Analitika podataka velikog obujma - Sustav za preporuku glazbe

Autor: Toni Baskijera

Datum: 3. lipnja 2023.

Tablica sadržaja

Uvod

Sustav za preporuku glazbe je sustav koji, na temelju određenih značajki, korisniku preporuča glazbu koja bi mu se mogla sviđati i biti zanimljiva. Za ovaj je projekt razvijen upravo takav sustav koji koristi filtriranje temeljeno na sadržaju (*engl.* content-based filtering) za generiranje preporuka.

Filtriranje temeljeno na sadržaju je tehnika koja se koristi u sustavima preporuka za izradu preporuka na temelju atributa i karakteristika samih stavki. U kontekstu glazbenih preporuka, filtriranje temeljeno na sadržaju fokusirano je na analizu glazbenih značajki i metapodataka pjesama, albuma ili izvođača kako bi se pružile personalizirane preporuke.

Za preporuku, odnosno izračunavanje sličnosti između pjesama korištena je kosinusna sličnost. Kosinusna sličnost je široko korištena mjera u znanosti o podacima i obradi prirodnog jezika (NLP) koja određuje sličnost između dva vektora, ali je vrlo prikladna i za numeričke vrijednosti. Izračunava kosinus kuta između ta dva vektora i proizvodi vrijednost koja se kreće od -1 do 1, pri čemu vrijednost 1 ukazuje na potpunu sličnost vektora, 0 ukazuje na nepostojanje sličnosti, a -1 ukazuje na potpunu različitost.

Svi podaci koje ovaj sustav za preporuku glazbe koristi dohvaćeni su sa Spoitfy API-ja. Spotify, popularna platforma za strujanje glazbe, sadrži golemu kolekciju pjesama koje obuhvaćaju različite žanrove, razdoblja i jezike. Korištena je biblioteka Spotipy. Spotipy je Python biblioteka koja omogućuje jednostavniju komunikaciju s Spotify platformom putem njenog web API-ja. Komunikacija je intuitivnija i ubrzava cijeli proces, kako ne bismo ručno pisali zahtjeve svakoga puta.

Nakon dohvaćanja, podaci su spremljeni u NoSQL bazu MongoDB kako bi bili lako pristupačni i spremni za daljnju analizu i upotrebu. Konkretno, koristi se MongoDB Atlas, inačica MongoDB baze u oblaku.

U ovom slučaju, sustav za preporuku glazbe oslanja se na 11 različitih audio značajki koje Spotify API pruža za svaku pjesmu u svojoj bazi. To su:

- plesnost (*engl.* danceability)
- energija (*engl.* energy)
- ključ (*engl.* key)

- glasnoća (*engl.* loudness)
- mod (*engl.* mode)
- govorljivost (*engl.* speechiness)
- akustičnost (*engl.* acousticness)
- instrumentalnost (*engl.* instrumentalness)
- živost (*engl.* liveness)
- vrijednost (*engl.* valence)
- tempo (*engl.* tempo)

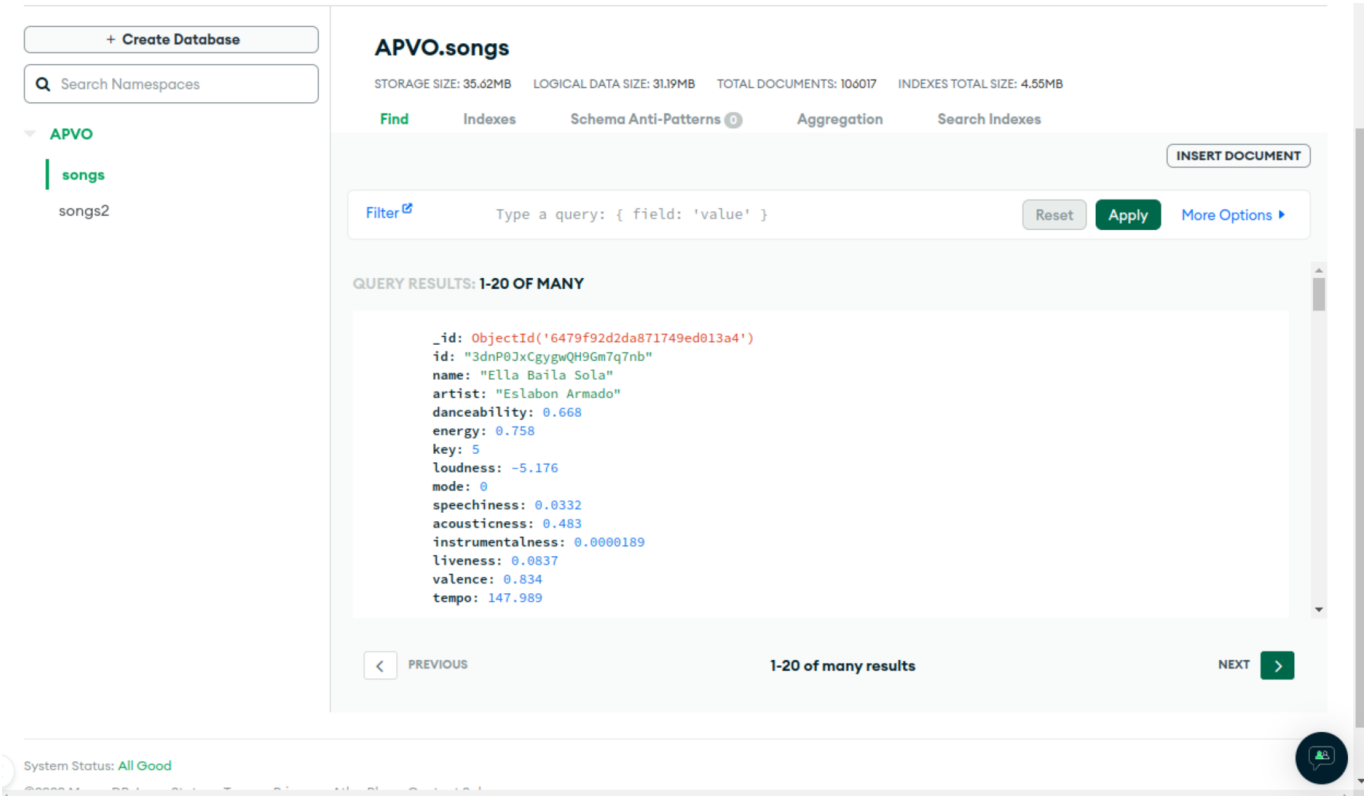
Struktura programa i baze

Cjeloupni projekt rasčlanjen je na 5 različitih python datoteka:

- main.py - glavni dio programa
- mongo.py - dio programa s funkcijama koje služe za interakciju sa MongoDB bazom
- api.py - dio programa s funkcijama koje služe za interakciju sa Spotify API-jem
- constants.py - dio programa u kojemu su navedene konstante koje se koriste
- utils.py - dio programa koji sadrži određene pomoćne funkcije (korištena je samo jedna)

U MongoDB bazu pohranjeni su podaci o 106 017 pjesama u, naravno, isto toliko dokumenata. Svaki dokument sadrži ukupno 15 polja, `_id`, `id`, `name`, `artist` i audio svojstva koja su prije spomenuta.

Polje `_id` je vrijednost koju MongoDB automatski dodjeli dokumentu, dok je polje `id` identifikator pjesme na Spotifyu.



Hodogram programa

Odvijanje programa, grubo (gledajući samo one bitnije procese, bez potprocesa):

1. Korisnik pokreće program naredbom `python main.py`

Program prepoznaje i dva argumenta:

- `-fetch` - dohvatiti još pjesama s API-ja i pohraniti ih u bazu ili ne (zadano False)
- `-iterations` - broj iteracija dohvaćanja pjesama (zadano 1, nema efekta ukoliko je `fetch` False)
- `-playlist` - playlista na temelju koje će se preporučiti glazba (ima zadanu vrijednost postavljenu na id playliste s house glazbom, dostupne na https://open.spotify.com/user/317ivi6d4nkvg6hme7due3bvz2tu/playlist/62vZL7XfkRNEd8K6wjGABT?si=H-At_sWmTJ-vajeHhuHQuA)

2. Inicijalizira se MongoDB klijent i pinga se MongoDB deployment u oblaku kako bi se provjerila veza s bazom, a inicijalizirane su i varijable s detaljima o nazivu baze i kolekcije

3. Inicijalizira se Spotipy klijent

4. Ukoliko je pri pokretanju programa proslijeđen argument `-fetch`, dohvaća se još pjesama, uzimajući i u obzir argument `-iterations`

   Dohvaćanje pjesama je djelomično nasumično - objašnjenje u sljedećem poglavlju

5. Nove dohvaćene pjesme spremaju se u bazu, ukoliko već ne postoji baza s istim ključem u bazi

6. Ipak, svejedno dolazi do pojavljivanja određenih duplikata u bazi, pa se oni uklanjaju

   Objašnjenje u sljedećem poglavlju

7. Dohvaća se id playliste ovisno o argumentu `playlist`

8. Pomoću URI-ja svake zasebne pjesme u playlisti, novim pozivima prema API-ju dohvaćaju se audio značajke koje su nam potrebne, isto tako za svaku zasebnu pjesmu, nakon čega se rezultati spremaju u pandas dataframe

9. Iz MongoDB se učitavaju svi dokumenti iz kolekcije, te se također spremaju u pandas dataframe, pritom izostavljajući polja koja nam nisu bitna

10. Sva se polja u oba dataframea skaliraju, koristeći `MinMaxScaler` iz biblioteke `sklearn`

11. Na skalirane vrijednosti, odnosno pojedine stupce u dataframeu, primjenjuju se određene težine (množe se), kako bi definirali važnost određenih atributa pri preporuci

    Vrijednosti po atributu definirane su sljedećim vrijednostima:

Audio Features	Weights
danceability	1.0
energy	1.5
key	0.8
loudness	1.2
mode	1.0

Audio Features	Weights
speechiness	0.5
acousticness	0.7
instrumentalness	0.5
liveness	0.8
valence	1.2
tempo	1.0

12. Izračunava se kosinusna sličnost između dvoje skaliranih težinskih dataframea, koji sadrži sve pjesme iz baze i pjesme iz proslijeđene playliste, respektivno.

13. Ispisuje se 10 preporuka na temelju izračunate sličnosti, silazno, a osim naziva i izvođaća pjesme prikazuje se i detektirana sličnost

Objašnjenja i zanimljivi djelovi

Nasumično dohvaćanje pjesama

Nažalost, Spotify API iz nekog razloga ne pruža endpoint za dohvaćanje pjesme nasumično. U svakom slučaju, otvoreno je mnogo Issuea na GitHubu od mnogo developera, čak i razvijeno dosta biblioteka koje to implementiraju, ali svejedno nisam bio baš zadovoljan s načinom na koji funkcioniraju. Stoga sam napravio vlastitu funkciju koja donekle nasumično dohvaća pjesme. Kada bi se funkcija vrtila dovoljno dugo, ipak bi se iskoristile sve pjesme i ne bi bilo moguće dohvaćati nove bez mijenjanja logike. Ipak, na ovaj sam način dohvatio nešto više od 110 000 (106 017 nakon uklanjanja duplikata), a mogao bih i još puno više ukoliko za to postoji potreba.

```
def fetch_songs(sp):
    print("Fetching songs")
    limit = 50
    results = []
    characters = 'abcdefghijklmnopqrstuvwxyz'

    for char in characters:
        genre = getRandomCategory(sp)
        try:
            response = sp.search(q=char+'%'+ genre, type='track',
limit=limit, offset=random.randint(0, 1000))
            results.extend(response['tracks']['items'])
        except spotipy.SpotifyException as e:
            continue
    print('End of fetching iteration')
    return results
```

Pojavljivanje duplikata u bazi

Unatoč tome što prije spremanja neke pjesme u bazu provjeravam postoji li već pjesma s istim ID-jem, primjetio sam da u bazi svejedno dolazi do određenih duplikata, gdje su naziv i izvođać pjesme isti kod više dokumenata, a samim time i audio značajke. Stoga sam iskoristio pipeline, odnosno agregaciju kako bih takve i izbrisao.

```
def remove_persisting_duplicates(collection):
    print('Removing duplicates')
    pipeline = [
        {
            '$group': {
                '_id': {
                    'name': '$name',
                    'artist': '$artist'
                },
                'duplicates': {'$push': '$_id'},
                'count': {'$sum': 1}
            }
        },
        {
            '$match': {
                'count': {'$gt': 1}
            }
        }
    ]

    duplicate_docs = collection.aggregate(pipeline)
    for duplicate in duplicate_docs:
        duplicate_ids = duplicate['duplicates']
        collection.delete_many({'_id': {'$in': duplicate_ids[1:]}})
```

Evaluacija sustava

Primjer izvođenja programa, bez dohvaćanja novih pjesama i s korištenjem zadane vrijednosti za playlistu(koja sadrži uglavnom house glazbu):

```
toni@toni-WRT-WX9:~/apvo$ python main.py
Pinged your deployment. You successfully connected to MongoDB!
Fetching user playlist
Loading database
Recommendations:
                                                    name            artist
similarity
71613                           Big Blow - Moodena Remix        Joey Negro
0.994447
105631  No More Looking Back (feat. Steffanie Christi'...        Idris Elba
0.991941
42791                              Just Can't Get Enough       MOMO Soundz
0.991034
71615                            Mood - Black Caviar Remix      Zack Martino
```

```
0.990954
19696   Into The Red Sky (feat. Aya) - Migs Extended M...       Miguel Migs
0.990872
65748                                    Baby Come Back       Pato Banton
0.990479
41994   Into The Red Sky (feat. Aya) - Migs Moodswing ...       Miguel Migs
0.989867
27292                   Free - Mood II Swing Radio Edit        Ultra Naté
0.989362
84006                                        Wiggle It     Poster Queens
0.989165
105957                 Pop Musik - Workout Remix 128 Bpm  Workout Music Tv
0.989066
```

Kako se u današnje vrijeme određeni recommenderi uglavnom ocjenjuju na temelju financijske dobiti i/ili korisničkih metrika, ne mogu dati konkretne brojke i donijeti evaluaciju preporuka objektivno. Ipak, subjektivno, mogu reći kako sam poprilično zadovoljan. Poslušao sam preporučene pjesme i mogu reći kako su vrlo slične kao sve pjesme koje se nalaze na korištenoj playlisti, i mislim da bi korisnik bio vrlo zadvoljan s preporukama. Isprobao sam više playlista, i rezultati su uvijek bili zadovoljavajući kada se koristi playlista na kojoj su sve pjesme iz istog žanra. U suprotnom kada nisu, potrebno je poboljšati model, ponajprije korištenjem i collaborative filtering tehnika,što bi mi bio i prvi korak kada bih krenuo s nadograđivanjem projekta.

Literatura

- [1] Analitika podataka velikog obujma - materijali s predavanja i vježbi