CACiC 2023
CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN

Experiencia de desarrollo de una aplicación de reconocimiento de canciones mediante la técnica de huella de audio

Rodrigo Lago[1], Veronica Scholz[1], Roman Bond[1],
Martin Morales[1,2], Diego Encinas[1,3]

[1] SimHPC-TICAPPS. Universidad Nacional Arturo Jauretche.
Florencio Varela, 1888, Argentina.

[2] Centro CodApli. FRLP. Universidad Tecnológica Nacional. La Plata, 1900, Argentina.

[3] Instituto de Investigación en Informática (III-LIDI). Facultad de Informática, Universidad Nacional de La Plata - Centro Asociado CIC. La Plata, 1900, Argentina.

{rodrigazo98, vero.scholz}@gmail.com, {rbond, martin.morales, dencinas}@unaj.edu.ar

Abstract. El artículo presenta el desarrollo de una aplicación para reconocer canciones, utilizando la tecnología de Reconocimiento Automático de Contenido (ACR). Se comparan las soluciones ACR Cloud y Music Recognition para el procesamiento en la nube en tiempo real. También se realizan pruebas de rendimiento con diferentes calidades de audio y se analizan los resultados. Se discuten los usos de la huella digital de audio en la gestión de derechos de autor y el reconocimiento de canciones. La aplicación se muestra efectiva en reconocimiento de música en tiempo real, lo cual permite aplicar esta tecnología en otras áreas.

Keywords: audio, fingerprint, aplicación, Nyquist-Shannon, muestreo.

1. Introducción

El reconocimiento automático de contenido o ACR por sus siglas en inglés (Automated Content Recognition) se refiere a la capacidad de los sistemas informáticos para identificar y analizar de manera automática el contenido presente en medios como videos, imágenes y audio. En este trabajo se presenta el desarrollo de una aplicación que pueda reconocer canciones, una herramienta para docentes y estudiantes de conservatorios o escuelas de música. En un comienzo se decidió entre dos Clouds: AWS con la app Music Recognition [1] y ACRCloud. Debido a los cortos plazos de prueba de Music Recognition de Sensifai [2] y a la escasa documentación fue imposible ponerla a prueba y se decidió continuar el desarrollo de la aplicación con ACRCloud.

El desarrollo se divide en dos etapas. En la primera etapa, se definen las pruebas a realizar y qué metodología utilizar. En la segunda etapa se utilizó React Native y Expo para la implementación de la aplicación de reconocimiento de canciones. A su vez, se obtienen las métricas para su posterior análisis. Luego se incorpora un simple afinador que reconoce notas musicales.

2. Trabajos relacionados

Google en la actualidad permite buscar una canción diciendo “Hey Google que canción estoy escuchando”, pero en este caso aparece el video de la canción que lleva a YouTube. Devuelve la información relacionada y sobre el artista. Claro Música también cuenta con reconocimiento de canciones en este caso es pago junto a la suscripción. Todas estas aplicaciones utilizan ACR Cloud para el desarrollo del reconocimiento de sonido, esto se puede ver en la página de ACR Cloud donde se encuentra una lista de empresas que utilizan su servicio.

La plataforma KKBox [3] se ha convertido en una de las bibliotecas de música Asiática. Cuenta con más de 40 millones de pistas para elegir y está disponible únicamente en algunas regiones de Asia. Mediante una colaboración con ACR Cloud incluyeron en su plataforma un reconocimiento de música que permite a los usuarios la opción de “reconocimiento de música”.

Shazam [4] es una aplicación para móvil que permite la identificación de música. Permite grabar una muestra de música y, al igual que ACR Cloud, crea una huella digital acústica a partir de dicha muestra y se compara con una base de datos para encontrar coincidencias. En Shazam, se brinda al usuario la siguiente información:

- Nombre de la canción.
- Información de la banda: conciertos, otras canciones.
- Letra.
- Información de la pista.

3. Desarrollo

ACRCloud brinda servicios automáticos de reconocimiento de contenido mediante algoritmos de Huellas Dactilares de Audio y Consulta por Zumbido. Posee 40 millones de bases de datos de huellas digitales de música. Los usuarios gratuitos pueden enviar 100 consultas por día en un tiempo de respuesta de 2 a 4 segundos en promedio. Brinda un listado de librerías compatibles con React Native para hacer la conexión con su api (pero no para React js) [5].

Soluciones ACRCloud:

- Soporte de carga de contenido personalizado y reconocimiento.
- Monitoreo de transmisión en tiempo real para música o anuncios.
- Admite detección de canales de TV en vivo e interacción de segunda pantalla.
- Identificación de contenido con derechos de autor y servicio cumplimiento.
- Soporte de servicio de identificación fuera de línea para dispositivos inteligentes y aplicaciones.
- Medición de audiencia.

ACR Cloud puede hacer coincidir la "huella digital" o ID de una canción de destino (características acústicas clave como el tempo y los tonos de una pieza) con una base de datos de referencia de millones de pistas (Figura 1).

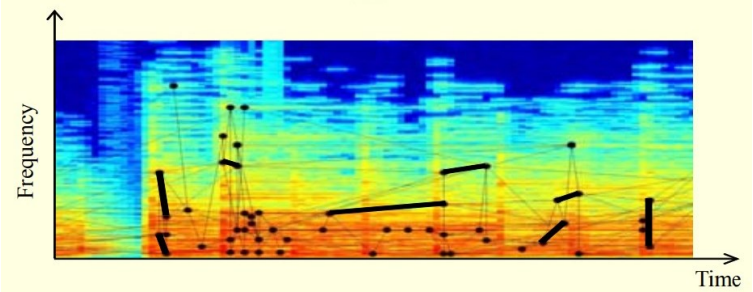

Fig. 1. Huella digital de audio [6].

El primer paso es la creación de la base de datos, donde se ingresan fragmentos de canciones, se extraen las huellas y se almacenan. En el siguiente paso, el usuario de la aplicación captura el audio, la app genera la huella y se envía la consulta a la base de datos. La respuesta indicará al usuario si hubo coincidencias. En la Figura 2 se puede observar los pasos a seguir.

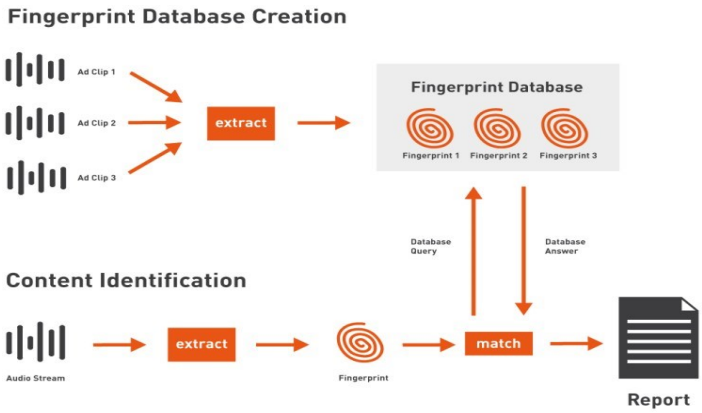

Fig. 2. Proceso de identificación de sonidos mediante huella digital. [7]

Entre los usos que se le pueden dar a la huella digital de audio se destacan:

- Derechos de autor: ACRCloud ayuda a controlar el uso de los derechos de autor de sellos discográficos. Las discográficas aplican los algoritmos de reconocimiento de contenido automatizado de la startup para monitorear obras de programas de radio y televisión, contenido generado por usuarios (UGC) en plataformas como YouTube o TikTok, o cualquier contenido que debería estar pagando derechos de autor.
- Reconocimiento de canciones: el reconocimiento de canciones de ACRCloud es una tecnología iniciada por Shazam, que los fabricantes de teléfonos (Huawei, Xiaomi) han integrado en sus dispositivos.

En [8] señalan que una huella digital de audio es una forma compacta basada en contenido que resume una grabación de audio.

La gráfica en el dominio del tiempo muestra los cambios de la amplitud de la señal con respecto al tiempo (Fig. 3.). Es una gráfica de la amplitud en función del tiempo, donde la fase y la frecuencia no se miden explícitamente. Un gráfico frecuencial muestra las componentes de la señal según la frecuencia en la que oscilan dentro de un rango determinado. Esta gráfica se tiene en cuenta para crear la huella digital.

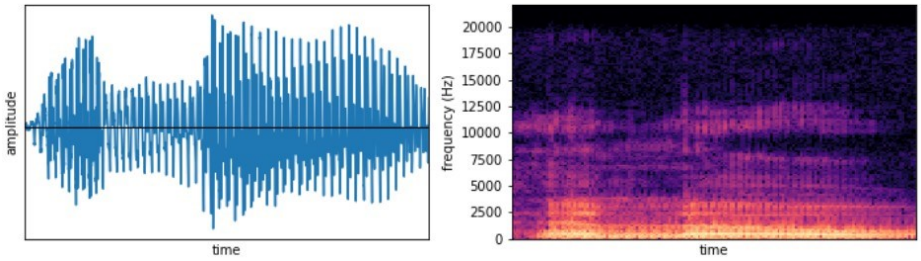

Fig. 3. Gráficas de amplitud en función del tiempo [5].

3.1 Primera etapa

Durante esta etapa se agregó React Native a la aplicación ya que, además de tener un buen desempeño en Mobile, permitió utilizar las librerías brindadas por ACRCloud. En conjunto con React Native, se incluyó Expo JS, que es un framework para poder trabajar con React Native el cual permite la emulación de dispositivos y la instalación de la aplicación en dispositivos físicos, lo que facilita el desarrollo.

3.2 Pruebas de rendimiento

Durante las pruebas de rendimiento se utilizó Python. Se trabajó con archivos locales de diferentes calidades para obtener los tiempos de respuesta favorables del cloud en función de las calidades de archivos enviados.

Se calcula el tiempo promedio que tarda en procesar archivos según la calidad del archivo para estimar la calidad de audio mínima y así obtener una respuesta en el menor tiempo posible. Entre las métricas que servirán de ayuda para obtener conclusiones se encuentran:

1. Tasa de error de detección: Esta métrica mide la cantidad de veces que el cloud no puede identificar correctamente una canción en función de la calidad del archivo.
2. Tiempo de procesamiento: El tiempo que tarda el algoritmo en procesar el archivo de audio en milisegundos.
3. Calidad de audio (baja, media, alta) depende del formato de audio y la frecuencia de muestreo en Hz.
4. Tamaño de archivo en bytes.
5. Duración del archivo: La duración del archivo de audio en segundos.

Durante estos cálculos surgieron los siguientes interrogantes: ¿La calidad del audio afecta significativamente la precisión de la detección de canciones? ¿El tiempo de procesamiento de un archivo de audio está correlacionado con su calidad?

3.2.1 Objetivo de las pruebas:

Medir los tiempos (en segundos) que tarda en procesar archivos de audio, el cloud de detección de canciones, según la calidad del archivo. Con el objetivo de estimar la calidad de audio mínima requerida para obtener una respuesta favorable y en el menor tiempo posible.

3.2.2 Metodología empleada:

Para construir el indicador se seleccionaron 4 calidades de audio (excelente, buena, regular y baja) por cada canción y se cargan carpetas por fragmento de canciones de 7 segundos. Se hace la solicitud y se reportan los resultados de *éxito* o *error.* También se obtiene el *coste de tiempo de procesamiento* de los casos favorables y se contabilizan los casos no favorables.

3.2.3 Fuente de información y tipos de registros:

Se envían archivos de audio (wav, mp3, ogg), luego se almacena el reporte (en formato json) y se analizan los resultados en un gráfico comparativo.

3.2.4 Tiempos esperados:

Se clasifica el tiempo de procesamiento a esperar en 3 categorías:

- Óptimo: Un tiempo de procesamiento de hasta 0.5 segundos
- Aceptable: Un tiempo de procesamiento entre 0.5 y 2 segundos
- No aceptable: Timeout, sucede cuando no se reconoce la canción.

3.2.5 Calidades empleadas:

Grupo 1: Excelente calidad

- Formato WAV (frecuencia de muestreo de 96000 KHz - 96 KHz, 2 canales).
- Formato FLAC (frecuencia de muestreo de 44100 KHz, 2 canales).

Grupo 2: Buena calidad

- WAV (frecuencia de muestreo de 48000 KHz - 48 KHz, 2 canales).
- MP3 (tasa de bits de 256Kbps, 2 canales).

Grupo 3: Calidad regular

- WAV (frecuencia de muestreo de 22050 KHz - 20 KHz, 2 canales).
- MP3 (tasa de bits de 128Kbps, 2 canales).
- FLAC (frecuencia de muestreo: 8000 KHz, 11025 KHz, 12000 KHz, 16000 KHz, 22050 KHz, 24000 KHz, 32000 KHz).

Grupo 4: Baja calidad

- MP3 (tasa de bits de 64 Kbps, 1 canal).
- AMR (tasas de bits constantes: 12.20 Kbps, 10.20 Kbps, 7.95 Kbps, 7.40 Kbps, 6.70 Kbps, 5.90 Kbps, 5.15 Kbps, 4.75 Kbps)

3.2.6 Resultados obtenidos

Entre los resultados de la medición se observó que algunos archivos de mayor calidad consumen tiempos de procesamiento más largos. Esto se debe, en primer lugar, a que algunos archivos de mayor calidad tienden a ser más grandes en tamaño, lo que conlleva un tiempo adicional de carga, como se puede apreciar en la Figura 4. A su vez, estos archivos pueden contener mayor cantidad de información, aumentando la complejidad del procesamiento y originando tiempos de detección más extensos en comparación con archivos de menor calidad.

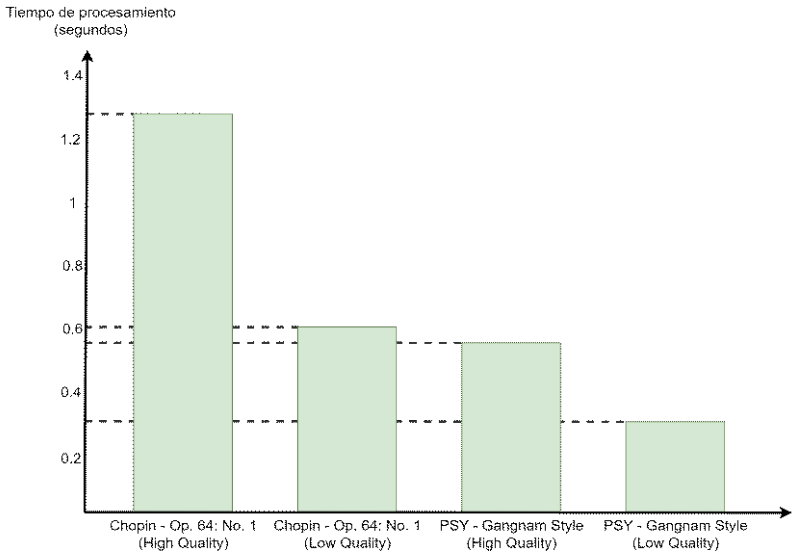

Fig. 4. Gráfico de tiempos de procesamiento de pruebas realizadas

Sin embargo, se puede observar que en casos donde las grabaciones de las canciones poseen más cantidad de ruido y menos definición en los instrumentos, como en Sodom - M 16 (ver Figura 5), al reducir la calidad del archivo, no se realiza el match con la canción original.

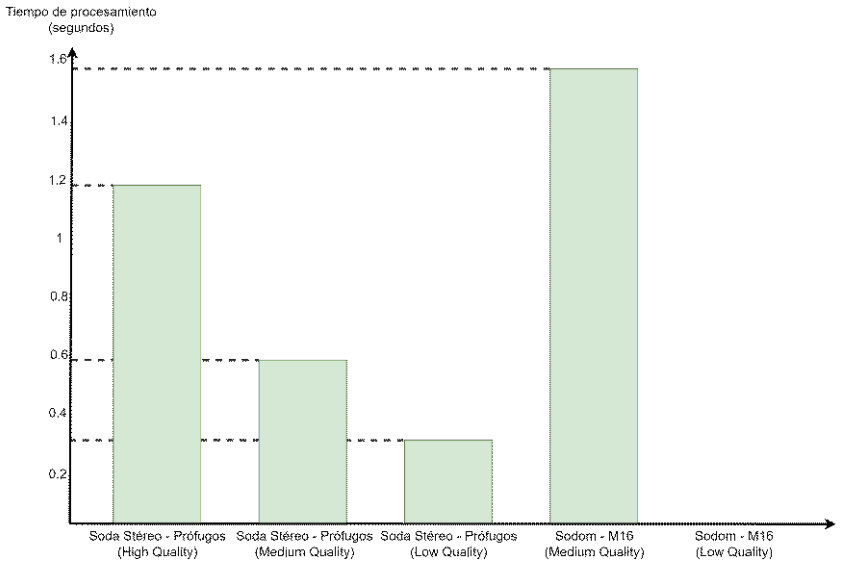

Fig. 5. Gráfico de tiempos de procesamiento de pruebas realizadas.

3.3 Muestreo del audio

El sonido es una vibración que se propaga como una onda mecánica de presión y se desplaza en el medio (ej: aire). El micrófono traduce esta señal de presión continua en una tensión analógica continua [9]. Luego se realiza una conversión analógico-digital. Como siempre se introduce una pequeña cantidad de errores al cuantificar la entrada, se realizan muchas conversiones en pedazos muy pequeños de la señal, el "sampling".

Según el teorema de Nyquist-Shannon [10] la tasa de muestreo necesaria para capturar determinada frecuencia debe ser el doble de la frecuencia muestreada, es decir, para capturar todas las frecuencias del oído humano (20 Hz-20 KHz) se debe realizar el el muestreo a 44.100 Hz.

Como resultado del sampling obtenemos una señal grabada en función del tiempo, el cambio de amplitud de la señal a lo largo del tiempo. A través de Fourier sabemos que una señal puede descomponerse en varias señales sinusoidales simples, cuya sumatoria es igual a la señal descompuesta. La serie de los sinusoides que juntos forman la señal en el dominio del tiempo original se conocen como serie de Fourier. La transformada de Fourier deconstruye una representación en el dominio de tiempo de una señal en la representación del dominio de frecuencia.

Trabajar con funciones en términos de la frecuencia facilita el análisis de la señal digital para estudiar el espectro y determinar las frecuencias que están presentes y/o faltan, hacer filtrados, aumentar frecuencias o reconocer el tono exacto de las frecuencias.

En el caso del afinador, para calcular la frecuencia en tiempo real, se tiene un bucle de grabación (Figura 6) donde se almacenan muestras de audio en el buffer, luego se procesan las muestras de audio y finalmente, se vacía el buffer.

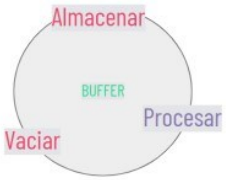

Fig. 6. Bucle de grabación.

El algoritmo recibe la muestra de audio y la multiplica por una función ventana para evitar bordes y aplica la transformada rápida de fourier para obtener la señal en su representación en el dominio de la frecuencia.

Es la misma representación estática que, al igual que en la huella de audio digital, actúa como un tipo de firma dactilar de la señal.

3.4 Segunda etapa

Luego de esta primera fase se utilizó el framework de front-end React Native para el desarrollo móvil e híbrido además de Expo JS para la emulación y pruebas en los distintos dispositivos. Utilizados ambos en conjunto brindan un ecosistema bastante completo para el desarrollo y pruebas de una aplicación móvil híbrida.

Una vez implementado el código, ACR Cloud proporciona librerías las cuales son la configuración para obtener el reconocimiento de lista de canciones. Estas librerías brindan las herramientas para consultar en la nube y traer la información deseada.

ACRCloud facilita que las consultas sean a la nube o de modo offline pero el objetivo es ver la funcionalidad de la nube. En el caso de querer implementar un sistema de reconocimiento de canciones en modo offline, todo el procesamiento se realiza en el dispositivo móvil.

3.5 Cálculos

Para el proyecto se evaluó distintas calidad de reproducción y el tiempo de respuesta por cada tema se tomó 50 muestras con variación de calidad. Para hacer el reconocimiento, se hizo en ambiente libre de sonidos externos, con variación de partes de la canción este programa toma 10 segundos en cada muestra. Resultó útil realizar la probabilidad que se encuentre cierta canción variando sus calidades de reproducción, de estos casos favorables se realizó un promedio de respuesta del software lo cual se aprecia en las Figuras 4 y 5 . La siguiente ecuación se utiliza para saber la probabilidad de encontrar con éxito una canción. P(A) es la probabilidad de que encuentre la canción, casos favorables son las veces que se encontró la canción y los casos posibles son la cantidad de casos analizados.

$$P(A) = \frac{Casos\ Favorables}{Casos\ Posibles} \quad (1)$$

De los aciertos se realizó el promedio para representar en los gráficos por lo que:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad (2)$$

Donde X es el promedio. En este caso la canción es poco probable de encontrarla en baja calidad, pero el tiempo de respuesta es menor al de mayor.

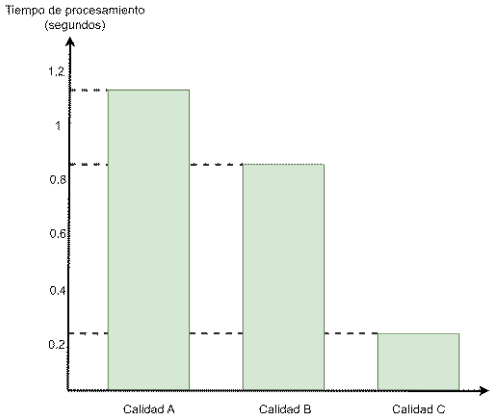

Fig. 7. Gráfico comparativo de calidades.

En la Figura 7 se analizan tres calidades de sonido. En la barra A se obtuvo un promedio de 1,17 de los casos favorables que fueron un total de 28 de 50 pruebas por la ecuación 1.1 la probabilidad de encontrarla es 0,56. La barra B se trata de una calidad media, con un total de 43 casos favorables de 50, y su probabilidad es de 0,86. En la barra C el tiempo de respuesta promedio es de 0,24, los casos favorables fueron 30 lo que da 0,6 en este caso la probabilidad de encontrar el tema tuvo mayor éxito con la producción regular y baja.

Luego de iniciar la aplicación aparecerá un botón que dice buscar, se pasa a detallar los pasos que sigue el algoritmo implementado.

1. Primero se pide el permiso para poder acceder al micrófono.
2. Se debe presionar el botón empieza a grabar durante 7 segundos la canción.
3. La grabación se envía al servidor de ACRCloud.
4. Si encuentra la canción devuelve un archivo JSON con varios datos.
5. Luego se renderiza el nombre de artista, canción y tiempo de procesamiento del cloud.
6. Si no la encuentra devuelve null y en pantalla se ve “canción no encontrada”.

4. Conclusión y trabajos futuros

Se puede observar que al desarrollar una aplicación en tiempo real los tiempos de respuestas son mínimos con respecto a otros. En cuanto a los cálculos se vio una variación en los aciertos a la calidad y esto se puede asociar a que el tema analizado

pudo tener mayor reproducción que otros. Con este tipo de aplicación se puede ayudar a un grupo de personas, de una determinada actividad, pero esta aplicación sólo podría ser un ejemplo para futuros desarrolladores que quieran facilitar en un aspecto la vida de otros usuarios. Considerando que un sistema similar podría reconocer diferentes tipos de ruidos, se puede tomar como ejemplo la detección de ruidos en un motor o mecanismo. Al realizar pequeñas modificaciones al código actual, se podrían resolver problemas adicionales para los usuarios, abordando diversas problemáticas. De esta manera, se mejoraría la funcionalidad del sistema, permitiendo su aplicabilidad en distintos contextos y maximizando su utilidad para beneficio general.

Referencias

1. https://aws.amazon.com/marketplace/pp/prodview-tnt5od5jzxct2
2. https://sensifai.com/
3. Zheng, Hanyue. KKBox subscription prediction: an application of machine learning methods. Diss. 2018.
4. Wang, Avery. "The Shazam music recognition service." Communications of the ACM 49.8
5. (2006).
6. https://docs.acrcloud.com/
7. Kim, Hyoung-Gook, and Jin Young Kim. "Robust audio fingerprinting method using prominent peak pair based on modulated complex lapped transform." ETRI Journal 36.6 (2014).
8. https://www.mufin.com/company/technology/
9. Cano, Pedro, et al. "Audio fingerprinting: concepts and applications." Computational intelligence for modelling and prediction (2005).
10. Oliver Gil, José Salvador. "Captura de señales de audio y preparación para su compresión." (2018).
11. Por, Emiel, Maaike van Kooten, and Vanja Sarkovic. "Nyquist–Shannon sampling theorem." Leiden University 1.1 (2019).