HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

NHẬP MÔN KHOA HỌC DỮ LIỆU

Đề tài: Xây dựng hệ thống khuyến nghị âm nhạc

Thành viên nhóm

Hà Nội, 2022

https://towardsdatascience.com/part-iii-building-a-song-recommendation-system-with-spotify-cf76b52705e7

Phần 1 : Giới thiệu về hệ tư vấn

https://viblo.asia/p/introduction-to-recommender-systems-aWj53LQ8K6m

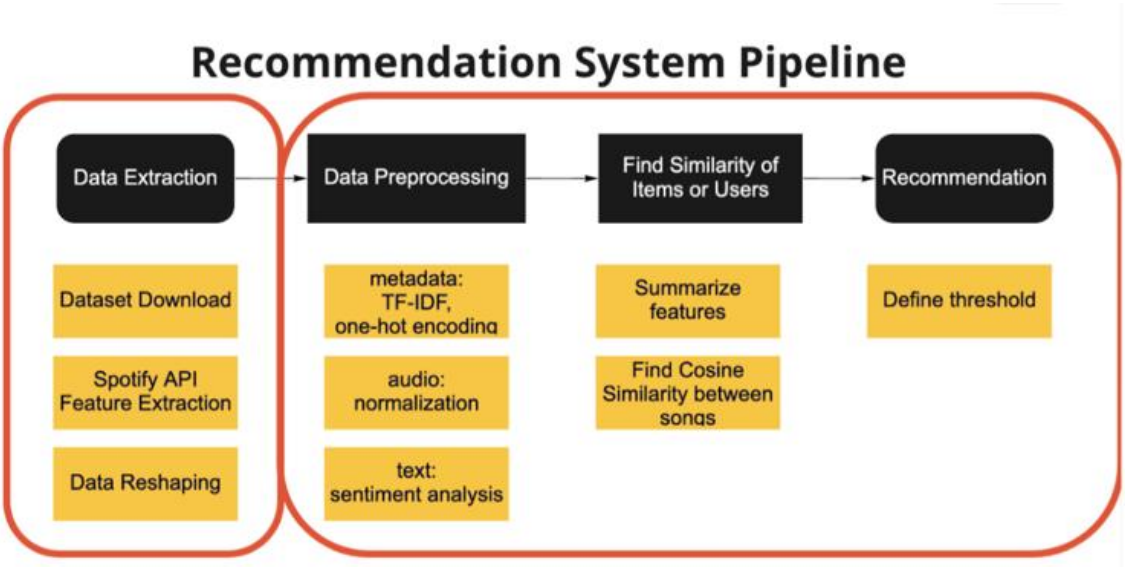
1. Khái niệm

Hệ tư vấn (recommender system) là một dạng công cụ lọc thông tin (information filtering) cho phép suy diễn, dự đoán các sản phẩm, dịch vụ, nội dung mà người dùng có thể quan tâm dựa trên những thông tin thu thập được về người dùng, về các sản phẩm, dịch vụ, về các hoạt động, tương tác cũng như đánh giá của người dùng đối với các sản phẩm, dịch vụ trong quá khứ.

2.Các cách tiếp cận

1. Tiếp cận dựa trên luật hay tri thức (rule/knowledge-based)
2. Tiếp cận dựa trên lọc cộng tác (collabrative - CF)
3. Tiếp cận dựa trên nội dung (content-based)
4. Tiếp cận dựa trên sự kết hợp giữa CF và content-based (hybrid)

3.Pipeline bài toán



Phần 2**: Trích xuất dữ liệu bài hát từ API của Spotify bằng Python**

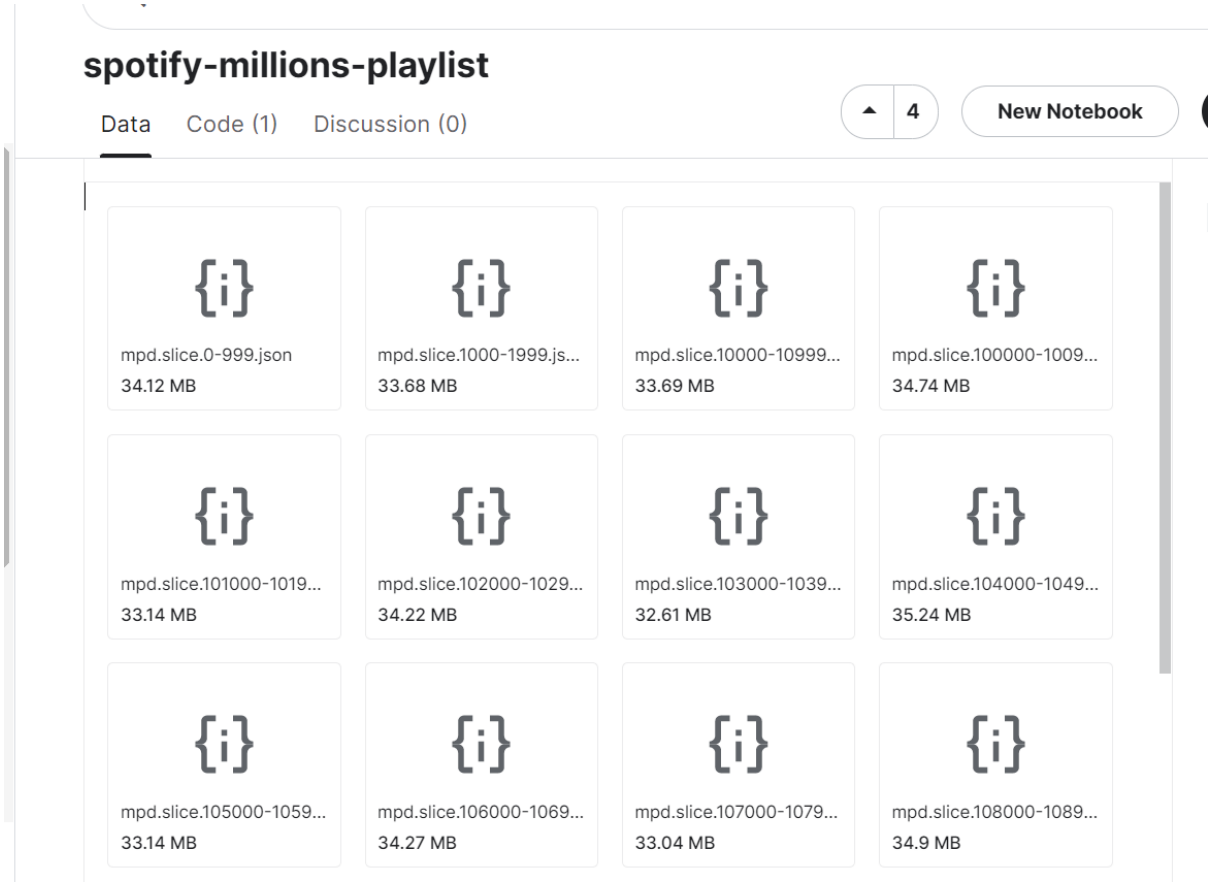
1.Giới thiệu về API Spotify

Spotify lưu giữ nhiều dữ liệu nội bộ và cho phép chúng ta truy cập thông qua API của họ. Điều này cực kỳ hữu ích khi chúng ta muốn sử dụng dữ liệu của chính mình để xây dựng bộ dữ liệu để phân tích. Trong bộ dữ liệu hàng triệu danh sách phát , sẽ cực kỳ hữu ích khi có thể trích xuất các tính năng về các bài hát có trong đó, nhờ đó chúng ta có thể hiểu rõ hơn về cách các bài hát liên quan với nhau và thực hiện phân cụm để xây dựng công cụ đề xuất của riêng chúng tôi.

2. Dataset:

Dataset gồm 1 triệu các bài hát được định dạng ở file json.

https://www.kaggle.com/datasets/adityak80/spotify-millions-playlist?fbclid=IwAR2Hw7xO8VWZOlKDVhwD6e6B8KOtxMVGK69b4FhzynX_VzLWymThxvo6O5k



data > {} First_1000.json


```
{
    "info": {
        "generated_on": "2017-12-03 08:41:42.057563",
        "slice": "0-999",
        "version": "v1"
    },
    "playlists": [
        {
            "name": "Throwbacks",
            "collaborative": "false",
            "pid": 0,
            "modified_at": 1493424000,
            "num_tracks": 52,
            "num_albums": 47,
            "num_followers": 1,
            "tracks": [
                {
                    "pos": 0,
                    "artist_name": "Missy Elliott",
                    "track_uri": "spotify:track:0UaMYEvWZi0ZqiDOoHU3YI",
                    "artist_uri": "spotify:artist:2wIVse2owClT7go1WT98tk",
                    "track_name": "Lose Control (feat. Ciara & Fat Man Scoop)",
                    "album_uri": "spotify:album:6vV5UrXcfyQD1wu4Qo2I9K",
                    "duration_ms": 226863,
                    "album_name": "The Cookbook"
                },
                {
                    "pos": 1,
                    "artist_name": "Britney Spears",
                    "track_uri": "spotify:track:6I9VzXrHxO9rA9A5euc8Ak",
                    "artist_uri": "spotify:artist:26dSoYclwsYLMAKD3tpOr4",
                    "track_name": "Toxic",
```

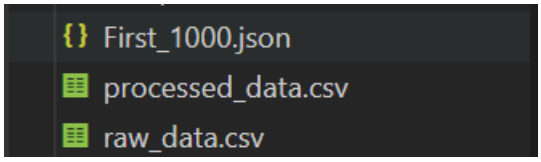
Ta sẽ chuyển sang csv :

convert_to_csv.py

process_data > convert_to_csv.py > ...

```python
from pprint import pprint
import json
import pandas as pd

data_path = "../data/First_1000.json"
raw_json = json.loads(open(data_path).read())

playlists = raw_json["playlists"]
df = pd.json_normalize(playlists, record_path='tracks', meta=['name'])

df.to_csv("../data/raw_data.csv")
```

Và có được kết quả là file processed_data.csv



3. Xử lý dữ liệu sau khi dữ liệu thô được trích xuất.

Thủ tục lựa chọn dữ liệu bao gồm hai nhiệm vụ. Đầu tiên là loại bỏ các bài hát trùng lặp. Vì dữ liệu được nhập ban đầu là dữ liệu danh sách phát Spotify, nên điều quan trọng là phải xóa các bài hát trùng lặp tồn tại giữa nhiều danh sách phát. Quá trình này bao gồm việc thu thập tên nghệ sĩ và tên bài hát để chúng ta không vô tình xóa các bài hát có cùng tên nhưng của các nghệ sĩ khác nhau.

```python
# Select useful columns
def select_cols(df):
    '''
    Select useful columns
    '''
    return df[['artist_name','id','track_name','danceability', 'energy', 'key', 'loudness', 'mode',
    'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', "artist_pop", "genres", "track_pop"]]
songDF = select_cols(songDF)
songDF.head()
```

[53] Python

	artist_name	id	track_name	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	artist_
0	Missy Elliott	0UaMYEvWZi0ZqiDOoHU3YI	Lose Control (feat. Ciara & Fat Man Scoop)	0.904	0.813	4	-7.105	0	0.1210	0.03110	0.006970	0.0471	0.810	125.461	
6	Britney Spears	6l9VzXrHxO9rA9A5euc8Ak	Toxic	0.774	0.838	5	-3.914	0	0.1140	0.02490	0.025000	0.2420	0.924	143.040	
19	Beyoncé	0WqIKmW4BTrj3eJFmnCKMv	Crazy In Love	0.664	0.758	2	-6.583	0	0.2100	0.00238	0.000000	0.0598	0.701	99.259	
46	Justin Timberlake	1AWQoqb9bSvzTjaLralEkT	Rock Your Body	0.892	0.714	4	-6.055	0	0.1410	0.20100	0.000234	0.0521	0.817	100.972	
55	Shaggy	1lzr43nnXAijIGYnCT8M8H	It Wasn't Me	0.853	0.606	0	-4.596	1	0.0713	0.05610	0.000000	0.3130	0.654	94.759	

Sau khi chọn dữ liệu hữu ích, do định dạng nhập của khung dữ liệu, chúng ta cần chuyển các genres cột trở lại thành danh sách.

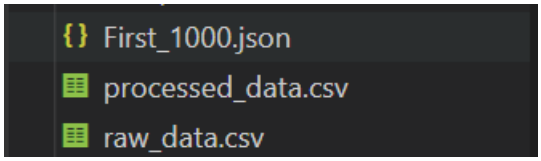
	id	track_name	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	artist_pop	genres	track_pop
	:i0ZqiDOoHU3YI	Lose Control (feat. Ciara & Fat Man Scoop)	0.904	0.813	4	-7.105	0	0.1210	0.03110	0.006970	0.0471	0.810	125.461	74	dance_pop hip_hop hip_pop pop pop_rap r&b rap ...	69
	O9rA9A5euc8Ak	Toxic	0.774	0.838	5	-3.914	0	0.1140	0.02490	0.025000	0.2420	0.924	143.040	84	dance_pop pop post-teen_pop	83
	3Trj3eJFmnCKMv	Crazy In Love	0.664	0.758	2	-6.583	0	0.2100	0.00238	0.000000	0.0598	0.701	99.259	86	dance_pop pop r&b	25
	9bSvzTjaLralEkT	Rock Your Body	0.892	0.714	4	-6.055	0	0.1410	0.20100	0.000234	0.0521	0.817	100.972	82	dance_pop pop	79
	AijIGYnCT8M8H	It Wasn't Me	0.853	0.606	0	-4.596	1	0.0713	0.05610	0.000000	0.3130	0.654	94.759	75	pop_rap reggae_fusion	2

```python
def genre_preprocess(df):
    df['genres_list'] = df['genres'].apply(lambda x: x.split(" "))
    return df
songDF = genre_preprocess(songDF)
songDF['genres_list'].head()
```

54]

```
0     [dance_pop, hip_hop, hip_pop, pop, pop_rap, r&...
6                    [dance_pop, pop, post-teen_pop]
19                          [dance_pop, pop, r&b]
46                               [dance_pop, pop]
55                        [pop_rap, reggae_fusion]
Name: genres_list, dtype: object
```

⇨ Kết quả cuối cùng ta được file raw_data.csv:



Phần 3 **: Xử lý và trích đặc trưng trên tập dữ liệu**

Chúng ta có thể phân loại các biến trong dữ liệu thành ba loại dựa trên nguồn dữ liệu, đó là metadata, dữ liệu âm thanh và dữ liệu văn bản.

1. Đặc trưng metadata

-Siêu dữ liệu đề cập đến các thuộc tính liên quan đến bài hát nhưng không liên quan đến chính bài hát (ví dụ: mức độ phổ biến và thể loại)

-Các thể loại trong Spotify không được phân bổ cân bằng với một số thể loại thịnh hành hơn trong khi những thể loại khác lại khó hiểu hơn. Ngoài ra, một nghệ sĩ hoặc bản nhạc có thể được liên kết với nhiều thể loại. Do đó, chúng ta cần cân nhắc tầm quan trọng của từng thể loại để chống lại việc đặt nặng các thể loại cụ thể trong khi đánh giá thấp những thể loại khác.

-Do đó, **các biện pháp TF-IDF** được giới thiệu và áp dụng cho dữ liệu thể loại. TF-IDF (thuật ngữ tần suất nghịch đảo tần suất tài liệu) có thể được coi là một thước đo số phản ánh tầm quan trọng của một từ trong một tập hợp ngữ liệu. Các từ thường xuyên xuất hiện trong một tài liệu nhưng không xuất hiện trong các tài liệu có xu hướng có điểm TF-IDF cao.

https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0

https://thanhvie.com/tf-idf-va-cosine-similarity/

Term Frequency × Inverse Document Frequency

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

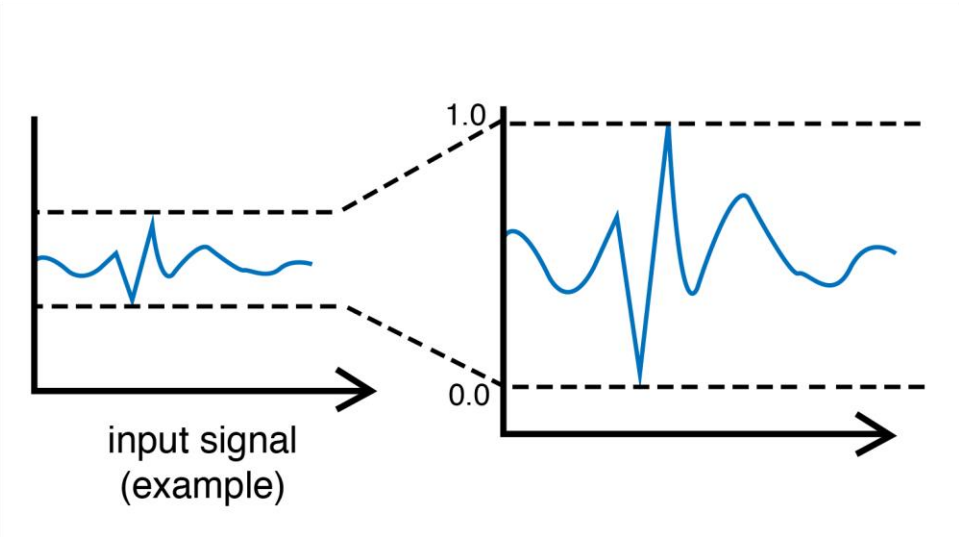
$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

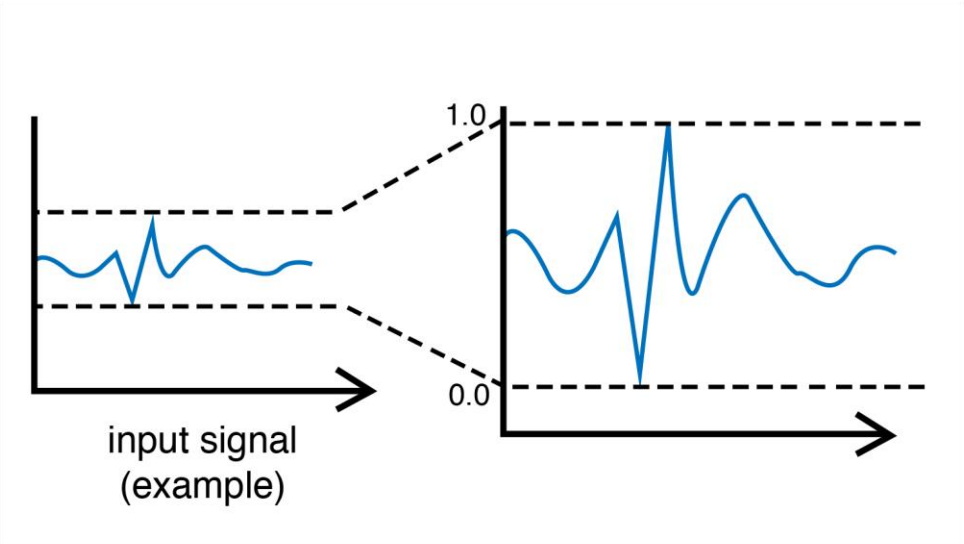
Trong dự án này, các tài liệu tương tự như các bài hát. Do đó, chúng ta cần tính toán thể loại nổi bật nhất trong mỗi bài hát và mức độ phổ biến của chúng trong các bài hát để xác định mức độ quan trọng của thể loại đó. Điều này tốt hơn nhiều so với mã hóa đơn thuần vì không có trọng số để xác định mức độ quan trọng và phổ biến của mỗi thể loại, dẫn đến việc đặt trọng số quá mức vào các thể loại không phổ biến.

Đối với dữ liệu về mức độ phổ biến, chúng ta sử dụng được coi là một biến liên tục và chỉ **chuẩn hóa** nó thành một phạm vi từ 0 đến 1. Ý tưởng là những bài hát phổ biến có khả năng được nghe bởi những người thích bài hát nổi tiếng, trong khi những bài hát ít phổ biến hơn thì có khả năng để được lắng nghe bởi những người có cùng sở thích.
Điều này được thực hiện thông qua hàm ***MinMaxScaler ()*** trong ***scikit learning***

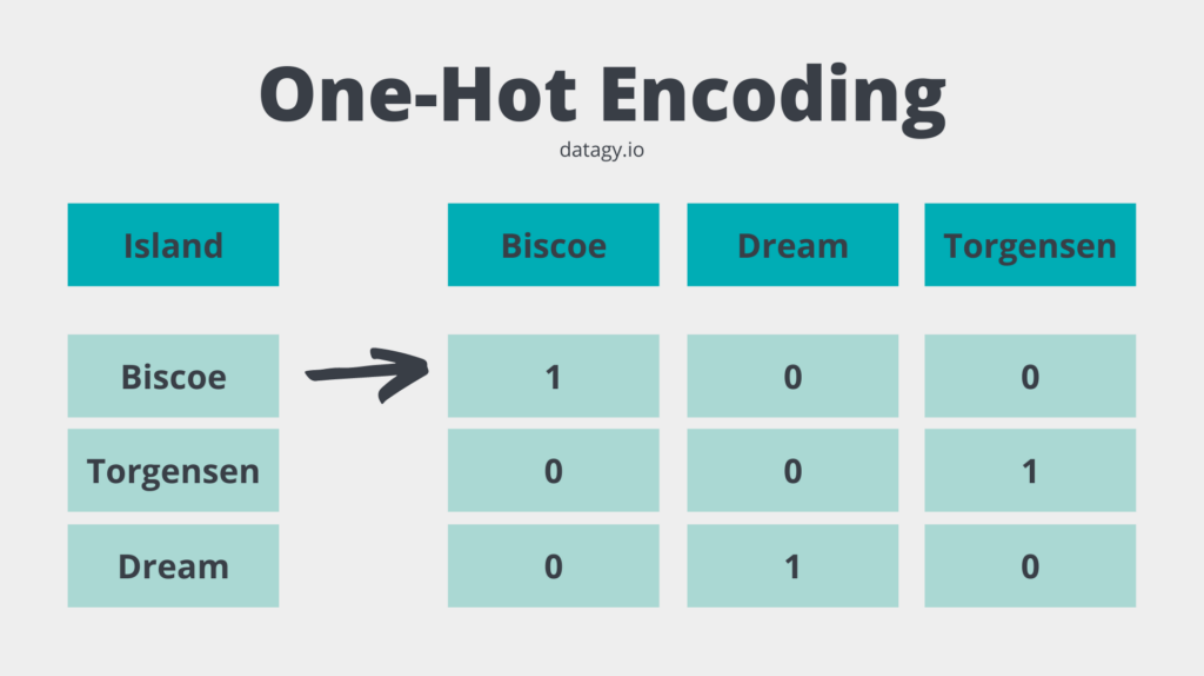


2. Đặc trưng âm thanh

-Dữ liệu âm thanh đề cập đến các tính năng âm thanh của bài hát được trích xuất bằng API Spotify. Ví dụ: độ lớn, nhịp độ, năng lượng, giọng nói, âm thanh, nhạc cụ, độ sống động và thời lượng,…. Trong dự án này, thao tác duy nhất mà ta phải tiến hành đối với những dữ liệu này là chuẩn hóa dựa trên các giá trị tối đa và nhỏ nhất của mỗi biến.



-Ngoài ra, mã hóa one-hot đã được tiến hành trên một số tính năng âm thanh khác, chẳng hạn như *khóa key* của bản nhạc. Bằng cách giả sử mọi khóa đều có trọng số như nhau, ít có khả năng có được sự biểu diễn dữ liệu tốt nhất về mặt toán học. Do đó, có thể cần điều chỉnh siêu tham số để cải thiện dự đoán.

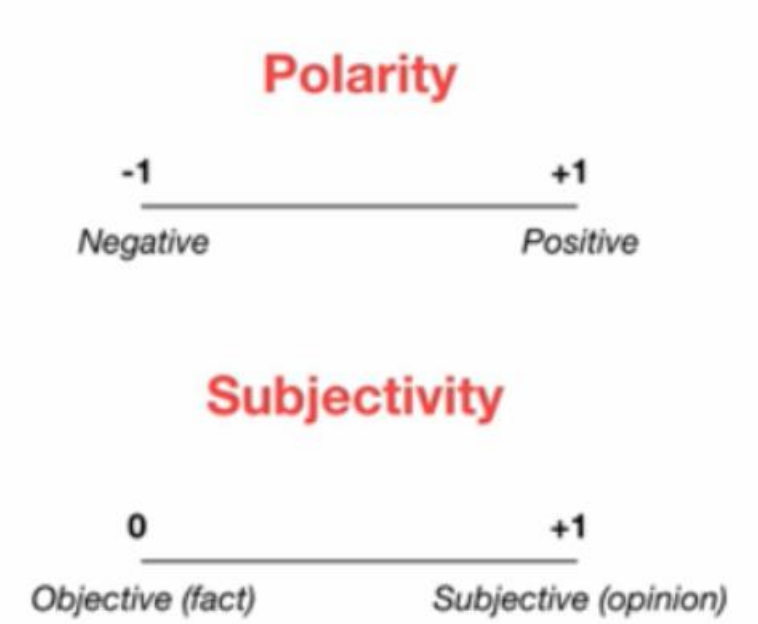


3. Đặc trưng về Chữ

Tính năng văn bản duy nhất mà đề tài đã sử dụng là tên bản nhạc. Nên chúng ta phải tiến hành **phân tích tình cảm** để tìm ra tính phân *cực* và *tính chủ quan* của tên bản nhạc.

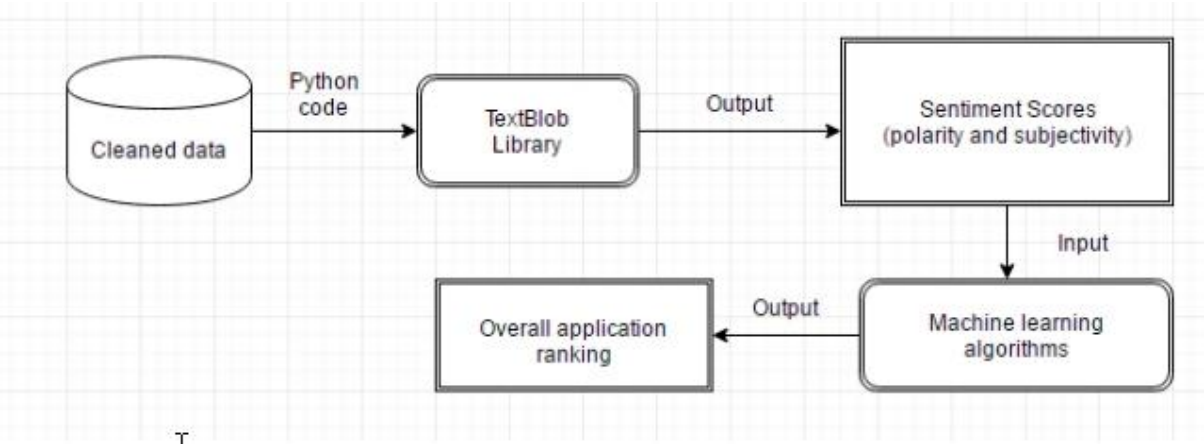
https://medium.com/analytics-vidhya/sentiment-analysis-on-ellens-degeneres-tweets-using-textblob-ff525ea7c30f

+ Tính phân cực quyết định tình cảm của văn bản. Giá trị của nó nằm ở [-1,1] trong đó -1 biểu thị tình cảm tiêu cực cao và 1 biểu thị cảm xúc tích cực cao.
+ Tính chủ quan xác định xem đầu vào văn bản là thông tin thực tế hay là ý kiến cá nhân. Giá trị của nó nằm giữa [0,1] trong đó giá trị gần 0 biểu thị một phần thông tin thực tế và giá trị gần 1 biểu thị ý kiến cá nhân.



-Mục tiêu của phân tích tình cảm là trích xuất các tính năng bổ sung từ các bản nhạc. Bằng cách đó, chúng tôi có thể trích xuất dữ liệu tình cảm các tính năng âm thanh khác thông qua thông tin văn bản. Ví dụ: nếu tâm trạng chung của tên bài hát trong danh sách phát là tích cực, thì điều này có thể được sử dụng để đề xuất các bài hát tích cực. Tuy nhiên, do độ dài ngắn của tiêu đề, hai thước đo không thể tạo ra kết quả tối ưu. Do đó, trọng số của hai chỉ số được đánh giá là thấp.

Để tiến hành phân tích tình cảm, ***TextBlob.sentiment*** đã được sử dụng:



=> Tổng hợp

Summarization of all songs features in a playlist

	Date	Song Genre					Song Year/Popularity					Liveness, Energy, etc			
Song 1	7/17/2019	0	0.1	0.3	0	0	0	0.1	0.3	0	0	0.5	0.2	0.8	
Song 2	7/02/2019	0	0	0	0.8	0	0	0	0	0.8	0	0.7	0.23	0.3	
Song 3	5/13/2019	0.9	0	0	0.7	0	0.9	0	0	0.7	0	0.6	0.1	0.8	
Song 4	2/12/2019	0.6	0	0	0.2	0	0.6	0	0	0.2	0	0.1	1.2	1.4	...
Song 5	11/23/2018	0.5	0.6	0.4	0	0	0.5	0.6	0.4	0	0	1.2	1.7	1.2	
Song 6	11/23/2018	0	0	0	0.9	0	0	0	0	0.9	0	1.4	1.7	1.2	
Song 7	10/20/2018	0.2	0.1	0	0	0	0.2	0.1	0	0	0	1.5	1.5	1.9	
Song 8	8/03/2018	0	0	0	0.1	0	0	0	0	0.1	0	1.2	3.2	1.7	
Final Playlist Vector		2.2	0.8	0.7	2.7	0.0	2.2	0.8	0.7	2.7	0.0	...	7.2	11.63	9.3

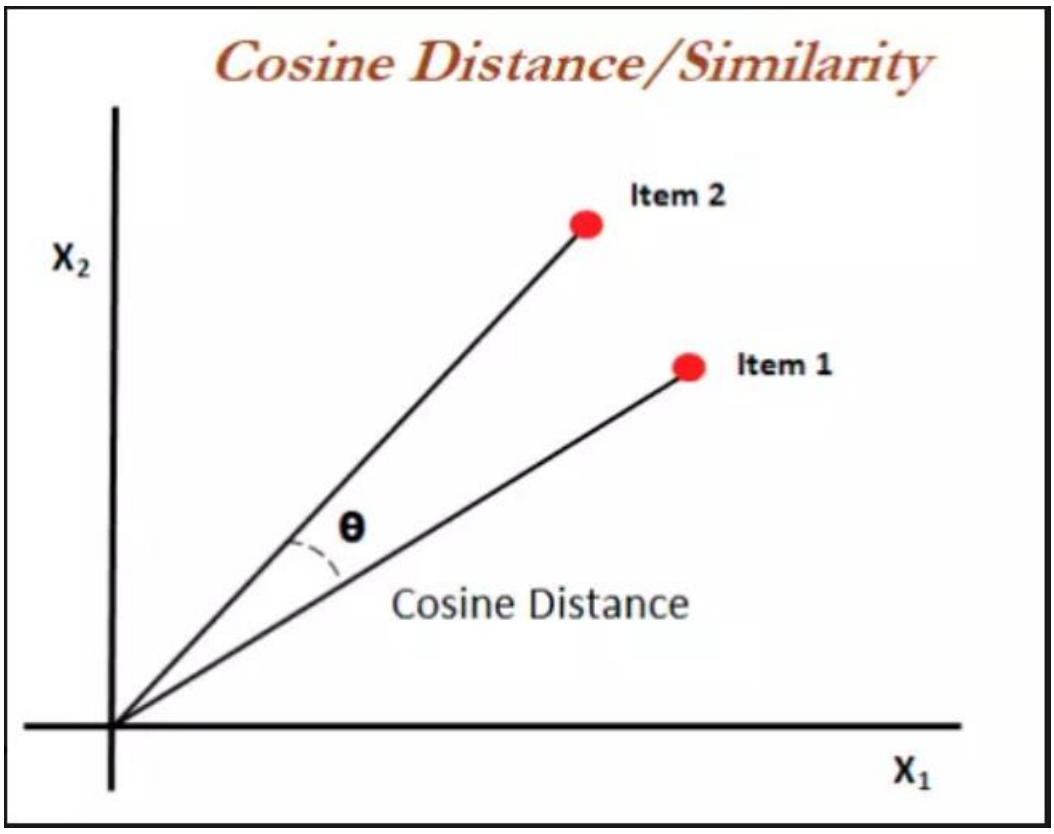
Cuối cùng, chúng tôi thêm tất cả các giá trị tính năng của từng bài hát trong danh sách phát lại với nhau dưới dạng vectơ tóm tắt.

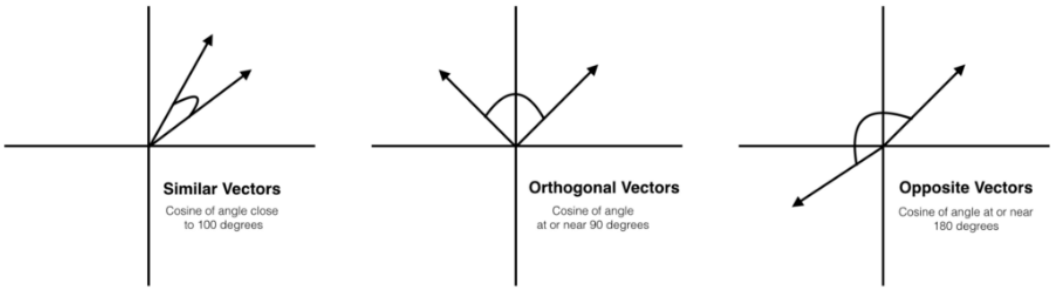
```python
def create_feature_set(df, float_cols):
    # Tfidf genre lists
    tfidf = TfidfVectorizer()
    tfidf_matrix =  tfidf.fit_transform(df['genres_list'].apply(lambda x: " ".join(x)))
    genre_df = pd.DataFrame(tfidf_matrix.toarray())
    genre_df.columns = ['genre' + "|" + i for i in tfidf.get_feature_names()]
    genre_df.drop(columns='genre|unknown') # drop unknown genre
    genre_df.reset_index(drop = True, inplace=True)

    # Sentiment analysis
    df = sentiment_analysis(df, "track_name")

    # One-hot Encoding
    subject_ohe = ohe_prep(df, 'subjectivity','subject') * 0.3
    polar_ohe = ohe_prep(df, 'polarity','polar') * 0.5
    key_ohe = ohe_prep(df, 'key','key') * 0.5
    mode_ohe = ohe_prep(df, 'mode','mode') * 0.5

    # Normalization
    # Scale popularity columns
    pop = df[["artist_pop","track_pop"]].reset_index(drop = True)
    scaler = MinMaxScaler()
    pop_scaled = pd.DataFrame(scaler.fit_transform(pop), columns = pop.columns) * 0.2

    # Scale audio columns
    floats = df[float_cols].reset_index(drop = True)
    scaler = MinMaxScaler()
    floats_scaled = pd.DataFrame(scaler.fit_transform(floats), columns = floats.columns) * 0.2

    # Concanenate all features
    final = pd.concat([genre_df, floats_scaled, pop_scaled, subject_ohe, polar_ohe, key_ohe, mode_ohe], axis = 1)

    # Add song id
    final['id']=df['id'].values

    return final
```

Phần II **: Xây dựng hệ thống đề xuất bài hát với Spotify**

-Sau khi truy xuất vectơ tóm tắt danh sách phát và các bài hát không thuộc danh sách phát, chúng ta có thể tìm thấy điểm giống nhau giữa từng bài hát riêng lẻ trong cơ sở dữ liệu và danh sách phát. Số liệu độ tương tự được chọn là **độ tương tự cosin** .
https://viblo.asia/p/chuc-nang-tu-dong-suggest-tag-su-dung-cosine-similarity-YWOZrpRR5Q0

Độ tương tự cosine là một giá trị toán học đo lường sự giống nhau giữa các vectơ. Tưởng tượng các vectơ bài hát của chúng ta chỉ là hai chiều, hình ảnh đại diện sẽ tương tự như hình bên dưới.





Một khi hai vectơ nói chung hướng về cùng một hướng, thì chúng tương tự nhau. Đây cũng là lý do tại sao ta không tìm thấy ý nghĩa của các bài hát mà chỉ đơn giản là thêm chúng vào. Trong tình huống này, các vectơ của bài hát là siêu chiều nên chúng ta không thể minh họa nó một cách độc đáo bằng đồ thị. Tuy nhiên, trực giác toán học vẫn vậy.
Về mặt hình thức, công thức toán học có thể được biểu thị như sau:

$$\text{Cosine Sim}(A, B) = \frac{A \cdot B}{||A|| \times ||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Ta có thể thực hiện code bằng cách gọi hàm cosine_similarity() từ scikit learn để đo mức độ giống nhau giữa mỗi bài hát và vectơ danh sách phát tóm tắt.

```python
from sklearn.metrics.pairwise import cosine_similarity


def generate_playlist_recos(df, features, nonplaylist_features):
    non_playlist_df = df[df['id'].isin(nonplaylist_features['id'].values)]
    non_playlist_df['sim'] = cosine_similarity(nonplaylist_features.drop('id', axis = 1).values, features.values.reshape(1, -1
    non_playlist_df_top_40 = non_playlist_df.sort_values('sim',ascending = False).head(40)

    return non_playlist_df_top_40
```

Python

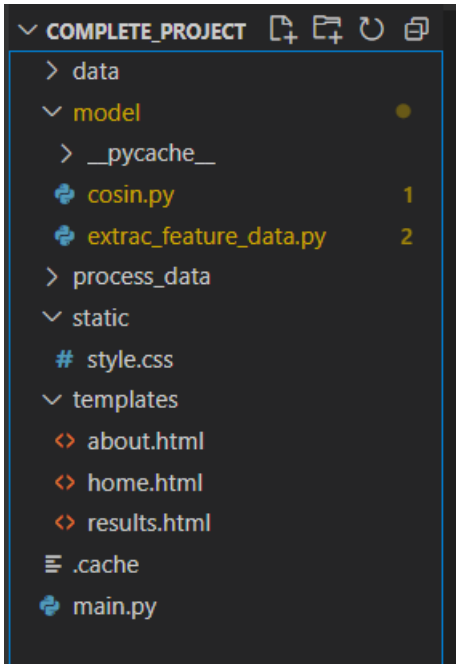
```python
recommend = generate_playlist_recos(songDF, complete_feature_set_playlist_vector, complete_feature_set_nonplaylist)
recommend.head(10)
```

Python

Phần III : **Triển khai Mô hình Đề xuất Spotify với Flask**

https://towardsdatascience.com/deploying-a-spotify-recommendation-model-with-flask-20007b76a20f

1.Cấu trúc thư mục



Để chạy ứng dụng đề xuất của chúng tôi cục bộ, chúng ta có thể chạy '$ python main.py' trong thư mục gốc - điều này sẽ đảm bảo rằng máy chủ đang ở chế độ phát triển và chúng tôi có thể gỡ lỗi và làm mới nó trong khi thực hiện bất kỳ thay đổi nào.

2. Test web với trường hợp đưa URL danh sách phát

Code:

```python
def extract(URL):
    client_id = "5356afb958c84e71a2c37c43e2a2cbf2"
    client_secret = "83e531491e9c458ba658ac30c4c56bc0"

    client_credentials_manager = SpotifyClientCredentials(client_id=client_id,client_secret=client_secret)
    sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

    playlist_id = URL.split("/")[4].split("?")[0]
    playlist_tracks_data = sp.playlist_tracks(playlist_id)

    playlist_tracks_id = []
    playlist_tracks_titles = []
    playlist_tracks_artists = []
    playlist_tracks_first_artists = []

    for track in playlist_tracks_data['items']:
        playlist_tracks_id.append(track['track']['id'])
        playlist_tracks_titles.append(track['track']['name'])
        artist_list = []
        for artist in track['track']['artists']:
            artist_list.append(artist['name'])
        playlist_tracks_artists.append(artist_list)
        playlist_tracks_first_artists.append(artist_list[0])

    features = sp.audio_features(playlist_tracks_id)
    features_df = pd.DataFrame(data=features, columns=features[0].keys())
    features_df['title'] = playlist_tracks_titles
    features_df['first_artist'] = playlist_tracks_first_artists
    features_df['all_artists'] = playlist_tracks_artists
    features_df = features_df[['id', 'title', 'first_artist', 'all_artists',
                               'danceability', 'energy', 'key', 'loudness',
                               'mode', 'acousticness', 'instrumentalness',
                               'liveness', 'valence', 'tempo',
                               'duration_ms', 'time_signature']]

    return features_df
```

```python
def generate_playlist_feature(complete_feature_set, playlist_df):
    complete_feature_set_playlist = complete_feature_set[complete_feature_set['id'].isin(playlist_df['id'].values)]
    complete_feature_set_nonplaylist = complete_feature_set[~complete_feature_set['id'].isin(playlist_df['id'].values)]
    if complete_feature_set_playlist.empty:
        return None, complete_feature_set_nonplaylist
    else:
        complete_feature_set_playlist_final = complete_feature_set_playlist.drop(columns = "id")
        return complete_feature_set_playlist_final.sum(axis = 0), complete_feature_set_nonplaylist


def generate_playlist_recos(df, features, nonplaylist_features):
    non_playlist_df = df[df['id'].isin(nonplaylist_features['id'].values)]
    non_playlist_df['sim'] = cosine_similarity(nonplaylist_features.drop('id', axis = 1).values, features.values.reshape(1, -1))[:,0]
    non_playlist_df_top_40 = non_playlist_df.sort_values('sim',ascending = False).head(40)

    return non_playlist_df_top_40


def recommend_from_playlist(songDF,complete_feature_set,playlistDF_test):
    complete_feature_set_playlist_vector, complete_feature_set_nonplaylist = generate_playlist_feature(complete_feature_set, playlistDF_test)
    print(complete_feature_set_playlist_vector)
    if complete_feature_set_playlist_vector is None:
        return None
    else:
        top40 = generate_playlist_recos(songDF, complete_feature_set_playlist_vector, complete_feature_set_nonplaylist)
        return top40
```

3. Test web với trường hợp đưa URL 1 bài hát

Code :

```
def extract_song(URL):
    client_id = "5356afb958c84e71a2c37c43e2a2cbf2"
    client_secret = "83e531491e9c458ba658ac30c4c56bc0"
    client_credentials_manager = SpotifyClientCredentials(client_id=client_id,client_secret=client_secret)
    sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

    track_id=[]
    track_titles=[]
    track_artist=[]
    track_artist_first=[]

    track_titles=sp.track(URL)['name']
    artist=sp.track(URL)['artists']
    artist_list=[]
    for ar in artist:
        artist_list.append(ar['name'])
    track_artist.append(artist_list)
    track_artist_first.append(artist_list[0])

    features=sp.audio_features(URL)
    features_df = pd.DataFrame(data=features, columns=features[0].keys())
    features_df['title']=track_titles
    features_df['first_artist']=track_artist_first
    features_df['all_artists']=track_artist
    features_df = features_df[['id', 'title', 'first_artist', 'all_artists',
                               'danceability', 'energy', 'key', 'loudness',
                               'mode', 'acousticness', 'instrumentalness',
                               'liveness', 'valence', 'tempo',
                               'duration_ms', 'time_signature']]
    return features_df
```

```
def generate_playlist_feature(complete_feature_set, playlist_df):
    complete_feature_set_playlist = complete_feature_set[complete_feature_set['id'].isin(playlist_df['id'].values)]
    complete_feature_set_nonplaylist = complete_feature_set[~complete_feature_set['id'].isin(playlist_df['id'].values)]
    if complete_feature_set_playlist.empty:
        return None, complete_feature_set_nonplaylist
    else:
        complete_feature_set_playlist_final = complete_feature_set_playlist.drop(columns = "id")
        return complete_feature_set_playlist_final.sum(axis = 0), complete_feature_set_nonplaylist


def generate_playlist_recos(df, features, nonplaylist_features):
    non_playlist_df = df[df['id'].isin(nonplaylist_features['id'].values)]
    non_playlist_df['sim'] = cosine_similarity(nonplaylist_features.drop('id', axis = 1).values, features.values.reshape(1, -1))[:,0]
    non_playlist_df_top_40 = non_playlist_df.sort_values('sim',ascending = False).head(40)

    return non_playlist_df_top_40


def recommend_from_playlist(songDF,complete_feature_set,playlistDF_test):
    complete_feature_set_playlist_vector, complete_feature_set_nonplaylist = generate_playlist_feature(complete_feature_set, playlistDF_test)
    print(complete_feature_set_playlist_vector)
    if complete_feature_set_playlist_vector is None:
        return None
    else:
        top40 = generate_playlist_recos(songDF, complete_feature_set_playlist_vector, complete_feature_set_nonplaylist)
        return top40
```