

Audio fingerprinting in WebAssembly per l'esecuzione in browser web

Candidato: Davide Pisanò *Relatore:* Antonio Servetti

Politecnico di Torino

Laurea Magistrale in Ingegneria Informatica
25 Luglio 2023



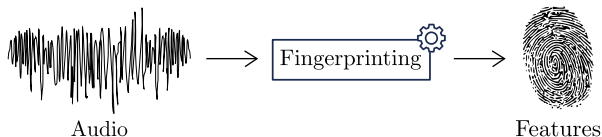
Obiettivo: fingerprinting di segnali audio all'interno del browser

Il Fingerprinting Audio

- ▶ Il **fingerprint audio** è l'impronta digitale di un audio
 - ▶ Ogni fingerprint identifica univocamente un audio
- ▶ Il **fingerprinting audio** crea il fingerprint dell'audio
 - ▶ L'algoritmo estrae delle features *riassuntive* dell'audio
 - ▶ Un audio può essere ricercato note *alcune* sue features

Il Fingerprinting Audio

- ▶ Il **fingerprint audio** è l'impronta digitale di un audio
 - ▶ Ogni fingerprint identifica univocamente un audio
- ▶ Il **fingerprinting audio** crea il fingerprint dell'audio
 - ▶ L'algoritmo estrae delle features *riassuntive* dell'audio
 - ▶ Un audio può essere ricercato note *alcune* sue features



▶ **Riconoscimento audio**

- ▶ Audio simili condivideranno molte features
- ▶ Es: *Shazam*, *ACRCloud*

▶ **Sincronizzazione** di fonti multimediali

- ▶ Più stream provenienti da fonti differenti
- ▶ Es: second screen application

▶ **Riconoscimento** di eventi

- ▶ Reagire alla presenza di un *marker* audio
- ▶ Es: inserimento di un segmento audio personalizzato

▶ **Riconoscimento audio**

- ▶ Audio simili condivideranno molte features
- ▶ Es: *Shazam*, *ACRCloud*

▶ **Sincronizzazione** di fonti multimediali

- ▶ Più stream provenienti da fonti differenti
- ▶ Es: second screen application

▶ **Riconoscimento** di eventi

- ▶ Reagire alla presenza di un *marker* audio
- ▶ Es: inserimento di un segmento audio personalizzato

- ▶ **Riconoscimento audio**

- ▶ Audio simili condivideranno molte features
- ▶ Es: *Shazam*, *ACRCloud*

- ▶ **Sincronizzazione** di fonti multimediali

- ▶ Più stream provenienti da fonti differenti
- ▶ Es: second screen application

- ▶ **Riconoscimento** di eventi

- ▶ Reagire alla presenza di un *marker* audio
- ▶ Es: inserimento di un segmento audio personalizzato

Fingerprinting Audio nel Browser

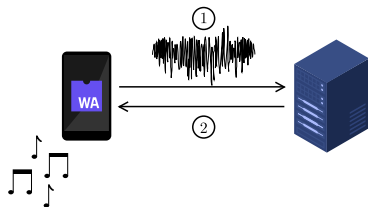
- ▶ Per molti anni capacità dei browser limitate
 - ▶ Scarse performance con Javascript
 - ▶ Niente operazioni in real time
 - ▶ Computazioni a carico del server
- ▶ Ora browser più avanzati
 - ▶ Computazioni parzialmente sul client
 - ▶ Server più scarichi

Fingerprinting Audio nel Browser

- ▶ Per molti anni capacità dei browser limitate
 - ▶ Scarse performance con Javascript
 - ▶ Niente operazioni in real time
 - ▶ Computazioni a carico del server
- ▶ Ora browser più avanzati
 - ▶ Computazioni parzialmente sul client
 - ▶ Server più scarichi

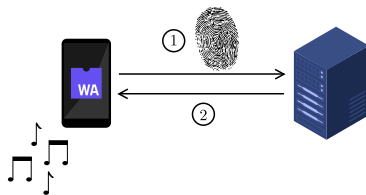
Rivedere il modello C/S

C/S classico



1. Client invia audio
2. Server risponde

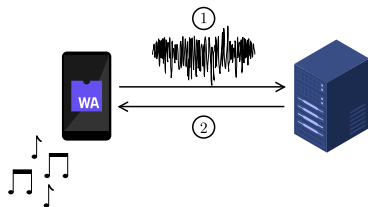
C/S rivisto



0. Client calcola fingerprint
1. Client invia fingerprint
2. Server risponde

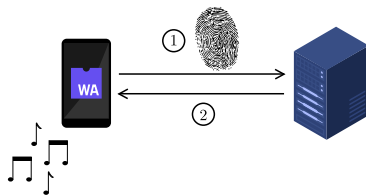
Rivedere il modello C/S

C/S classico



1. Client invia audio
2. Server risponde

C/S rivisto



0. Client calcola fingerprint
1. Client invia fingerprint
2. Server risponde

WebAssembly e WebAudio

È possibile eseguire l'algoritmo nel browser grazie a:

WebAssembly



- ▶ Formato per codice binario
- ▶ Eseguitibile nel browser
- ▶ Target per linguaggi di basso livello

WebAudio



- ▶ API Javascript
- ▶ Manipolazione audio

WebAssembly e WebAudio

È possibile eseguire l'algoritmo nel browser grazie a:

WebAssembly



- ▶ Formato per codice binario
- ▶ Eseguitibile nel browser
- ▶ Target per linguaggi di basso livello

WebAudio



- ▶ API Javascript
- ▶ Manipolazione audio

WebAssembly e WebAudio

È possibile eseguire l'algoritmo nel browser grazie a:

WebAssembly



- ▶ Formato per codice binario
- ▶ Eseguitibile nel browser
- ▶ Target per linguaggi di basso livello

WebAudio

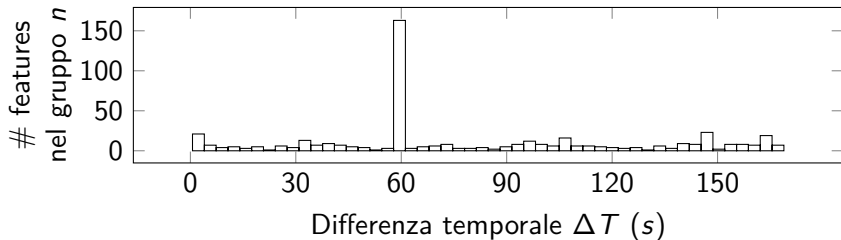
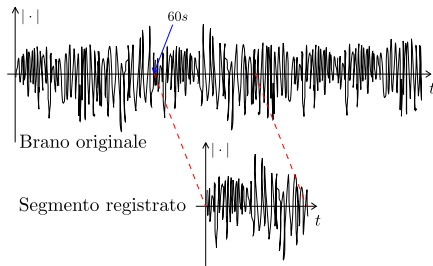


- ▶ API Javascript
- ▶ Manipolazione audio

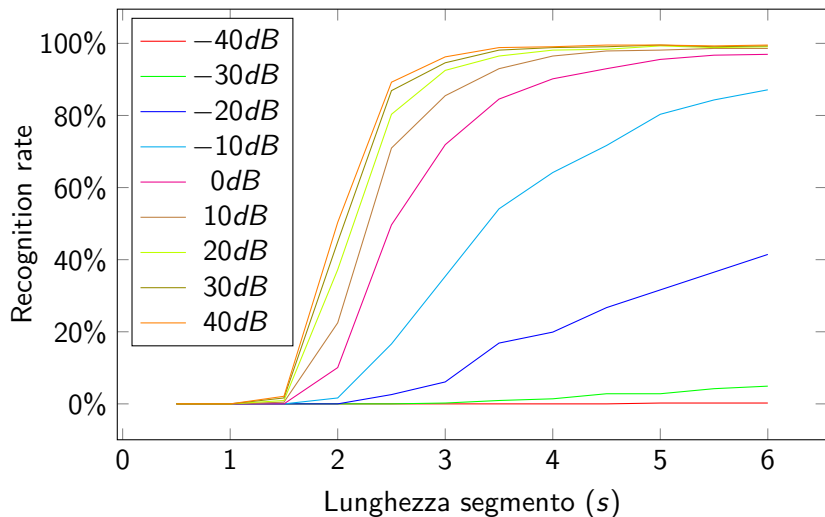
Matching

- ▶ Basato su tre idee:
 1. $\Delta T \geq 0$ tra features brano originale e registrazione
 2. ΔT costante tra features brano originale e registrazione
 3. Uguaglianza tra features basata su hash
- ▶ Quindi bisognerà:
 1. Raggruppare per ΔT e per id brano originale
 2. Ogni gruppo da n features
 3. Ordinare per n crescente
 4. **Miglior match:** gruppo con n maggiore

Riconoscimento: Visualizzazione



Recognition Rate



Second Screen Application

