노래 가사 분석을 통한 인기 차트 Top10 예측하기

- 멜론의 시대별 인기차트를 이용하여

AI빅데이터융합경영학과
20202658
이 태 범

Index :

1. 프로젝트 소개

- 국내 최고 음원 스트리밍 사이트인 멜론에서 제공하는 시대별 차트 페이지를 크롤링해 1990년부터 2021년까지의 Top 30 노래 가사 데이터를 얻음

- 노래 가사 데이터에 대한 기본적인 텍스트 분석을 진행

- 텍스트 분석을 진행한 결과를 이용해 1990 ~ 2019년 까지의 데이터를 Train, 2020 ~ 2021년의 데이터를 Test로 머신러닝 과정을 거쳐 최종적으로 Top10 여부를 예측

2. 데이터 크롤링

- 직접 짠 코드로 멜론 사이트에서 시대별 차트 데이터를 크롤링 (1990 ~ 2021 년)

- 크롤링한 데이터를 바탕으로 '제목', '가수', '가사' 형태의 DataFrame을 만들어 저장

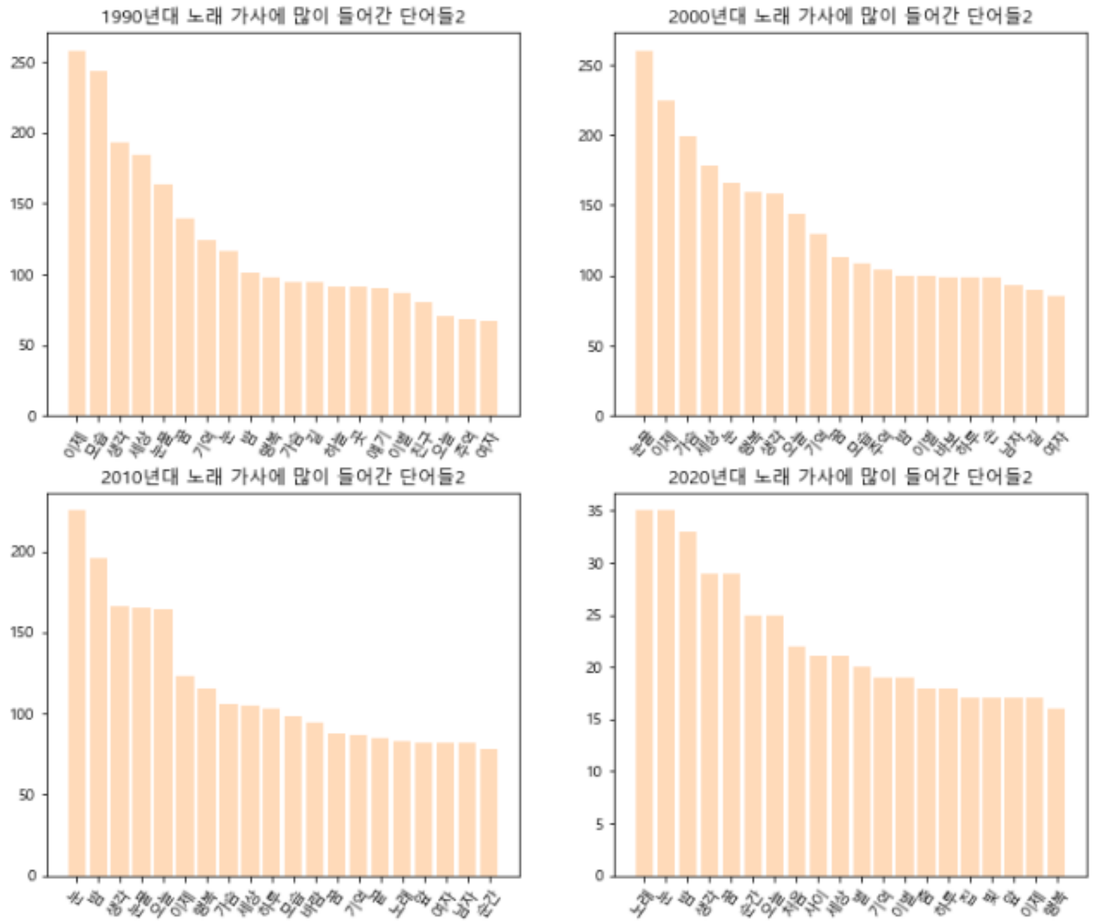
- 너무 많은 데이터를 가져올 경우 IP 밴을 먹을 수 있어 각 년도별 Top30곡만을 크롤링해 가져옴

```
def melon_collector(url):
    time.sleep(5)
    driver.get(url)
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')

    # 제목 가져오기
    title=driver.find_elements(by=By.CLASS_NAME, value='ellipsis.rank01') # 제목

    title2=[]
    for i in title:
        title2.append(i.text)

    del title2[30:]

    # 가수 가져오기
    singer=driver.find_elements(by=By.CLASS_NAME, value='ellipsis.rank02') # 가수

    singer2=[]
    for i in singer:
        singer2.append(i.text)

    del singer2[30:]

    # 가사 가져오기

    song_info = soup.find_all('div', attrs={'class': 'ellipsis rank01'})

    # Top 30의 추출
    songid = []

    for i in range(30):
        try:
            songid.append(re.sub(r'[^0-9]', '', song_info[i].find("a")["href"][43:])) # 곡 id정보 추출
        except:
            songid.append('')
            continue # 1992년 원지가 되어와 같이 멜론에서 현재 들을 수 없는 노래 제외

    lyrics=[]

    for i in songid:
        try:
            driver.get("https://www.melon.com/song/detail.htm?songId=" + i)
            time.sleep(2)
            lyric=driver.find_element(by=By.CLASS_NAME, value="lyric")
            lyrics.append(lyric.text)
        except: # 가사가 없다.
            lyrics.append('')
            continue
    lyrics2 = []

    for i in lyrics:
        lyrics2.append(i.replace("₩n"," "))

    df=pd.DataFrame({"제목":title2,"가수":singer2,"가사":lyrics2})

    # 저장하기
    df.to_csv(f'멜론{start}.csv', index = False)
```

3. 데이터 분석

3-1. 빈도 분석

- 토큰화, 불용어 처리를 거쳐 빈도 분석을 진행
- Komoran 분석기를 선택



3. 데이터 분석

3-2. TF-IDF

- 많이 등장하는 단어에 계산 과정에서 역 가중치를 주기 때문에 별다른 명사 추출이나 불용어 처리를 거치지 않은 상태에서 분석 진행

- 임베딩 과정을 거치기 때문에 추후에 머신러닝 과정에서의 Feature로 사용

```
tfidf_mean = pd.DataFrame([sorted(mean_1990s.items(), key = lambda item: item[1], reverse = True)[:10],
                           sorted(mean_2000s.items(), key = lambda item: item[1], reverse = True)[:10],
                           sorted(mean_2010s.items(), key = lambda item: item[1], reverse = True)[:10],
                           sorted(mean_2020s.items(), key = lambda item: item[1], reverse = True)[:10]])

tfidf_mean.set_index(keys = [[1990,2000,2010,2020]], inplace = True)
tfidf_mean.columns = np.arange(1,11)
tfidf_mean
```

	1	2	3	4	5	6	7	8	9	10
1990	(사랑, 0.111)	(그대, 0.086)	(마음, 0.047)	(모습, 0.045)	(다시, 0.043)	(우리, 0.043)	(사람, 0.04)	(세상, 0.037)	(눈물, 0.036)	(생각, 0.036)
2000	(사랑, 0.139)	(그대, 0.09)	(사람, 0.053)	(다시, 0.049)	(눈물, 0.043)	(가슴, 0.038)	(하나, 0.035)	(우리, 0.033)	(마음, 0.032)	(이제, 0.032)
2010	(사랑, 0.08)	(그대, 0.052)	(다시, 0.038)	(우리, 0.038)	(사람, 0.032)	(눈물, 0.031)	(지금, 0.029)	(오늘, 0.028)	(마음, 0.027)	(생각, 0.027)
2020	(사랑, 0.067)	(우리, 0.043)	(모든, 0.04)	(그대, 0.036)	(매일, 0.036)	(지금, 0.034)	(다시, 0.033)	(이제, 0.033)	(노래, 0.029)	(순간, 0.028)

3. 데이터 분석

3-3. 토픽 모델링

- 몇 개의 토픽으로 가사를 분류하는 것이 적절한지 Perplexity와 Coherence를 기준으로 선택 (최종적으로 12개의 토픽으로 지정)

- 결과를 확인했을 때 비슷한 의미의 단어들이 잘 묶여있는 것을 확인 가능

- LSA 방식보다 LDA 방식을 사용했을 때 토픽이 더 잘 구분됨

Topic ID: 0

그녀	0.0929611474275589
밤하늘	0.038122132420539856
벌써	0.03580517321825027 5
하나로	0.024322424083948135
송이	0.023669792339205742
베이비	0.022097760811448097
다운타운	0.022097760811448097
으르렁	0.02204214595258236
고백	0.019669337198138237
그림자	0.019368356093764305
야야	0.01899593323469162
꽃잎	0.018342584371566772
마주	0.01831807568669319
용기	0.01747623272240162
소식	0.016938503831624985
장미꽃	0.01390683464705944
새벽	0.013855669647455215
길이	0.013608208857476711
달이	0.013573571108281612
외치	0.013087167404592037

Topic ID: 1

매일	0.14634287357330322
그땐	0.03544526919722557
하나요	0.029888221994042397
어서	0.029080504551529884
대도	0.027957595884799957
물보라	0.02693706378340721
취해	0.023344894871115685
제일	0.021215813234448433
우연	0.02110127918422222
진짜	0.018253756687045097
먼저	0.01518103014677763
온통	0.014657323248684406
시선	0.01403206866234541
히라기	0.013483995571732521
루루	0.012351338751614094
행동	0.010682424530386925
오지	0.010355481877923012
여행	0.00961446762084961
나인	0.009506963193416595
불쑥	0.009260859340429306

토픽 0 : 여러 단어들이 0.1%가 안되는 낮은 비중을 보이며 그녀, 고백, 꽃잎같이 설렘을 느낄 수 있는 단어들이 위치해있다.
토픽 1 : 매일, 그땐, 제일 등 명사를 앞에서 꾸며주는 단어들이 위치해있다.
토픽 2 : 겨울, 바다, 파도, 여름 등 계절과 바다가 떠오르는 단어들이 위치해있다.
토픽 3 : 마음, 생각, 가슴 등 누군가를 기억하거나 떠올리는 데 사용되는 단어들이 위치해있다.
토픽 4 : 점핑, 빠빠빠, 외톨이야와 같이 특정 노래가 떠오르는 단어들이 엿보인다.
토픽 5 : 남자, 여자, 오빠와 같이 성별이나 머리, 정신같이 생각과 관련된 단어들이 위치해있다.
토픽 6 : 무릎, 어깨같은 신체부위, 담배, 샴푸와 같은 생활용품 단어들이 위치해있다.
토픽 7 : 다시, 세상, 하늘같이 노래 가사에 많이 사용될법한 단어들이 많이 보인다.
토픽 8 : 보고, 운명, 걱정 등 누군가를 기다리는데 사용될만한 단어들이 위치해있다.
토픽 9 : 노래, 가사 등과 후렴구에 나올법한 단어들이 많이 위치해있다.
토픽 10 : 향기, 세월, 바람과 같이 지나가버린 세월, 인연 등이 느껴지는 단어들이 엿보인다.
토픽 11 : 달라, 살짝, 벚꽃과 같이 희망이나 포기하지 않는 느낌이 드는 단어들이 위치해있다.

4. Feature 데이터 생성

- 빈도 분석을 통해 각 년도에 가장 많이 등장한 Top 10 단어들의 빈도 (10 columns)

- TF-IDF 분석을 통해 90년대, 00년대, 10년대 각각에 주로 사용된 단어 Top 10을 지정해 모든 년도에 적용시켜 Feature로 활용 (30 columns)

- 가사 데이터를 W2V를 통해 임베딩하여 Feature로 활용 (30 columns)

	제목	빈도_1등	빈도_2등	빈도_3등	빈도_4등	빈도_5등	빈도_6등	빈도_7등	빈도_8등	빈도_9등	...	w2v_20	w2v_21	w2v_22	w2v_23	w2v_24	w2v_25	w2v_26	w2v_27	w2v_28	w2v_29
0	희망사항	17	2.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	0.443735	0.104548	-0.101639	-0.147584	0.592460	-0.153130	0.034033	0.363047	0.505773	-0.147177
1	사랑일뿐야	2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	...	0.614079	0.018795	0.056959	0.026999	0.319842	0.031030	0.014634	0.263643	0.470032	-0.225170
2	유리창엔비	4	4.0	3.0	3.0	3.0	3.0	2.0	2.0	2.0	...	0.537809	0.026070	-0.052163	0.022258	0.431383	0.148869	0.142811	0.423212	0.513870	-0.151350
3	비오는날수채화	6	4.0	4.0	4.0	4.0	2.0	2.0	2.0	2.0	...	0.437628	0.059382	0.032185	0.000137	0.461525	0.085306	0.062933	0.335889	0.414543	-0.171855
4	마지막콘서트	5	4.0	2.0	2.0	2.0	2.0	2.0	1.0	1.0	...	0.492598	0.229535	0.061022	0.026889	0.519262	0.221837	0.114534	0.310134	0.466154	-0.221947

5 rows × 71 columns

5. 데이터 모델링

- 만든 Feature 데이터에 대한 y 값을 Top10 안에 들면 1, 들지 않으면 0으로 지정

- LGBM, RandomForest, ExtraTree 모델을 위주로 모델링 (Bayesian 튜닝)

- 평가 지표로는 roc_auc_score를 활용 (전부 0으로 찍어도 0.5가 나오도록)

- 튜닝 결과 가장 높은 점수는 LGBM (0.5956937799043063)

```
from sklearn.model_selection import cross_val_score
print('#### 만들어진 피처의 결과는....? ####')
cv_score1 = cross_val_score(lgbm, X_dev, y_dev, cv=skf, scoring = 'roc_auc')
print('lgbm : ',cv_score1)
print('평균 : ',np.mean(cv_score1))
cv_score2 = cross_val_score(rf, X_dev, y_dev, cv=skf, scoring = 'roc_auc')
print('rf : ',cv_score2)
print('평균 : ',np.mean(cv_score2))
cv_score3 = cross_val_score(lr, X_dev, y_dev, cv=skf, scoring = 'roc_auc')
print('lr : ',cv_score3)
print('평균 : ',np.mean(cv_score3))
cv_score4 = cross_val_score(knn, X_dev, y_dev, cv=skf, scoring = 'roc_auc')
print('knn : ',cv_score4)
print('평균 : ',np.mean(cv_score4))
cv_score5 = cross_val_score(et, X_dev, y_dev, cv=skf, scoring = 'roc_auc')
print('et : ',cv_score5)
print('평균 : ',np.mean(cv_score5))
```

```
#### 만들어진 피처의 결과는....? ####
lgbm :  [0.55523256 0.50857143 0.47904762 0.53670635]
평균 :  0.5198889488741233
rf :  [0.48885659 0.50095238 0.50333333 0.51140873]
평균 :  0.5011377583979328
lr :  [0.58042636 0.48       0.43238095 0.3452381 ]
평균 :  0.45951135105204877
knn :  [0.42005814 0.50904762 0.52428571 0.60218254]
평균 :  0.5138935031376891
et :  [0.50145349 0.43809524 0.48190476 0.59275794]
평균 :  0.50355285622000073
```

```
# lgbm
pred_lgbm = pd.DataFrame(lgb_tune.predict_proba(test)[:,1])
pred_lgbm.columns = ['Top10']
```

```
roc_auc_score(y_test,pred_lgbm)
```

```
0.5956937799043063
```

6. 결론

- 각 년도 별 Top 30 노래 안에서만 Top 10 노래를 구분해내는 모델링을 수행했기 때문에 예측이 쉽지 않음
(토픽 모델링 결과를 살펴보면 적절한 토픽 분류 군집의 수가 12개로 많은 곡에도 불구하고 비슷한 단어들이 많음을 알 수 있다)

- 상위권에 위치한 노래 가사들에서 사용된 단어들이 비슷하다보니 빈도 분석, TF-IDF 분석을 통해 Feature를 생성할 때 비슷한 값을 가진 Column들이 많이 만들어져 Top 10을 구분하는데 방해가 됨

- 따라서 Top30이 아닌 더 많은 데이터를 가지고 Train, Test 셋을 만들어서 모델링을 진행할 경우 더 높은 score를 기대할 수 있음

- 또한 방대한 양의 가사데이터를 모아 딥러닝을 활용한 RNN, LSTM 등의 모델을 활용하면 훨씬 더 높은 score가 기대된다


Thank
You!