Just Repeat-Hit: Measuring the nowadays music repetitiveness using textual compression

Alessandro Borroni[1], Mirko Giugliano[1]*, Angela Prade[1]

Sommario

Il mondo musicale degli ultimi anni sembra essere caratterizzato da hit sempre più ripetitive, scontate e banali. Tale progetto, quindi, si propone di misurare la ripetitività delle canzoni, analizzandole da un punto di vista testuale, con il fine di valutare l'esistenza o meno di alcune tendenze. L'approccio utilizzato sfrutta il calcolo di un indicatore, definito "Indice di Ripetitività", il quale corrisponde al complementare del rapporto fra le parole uniche all'interno di un testo, ossia quelle parole che sono presenti una e una sola volta, e le parole totali dello stesso. I dati necessari allo sviluppo di quanto suddetto sono stati ottenuti attraverso l'implementazione di alcuni script in linguaggio Python; questi sono stati utilizzati principalmente per effettuare specifiche richieste tramite le API di Spotify e di Genius, le quali hanno permesso di ottenere, rispettivamente, una lista di artisti con tutte le caratteristiche dei testi delle canzoni per ogni artista (lyrics) con annesse peculiarità audiometriche o, più semplicemente, informazioni relative all'artista o relative a ogni canzone. Si è posta maggiore attenzione sullo studio dell'andamento temporale dell'Indice di Ripetitività medio e sull'analisi del medesimo indice, focalizzandosi principalmente su un'aggregazione in base al genere musicale. Il primo obiettivo che è stato perseguito con questo progetto riguarda l'individuazione dei trend evidenti nell'andamento dell'indice; come secondo obiettivo, invece, ci si è proposti di individuare quali generi musicali fossero più o meno ripetitivi, cercando anche di capire come i testi delle canzoni potessero influire su tali risultati. Mediante l'utilizzo del software Tableau, inoltre, sono state proposte alcune infografiche che permettono di ottenere un insight di maggior dettaglio sui risultati, con l'intento di comprenderli al meglio. Infine, sono stati esposti possibili miglioramenti al progetto ed eventuali sviluppi futuri.

Keywords

— MongoDB — TextualCompression — DataManagement — DataVisualization — Spotify — Genius — Mongos — Python — R — Music

[1]*Data Science M.Sc., Department of Computer Science, Systems and Communication, University of Milano-Bicocca, Milan, Italy*

***Corresponding author**: m.giugliano@campus.unimib.it

Indice

1. introduzione

Il progetto ha avuto origine a partire da due principali quesiti:

1. La musica sta effettivamente diventando sempre più ripetitiva?
2. In caso affermativo, sono soprattutto i nuovi generi, quali ad esempio la Trap, a trainare questa tendenza?

Tali tematiche sono state affrontate in uno studio di Colin Morris, programmatore americano che ha analizzato la ripetitività di un dataset di quindicimila canzoni che hanno occupato la top 100 secondo Billboard tra il 1958 e il 2017. Il suo progetto mirava a capire se i testi delle canzoni Pop fossero sempre più ripetitivi con il passare del tempo. Tali limiti vengono qui superati, andando a indagare su un più ampio numero di canzoni (circa 26 volte maggiore), ricoprendo il maggior numero possibile di generi e conseguentemente analizzando un numero di artisti ben più alto, senza fare distinzione tra più popolari e meno famosi.

A partire dalla volontà di trovare una risposta alle domande citate precedentemente e al fine di rendere possibile l'integrazione dei dati, sono state effettuate due tipologie di richieste. La prima è stata rivolta all'API di Spotify, un servizio musicale che offre lo streaming on demand di una vasta selezione di brani; la seconda, invece, all'API di Genius, un sito dedicato alla lettura, all'interpretazione e alla spiegazione dei testi musicali dei più svariati artisti.

La prima difficoltà riscontrata è sorta in relazione alle richieste tramite API di Spotify e di Genius di cui sopra. Tali interrogazioni richiedevano, infatti, diversi giorni per il completamento, pertanto è stato necessario utilizzare le Macchine Virtuali di Azure, messe a disposizione dall'Ateneo, con l'obiettivo principale di evitare che il flusso di dati venisse interrotto durante l'esecuzione in locale degli script.

Al fine di rendere possibile l'implementazione della *Data Ingestion*, si è optato per l'utilizzo di Python, il quale è stato utilizzato anche durante la fase di pre-processing. Per lo storage, invece, è stato utilizzato MongoDB, il quale è stato impiegato congiuntamente a Tableau per lo sviluppo delle query atte a ottenere i risultati necessari. Più nello specifico, attraverso MongoDB sono stati analizzati i dati al fine di rispondere ai quesiti iniziali, mentre Tableau ha permesso di interrogare i dati ed estrarre quelli necessari alla creazione delle infografiche.

Si è scelto di utilizzare Python poiché le librerie *Pandas* e *Re* sono risultate particolarmente adatte alla trattazione di grandi quantità di dati e alla soddisfazione delle esigenze delle fasi di pre-processing. Inoltre, anche l'utilizzo della libreria *Collection* si è dimostrato utile al calcolo dell'Indice di Ripetitività.

MongoDB, un sistema di gestione basato sui documenti (Document Based Management System, DBMS), in cui i dati vengono archiviati in formato BSON (Binary JSON) e letti tramite indici, è risultato lo strumento migliore per le sue capacità di indicizzare i dati, velocizzare la creazione delle query e implementare la tecnica dello Sharding, ovvero rendere possibile la creazione di un database distribuito in cui ogni nodo contiene una porzione del database stesso.

Come menzionato precedentemente, Tableau è stato utilizzato per lo sviluppo delle infografiche. Grazie alla sua intuitività e agevolezza nella creazione di contenuti, ha permesso di ottenere infografiche chiare, esplicative e che rispettassero le euristiche classiche del mondo delle *Data Visualization*.

I problemi riscontrati sono stati generati perlopiù dal fatto di aver utilizzato, inizialmente, due differenti fonti di dati, ossia Spotify e Genius. Nel tentativo di porre rimedio agli stessi, per ottenere il massimo livello di informazioni e per riuscire ad adempiere agli obiettivi posti, sono state implementate manipolazioni tipiche della *Data Integration*.

2. Approccio metodologico

Oltre alle domande poste nel paragrafo precedente, di cui si vuole trovare una risposta, il progetto si pone come obiettivi didattici quelli di affrontare due delle tre **V** caratteristiche dei *Big Data*:

1. **V**olume: la raccolta dei dati ha prodotto un quantitativo di notevole importanza, comportando la costruzione di un dataset di dimensione maggiore di 2 GB, come richiesto.

2. **V**ariety: da un lato, l'utilizzo di differenti fonti di dati ha generato delle complicazioni non di poco conto, dall'altro ha permesso di soddisfare tale requisito.

2.1 Calcolo indice

Durante la fase iniziale del progetto, è stato necessario valutare se l'obiettivo finale e, quindi, l'individuazione di un Indice di Ripetitività, fosse perseguibile in maniera efficace e valida. A tale proposito, la libreria *Collections* si è rivelata particolarmente utile; dando in input alla sua funzione *Counter* un testo, infatti, questa ha permesso di ricavare un dizionario che avesse come chiave una parola del testo e come valore il numero di volte in cui essa si ripeteva. A partire da questo dizionario, è stato poi possibile ricavare il valore dell'Indice di Ripetitività, dato dal complementare del rapporto tra il numero di parole uniche all'interno di un testo di una canzone e le parole totali del testo stesso. Calcolare l'indice in questo modo avrebbe richiesto poca memoria e poca potenza computazionale. Successivamente si è poi passati all'acquisizione dei dati necessari.

Per avere un'idea più precisa del calcolo dell'indice, se ne riporta qui di seguito la formula:

$$Indice\,di\,Ripetitivit = 1 - \frac{parole\_uniche}{parole\_totali}$$

2.2 Prima fase di raccolta dati (*Spotify*)

Durante la prima fase di ottenimento dei dati, è stato perseguito l'obiettivo di ottenere una lista di nomi di artisti attraverso l'API di Spotify. L'utilizzo della libreria Spotipy ha reso possibile la connessione al server di Spotify e ha permesso di procedere, successivamente, con le richieste. Tuttavia, nonostante l'ampia documentazione, sono sorte diverse difficoltà causate dai limiti imposti al numero di richieste effettuabili e dalla lentezza che ne ha caratterizzato l'elaborazione; ciò ha rallentato la prosecuzione dei lavori, prolungando l'esecuzione di due settimane circa.

Vista l'impossibilità di sottoporre interrogazioni dettagliate, mirate all'ottenimento di uno specifico numero o di una particolare tipologia di artisti (ad esempio i più popolari), ed essendo il progetto orientato a raccogliere dati e informazioni relativi a un'ampia varietà di cantanti appartenenti al panorama mondiale, la query di ottenimento dei dati è stata formulata iterando le lettere dell'alfabeto; per ogni lettera dell'alfabeto, sono stati ottenuti gli artisti il cui nome iniziava per quella specifica lettera.

I risultati ottenuti per ogni artista (nome, popolarità, follower, uri di Spotify e genere), sono stati successivamente memorizzati direttamente su MongoDB attraverso la libreria *Pymongo*, dal momento che l'interesse verteva sullo storing di un database che rendesse possibile una scalabilità orizzontale. Una volta terminata la query, si è subito notato come nei dati ottenuti, vale a dire otto milioni di istanze, uno stesso artista si ripeteva anche molteplici volte; inoltre, alcuni record risultavano incompleti o addirittura privi di informazione e il campo nome, in alcuni casi, conteneva caratteri speciali. È chiaro che tali problemi, in special modo l'ultimo, avrebbero creato non poche difficoltà durante la fase successiva del progetto, ovvero quella relativa all'acquisizione, tramite Genius, dei testi delle canzoni degli artisti.

Considerate tutte queste difficoltà, fattasi nota la grande quantità di pre-processing necessario e ritenendo Python la miglior soluzione alle problematiche suddette, è stato estratto un file CSV a partire dalla collezione salvata su MongoDB. Con l'aiuto della libreria *Pandas* è stata svolta la pulizia dei dati sullo stesso file: una volta corrette le incongruenze e le cause dei problemi precedentemente esposti, è stato possibile ricavare una lista di artisti di numerosità decisamente minore, per un totale di circa 255.000 osservazioni, ma di qualità decisamente migliore.

2.3 Seconda fase di raccolta dati (*Genius*)

Durante la seconda fase, la lista di artisti precedentemente ottenuta è stata sottoposta all'API di Genius, connettendosi a esso attraverso un account appositamente creato, al fine di dar vita a un secondo dataset composto dalle canzoni di un artista e dai relativi testi. Anche durante questa fase sono stati rilevati problemi di lentezza computazionale e, di conseguenza, si è scelto di limitare il numero di canzoni per artista a dieci; una riduzione abbastanza significativa che, però, ha reso comunque necessario l'impiego delle macchine virtuali, le quali hanno restituito i dati richiesti dopo circa una settimana.

Un ulteriore problema è subentrato durante la fase di ottenimento delle canzoni e dei relativi testi. Infatti, dando in input all'API di Genius il nome di un'artista ricavato da Spotify, qualora esso fosse mancato nell'archivio di Genius sarebbe stato sostituito, attraverso il processo di "*Changing Name*", dal nome di un altro artista simile a quello cercato inizialmente. In alcuni casi, questo procedimento ha portato a dei vantaggi poiché, ad esempio, "fabrizio de andrè" veniva cambiato e corretto in "Fabrizio De André" permettendo di ottenere i relativi dati necessari; tuttavia, in molti altri casi questo ha rappresentato un ostacolo: alcuni nomi di cantanti, infatti, sono stati confusi con di nomi di scrittori, attori e/o politici presenti nell'archivio di Genius e hanno restituito interi copioni di film, libri e discorsi che chiaramente differivano dagli scopi di questo progetto.

Molto spesso, i testi riportavano informazioni quali "Testo non disponibile", "Le lyrics verranno pubblicate a breve", e altre stringhe simili. Dopo aver analizzato alcune istanze di questi casi particolari, si è deciso di non considerare quelle che presentavano un numero di parole nel campo "Lyrics" inferiore a 20 e superiore a 2000. Infine, è stato svolto un processo di pulizia necessario a eliminare dal campo contenente il testo delle canzoni i caratteri e le parole superflue all'analisi, quali [Chorus], [Verse] e altri caratteri di escape (/n, /r, et similia).

2.4 Integrazione dei due dataset

Una volta ottenuti i due dataset, si è continuato con la fase di integrazione. Il match è stato effettuato sul campo "Artista", presente in entrambi i dataset, con l'accortezza di valutare come i nomi fossero cambiati dopo il processo di "*Changing Name*". Sono stati studiati due diversi tipi di approccio. Il primo, poi scartato a causa di un'eccessiva lentezza, prevedeva di effettuare operazioni di matching tramite la libreria "*FuzzyWuzzy*" e di accettare i valori di tale match che si collocassero al di sopra di una determinata soglia empiricamente stabilita. Il secondo, invece, nonché quello effettivamente utilizzato, ha richiesto un processo di pulizia, attraverso la libreria "Re", dei nomi degli artisti da tutto ciò che non fosse carat-

teri alfabetici, accenti compresi, per poi effettuare il match.

2.5 Terza fase di raccolta dati (*Spotify*)

Una volta ottenuto il dataset nella sua interezza, i titoli delle canzoni ricavati precedentemente attraverso Genius sono stati sottoposti all'API di Spotify, al fine di integrare ulteriori caratteristiche delle canzoni e permettere ulteriori indagini sui dati. Le informazioni così ricavate riguardano, per ogni canzone, la durata, l'indice di danzabilità, l'energia, l'acusticità, la strumentalità, la presenza o meno del pubblico nella traccia sonora (*liveness*), l'intensità sonora (in decibel, dB), la presenza di testo parlato (*speechiness*), la positività di una canzone (*valence*), la sua popolarità, lo Uniform Resource Locator (URL) e la data di realizzazione della canzone.

Anche il campo genere ha necessitato di un processo di pulizia e di elaborazione. Infatti, i generi presentavano eccessive specificazioni che sono state ignorate per motivi di praticità (ad esempio, il "*Chilenian Rap*" è stato trasformato e considerato solamente come "rap"). Inoltre, erano memorizzati come stringhe non ben strutturate, contenenti parentesi e/o caratteri non informativi (ad esempio: "[alternative indie, pop]"). Si è resa quindi necessaria l'eliminazione dei suddetti caratteri al fine di ottenere una lista di generi più pulita (esempio: "alternative indie, pop") e un'elaborazione più approfondita in modo da rendere il campo ulteriormente informativo. A tale scopo, è stato necessario creare un nuovo dataset in cui ogni riga è stata duplicata tante volte quanti erano i generi contenuti nella lista di generi per ogni artista (Example 1.).

Example 1.

Tabella 1. - Prima

Artista	Genere
Drake	Rap, Pop

Tabella 2. - Dopo

Artista	Genere
Drake	Rap
Drake	Pop

Quest'ultima elaborazione è stata utile al fine di creare delle query che riuscissero a elaborare una corretta aggregazione per genere. Il dataset completo si compone, quindi, dei seguenti campi:

```
nome_artista, popolarità_artista
url_spotify_artista, follower_artista
genere_artista, titolo_traccia
lyrics_traccia, popolarità_traccia
markets_traccia, data_realizzazione_traccia
duration_traccia, url_spotify_traccia
acousticness_traccia, speechiness_traccia
valence_traccia, danceability_traccia
energy_traccia, instrumentalness_traccia
liveness_traccia, loudness_traccia
```

2.6 Calcolo Indice di Ripetitività

Dopo aver ottenuto le lyrics, per ogni testo del dataset è stato calcolato l'Indice di Ripetitività, secondo i criteri già menzionati. In seguito a ciò, sono state aggiunte le seguenti colonne:

```
parole_uniche_traccia
parole_totali_traccia
indice
```

3. Query

Ottenuti i due dataset, quello completo e quello che ha reso possibile l'aggregazione per genere, entrambi sono stati caricati su MongoDB così da poter procedere in maniera efficiente con le interrogazioni. Dopo aver indicizzato i campi su cui la query operava, al fine di velocizzare la computazione, le canzoni dapprima sono state raggruppate in base all'anno di realizzazione per verificare l'andamento dell'Indice di Ripetitività nel corso del tempo. Successivamente, è stato interrogato il dataset appositamente predisposto per le interrogazioni sui generi, in modo da ottenere la media dell'Indice di Ripetitività per ogni genere musicale.

1. `db.songs1.aggregate([{$group:{_id: "$Date",indice_avg: {$avg: "$indice" }}},{$sort:{"_id":1}}])`
2. `db.genres.aggregate([{$group:{_id: "$GENERI_NOSPAZIO", indice_avg: {$avg: "$indice"}}}])`

4. Discussion

I risultati ottenuti rispondono in maniera soddisfacente ai quesiti che hanno dato origine allo sviluppo del progetto. Infatti, aggregando i risultati per anno, è stato possibile notare un leggero ma chiaro trend in salita (Figura 1). Quest'ultimo sembrerebbe confermare che la tendenza musicale degli ultimi anni spinga gli artisti ad essere più ripetitivi rispetto agli anni precedenti.

Tuttavia, effettuando un'aggregazione per genere (Figura 2), si può notare che, contrariamente a quanto ipotizzato inizialmente, non sono la musica Trap e

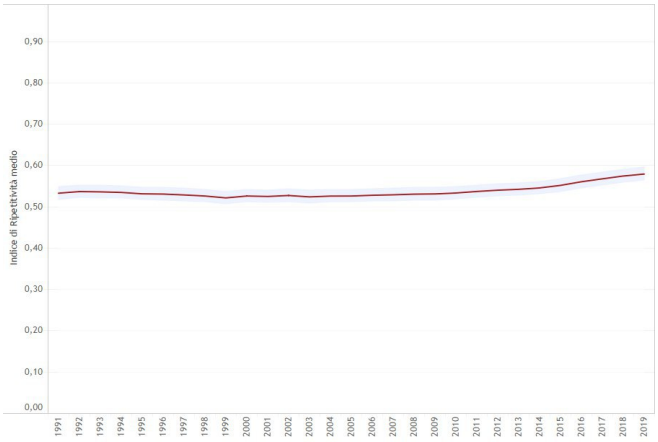

Figura 1. Analisi dell'andamento dell'Indice di Ripetitività nel tempo

gli altri generi in voga attualmente a trainare questa tendenza alla ripetitività, bensì sono perlopiù generi come la musica Dance, quella Elettronica e quella House ad alzare il valore medio dell'Indice di Ripetitività.


```
{ "_id" : "reggaeton", "indice_avg" : 0.6055980253287762 }
{ "_id" : "blues", "indice_avg" : 0.5649127637944101 }
{ "_id" : "jazz", "indice_avg" : 0.5206429419470203 }
{ "_id" : "electro", "indice_avg" : 0.6172337978076158 }
{ "_id" : "dance", "indice_avg" : 0.6289530274001086 }
{ "_id" : "folk", "indice_avg" : 0.5281128227359901 }
{ "_id" : "r&b", "indice_avg" : 0.6104954287442547 }
{ "_id" : "rock", "indice_avg" : 0.5452343224504137 }
{ "_id" : "funky", "indice_avg" : 0.6826091544053159 }
{ "_id" : "punk", "indice_avg" : 0.5224614952930755 }
{ "_id" : "pop", "indice_avg" : 0.5796828226042873 }
{ "_id" : "latin", "indice_avg" : 0.5438495909575778 }
{ "_id" : "hiphop", "indice_avg" : 0.5503040283314597 }
{ "_id" : "house", "indice_avg" : 0.646381377775793 }
{ "_id" : "alternative", "indice_avg" : 0.5549017734974601 }
{ "_id" : "trap", "indice_avg" : 0.5819826791798626 }
{ "_id" : "disco", "indice_avg" : 0.6467151418261288 }
{ "_id" : "indie", "indice_avg" : 0.5543331215659414 }
{ "_id" : "country", "indice_avg" : 0.5647819517506778 }
{ "_id" : "reggae", "indice_avg" : 0.583926204850237 }
```


Figura 2. Output della query di aggregazione per Genere ottenuto su MongoDB

5. Conclusione e sviluppi futuri

5.1 Conclusione

Da una prima riflessione, emerge che i generi più ripetitivi sono quelli che pongono il loro focus sulla base musicale e sull'originalità delle sonorità piuttosto che sul testo. Inoltre, nei generi più ripetitivi, spesso il testo risulta essere molto breve o addirittura composto da versi e parole che, più che rappresentare un testo strutturato vero e proprio, fungono da supporto alla base, la quale rappresenta l'elemento principale del brano musicale. Al contrario, anche da un punto di vista storico, i generi meno ripetitivi trovano la loro stessa essenza nel significato del testo. Risulta, infatti, facile imbattersi in testi dal carattere politico e/o sociologico. Senza dimenticare la forte componente di improvvisazione musicale, tipica di questi generi, la quale si contretizza in un alto tasso di originalità (Figura 3).

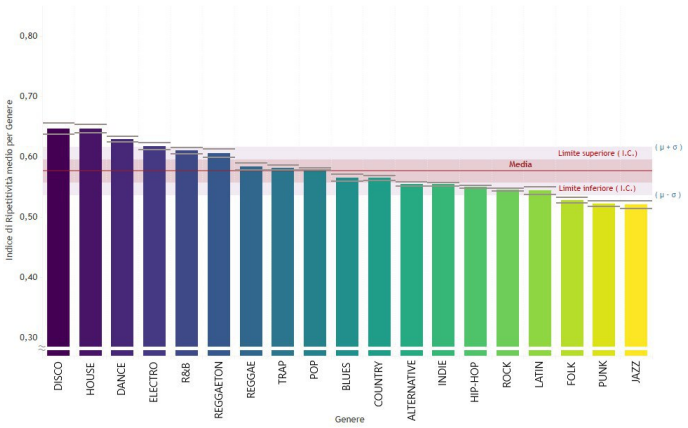

Figura 3. Analisi dell'Indice di Ripetitività per Genere

5.2 Sviluppi futuri

Durante l'implementazione del problema, ci si è accorti che tra i risultati ottenuti dalle richieste avanzate a Spotify, sono stati ottenuti anche nomi di scrittori, giornalisti e registi di cui Genius he restituito i testi dei relativi libri, podcast o copioni. Così, è sorto spontaneo pensare che in futuro si potrebbe estendere la valutazione dell'Indice di Ripetitività anche alle summenzionate categorie di testi e non solo alle canzoni.

Il focus del progetto, è stato posto su un'analisi di tipo temporale e per genere dell'Indice di Ripetitività. Tuttavia non si esclude la possibilità di ottenere risultati significativi utilizzando altri tipi di informazioni, quali le caratteristiche della traccia audio (già presenti nel dataset e inutilizzate) o informazioni anagrafiche riguardanti gli artisti.

Un notevole improvement dell'Indice di Ripetitività sarebbe quello di approcciarsi al testo delle canzoni tramite tecniche di analisi tipiche del mondo della *Text Mining*; in questo modo si terrebbe conto anche della semantica delle parole nel calcolo dell'Indice. Inoltre, potrebbe anche essere interessante approfondire in che misura questa ripetitività dipenda dagli autori delle canzoni, dai produttori o se sia imputabile anche alla casa discografica, la quale stipula contratti solo con artisti che garanti-

scono un alto ritorno economico spesso a discapito dell'originalità dei testi.

Inoltre, avendo a disposizione dei dati rappresentanti la base musicale di ogni canzone, sarebbe interessante analizzare la canzone nella sua globalità. Questo permetterebbe di ottenere un maggiore dettaglio non solo sulla canzone, ma anche sull'artista: sarebbe possibile comprendere se la ripetitività dipenda maggiormente dal testo della canzone o dalla base musicale e, eventualmente, quanto quest'ultima sia simile ad altre già esistenti.

Sharding

Lo sharding è un metodo che viene utilizzato con il fine di distribuire i dati su più macchine. MongoDB sviluppa lo sharding per supportare implementazioni su grandi dataset e ad alto carico computazionale. Sempre in un'ottica di implementazioni future, le quali porterebbero a un aumento delle dimensioni del dataset, sarebbe utile effettuare uno scaling orizzontale, aggiungendo un nuovo nodo per ogni anno a venire. Inoltre, dividere il dataset su più server permetterebbe, da un lato, di aumentare la capacità di storage, e dall'altro, di aumentare le prestazioni in termini di tempo di risposta. In questo modo, ogni macchina, gestendo un sottoinsieme del carico di lavoro, migliorerebbe in termini di efficienza e velocità. Ovviamente, tutto ciò determinerebbe anche una maggiore difficoltà nella gestione dell'infrastruttura.

Lo sharding è molto semplice concettualmente, ma incredibilmente complesso in pratica. L'applicazione deve contenere la logica che permetta di capire la locazione di ogni particolare sottoinsieme di dataset e la logica di indirizzamento delle richieste allo shard corretto. Lo sharding inoltre è spesso associato a una rapida crescita del dataset e conseguentemente dell'architettura, quindi la logica di indirizzamento delle richieste deve essere dinamica. Ogni cluster è composto da:

- *Shard*: che contiene un subset di dati;
- *Mongos*: che fornisce un'interfaccia tra client application e il cluster;
- *Config Server*: memorizzano i metadati e le impostazioni di configurazione del server.

Di seguito viene illustrata una possibile strategia perseguibile: dopo avere abilitato lo sharding sul dataset, con il metodo *enableSharding*, la Shard Key verrebbe applicata al campo "`anno`", dal momento che la query richiedente più tempo è proprio l'aggregazione per anno, con il relativo calcolo della media dell'Indice di Ripetitività.

La Shard Key sarebbe implementata tramite un criterio di tipo *range-based*, partizionando la collezione in base ai valori. Prima di impostare l'anno come Shard Key, col comando *ensureIndex*, viene permesso lo sharding. Al fine di bilanciare i chunk ed evitare i colli di bottiglia, si implementerebbero 30 nodi contenenti ognuno uno shard. Ogni anno compreso tra il 1991 e il 2019 verrebbe assegnato a ognuno di questi shard, poichè il numero di canzoni per anno è elevato. Gli anni precedenti al 1991, poichè contengono un minore numero di brani, verranno invece aggregati in un unico shard. In questa maniera le richieste inoltrate, potranno essere eseguite in perfetto parallelismo, consentendo una maggiore efficienza in termini di tempo di esecuzione.

Riferimenti bibliografici

[1] Guy Harrison (2015) *"Next Generation Databases: NoSQL and Big Data"*, Apress, Berkely (CA), USA, 1st ed.