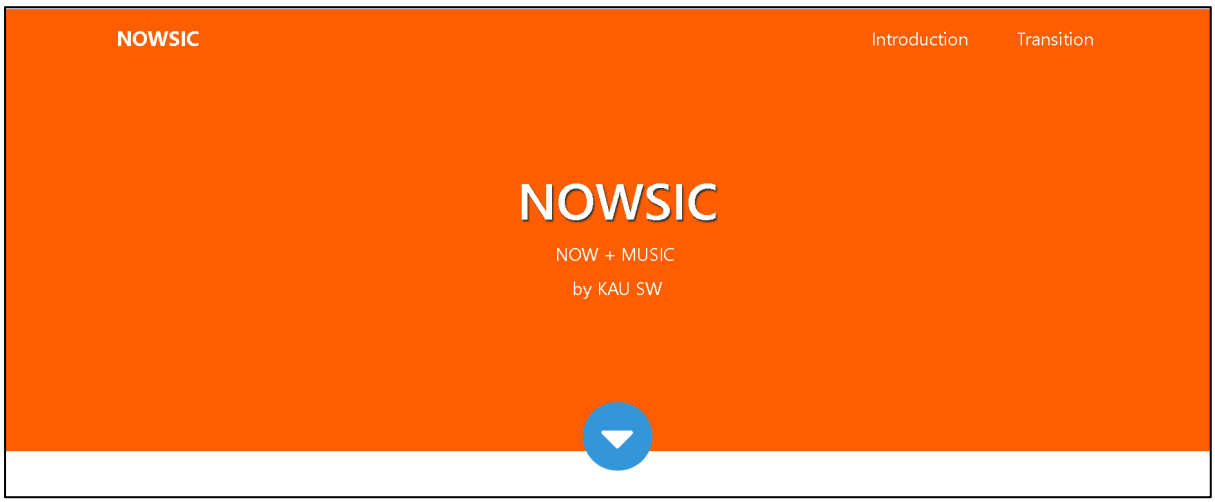
Clustering 결과가 충분히 강력하기 때문에, 한 감정이(Image Vector의 한 성분이) 강력하게 나타는 사진에 음악을 추천할 때는 문제가 나타나지 않지만, 인접한 Cluster('Happy' Cluster와 'Loud' Cluster, 'Sad' Cluster와 'Calm' Cluster)의 경계에 Image Vector가 위치할 때 일어난다. (Image Vector의 0번 성분과 1번 성분 혹은 2번 성분과 3번 성분이 거의 같은 값일 때)

가중치를 적절히 수정하기 위해, 매 사용자의 만족도 표현마다 Major Cluster와 선택된 음악이 속해 있는 Cluster의 쌍을 저장한다.

수학적으로 초기 가중치로부터 인접하지 않은 Cluster 쌍이 선택되는 일은 일어날 수 없다. 결국 인접한 Cluster끼리 각자가 Major Cluster일 때 어느 비율로 선택되었는지 파악할 수 있으며, 이 비율은 가중치 수정의 근거가 된다.

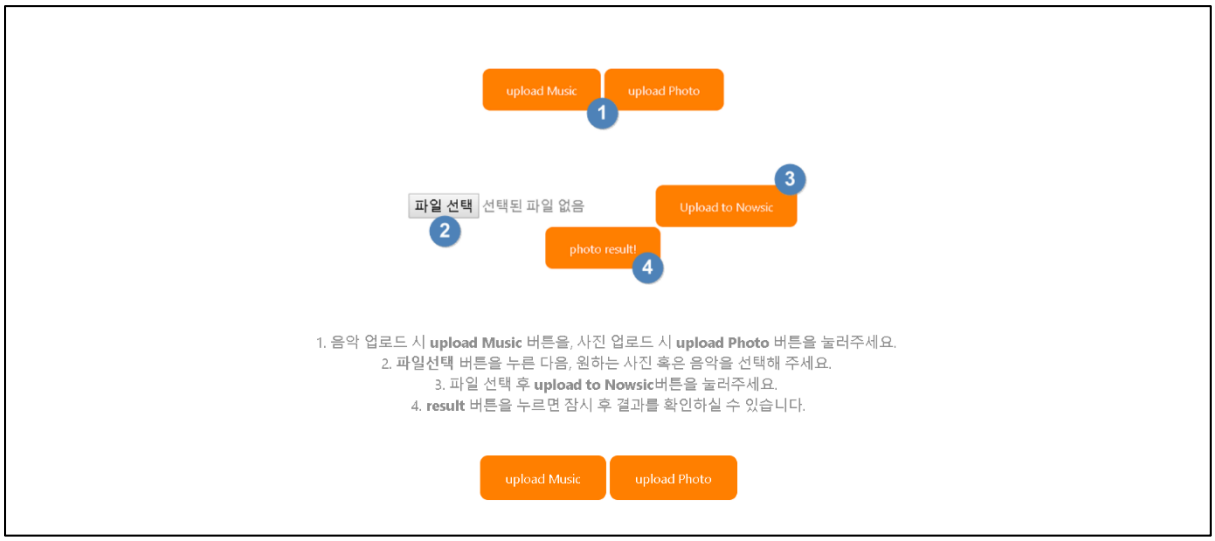
3-5 웹페이지

① 홈페이지



홈페이지 상단 부분에서는 왼쪽 위의 최상단으로 돌아오는 네비게이션과 우측 상단의 Nowsic 소개로 이동하는 네비게이션(Introduction), 추천할 파일을 업로드 하는 곳으로 이동하는 네비게이션(Transition), 그리고 Nowsic 소개 부분으로 이동하는 역삼각형 버튼을 확인할 수 있다.

홈페이지 중단은 Nowsic을 소개하는 section인 NOWSIC Introduction이다. 홈페이지 상단에서 역삼각형 버튼을 누르거나 Introduction 네비게이션을 눌러서 이동할 수도 있다. START 버튼을 눌러 추천 받을 사진 혹은 음악 파일을 업로드 하는 section으로 이동할 수 있다.

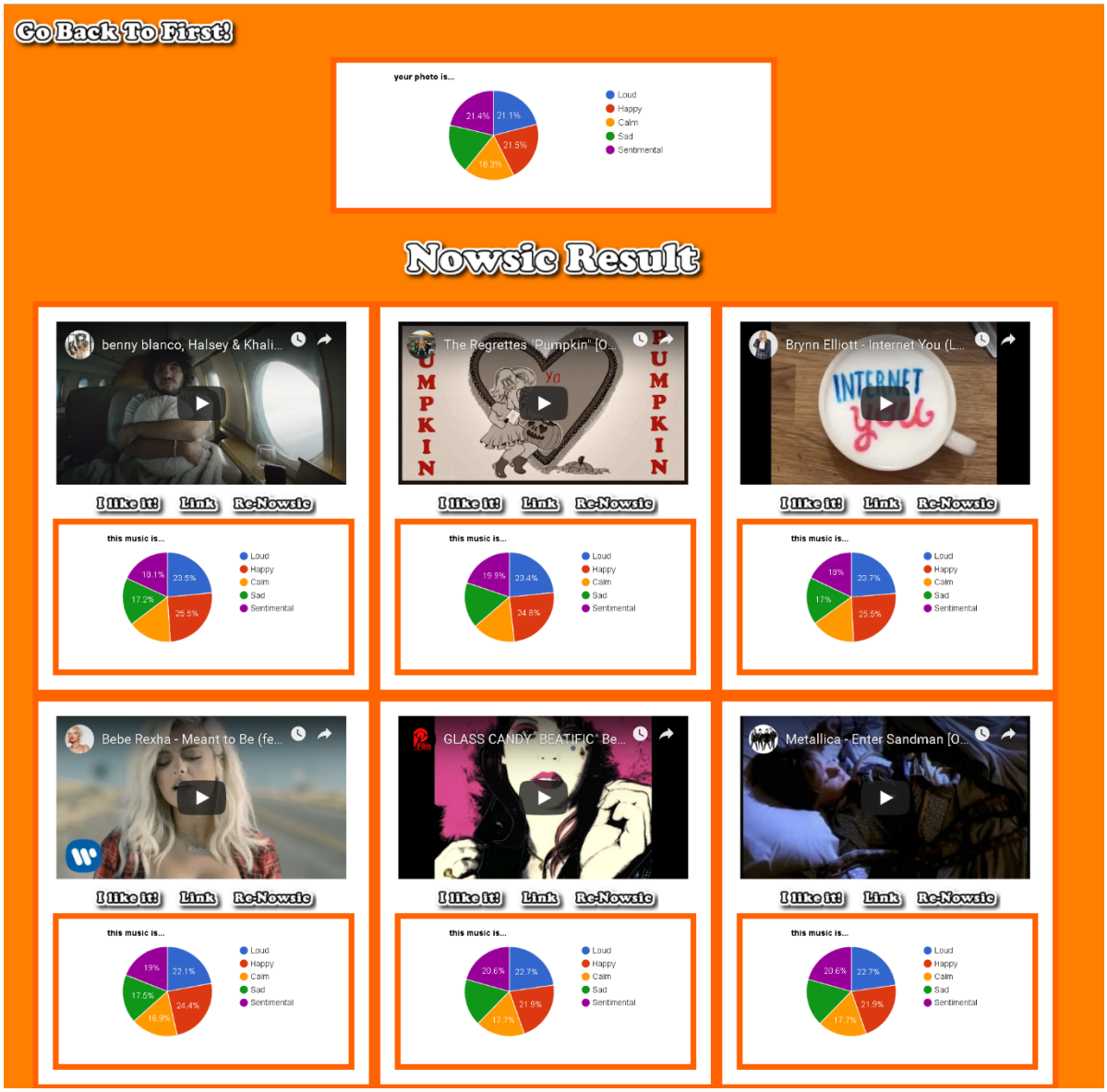


홈페이지 하단은 사진을 추천 받을지 음악을 추천 받을지 선택하고 업로드하는 section 이다. 사용하는 순서에 대한 설명이 있으며, Upload Music(혹은 Upload Photo) 버튼을 누르면 filechooser 버튼이 나타나고, 파일을 선택한 다음 우측의 Upload to Nowisc 버튼을 통해 Nowsic 에 업로드할 수 있다. 마지막으로, photo result!(music result!) 버튼을 눌러 결과를 확인할 수 있다.

② 로딩 페이지

추천 결과가 웹페이지에 전송될 때까지 사용자를 기다리도록 하는 화면이다. SQS 를 통해 결과 버킷 (nowsicresult, nowsicresult.phototomusic)에 추천 결과를 담은 json 파일이 업로드 된 것을 확인하면 결과페이지로 넘어간다.

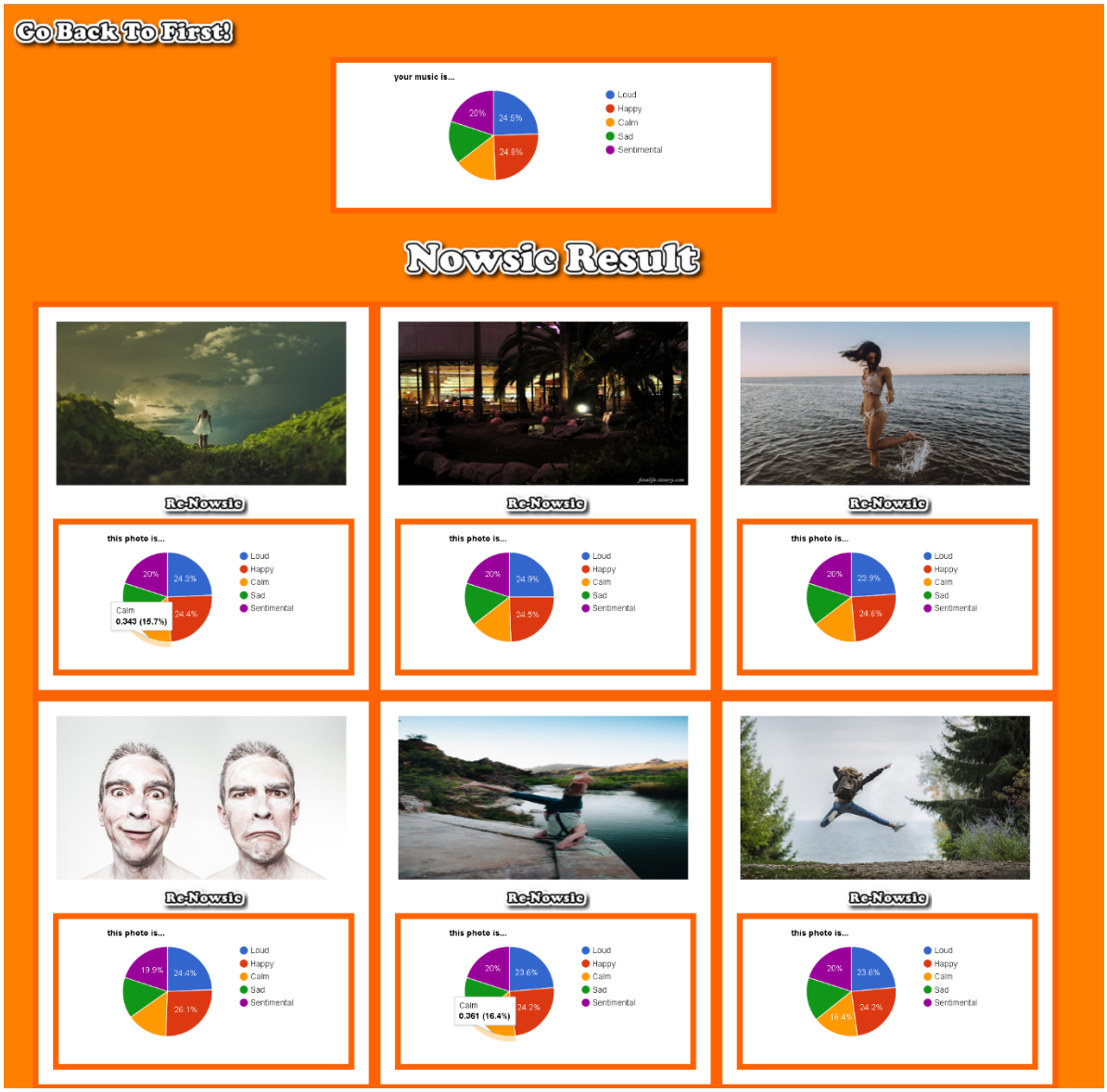
③ 사진-음악 결과 페이지



최상단의 파이차트는 사용자가 어울리는 음악을 찾길 원했던 사진의 분석 결과로, 어떤 분위기가 어느 비율로 섞여 있는 지를 나타낸다. (각 항목은 Image Vector의 성분이다.) 하단에는 youtube 소스코드를 이용한 6개의 음악 추천 결과가 표시되며, 각각의 추천 결과 또한 파이차트를 통해 어떤 비율로 감정이 섞여 있는 지를 나타낸다 (각 항목은 Music Vector의 성분이다.).

음악 재생버튼 아래에는 Nowsic에서 제공하는 3개의 버튼이 있다. youtube 저작권 정책에 의해 해당 페이지에서 음악을 재생할 수 없을 경우, Link 버튼을 누르면 해당 음악을 들을 수 있는 youtube 페이지로 이동한다. I like it 버튼을 통해 추천 결과에 대한 만족을 표현할 수 있으며, 동시에 AWS로 사용자 선택을 반영하기 위한 데이터가 전송된다. Re-Nowsic 버튼을 누르면 해당 음악에 어울리는 사진을 추천 받을 수 있다.

④ 음악-사진 결과 페이지



최상단의 파이차트는 사용자가 어울리는 사진을 추천 받기 원했던 음악의 분석결과로, 어떤 분위기가 어느 비율로 섞여 있는지 파이차트로 나타낸다. (각 항목은 Music Vector의 성분이다.)

사진-음악 추천과 마찬가지로 각 사진의 하단에도 사진의 분석결과가 파이차트로 그려져 있으며, (각 성분은 Image Vector의 성분이다.) Re-Nowsic 버튼을 통해 다시 사진으로부터 음악을 추천받을 수 있다.

사진의 분석결과는 AWS의 Rekognition을 사용하기 때문에 신뢰도가 보장되는 반면, 사용자가 업로드한 음악이 동요처럼 일반적인 음악들과 상이한 특징값을 가지거나 노이즈가 많을 경우, 또 K-Means Clustering으로 파악한 분류 모델과 지나치게 어긋날 경우 신뢰도를 보장할 수 없기 때문에, 음악-사진 추천결과에서는 피드백을 받지 않는다.

4. 문제점 및 해결내용

4-1 사진 분석

① 사진에 얼굴이 여러 개 있을 때, Rekogntion에서 문제가 발생 함.

- 신뢰도가 높은 3명의 얼굴만 분석하도록 해 문제 해결

② 사진에 대한 분석이 총 3단계로 이루어 지는데, SNS를 사용하면 동시 병렬로 진행되기 때문에 저장 완료 후 imageVector Lambda에 트리거를 보낼 때 문제가 생김.

- Lambda 트리거를 사용하여 순차적으로 진행한 다음 마지막 단계에서는 DB가 아닌 s3버킷으로 업로드를 해서 버킷 트리거로 imageVector Lambda 실행.

③ music_recommend 함수가 무한 반복되면서 최신에 업데이트 된 순으로 imageVector의 항목을 삭제하는 문제

- 사용자가 업로드한 사진에 대한 정보를 추천이 끝난 뒤 지우도록 했는데, Lambda함수의 DB트리거는 항목이 삭제될 때도 반응해서 생긴 문제였다. music_recommend 함수 시작 시 트리거 이벤트가 INSERT가 아니면 아무 일도 하지 않도록 함으로써 해결.

④ (4-2와 공통적으로) DynamoDB에 float 형 데이터를 저장할 수 없는 문제

- 공백으로 구분된 String으로 바꾸어 저장한 다음 불러올 때 파싱하여 사용

4-2 음악 분석

① 지나치게 긴 분석시간 문제

- 전체 구간 대신 30초에서 60초 구간만 분석하도록 함으로써 해결

② 분석에 사용할 특징값의 종류 및 형태 선택 문제

- 다양한 조건에서 테스트 해봄으로써 좋은 결과를 보이는 특징값의 종류 및 형태를 결정

4-3 추천

① K-Means Clustering이 동요를 제대로 분류하지 못함.

- 전체 Cluster와 어울리지 않는 동요가 항상 존재함을 통해 동요를 제외하고 다시 특징값의 형태 및 구간을 바꿔가며 Clustering을 시도, 강력한 결과를 얻어 해결했다.

② Image Vector 혹은 Music Vector가 5개의 성분이 비등비등할 때 종종 엉뚱한 사진 혹은 음악을 추천함

- 아무리 가중치를 수정하더라도 이런 경우는 발생할 것이기 때문에, 이런 경우에 한해서 코사인 유사도가 아닌 다른 추천 근거를 마련할 필요가 있음.

③ AWS Rekogniton 성능의 문제로 가끔 어린이가 아닌 사진을 어린이로, 혹은 어린이 사진을 어린이로 분석하지 못하거나 사진 속 인물의 감정을 잘못 파악하는 등 문제를 보임

- 사진 분석을 Rekognition에 대부분 의존하기 때문에 근본적으로 해결하기 힘든 문제.

④ Weight Table을 수정하면, 현재 저장중인 Image Vector를 전부 수정해야 하는 문제

- ImageData 테이블에 Image Vector로 변환 전의 자료들을 저장하고, update_imageVector Lambda 함수를 통해 가중치 변경이 있을 때 마다 한번에 업데이트 할 수 있도록 함..

4-4 웹페이지

① 한글이 깨지는 현상 발생

- UTF-8을 인코딩 하여 해결.

② 사용자가 동시에 추천 결과를 얻으려 할 때 로딩 화면 때문에 결과를 보지 못함

- 여러 사용자가 동시에 Nowsic을 사용할 수 있는 환경을 구현하려고 하였으나, 시간 부족으로 구현하지 못하였음.

5. 계획 대비 자체 평가

5-1 성공

① 사진과 음악을 연결 짓기 위한 기준 및 분석 방법 확립

사진과 음악은 전혀 다른 성질을 가지고 있기 때문에, 둘 사이를 연결 지을 기준을 세우는 것이 가장 중요했다. 팀원들의 자료 조사 및 테스트를 통해 사진의 감정 및 색채 정보, 음악의 특징 값 및 K-means Clustering과 코사인 유사도를 이용하여 사진과 음악을 추천할 기준을 세우는 데 성공했다.

② 발전 가능한 사진과 음악 간의 추천 프로세스 구현

사진과 음악 사이의 추천 정확도를 높이기 위해서는 사진으로부터 좀 더 다양한 특성을 추출하고 각각에 적합한 가중치를 부여해야 하며, 음악의 분류 결과를 발전시켜야 한다. Rekognition에서 사진의 객체정보를, OpenCV에서 사진의 색채 정보를 좀 더 상세하게 분석할 여지가 있으며, 음악의 분류 모델을 새롭게 구축한다면 추천 결과를 산출하는 Lambda 함수를 업데이트하여 반영할 수 있다. 또, 사용자의 만족도를 조사하는 방식으로 사진의 특성에 부여될 가중치를 수정할 데이터를 수집하고 잘못 분류된 음악을 검출할 수 있다.

5-2 실패

① 모호한 구간에서의 추천 정확도 및 분석의 세밀화

사진과 음악의 분석에서 강력한 특징을 얻지 못할 경우, 이에 대응되는 추천 결과가 낮은 정확도를 보인다. 많은 테스트를 통해 사진 특성에 대응되는 가중치를 조절하고, 좀 더 다양한 사진의 특성을 추출해 사용해야 하며, 동요와 같이 독특한 특성을 가지는 음악을 미리 파악할 방법이 필요하다. 또, 분석 기능을 강화하더라도 확실한 특성을 보이지 않는 음악과 사진이 있을 수 있기 때문에, 이를 대비한 추천 기능 강화 또한 필요할 것이다.

② 다수의 사용자를 대비한 환경 구축 실패

사전 지식 부족 및 일정 조절의 실패로 다수의 사용자가 동시에 서비스를 이용할 때 생기는 문제점을 해결하지 못하였다. 다른 부분에서도 문제가 있었지만, 특히 이 실패를 통해 구현 가능성 확인 및 계획 수립의 중요함을 느낄 수 있었다.