2306aml133-kritika-assignment15-1

August 30, 2023

[5]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[6]:
```
import numpy as np
import pandas as pd

import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
import matplotlib.pyplot as plt
from wordcloud import WordCloud

from collections import defaultdict
from scipy.spatial.distance import cdist
from sklearn.preprocessing import MinMaxScaler, StandardScaler

import warnings
warnings.filterwarnings("ignore")

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

[35]:
```
data = pd.read_csv("/content/data.csv")
genre_data = pd.read_csv('/content/data_by_genres.csv')
year_data = pd.read_csv('/content/data_by_year.csv')
artist_data = pd.read_csv('/content/data_by_artist.csv')
```

[45]:
```
def get_song_data(name, data):
    try:
        return data[data['name'].str.lower() == name].iloc[0]
        return song_data
    except IndexError:
        return None
```

[46]:
```
def get_mean_vector(song_list, data):
    song_vectors = []
    for song in song_list:
        song_data = get_song_data(song['name'], data)
        if song_data is None:
            print('Warning: {} does not exist in the dataset'.
 ↪format(song['name']))
            return None
        song_vector = song_data[number_cols].values
        song_vectors.append(song_vector)
    song_matrix = np.array(list(song_vectors))
    return np.mean(song_matrix, axis=0)
```

[40]:
```
def flatten_dict_list(dict_list):
    flattened_dict = defaultdict()
    for key in dict_list[0].keys():
        flattened_dict[key] = []
    for dictionary in dict_list:
        for key, value in dictionary.items():
            flattened_dict[key].append(value)
    return flattened_dict
```

[41]:
```
min_max_scaler = MinMaxScaler()
normalized_data = min_max_scaler.fit_transform(data[number_cols])

standard_scaler = StandardScaler()
scaled_normalized_data = standard_scaler.fit_transform(normalized_data)
```

[42]:
```
def recommend_songs(seed_songs, data, n_recommendations=10):
    metadata_cols = ['name', 'artists', 'year']
    song_center = get_mean_vector(seed_songs, data)

    if song_center is None:
        return []

    normalized_song_center = min_max_scaler.transform([song_center])

    scaled_normalized_song_center = standard_scaler.
 ↪transform(normalized_song_center)

    distances = cdist(scaled_normalized_song_center, scaled_normalized_data,␣
 ↪'euclidean')
    index = np.argsort(distances)[0]

    rec_songs = []
    for i in index:
        song_name = data.iloc[i]['name']
```

```
        if song_name not in [song['name'] for song in seed_songs] and song_name␣
 ↪not in [song['name'] for song in rec_songs]:
            rec_songs.append(data.iloc[i])
            if len(rec_songs) == n_recommendations:
                break

    return pd.DataFrame(rec_songs)[metadata_cols].to_dict(orient='records')
```

[43]:
```
seed_songs = [
    {'name': 'Paranoid'},
    {'name': 'Blinding Lights'},
]
seed_songs = [{'name': name['name'].lower()} for name in seed_songs]

n_recommendations = 15

recommended_songs = recommend_songs(seed_songs, data, n_recommendations)

recommended_df = pd.DataFrame(recommended_songs)

for idx, song in enumerate(recommended_songs, start=1):
    print(f"{idx}. {song['name']} by {song['artists']} ({song['year']})")
```

```
1. Infinity by ['One Direction'] (2015)
2. Secrets by ['OneRepublic'] (2009)
3. In My Blood by ['Shawn Mendes'] (2018)
4. Head Above Water by ['Avril Lavigne'] (2019)
5. Green Light by ['Lorde'] (2017)
6. My Wish by ['Rascal Flatts'] (2006)
7. Magic Shop by ['BTS'] (2018)
8. Good Things Fall Apart (with Jon Bellion) by ['ILLENIUM', 'Jon Bellion']
(2019)
9. Inside Out (feat. Griff) by ['Zedd', 'Griff'] (2020)
10. A.M. by ['One Direction'] (2015)
11. Love You Goodbye by ['One Direction'] (2015)
12. Story of My Life by ['One Direction'] (2013)
13. Perfect by ['Simple Plan'] (2018)
14. arms by ['Christina Perri'] (2011)
15. Breezeblocks by ['alt-J'] (2012)
```

[44]:
```
recommended_df['text'] = recommended_df.apply(lambda row: f"{row.name + 1}.␣
 ↪{row['name']} by {row['artists']} ({row['year']})", axis=1)
fig = px.bar(recommended_df, y='name', x=range(n_recommendations, 0, -1),␣
 ↪title='Recommended Songs', orientation='h', color='name', text='text')
fig.update_layout(xaxis_title='Recommendation Rank', yaxis_title='Songs',␣
 ↪showlegend=False, uniformtext_minsize=20, uniformtext_mode='show',␣
 ↪yaxis_showticklabels=False, height=1000)
```

```
fig.update_traces(width=1)
fig.show()
```