SpotiStats
Web app per l'analisi di un dataset di canzoni

Virginia Ranciaro - virginia.ranciaro@studio.unibo.it
Irene Vivani - irene.vivani@studio.unibo.it

1. Introduzione

Durante il corso 'Introduzione alla programmazione', svolto durante l'aa. 2023/2024, è stata posta particolare attenzione sulle tecnologie utili per l'elaborazione di dati testuali. Abbiamo quindi pensato di unire il nostro interesse per la musica con le conoscenze acquisite durante il corso per sviluppare un progetto basato sull'analisi di un dataset contenente dati testuali e audiometrici di canzoni presenti sulla piattaforma Spotify, e presentarlo su una web app creata con il framework Streamlit.

Il dataset è stato selezionato dalla piattaforma Kaggle in base alle sue caratteristiche, descritte nel dettaglio nel paragrafo 2.

La prima idea per l'elaborazione di questi dati è anche la prima che viene presentata nella web app, nella pagina 'Word Frequency Analysis', dove vengono analizzati i testi delle canzoni e mostrate le parole più frequenti in un dato intervallo di tempo.

Non volendo limitarci all'esecuzione delle tecniche apprese durante il corso, abbiamo provato a includere nell'analisi anche un task di text similarity e uno di clustering.

Infine, abbiamo impostato il codice affinché fosse compatibile con una presentazione ordinata in una web app costruita grazie al framework Streamlit.

Nella progettazione dell'app, abbiamo cercato di porre l'attenzione non solo sulla presentazione dei risultati delle nostre analisi, ma anche sulla possibilità da parte dell'utente di interagire e selezionare attivamente i contenuti di maggiore interesse, al fine di rendere l'esperienza più piacevole e interattiva.

2. Descrizione del dataset

Il dataset scelto contiene un vasto numero di informazioni di diversi tipi per ogni canzone. È possibile visionare la descrizione completa del dataset nell'apposita pagina Kaggle[1].

Trattandosi di un dataset molto ampio (più di 18.000 canzoni presenti sulla piattaforma Spotify), è stato necessario selezionare alcune specifiche informazioni per svolgere un'analisi significativa.

Innanzitutto, si è deciso di limitare l'analisi alle canzoni in lingua inglese, effettuando una prima scrematura in base ad un criterio oggettivo per alleggerire il dataset ed evitare tempi di elaborazione eccessivamente lunghi.

Nel complesso, le variabili considerate rilevanti per i task svolti sono le seguenti:

- track_name: titolo della canzone;
- track_artist: autore/autrice della canzone;

[1] https://www.kaggle.com/datasets/imuhammad/audio-features-and-lyrics-of-spotify-songs/data

- lyrics: testo della canzone;
- track_album_release_date: data di pubblicazione dell'album che contiene la canzone;
- playlist_genre: genere musicale della playlist in cui il brano è stato inserito;
- speechiness: valore compreso tra 0 e 1 che indica la presenza o meno di parole pronunciate nella traccia audio; i valori sopra 0.66 indicano che la traccia è probabilmente composta solo da parole in assenza di una componente puramente musicale;
- acousticness: valore di confidenza compreso tra 0 e 1 che indica se la traccia è acustica oppure no;
- instrumentalness: valore compreso tra 0 e 1 che indica se la traccia contiene del cantato oppure se è principalmente strumentale;
- language: lingua del testo della canzone.

3. Normalizzazione di testi e titoli

La normalizzazione delle colonne 'lyrics' e 'track_name' è stata effettuata in uno script separato che non fa parte della struttura della web app. Abbiamo dovuto prendere questa decisione a causa dell'ingente quantità di dati del dataset scelto, che quindi portava ad avere tempi di caricamento molto lunghi all'apertura di ogni pagina che avrebbero reso più complessa l'esplorazione della web app.

Abbiamo inoltre scelto di scaricare un file a parte per eliminare le stopwords dall'analisi da una repository GitHub[2]. L'elenco di stopwords per l'inglese incluso nel pacchetto nltk risultava infatti incompleto e insufficiente per poter evitare che casi come 'don' inquinassero i risultati. Alla lista sono state manualmente aggiunte le parole 'wan, gon, ooh, yeah, ya': è stato infatti constatato che, calcolando le cinque parole più frequenti nella pagina 'Word Frequency Analysis', questi cinque elementi, non essendo stati inclusi nemmeno nella più ampia lista di stopwords scaricata appositamente, tendevano ad essere predominanti, coprendo informazioni potenzialmente più interessanti.

Dopo aver ampliato la lista di stopwords, sono state definite due funzioni di normalizzazione diverse:

1. 'normalize_complete' prevede processi di tokenizzazione, rimozione della punteggiatura e delle maiuscole, lemmatizzazione e rimozione delle stopwords;
2. 'normalize_partial' prevede solamente tokenizzazione, rimozione della punteggiatura e delle maiuscole.

Queste funzioni sono state applicate rispettivamente alle colonne 'lyrics' e 'track_name' e 'track_artist'.

Infine, abbiamo salvato le suddette colonne e quella contenente l'anno di pubblicazione del brano in un nuovo file .csv che è poi stato usato per le fasi successive del progetto.

4. Struttura della web app

Per consentire una visualizzazione più ordinata, abbiamo deciso di organizzare la

[2] https://github.com/stopwords-iso/stopwords-en/blob/master/stopwords-en.txt

web app in quattro diverse pagine selezionabili dal menu laterale.
Il codice della pagina principale è contenuto nel file 'Home_page.py', mentre il resto delle pagine è raccolto nella cartella 'pages'.
Per l'integrazione del codice Python nel framework Streamlit, è stato necessario sostituire alcuni comandi standard di Python con quelli specifici previsti da Streamlit, ad esempio 'print()' è stato sostituito con 'st.write()'.
La web app presenta la seguente struttura:

- Home page: contiene il titolo del progetto, una breve presentazione dei contenuti della web app e il link alla pagina Kaggle del dataset di riferimento;
- Word Frequency Analysis: in questa pagina, l'utente può selezionare da un dropdown menu il decennio di suo interesse, avviare il calcolo delle cinque parole più frequenti e visualizzare i grafici che mostrano l'andamento delle occorrenze nell'arco di tempo selezionato;
- Text Similarity: l'utente può selezionare due canzoni da due diversi dropdown menu e avviare il calcolo per visualizzare la percentuale di similarità tra i due testi;
- Clustering: l'utente può visualizzare il numero ottimale di cluster trovati per le caratteristiche audio scelte, i valori medi di queste ultime e la distribuzione dei generi per ogni cluster e lo scatterplot raffigurante i cluster trovati.

4.1 Word Frequency Analysis

Questa sezione del sito permette all'utente di visualizzare un'analisi del contenuto testuale delle canzoni, già normalizzato in uno script separato.
Le canzoni sono state suddivise in base alla loro data di rilascio e raggruppate per decenni, dagli anni Sessanta agli anni Dieci del Duemila: si è optato per l'esclusione delle canzoni precedenti e successive a questo periodo in ragione del basso numero di canzoni a disposizione. A livello di script, l'operazione di raggruppamento è stata effettuata mediante la creazione di un dizionario.
Nella web app è possibile per l'utente selezionare il decennio desiderato da un dropdown menu; avviando poi il widget 'Calcola' vengono visualizzati i dati ad esso relativi.
Al decennio selezionato viene applicata la funzione 'five_most_freq_words', che crea un'unica stringa contenente i testi di tutte le canzoni e da essa estrae le cinque parole più frequenti, che vengono stampate sulla pagina come lista.
Viene successivamente applicata la funzione 'plot_word_occurrences', che itera sulla lista delle parole più frequenti nel decennio selezionato: per ogni parola, viene stampato un line graph che mostra l'andamento delle occorrenze per ogni anno nel decennio.
All'interno di 'plot_word_occurrences' è definita una funzione lambda, che prende come input i testi raggruppati per anno, crea una stringa unica dei testi per ogni anno e conta le occorrenze della parola corrente.
Per l'integrazione dello script nella web app, è stato necessario aggiungere un controllo sul tipo di dato preso in input, per evitare di includere valori numerici che venivano rilevati come int e float.

4.2 Text Similarity

In questa parte del progetto, vengono estratti i testi normalizzati dei due brani selezionati dall'utente per calcolarne la similarità.

Per la vettorizzazione dei testi è stato usato TF-IDF Vectorizer, presente nel modulo 'sklearn.feature_extraction.text', che assegna a ogni parola un peso in base alla sua importanza nel testo, attribuendo un'attenzione minore sulle più comuni e maggiore sulle più rare per poi rappresentare i testi come valori numerici.

Successivamente, dal modulo 'sklearn.metrics.pairwise' è stato importata la funzione 'cosine_similarity', che permette di calcolare attraverso la similarità coseno il grado di somiglianza dei testi.

Il risultato dell'operazione è stato poi presentato come percentuale per una più immediata interpretazione da parte dell'utente.

4.3 Clustering

In questa sezione viene presentata un'analisi di clustering delle canzoni presenti nel dataset, svolta con tecniche di machine learning sulla base di specifiche caratteristiche audio.

Nella prima parte di pre-processing del dataset viene effettuato un controllo per identificare eventuali valori nulli, che vengono rimossi per garantire l'integrità dei dati. Viene poi creato un dataframe contenente soltanto le caratteristiche audio rilevanti per il clustering, ovvero 'speechiness', 'instrumentalness' e 'acousticness': si è scelto di selezionare queste caratteristiche perché particolarmente utili nella distinzione tra generi ad alto contenuto strumentale rispetto a quelli a prevalenza vocale, e nella distinzione tra tracce più acustiche e canzoni con base primariamente elettronica[3].

Successivamente, viene calcolato lo Z-score per identificare e rimuovere gli outlier nelle colonne selezionate, mantenendo solo i dati con Z-score inferiore a 3; il dataframe viene filtrato mediante una maschera booleana che consente di individuare ed escludere le righe con outlier. I dati senza outlier vengono quindi standardizzati con la funzione 'StandardScaler()' del modulo 'sklearn.preprocessing', che implementa la riduzione in scala delle caratteristiche. Si applica poi la PCA (Principal Component Analysis) per ridurre la dimensionalità dei dati a due dimensioni; questo riduce la complessità dei dati e ne facilita la visualizzazione.

Ultimato il pre-processing, viene individuato il numero ottimale di cluster per il raggruppamento (in un range da 2 a 10) mediante il calcolo del Silhouette score, che misura la qualità del clustering. Quindi, all'interno di un ciclo 'for', si calcolano i Silhouette score per ogni diverso numero di cluster nel range stabilito, e la funzione 'np.argmax()' individua il numero di cluster con il valore del Silhouette score più alto.

Si è scelto di utilizzare l'algoritmo K-means per effettuare il clustering dei dati: la funzione 'KMeans()' del modulo 'sklearn.cluster' viene applicata al numero ottimale di cluster individuato, e le etichette dei cluster vengono assegnate ai dati.

Per associare ogni canzone al suo cluster, è stata aggiunta una colonna 'cluster' al

[3] Shirol, Santosh & Kathiresan, & Taqa, Amer. (2023). A Comprehensive Survey of Music Genre Classification Using Audio Files. International Journal of Enhanced Research In Science Technology & Engineering. 12. 2319-7463.

dataframe originario. Viene poi creato un dataframe che contiene il numero di canzoni appartenenti ad ogni genere presenti all'interno di ciascun cluster, ordinando i generi in ordine decrescente di occorrenze.
La pagina web mostra:

- i valori medi delle colonne 'acousticness', 'speechiness' e 'instrumentalness' per ogni cluster;
- la distribuzione dei generi all'interno di ciascun cluster;
- uno scatterplot che fornisce la rappresentazione grafica dei cluster ottenuti.

5. Suddivisione del lavoro

Il carico di lavoro per lo sviluppo del progetto è stato equamente distribuito e nessuno dei diversi compiti è stato svolto esclusivamente da una delle due partecipanti. Il confronto e la collaborazione sono stati essenziali per la creazione della web app nel suo complesso, resa possibile grazie a numerosi incontri sia a distanza sia in presenza.
Nello specifico, Ranciaro Virginia si è concentrata principalmente sui processi di normalizzazione, analisi delle frequenze, clustering ed elaborazione dei grafici per entrambi i task.
Vivani Irene, invece, ha lavorato su text similarity, normalizzazione, progettazione della web app e adattamento del codice Python per l'implementazione tramite Streamlit.

6. Conclusioni

In questo progetto, abbiamo cercato di mettere in pratica quanto appreso durante il corso, ma anche di spingerci oltre ed esplorare tecniche a noi nuove, come nel caso di text similarity e clustering.
Durante la fase di sviluppo sono emerse diverse sfide e difficoltà, soprattutto dovute alle dimensioni e alla struttura del dataset, e alcune parti del progetto hanno dovuto subire revisioni e modifiche. Questo progetto, infatti, ha rappresentato per noi un primo approccio a tecniche di NLP più avanzate. Pur consapevoli che, data la nostra limitata esperienza, possano esserci alcune imprecisioni, siamo interessate a sviluppare ulteriormente le nostre competenze e ad approfondire le metodologie utilizzate in prospettiva di eventuali progetti futuri.

Fonti

Matplotlib, https://matplotlib.org/stable/index.html
NLTK https://www.nltk.org/_modules/nltk.html
Pandas, https://pandas.pydata.org/docs/
Scikit-learn, https://scikit-learn.org/stable/
Scipy, https://docs.scipy.org/doc/scipy/
Streamlit, https://docs.streamlit.io/