|  |
| --- |
| Visualizzatore di features in HTML5 |
| Progetto di Computer Music 2016 |
|  |
| Realizzazione di un visualizzatore di features musicali basato su tecnologie web, principalmente HTML5 e Javascript. |
|  |

Visualizzatore di features in HTML5

Progetto di Computer Music 2016

# Introduzione

L’idea di partenza del progetto è di realizzare un visualizzatore di features usando tecnologie web, ovvero basandosi su javascript e HTML5, possibilmente evitando librerie di terze parti e usando le risorse messe a disposizione dagli standard web sopra citati. In particolare la richiesta comprendeva 4 realizzazioni:

1. Visualizzazione di un segnale monodimensionale: ricevendo un valore in ingresso alla volta costruisce una rappresentazione monodimensionale, come la forma d’onda. Quando lo spazio di rappresentazione termina, viene sovrascritta con un aggiornamento a scorrimento verso sinistra.
2. Visualizzazione di un segnale bidimensionale: da un ingresso vettoriale di valori, rappresenta lo spettro del segnale audio
3. Visualizzazione a istogramma: ricevendo un ingresso vettoriale visualizza un grafico a barre dello spettro.
4. Visualizzazione libera: scelta libera di rappresentazione con forme, colori e oggetti che si muovono seguendo dei parametri delle features.

La generazione delle features è offline, per cui bisogna usare Matlab e il toolbox “MirToolBox”, esportarle in JSON e importarle sulla pagina HTML. La dimostrazione del progetto comprende una pagina web con un database di canzoni tra cui scegliere e visualizzare le features.

Lo studio del problema è iniziato partendo dalla conoscenza dei metodi richiesti e vari tentativi di implementazione, studiando anche i problemi prestazionali riguardanti alcune realizzazioni e trovandone una abbastanza performante.

# Matlab – MirToolBox

La parte di codice su Matlab è molto essenziale e si commenta da sé, in buona parte. Ho costruito uno script che legge nella cartella di test tutti i file audio disponibili e ne fa l’analisi. Successivamente vengono creati dei file JSON per ogni audio con i dati delle analisi e del file.

La suite di metodi MirToolBox permette di operare in gran precisione sui segnali audio, semplificando l’analisi delle features. Sono inclusi anche dei metodi specifici per lo studio delle features in machine learning, ma sono caratteristiche troppo avanzate per quello che mi è stato richiesto. In particolare, il plugin tratta i dati delle analisi come “miraudio object”, struttura complessa in cui salva le caratteristiche analizzate. Per iniziare l’analisi di una traccia, dobbiamo innanzitutto caricarla in memoria attraverso il metodo “miraudio”: qui possiamo già specificare alcune caratteristiche dell’analisi, come Frames, Frequenza di campionamento, Canali ma ciò non toglie che possano esser ri-elaborate anche in futuro, semplicemente applicando un’ulteriore volta “miraudio” all’oggetto.

Una volta estratte le features non è possibile lavorarci direttamente in matlab, bisogna convertirle in variabili manipolabili, quindi vettori/matrici: usando il metodo “mirgetdata” o “get” è possibile convertire alcuni parametri dall’oggetto miraudio a variabile di matlab. I dati non sempre sono normalizzati o in ordine, per questo è sempre bene sperimentare per visualizzare il “come” vengono passati i dati di interesse a Matlab. Per questo, dopo l’analisi ho introdotto una serie di operazioni, per la normalizzazione/estrazione dei dati in un formato compatibile con Javascript e le rappresentazioni.

## JSON – JavaScript Object Notation

Per poter trasferire i dati analizzati da Matlab all’applicazione web, ho usato il formato JSON, come da richiesta. Questo formato di dato puro formattato permette di trascrivere in file di testo puro strutture di oggetti Javascript. Matlab ha un plugin molto comodo e preciso che costruisce un file Javascript a partire da qualunque (o quasi) oggetto Matlab.

Per caricare il file JSON bisogna fare una richiesta a-sincrona al server che carica il file da poi tradurre in oggetti attraverso il metodo nativo di Javascript : JSON.parse(), il quale riceve in ingresso una “stringa” e la trasforma in oggetti.

Ho incontrato dei problemi dal punto di vista di interpretazione e caricamento da parte di alcuni browser internet (in particolare Firefox) che non riuscivano a interpretare i file JSON correttamente attraverso il parser nativo, probabilmente a causa del gran numero di dati nel file. Ho trovato varie documentazioni che parlano del problema del caricamento di file JSON di notevoli dimensioni in browser (consigliano tutti lo split ogni 40mega circa), ma nonostante i miei fossero di dimensioni minori il parser dava problemi dichiarando che il file non era “ben formattato”. Per questo, sotto il benestare del professore, ho usato il metodo jQuery parseJSON che, non so bene per qual motivo, ha funzionato a dovere con tutti i browser e i file JSON che gli ho “dato in pasto”.

# JavaScript e HTML5

Con l’ultima versione di HTML, sono stati universalmente dichiarati dei tag specifici per elementi multimediali per elementi che potessero essere interpretati direttamente dal browser senza dover passare da plugin specifici. In particolare fanno comodo il tag <audio> che definisce un elemento audio riproducibile dal browser, il problema del formato del file audio è legato alle capacità del browser ma attualmente wav e mp3 sono pienamente supportati, e il tag <canvas> che rappresenta un area in cui ci sarà una rappresentazione webGL: questa rappresentazione è gestita in Javascript nativo o con librerie di terze parti che possono semplificare la gestione di elementi quali SHADERS e il calcolo matriciale complesso che sta dietro alle manipolazioni in 3D.

Per le prime tre rappresentazioni ho usato quello che offre JavaScript puro, le rappresentazioni sono veloci e in 2D. Per la rappresentazione personalizzata ho chiesto e ricevuto il permesso dell’utilizzo di una libreria esterna, THREE.js, così da potermi permettere una rappresentazione 3D nelle tempistiche chieste per la realizzazione del progetto.

## Struttura del programma

Lo script viene avviato attraverso un file js chiamato loader.js nel quale si fa il caricamento dei dati generali e la ricerca dei files JSON per ogni traccia audio. Una volta terminato il caricamento della struttura dati di analisi, viene lanciato lo script che crea le rappresentazioni.