



90 lines (53 loc) · 4.83 KB

Preview

Code

Blame

Raw



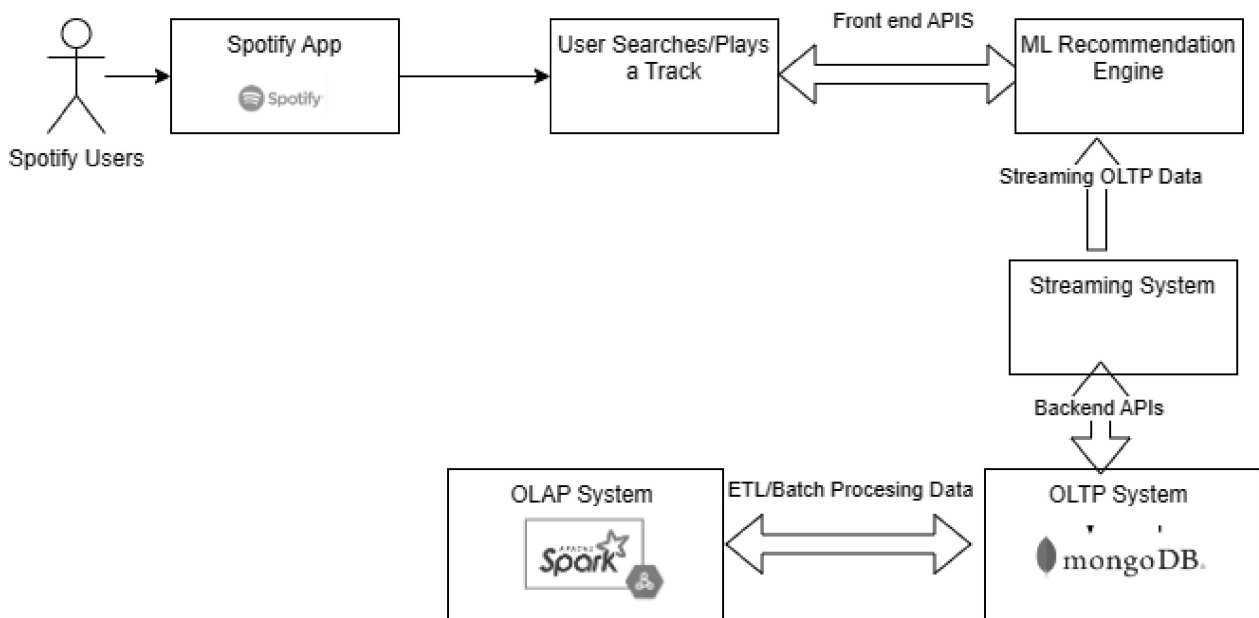
BITS_MTech_BDSAssignment_SpotifyRecommendation

Our project focuses on developing a Music Recommendation System using the Spotify Playlist Dataset. Users can search for songs by track name and artist name, select their preferred songs from the list, and receive relevant recommendations from the system.

Team Members

1. Bhaskar Kurada - 2022OG04023 - **Contribution:** Installing Spark on local, Connecting Sparkshell/Pyspark on local command line troubleshooting and the architecture diagram creation on draw.io, Github Creation
2. Mahesh Kammari - 2022OG04033 - **Contribution:** Creating MongoDB Cloud Instance, MongoDB and Pyspark connection troubleshooting, MongoDB Data Model and queries, Data Synthesization and loom video setup

Part1 - Architecture diagram of our system - BigData Recommendation Pipeline



Brief Overview of the System components and Rationale

Components Frontend APIs: APIs to connect the mobile App to the ML Recommendation

Engine and the underlying OLTP System **ML Recommendation Engine:** NLP based ML

Algorithm to find similar documents for recommendation basis the attributes in the

databset **Streaming Component:** Apache Kafka could be used here to stream the data from

OLTP databse related services **Backend APIs:** APIs to connect the MongoDB to the

streaming component **OLTP Sytem:** MongoDB to store the user details, playlists, search

history and recommendations provided by ML library **OLAP System:** SparkSQL to load the

master data as well as to run the analytics queries **Pyspark:** Utilized for preprocessing the

Spotify Dataset. **ETL/Batch processing jobs:** To collate the user search and playlist history to

dervie analytical queries for better sales and growth

Rationale for the design choices OLTP System: We have chosen the MongoDB which is a

document based NOSQL DB to store the search cum playlist history and also the associated

recommendations MongoDB natively provides Consistency and Performance tradeoffs with

scale and durability with various read and write choices

Considering this problem of recommendation only being a part of the overall spotify app,

consistency in the recommendations basis user playlist history and the search criteria takes

a notch high priority than the availability.

Availability could take precedence once the user accepts the recommendation and starts

playing the track which is not the scope of the project

OLTP Data Model:

Album MasterCollection Document			UserPlaylist Document	
Track_id			id	
Artists			Userid	
album_name			Trackid	
track_name				
popularity		User Document		
duration		id	UserPlayHistory Document	
explicit		name	id	
danceability			Userid	
energy			Trackid	
key				
loudness			UserSearchHistory Document	
mode			id	
speechiness			Userid	
acousticness			SearchItem	
instrumentalness				
liveness			UserPlayRecommendations Document	
valence			id	
tempo			playhistoryid	
time_signature			RecommendationStatus	
track_genre				
			UserSearchRecommendations Document	
			id	
			searchhistoryid	
			RecommendationStatus	

OLAP System: Spark provides a powerful and flexible framework that can integrate well with various NoSQL databases. like Mongo, Cassandra, Dynamo and also provides RDD Dataframes which can handle huge load of data from both batch processing and also running ML based analytics on them.

End of Part-1

Part-2 - OLTP Read/Write Queries on Mongo DB

Loom Video Link for Pymongo - Read/Write Queries

<https://www.loom.com/share/1398b11cc5c84cd6afd3c2fdc29be7df?sid=9abbf41c-17a6-4f98-9673-26840758af39>

Also Find Mongo Compass IDE & Pymongo Screenshots

The screenshot shows the MongoDB Compass interface. On the left, the 'Databases' panel lists the 'SpotifyMusicRecommDB' database and its collections: 'MasterAlbumInfo', 'UserInfo', 'UserPlayHistoryInfo', 'UserPlayRecommendationInfo', 'UserPlaylistInfo', 'UserSeachRecommendationInfo', and 'UserSearchHistoryInfo'. The main panel displays the 'SpotifyMusicRecommDB.MasterAlbumInfo' collection. The 'Documents' tab is active, showing a query with the following parameters:

- Filter:** {}
- Project:** {"track_name": 1, "popularity": 1}
- Sort:** {"popularity": -1}
- Collation:** {"locale": "simple"}
- Limit:** 10

The query is described as 'Find top 10 most popular tracks'. The 'EXPORT DATA' button is visible at the bottom. The first document in the result set is shown:

```
{
  "_id": ObjectId('658803cda5ceb73902f5f30'),
  "track_name": "Unholy (feat. Kim Petras)",
  "popularity": 100
}
```

End of Part-2

Part-3 - Pyspark & MongoDB Connection Setup & OLAP Queries

Loom Video Link for Spark - OLAP Queries

<https://www.loom.com/share/d26e9a49c8ca497788a5fc1d053a6fb6>

Commands to connect to Pyspark and connect to Mongo DB and then finally run the OLAP aggregate queries

```
pyspark --packages org.mongodb.spark:mongo-spark-connector_2.12:3.0.1
```

```
df = spark.read.format("mongo").option("uri",  
"mongodb+srv://kuradabhaskar:fiYR58qHxeMqle1P@cluster0.8jajr3e.mongodb.net/Spotify  
MusicRecommDB.MasterAlbumInfo").load()
```

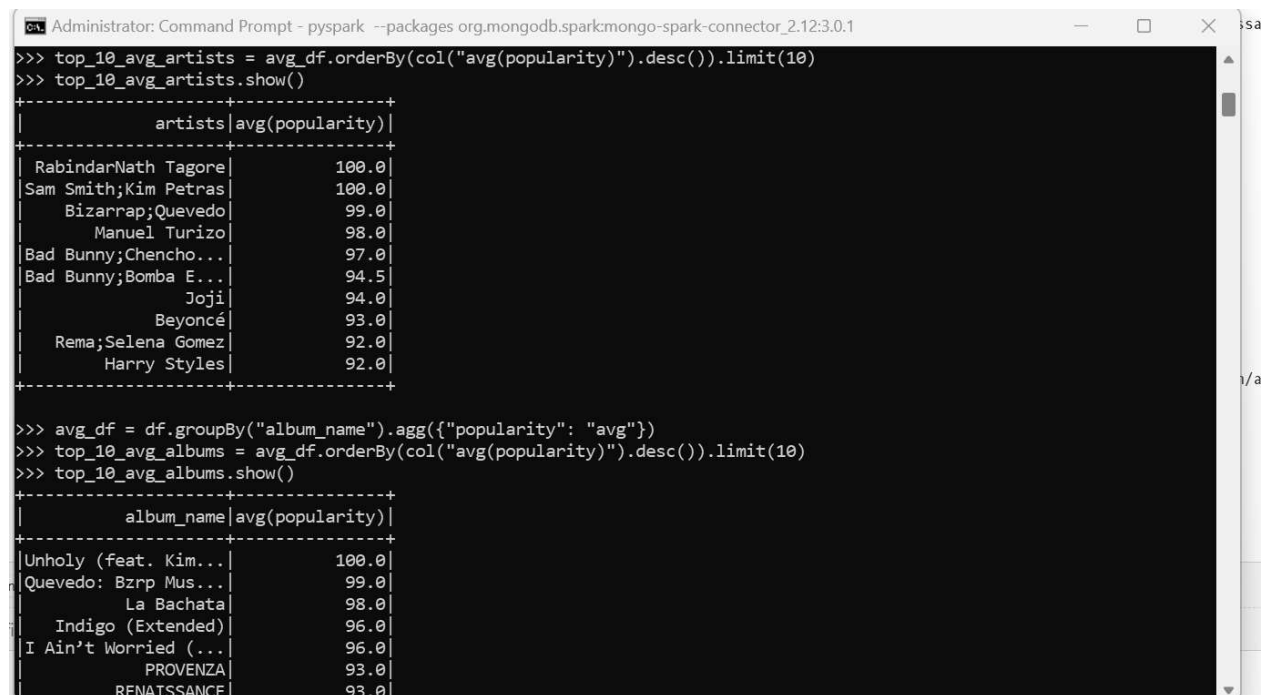
```
df.show()
```

```
from pyspark.sql.functions import col
```

```
avg_df = df.groupBy("album_name").agg({"popularity": "avg"}) top_10_avg_albums =  
avg_df.orderBy(col("avg(popularity)").desc()).limit(10) top_10_avg_albums.show()
```

```
avg_df = df.groupBy("artists").agg({"popularity": "avg"}) top_10_avg_artists =  
avg_df.orderBy(col("avg(popularity)").desc()).limit(10) top_10_avg_artists.show()
```

```
avg_df = df.groupBy("track_genre").agg({"popularity": "avg"}) top_10_avg_genres =  
avg_df.orderBy(col("avg(popularity)").desc()).limit(10) top_10_avg_genres.show()
```



```
>>> top_10_avg_artists = avg_df.orderBy(col("avg(popularity)").desc()).limit(10)
>>> top_10_avg_artists.show()
+-----+-----+
| artists | avg(popularity) |
+-----+-----+
| RabindarNath Tagore | 100.0 |
| Sam Smith;Kim Petras | 100.0 |
| Bizarrap;Quevedo | 99.0 |
| Manuel Turizo | 98.0 |
| Bad Bunny;Chencho... | 97.0 |
| Bad Bunny;Bomba E... | 94.5 |
| Joji | 94.0 |
| Beyoncé | 93.0 |
| Rema;Selena Gomez | 92.0 |
| Harry Styles | 92.0 |
+-----+-----+

>>> avg_df = df.groupBy("album_name").agg({"popularity": "avg"})
>>> top_10_avg_albums = avg_df.orderBy(col("avg(popularity)").desc()).limit(10)
>>> top_10_avg_albums.show()
+-----+-----+
| album_name | avg(popularity) |
+-----+-----+
| Unholy (feat. Kim... | 100.0 |
| Quevedo: Bzrp Mus... | 99.0 |
| La Bachata | 98.0 |
| Indigo (Extended) | 96.0 |
| I Ain't Worried (... | 96.0 |
| PROVENZA | 93.0 |
| RENAISSANCE | 93.0 |
+-----+-----+
```

Github Link

https://github.com/Bhaskarkurada/BITS_MTech_BDSAssignment_SpotifyRecommendation.git