Does the visual matter? Creazione di un dataset di Spotify e YouTube

MARCO SALLUSTIO 906149 , MARCO GUARISCO 789244 , AND SALVATORE RASTELLI 903949

Dipartimento di Informatica Sistemistica e Comunicazione, DISCO, Università degli Studi di Milano Bicocca

Piazza dell'Ateneo Nuovo, 1, 20126 Milano MI

Compiled February 8, 2023

Oggi Spotify è il servizio di musica in streaming più famoso al mondo che permette di accedere a migliaia di brani musicali (italiani e internazionali e di qualsiasi epoca) in modo completamente legale da computer, smartphone, tablet e tanti altri device senza dover acquistare singolarmente canzoni o album. Inoltre, propone una vasta selezione di podcast in tutte le lingue, compresi alcuni in esclusiva per la piattaforma. Grazie alla popolarità di Spotify, i brani caricati sulla piattaforma possono contare milioni di riproduzioni e a tal proposito ci siamo chiesti se un brano su Spotify conserverebbe la stessa popolarità confrontandolo con il suo video musicale caricato sulla celebre piattaforma di Youtube. Il nostro obiettivo, quindi, in questo progetto è capire se il successo di un brano musicale possa dipendere dalla presenza di un video musicale o meno.

1. INTRODUCTION

Per raggiungere l'obiettivo prefissato è stato, ovviamente, necessario raccogliere sia i dati di Spotify e sia i dati di Youtube. Per fare ciò sono state utilizzate diverse tecniche di estrazioni dati: per quanto riguarda Spotify è stata utilizzata sia una tecnica di acquisizione tramite API con le diverse funzionalità messe a disposizione dagli sviluppatori e sia una tecnica di acquisizione tramite Web Scraping; per quanto concerne, invece, Youtube è stata utilizzata unicamente l'API messa a disposizione da Google che ci ha permesso di raccogliere tutti i dati che ritenevamo necessari per il raggiungimento del nostro obiettivo iniziale. Dopo questa prima fase di acquisizione dati sono stati integrati i due dataset in modo che potessimo effettuare un'analisi comparativa tra le due piattaforme.

Di seguito, quindi, saranno descritte dettagliatamente tutte le fasi che sono state effettuate e che si concludono con la valutazione attraverso diverse misure di qualità che hanno permesso di ottenere una panoramica generale dell'integrazione effettuata.

2. DATA ACQUISITION

La prima fase del progetto è stata la raccolta di tutti i dati che ritenevamo potessero essere utili al fine del progetto e per fare ciò, come già anticipato, sono state utilizzate diverse tecniche per raccogliere i dati di Spotify e Youtube: API e Scraping.

Un'API (Application Programming Interface) è un'interfaccia standardizzata e sicura che consente alle applicazioni di comunicare e lavorare tra loro. L'API consente di richiedere dati a un fornitore di terze parti in modo da poter utilizzare queste informazioni come e quando necessario. L'API funge quindi da elemento di intermediazione tra gli utenti o i clienti e le risorse o servizi web che questi intendono ottenere. È anche un mezzo con il quale un'organizzazione può condividere risorse e informazioni assicurando al contempo sicurezza, controllo e autenticazione, poiché stabilisce i criteri di accesso. L'utilizzo dell'API inoltre non impone all'utente di conoscere le specifiche con cui le risorse vengono recuperate o la loro provenienza.

Al contrario, lo scraping è il processo per estrarre i dati da un sistema senza utilizzare l'API; invece, il Web Scraping è il processo per estrarre dati da pagine web. Ci possono essere diversi modi di fare Web Scraping: si possono utilizzare estensioni o tool appositi. A volte, però, quest'ultimi potrebbero non essere in grado di completare il processo di Scraping a causa di limiti di licenza o problemi relativi alla complessità della pagina web. Per ovviare a questi problemi ci sono molteplici modi: si potrebbe, ad esempio, utilizzare Python e la libreria Beautifulsoup che permette di leggere il codice sorgente HTML della pagina web ed estrarre le informazioni desiderate. Alcune volte, però, leggere il codice sorgente di una pagina web potrebbe non essere sufficiente e in tal caso si potrebbe utilizzare un'ulteriore libreria di Python chiamata Selenium; per il nostro caso d'uso, infatti, è stata utilizzata questa libreria il cui utilizzo è stato spiegato nei paragrafi successivi.

A. Spotify - API

Per raccogliere i dati di Spotify è stata utilizzata la sua API (https://developer.spotify.com/documentation/web-api/) in cui la ricerca di tutti i brani è avvenuta in base al tipo di query effettuata. Per utilizzare l'API è stata utilizzata la libreria Spotipy di Python. Spotipy supporta tutte le funzionalità dell'API Web di Spotify, incluso l'accesso a tutti gli endpoint e il supporto per l'autorizzazione dell'utente. Abbiamo deciso di effetturare una ricerca di artisti usando il metodo search. Nella nostra ricerca abbiamo ottenuto 2079 artisti con canzoni pubblicate dal 2000 al 2022. Di questi artisti abbiamo richiesto le prime dieci canzoni più popolari secondo spotify.

```
#Utilizzo il metodo search per cercare gli artisti
for anni in range(2000,2023):
    for i in range(0,800,50):
        track_results = sp.search(q=f'year:{anni}', type='artist', limit=50, offset=i , market = mkt)
        for i, t in enumerate(track_results['artists']['items']):
            if t['name'] not in artist_name:
                artist_name.append(t['name'])
                link.append(t['external_urls']['spotify'])
                uri.append(t['uri'])
```

Per ogni brano trovato sono stati raccolti i seguenti dati che comprendono le informazioni generali e le diverse Audio Features del brano:

- Artist: Nome degli artisti che hanno eseguito il brano;
- Url: L'URL di Spotify della pagina dell'artista
- Track: Nome della canzone;
- Album: L'album in cui è presente il brano;
- Album_type: Il tipo di album (album, single, compilation);
- Uri: L'URI di Spotify per la traccia;
- Danceability: La danceability descrive quanto sia adatta una traccia per "ballare" sulla base di una combinazione di elementi musicali, tra cui tempo, stabilità del ritmo, forza del ritmo e regolarità generale. Un valore di 0.0 è meno "ballabile" e 1.0 è più "ballabile";
- Energy: L'energia è una misura da 0,0 a 1,0 e rappresenta una misura percettiva dell'intensità e dell'attività. In genere, le tracce energiche sembrano veloci e rumorose. Ad esempio, il death metal ha un'energia elevata, mentre un preludio di Bach ha un punteggio basso sulla scala. Le caratteristiche percettive che contribuiscono a questo attributo includono la gamma dinamica, il volume percepito, il timbro, la frequenza di insorgenza e l'entropia generale;
- Key: La tonalità in cui si trova la traccia. I numeri interi vengono mappati alle altezze utilizzando la notazione standard della classe di altezza. Per esempio. 0 = DO, 1 = DO$^{\#}$, 2 = RE, e così via. Se non è stata rilevata alcuna chiave, il valore è -1;

- Loudness: Il volume complessivo di una traccia in decibel (dB). I valori del volume vengono calcolati in media sull'intera traccia e sono utili per confrontare il volume relativo delle tracce. La loudness è la qualità di un suono che è il correlato psicologico primario della forza fisica (ampiezza). I valori in genere variano tra -60 e 0 db;

- Speechiness: Rileva la presenza di parole pronunciate in una traccia. Più la registrazione è esclusivamente vocale (ad es. talk show, audiolibri, poesie), più il valore dell'attributo si avvicina a 1.0. I valori superiori a 0,66 descrivono tracce che probabilmente sono costituite interamente da parole pronunciate. I valori compresi tra 0,33 e 0,66 descrivono tracce che possono contenere sia musica che parlato, sia in sezioni che sovrapposte, inclusi casi come la musica rap. I valori inferiori a 0,33 molto probabilmente rappresentano musica e altre tracce non vocali;

- Acousticness: Una misura di confidenza da 0,0 a 1,0 del fatto che la traccia sia acustica. 1.0 rappresenta un'elevata sicurezza che la traccia sia acustica;

- Instrumentalness: Prevede se una traccia non contiene voci. I suoni "Ooh" e "aah" sono trattati come strumentali in questo contesto. Le tracce rap o di parole sono chiaramente "vocali". Più il valore di strumentalità è vicino a 1.0, maggiore è la probabilità che la traccia non contenga contenuto vocale. I valori superiori a 0,5 intendono rappresentare tracce strumentali, ma la confidenza è maggiore quando il valore si avvicina a 1,0;

- Liveness: Rileva la presenza di un pubblico nella registrazione. Valori di vivacità più elevati rappresentano una maggiore probabilità che la traccia sia stata eseguita dal vivo. Un valore superiore a 0,8 fornisce una forte probabilità che la traccia sia live:

- Valence: Una misura da 0.0 a 1.0 che descrive la positività musicale trasmessa da un brano. I brani con valenza alta suonano più positivi (ad es. felice, allegro, euforico), mentre i brani con valenza bassa suonano più negativi (ad es. triste, depresso, arrabbiato);

- Tempo: Il tempo complessivo stimato di una traccia in battiti al minuto (BPM). Nella terminologia musicale, il tempo è la velocità o il ritmo di un dato brano e deriva direttamente dalla durata media del battito;

- Duration_ms: La durata della traccia in millisecondi.

B. Spotify - Scraping

Si può notare che nella precedente sezione, non è presente un dato fondamentale ai fini del progetto: il numero di stream di una canzone. Infatti, l'API di Spotify non permette ancora di acquisire questo dato; per sopperire a questo problema è stato pensato di recuperare il dato utilizzando tecniche di web scraping, in particolare utilizzando la libreria Selenium. Selenium è un tool open source per l'automazione del browser in grado di svolgere numerosi task. Uno di questi è il web scraping per estrarre dati e informazioni utili che altrimenti potrebbero non essere disponibili. Supporta collegamenti per tutti i principali linguaggi di programmazione, incluso il linguaggio utilizzato per l'intero progetto: Python. La nostra idea è stata, quindi, quella di collegarsi automaticamente alla pagina dell'artista di Spotify, utilizzando l'URL estratto dall'acquisizione dei dati tramite API, ed estrarre il numero di stream per i 10 brani più popolari dell'artista. Per fare ciò sono stati utilizzati tutti gli URL degli artisti ottenuti precedentemente tramite l'API e per ottenere il numero di stream è stata utilizzata una funzione di Selenium in grado di accedere al dato desiderato in base al valore del suo selettore CSS.

```
options = Options()
options.page_load_strategy = 'normal'
driver = webdriver.Chrome(options=options)

driver.get("https://open.spotify.com")

WebDriverWait(driver, 120).until(EC.element_to_be_clickable((By.XPATH, "//button[@id='onetrust-accept-btn-handler']"))).click
for i in range(0,len(url)):

        driver.get(url[i])
        WebDriverWait(driver, 120).until(EC.element_to_be_clickable((By.XPATH, "//div[@class='Type__TypeElement-sc-goli3j-0 c
        elem=driver.find_elements(By.CSS_SELECTOR, ".Type__TypeElement-sc-goli3j-0.hGXzYa.nYg_xsOVmrVE_8qk1GCW")
        elem1=driver.find_elements(By.CSS_SELECTOR, ".Type__TypeElement-sc-goli3j-0.kHHFyx.t_yrXoUO3qGsJS4Y6iXX.standalone-el
        if(len(elem) or len(elem1)) < 10:
            print("problema con",i)
            remove.append(art[i])
        else:
            for value in elem:
                stream.append(value.text)
            for value in elem1:
                name.append(value.text)
```

C. YouTube - Scraping

Per acquisire i dati relativi a YouTube, è stata utilizzata la sua API, combinata con tecniche di Web Scraping.

Conoscendo già gli artisti e i titoli delle canzoni ottenuti da spotify, si è inizialmente provato ad utilizzare l'API di YouTube per effettuare la ricerca di ciascuna di esse. Tuttavia, è stata riscontrata una limitazione importante: l'API di YouTube mette a dispo-

sizione solo 10.000 punti al giorno di quota, con ogni ricerca tramite la funzione "search" vengono spesi 100 punti. Questo significava che si potevano effettuare solo 100 ricerche al giorno, rendendo questa soluzione troppo lenta per raggiungere l'obiettivo.

Per ovviare a questo problema, si è deciso di utilizzare anche Selenium. Il codice è stato scritto in Python e eseguito in un ambiente Jupyter Notebook. Utilizzando Selenium, si è effettuata una ricerca su YouTube per ottenere l'URL del primo risultato della ricerca *"Artista" + "Nome Canzone" + "Official Video"*.

```
#open Chrome
options = Options()
options.page_load_strategy = 'normal'
driver = webdriver.Chrome(options=options)

#Load youtube
driver.get("https://www.youtube.com")

wait = WebDriverWait(driver, 10)
#reject cookies
reject = wait.until(EC.presence_of_element_located((By.XPATH,'//*[@id="content"]/div[2]/div[6]/div[1]/ytd-button-renderer[1]
reject.click()
#iterate on df
for artist,track in zip(df['Artist'], df['Track']):
    #search for the video
    query = artist + " " + track + " official video"
    driver.get(f"https://www.youtube.com/results?search_query={query}")
    #click on the first result and get the URL
    try:
        element = wait.until(EC.presence_of_element_located((By.XPATH, '//*[@id="thumbnail"]/yt-image/img')))
        element.click()
    except:
        element1 = wait.until(EC.presence_of_element_located((By.XPATH, '//*[@id="video-title"]/yt-formatted-string')))
        element1.click()

    url.append(driver.current_url)
    traccia.append(track)
    artista.append(artist)
```

D. YouTube - API

Una volta ottenuta la lista di tutti gli URL è stato possibile utilizzarli per ottenere le informazioni necessarie dall'API (https://developers.google.com/youtube/v3), in quanto una richiesta di informazioni a partire dal video-id costa solo 1 punto. Per ogni video sono stati raccolti i seguenti dati:

- Channel: Il nome del canale che ha pubblicato il video;
- Title: Il titolo del video;
- Views: Il numero di visualizzazioni del video;
- Likes: Il numero di likes ricevuti dal video;
- Comments: Il numero di commenti ricevuti dal video;
- Description: Il testo dell'intera descrizione del video;
- Licensed: Valore booleano che indica se il video rappresenta contenuti concessi in licenza, il che significa che i contenuti sono stati caricati su un canale collegato a un partner per i contenuti di YouTube e quindi rivendicati da tale partner.

```
youtube = build("youtube", "v3", developerKey=API_KEY)
for url in URL[0:]:
    #check for valid url
    try:
        video_id=extract_video_id(url)
    except:
        #if the video is a youtube short the API won't recognize its id, so we won't consider those.
        channels.append(None)
        titles.append(None)
        views.append(None)
        comments.append(None)
        likes.append(None)
        descriptions.append(None)
        licensed.append(None)
        print(f'short on {url}')
    else:

        request = youtube.videos().list(part="statistics", id=video_id)
        request1 =youtube.videos().list(part="snippet", id=video_id)
        request2 = youtube.videos().list(part="contentDetails", id=video_id)

        response = request.execute()
        response1 =request1.execute()
        response2 = request2.execute()
```

3. DATA PREPARATION

A. Data Quality

A.1. Currency

Per valutare se un dato sia aggiornato oppure no bisogna utilizzare come misura di qualità la currency. La currency misura con quale rapidità i dati sono aggiornati rispetto al corrispondente mondo reale. Dalla definizione appena data possiamo affermare, quindi, che nel nostro progetto i dati relativi al numero di stream di ogni canzone su Spotify e i dati relativi al numero di visualizzazioni di ogni video su YouTube, non sono aggiornati. Questo perché i dati sono stati raccolti solo nel momento in cui è stata effettuata la fase di Data Acquisition e di conseguenza i dati non rispecchiano la realtà attuale, dato che il numero di stream di una canzone o il numero di visualizzazioni di un video possono cambiare ogni giorno. Questo non riguarda solo i dati appena descritti, ma riguarda anche le canzoni presenti all'interno del dataset creato perché, bisogna ricordare, che sono state estratte le canzoni più popolari di ogni artista e anche in questo caso ci potrebbe essere la possibilità che una canzone di un'artista non sia più presente tra le sue dieci canzoni più popolari.

A.2. Accuracy

Considerando come riferimento i nomi delle canzoni che abbiamo recuperato dall'API, siamo andati a verificare se le stream recuperate tramite scraping su spotify siano effettivamente accurati rispetto ai dati iniziali. Per quanto riguarda le stream, ci siamo trovati di fronte a 686 tuple per cui il titolo della canzone data dall'API era diverso dal valore acquisito tramite scraping; alcune di queste erano diverse per la presenza (o assenza) di parole come Remastered o Remaster nel titolo, mentre l'altro motivo da ricercare nella modifica dell'ordine delle canzoni più popolari nel lasso di tempo, seppur breve, dal momento in cui abbiamo acquisito i dati con l'API al momento in cui abbiamo finito di acquisire i dati tramite Scraping.

Per rimediare al primo caso, abbiamo semplicemente diviso in due il dataset e accoppiato i dati in base all'artista e alla track, mentre per il secondo caso abbiamo proceduto per step:

- Per gli artisti con una sola canzone sbagliata, abbiamo usato la funzione *SequenceMatcher* della libreria *difflib* per valutare la somiglianza tra i due titoli su una scala da 0 a 1 e abbiamo scelto 0.5 come soglia per essere considerate 'giuste';
- Per gli artisti con più di una canzone sbagliata, abbiamo definito una funzione che controllasse che Track_Api e Track che stanno sulla stessa riga siano le più simili tra loro in relazione alle altre canzoni dello stesso artista e che ci restituisse un valore True o False;
- In base a questa procedura, abbiamo poi assegnato un valore nullo a quelle canzoni per cui abbiamo ritenuto non avere il numero delle Stream

Usando sempre i nomi delle tracce acquisite con l'API di spotify,abbiamo verificato che i video trovati tramite scraping su youtube siano effettivamente quelli relativi alla canzone. Anche in questo caso abbiamo proceduto per step:

- Abbiamo inizialmente usato la funzione *SequenceMatcher* per valutare la somiglianza tra il nome della canzone e il titolo della traccia. Essendo però i titoli dei video, nella maggior parte dei casi, molto più lunghi dei titoli e contenendo anche il titolo dell'artista o degli artisti che hanno partecipato, abbiamo scelto come soglia in questo caso 0.3;
- per i brani con una somiglianza minore di 0.3, abbiamo definito un'ulteriore funzione che ci restituisse il valore True se una di queste condizioni veniva soddisfatta:
    1. Il nome dell'artista è presente nel titolo del video;
    2. Il nome dell'artista corrisponde al nome del canale che ha pubblicato il video
    3. Il titolo della canzone è contenuto nel titolo del video su youtube
- Nel caso il titolo del video avesse una somiglianza minore di 0.3 e non verificasse nessuna di queste condizioni, abbiamo deciso di considerare il video sbagliato e sostituire i dati recuperati dall'API con dei valori nulli.

Infine abbiamo aggiunto una colonna con un valore booleano che indicasse se il video in questione fosse o meno un video ufficiale, in base a questa selezione:

- Se l'Attributo Licensed è True, allora il video è stato rivendicato dall'artista e può essere considerato un video ufficiale;
- Nel caso in cui Licensed è False, allora abbiamo aggiunto un secondo controllo: se il nome del canale corrisponde al nome dell'artista o è contenuto nel titolo del video, allora anche in questo caso il video può essere considerato un video ufficiale.

Dopo questa selezione abbiamo contato il numero di video con attributo Official_Video True, che sono risultati essere 15737, più di 3/4 della grandezza del dataset.

B. Data Cleaning

Innanzitutto, è stato notato che i dati relativi alle visualizzazioni dei video di YouTube e agli streams delle canzoni di Spotify non erano dello stesso formato: per le visualizzazioni di YouTube erano di tipo numpy.float64, mentre gli streams di Spotify erano di tipo String. Si è scelto, quindi, di cambiare il tipo di formato degli streams di Spotify. Successivamente, abbiamo notato come ci fossero alcune canzoni con valore di Artist, Track e Album uguali, ma conteggio delle Stream diverse. Abbiamo quindi deciso di tenere il valore più alto per ognuna di queste, ed eliminare le restanti.

C. Data Integration ed Enrichment

A questo punto siamo passati all'unione di questi 3 datasets: abbiamo prima di tutto unito i datasets ottenuti tramite l'API di Spotify e quella di Youtube attraverso un Full Outer Join, matchando i valori in base all'indice; successivamente, abbiamo unito questo dataset con quello del conteggio delle stream attraverso un Left Outer Join, matchando i valori in base al valore della colonna Artist, Track e Album. Abbiamo poi fatto un controllo finale per la presenza di duplicati, che abbiamo poi eliminato attraverso il metodo *drop_duplicated* della libreria *pandas*.

D. Completezza

Dopo l'integrazione, abbiamo controllato la completezza di Attributo del dataset integrato, con un risultato di:

- 0 valori nulli per per gli attributi Artist, Url_spotify, Track,Album,Album_Type e Uri;
- 2 valori nulli per gli attributi Danceability, Energy, Key, Loudness, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Temp and Duration_ms
- 470 valori nulli per gli attributi Url_youtube, Title, Channel e Views
- 541 valori nulli per l'attributo Likes
- 569 valori nulli per l'attributo Comments
- 876 valori nulli per l'attributo Description
- 470 valori nulli per l'attributo Licensed
- 470 valori nulli per l'attributo official_video
- 576 valori nulli per l'attributo Stream

4. STORAGE DEL DATABASE INTEGRATO

Una volta che i dati raccolti sono stati integrati e ripuliti è stato scelto di salvarli in MongoDB. La scelta è ricaduta sul modello documentale soprattutto per la flessibilità che questo tipo di modello offre; MongoDB, infatti, ha un'architettura a schema dinamico che funziona con dati e storage non strutturati. Poiché i dati sono memorizzati in documenti flessibili, lo schema del database non deve essere predefinito e gli schemi possono essere modificati in modo dinamico. Quindi, con il formato di dati BSON di MongoDB, gli oggetti di una raccolta possono avere diversi insiemi di campi e quasi ogni tipo di struttura di dati può essere modellata e manipolata. Un altro importante motivo per il quale è stato scelto di utilizzare MongoDB riguarda le performance più elevate che questo tipo di DB offre. MongoDB, infatti, memorizza i dati nella RAM per accelerarne l'accesso e ottimizzare le performance durante l'esecuzione delle query. Raccoglie i dati direttamente dalla RAM anziché dal disco rigido, rendendo più veloci le letture e le scritture dei dati. La struttura non relazionale dei dati di MongoDB richiede inoltre una minore potenza di elaborazione per la ricerca e il recupero dei dati rispetto a un database relazionale.

Inoltre, i dati prima di essere salvati su MongoDB sono stati convertiti nel formato JSON in una struttura scelta da noi che prevedeva di avere per ogni documento la canzone, l'artista e i dati relativi a Spotify all'interno di un' array e i dati relativi a YouTube all'interno di un altro array. Di seguito viene riportato l'estratto di codice che ha permesso di creare la struttura appena descritta e un esempio di documento salvato in MongoDB:


```
_id: ObjectId('63e2d062175cd3f6d2dff145')
Artist: "Gorillaz"
Track: "Feel Good Inc."
▾ Spotify: Object
   Stream: 1040234854
   Url artista: "https://open.spotify.com/artist/3AA28KZvwAUcZuOKwyblJQ"
   Album: "Demon Days"
   Album Type: "album"
   Uri: "spotify:track:0d28khcov6AiegSCpG5TuT"
   Danceability: 0.818
   Energy: 0.705
   Key: 6
   Loudness: -6.679
   Speechiness: 0.177
   Acousticness: 0.00836
   Instrumentalness: 0.00233
   Liveness: 0.613
   Valence: 0.772
   Tempo: 138.559
   Duration_ms: 222640
▾ Youtube: Object
   Views: 693555221
   Likes: 6220896
   Channel: "Gorillaz"
   Url video: "https://www.youtube.com/watch?v=HyHNuVaZJ-k"
   Title: "Gorillaz - Feel Good Inc. (Official Video)"
   Description: "Official HD Video for Gorillaz' fantastic track Feel Good Inc.

                 Follow…"
   Licensed: true
   Official_video: true
```


A. Query effettuate

A.1. Query 1: Trovare il numero di documenti totali presenti nella collection

E' stata effettuata una query in grado di restituire il numero di documenti presenti nella collection.

```
n_documents=[]
for doc in coll.find():
    n_documents.append(doc)
```

Il risultato di questa query è: 20718 documenti

A.2. Query 2 e 3: Trovare il numero di documenti le cui views e Streams sono maggiori di zero

Sono state effettuate due query in grado di restituire il numero di documenti in cui il campo Views di YouTube e il campo Stream di Spotify è maggiore di zero (quindi non un valore nullo)

```
non_null_streams=[]
for doc in coll.find({"Spotify.Stream":{"$gt":0}},{"Track","Artist"}):
    non_null_streams.append(doc)
    print(doc)
```

```
non_null=[]
for doc in coll.find({"Youtube.Views":{"$gt":0}},{"Track","Artist"}):
    if doc in non_null_streams:
        non_null.append(doc)
        print(doc)
```

che ci hanno restituito 19692 documenti.

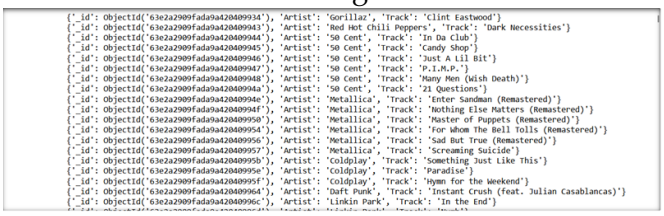
A.3. Query 4: Trovare i documenti che hanno un numero di views(YouTube) maggiori del numero di streams(Spotify)

È stata effettuata la query in grado di rispondere alla nostra domanda principale: un video ufficiale di una canzone potrebbe ottenere un numero maggiore di visualizzazioni

rispetto alla sua versione musicale su Spotify?

```
results=[]
for doc in coll.find({"$and":[{"$expr": {"$gt": ["$Youtube.Views","$Spotify.Stream"]}},
                              {"Youtube.Official_video":True}]},{"Track","Artist"}):
    results.append(doc)
    print(doc)
```

Ottenendo i seguenti documenti:



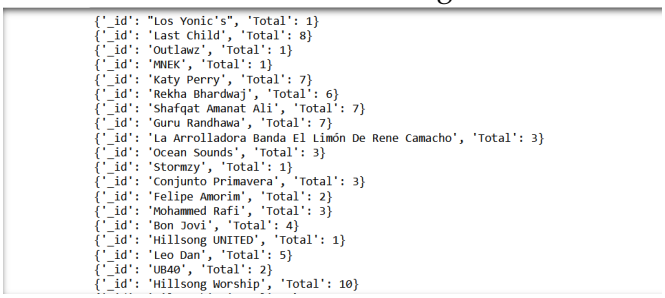
A questo, abbiamo contato il numero di documenti restituiti e abbiamo ottenuto un risultato pari a 4369.

A.4. Query 5: Numero di canzoni per ogni artista trovato nella precedente query

In questa seconda query abbiamo deciso di trovare il numero di canzoni eseguite da ogni artista in base ai risultati della precedente query. Per fare ciò è stato salvato ogni documento trovato nella precedente query in una nuova collection e su quest'ultima è stato utilizzato l'operatore *aggregate* contando quindi il numero di documenti per ogni artista

```
for doc1 in coll_new.aggregate([{"$group":{ "_id":'$Artist',"Total":{ "$sum" : 1 } } }]):
    print(doc1)
```

E' stato così ottenuto il seguente risultato:



```
{'_id': "Los Yonic's", 'Total': 1}
{'_id': 'Last Child', 'Total': 8}
{'_id': 'Outlawz', 'Total': 1}
{'_id': 'MNEK', 'Total': 1}
{'_id': 'Katy Perry', 'Total': 7}
{'_id': 'Rekha Bhardwaj', 'Total': 6}
{'_id': 'Shafqat Amanat Ali', 'Total': 7}
{'_id': 'Guru Randhawa', 'Total': 7}
{'_id': 'La Arrolladora Banda El Limón De Rene Camacho', 'Total': 3}
{'_id': 'Ocean Sounds', 'Total': 3}
{'_id': 'Stormzy', 'Total': 1}
{'_id': 'Conjunto Primavera', 'Total': 3}
{'_id': 'Felipe Amorim', 'Total': 2}
{'_id': 'Mohammed Rafi', 'Total': 3}
{'_id': 'Bon Jovi', 'Total': 4}
{'_id': 'Hillsong UNITED', 'Total': 1}
{'_id': 'Leo Dan', 'Total': 5}
{'_id': 'UB40', 'Total': 2}
{'_id': 'Hillsong Worship', 'Total': 10}
```

5. CONCLUSIONI E SVILUPPI FUTURI

In questo progetto è stato presentato un metodo per acquisire e preparare i dati da fonti diverse (Spotify API, Web Scraping su Spotify e YouTube, YouTube API) per analizzare la popolarità di artisti e brani musicali. L'utilizzo di tecniche di web scraping e API è stato essenziale per raggiungere l'obiettivo di raccogliere informazioni sul numero di stream, visualizzazioni, commenti e like di un brano musicale. La qualità dei dati è stata verificata e eventualmente migliorata durante la fase di preparazione. Utilizzando il nostro dataset, è possibile misurare l'impatto che un video ufficiale può avere sulla popolarità di un brano. La scelta di salvare i dati in MongoDB ci ha permesso di accedere e manipolare facilmente i dati,fornendo una solida base per continuare a esplorare questo argomento e a comprendere meglio il ruolo dei video nella promozione musicale. La quantità di canzoni con più views su youtube che su spotify si è rivelata meno di un terzo della quantità di canzoni con un video ufficiale. Il risultato della query 4 sembrerebbe tendere alla risposta che no, la presenza di un video musicale non aiuta al successo di un brano in generale. Questo non esclude la possibilità che, sotto certe condizioni, la presenza di un video ufficiale possa avere l'effetto sperato, considerando che ci sono effettivamente degli artisti con un gran numero di canzoni con più views su youtube che riproduzioni su spotify (ad esempio Katy Perry, con 7 canzoni su 10 considerate), al fronte di altri con poche canzoni (Bon Jovi, con solo 4). Questo ci porta a suggerire, come sviluppo futuro di questo dataset, l'aggiunta di campi che possano andare a migliorare la ricerca in questo senso. Si potrebbe pensare, ad esempio, di:

- Aggiungere all'interno di ogni documento i nomi degli artisti che hanno partecipato alla canzone nel caso di Featuring;
- Estrarre le informazioni che compongono il nostro dataset in tempo reale, aggiornando quindi periodicamente i dati per evitare di avere dati non aggiornati;
- Aggiungere la data di pubblicazione della canzone.

REFERENCES

1. https://www.mongodb.com/docs/
2. https://developer.spotify.com/documentation/web-api/
3. https://developers.google.com/youtube/v3
4. https://pandas.pydata.org/docs/
5. https://selenium-python.readthedocs.io/
6. https://pymongo.readthedocs.io/en/stable/