Introducción al Reconocimiento de Patrones 2011

Proyecto final

Reconocimiento de canciones usando DTW

María Inés Camacho

4.197.302-6

Gonzalo Pereira

3.809.948-1

Objetivo

Reconocimiento de piezas musicales utilizando la técnica de Dynamic Time Warping

Resumen

Se busca generar un método que permita a partir de unos pocos segundos de audio desconocido, identificar una pieza musical. Se utilizará la técnica de clasificación Deformación Temporal Dinámica (DTW, Dynamic Time Warping).

Descripción del problema

Se creó una base de datos de canciones etiquetadas y una base de consultas de audios a identificar. Debido a que la idea es que el sistema pueda utilizarse con audios obtenidos de forma cotidiana, se trató de simular situaciones parecidas a las mismas tanto a nivel domestico como comercial por ejemplo:

- Escuchar una canción en la radio y grabarla con un celular, mp3 o similar.
- Grabar una canción y que haya una voz u otro sonido interfiriendo
- Estiramiento y compresión de la pieza musical.

Metodología

En vez de trabajar con las piezas musicales, lo haremos con fingerprints. Estas consisten en un mapeo de las canciones a elementos menos complejos y por lo tanto más fáciles de manejar. La relación de igualdad de dos piezas musicales quedará determinada por la igualdad o no de sus fingerprints. Este proceso es análogo al realizado por los hash en criptografía, donde, se mapean objetos largos (X e Y) en objetos H(X) y H(Y) y la comparación entre X e Y queda definida por la comparación de H(X) y H(Y).

El sistema de fingerprints tendrá los siguientes parámetros:

- Robustez: Se busca poder identificar una pieza aun cuando la misma este degradada. Esto implica que el mapeo debe estar basado en características que se mantengan invariantes para cierto grado de degradación
- Confiabilidad: Se busca que la tasa de confusión entre dos pistas diferentes sea baja, es decir que dado una pista musical sea baja la probabilidad de que sea identificada como otra pista.
- Tamaño de la fingerprint: Cuánta memoria es necesaria para casa fingerprint
- Granularidad: Cuántos segundos por canción son necesarios para poder identificarla

- Velocidad de procesamiento: Cuánto tiempo se demora en encontrar una fingerprint en una base de datos

Cada uno de estos parámetros pesa sobre los otros. Por ejemplo si se quiere poca granularidad es necesario fingerprints mas largas para poder mantener la confiabilidad ya que los falsos positivos se reducen al aumentar el tamaño de la fingerprint.

Las fingerprints tratan de capturar las características más relevantes teniendo en cuenta que tanto la obtención como la búsqueda de las mismas debe ser fácil. Es necesario establecer que características son útiles, en este caso las características del tipo semánticas quedan descartadas ya que entre otras razones, son ambiguas (por ejemplo puede haber varias opiniones respecto al género de una pieza), y por lo general son más difíciles de computar. Por lo tanto se tendrán en cuenta solo las características no semánticas, las cuales son de naturaleza más matemática.

Obtención de la fingerprint:

Siguiendo lo presentado en los papers: "A Highly Robust Audio Fingerprinting System" de Jaap Haitsma y Ton Kalker y "Robust Audio Hashing for Content Identification" de Jaap Haitsma, Ton Kalker y Job Oostveen se realizó el siguiente procedimiento para obtener la fingerprint

Primero se divide la pieza de audio en tramas. A cada una de esas tramas le corresponderá una subfingerprint, y el conjunto de todas las subfingerprint será la fingerprint del audio a analizar.

Las tramas se eligen solapadas entre sí. Cada trama tiene un largo de 0.4 segundos ponderada por una ventana Hann de 31/32. El solapamiento permite que dos subfingerprints consecutivas sean muy similares y por lo tanto la fingerprint varíe lentamente en el tiempo.

Debido a que las características del audio más relevantes surgen del espectro del mismo, se realiza la transformación de Fourier en cada trama y solo se toma en cuenta el módulo de la misma ya que el sistema auditivo humano es insensible a la fase.

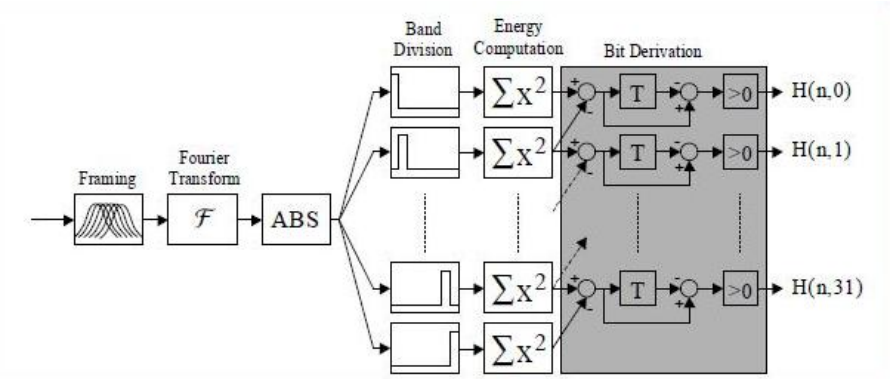
Cada subfingerprint es un vector de 32 bits, para obtenerlas se toma el rango de frecuencias audibles (300 a 3000 Hz) y se lo divide en 33 bandas no solapadas tomadas logarítmicamente, esto último surge de que el sistema auditivo humano también opera en bandas aproximadamente logarítmicas.

Ambos papers expresan que experimentalmente se prueba que una característica muy robusta es tomar el signo de la diferencias de energía simultáneamente en el eje de la frecuencia como en el del tiempo.

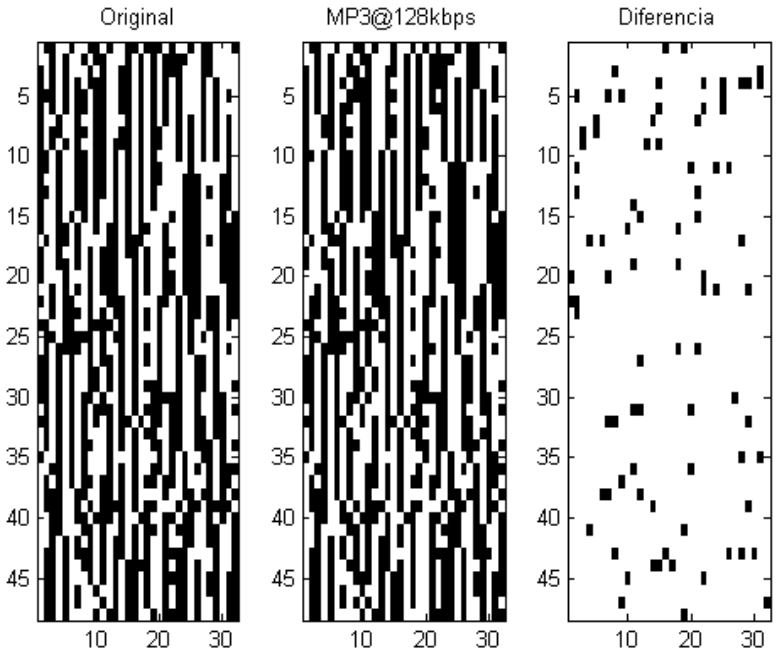
Esto se traduce en lo siguiente, llamamos F(n, m) al valor del bit m-ésimo de la subfingerprint n-ésima

$$F(n,m) = \begin{cases} 1 \text{ si } E(n,m) - E(n.m+1) - \big(E(n-1,m) - E(n-1,m+1)\big) > 0 \\ 0 \text{ si } E(n,m) - E(n.m+1) - \big(E(n-1,m) - E(n-1,m+1)\big) \leq 0 \end{cases}$$

Donde $E(n,m)$ corresponde a la energía de la trama n y banda de frecuencia m.
El proceso de la generación de fingerprints se ilustra en la siguiente figura:



A continuación se muestra un ejemplo de la fingerprint obtenida para 1 segundo de la canción "Auto Rojo" de Vilma Palma e Vampiros, donde se observa la comparación de la canción original, con la compresión MP3 a 128 Kbps, y la diferencia entre ambas:



Para poder establecer cuál es la distancia entre dos fingerprints se utilizó distancia de edición, en particular por tratarse de matrices binaria se toma la distancia de Hamming.

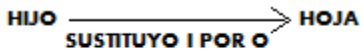
Distancia de edición:

Busca definir una medida de distancia entre dos strings, la distancia de edición cuenta la cantidad de operaciones son necesarias para transformar un string en el otro. Las operaciones que se toman en cuenta entre los strings *x* e *y* son:

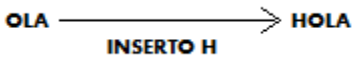
- Sustitución: Donde se cambia un carácter en el string *x* por el correspondiente en el string *y*
- Inserción: Se introduce en *x* un carácter presente en *y*, lo cual aumenta el tamaño de *x* en 1.
- Eliminación: Se borra un carácter en *x* y en consecuencia se reduce en uno el tamaño de *x*

Ejemplos:

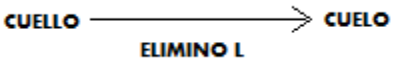
- La distancia de edición entre el string x= "HIJO" e B="HOJA" es uno ya que con una sustitución de la O de B por una I obtengo A



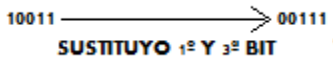
- En el caso que quisiéramos ver la distancia de edición entre el string A= "OLA" y el string B="HOLA" observamos que es necesario hacer una inserción al comienzo de A para obtener B y por lo tanto la distancia de edición es 1.



- La distancia de edición es uno para el caso de A="CUELLO" y B="CUELO" ya que alcanza con borrar una de las L de A para obtener B



- Buscamos la distancia entre los vectores A=00111 B=10011: vemos que es necesario realizar una sustitución en el primer y tercer bit de B para transformarlo en A, y por lo tanto la distancia entre A y B es dos.



- La distancia entre los strings “exhausted” y “excused” es 3 ya que “excused” se convierte en “exhasuted” si insertamos una h luego de ex, cambiamos la c por una a y luego volvemos a insertar una t luego de la s y antes de ed. Es decir:

“excused”➔ “exhcused”➔”exhaused”➔”exhausted”

Como estamos trabajando con vectores binarios de igual largo, la única operación que tendrá lugar será la de sustitución, es por esto que en este caso la distancia entre dos fingerprints será simplemente el número de bits en que difieren. Es decir, que la distancia entre fingerprints la podemos calcular realizando un XOR y luego sumando los unos obtenidos, lo que se conoce como distancia de Hamming.

Análisis de falsos positivos:

Consideraremos que dos piezas musicales tienen el mismo origen si sus fingerprints difieren en menos de un cierto umbral T de bits. Este umbral determina la tasa de falsos positivos obtenidos, es decir la cantidad de veces que se establece que son iguales dos audios que en realidad no lo son. Es claro que a menor umbral, menos falsos positivos tendremos, pero por otro lado, se incrementa la tasa de falsos negativos, es decir que aumenta la tasa de tener dos audios iguales pero no identificarlos como tales.

Asumiremos que la obtención de la fingerprint se basa en bits iid (independientes e idénticamente distribuidos). El número de bits errados tendrá una distribución binomial (n, p) con n el número total de bits de la fingerprint y p =0.5 la probabilidad de que el bit sea 0 o 1. El número de tramas obtenidas en un audio de T segundos viene dado por la ecuación:

$$N = 32 \times (\frac{T}{0.4} - 1)$$

Esto se debe a de que el tiempo T necesario para tener N tramas es $T = 0.4 \times (1 + \frac{N}{32})$ debido a que 0.4 es la duración de la trama y el sumando $\frac{N}{32}$ corresponde al solapamiento de 31/32 que se habían mencionado anteriormente.

Por lo tanto el número de bits que tendremos en un audio de 3 segundos es

$$n = 32 \times \left(32 \times \left(\frac{3}{0.4} - 1\right)\right) = 32 \times 208 = 6656$$

Este número es grande para nuestra aplicación y por lo tanto podemos aproximar la distribución binomial por una normal media $\mu = np$ y desviación estándar $\sigma = \sqrt{np(1-p)}$.

Entonces la probabilidad de falsos positivos para un umbral $T = \alpha n$ es

$$P_f(\alpha) = \frac{1}{2\pi}\int_{(1-2\alpha)\sqrt{n}}^{\infty} e^{-\left(\frac{x^2}{2}\right)}\,dx = \frac{1}{2}Q\left(\frac{1-2\alpha}{\sqrt{2}}\sqrt{n}\right)$$

, siendo α la tasa de errores de bit (Bit Error Rate, BER)
Debido a que las subfingerprints tienen correlación temporal alta por el solapamiento que hay entre ellas, la desviación estándar será mayor. En los papers mencionados anteriormente se concluye experimentalmente que la desviación estándar es en realidad el triple que la correspondiente a una distribución iid y por lo tanto:

$$P_f(\alpha) = \frac{1}{2}Q\left(\frac{1-2\alpha}{3\sqrt{2}}\sqrt{n}\right)$$

En este trabajo se consideraron los umbrales T definidos por BER=0.25 y BER=0.35, en audios de 3 y 5 segundos esto se traduce en las siguientes probabilidades de falsos positivos:

	3 seg	5 seg
BER=0.25	$P_f = 1.3 \times 10^{-42}$	$P_f = 2.0 \times 10^{-73}$
BER=0.35	$P_f = 1.4 \times 10^{-16}$	$P_f = 9.8 \times 10^{-28}$

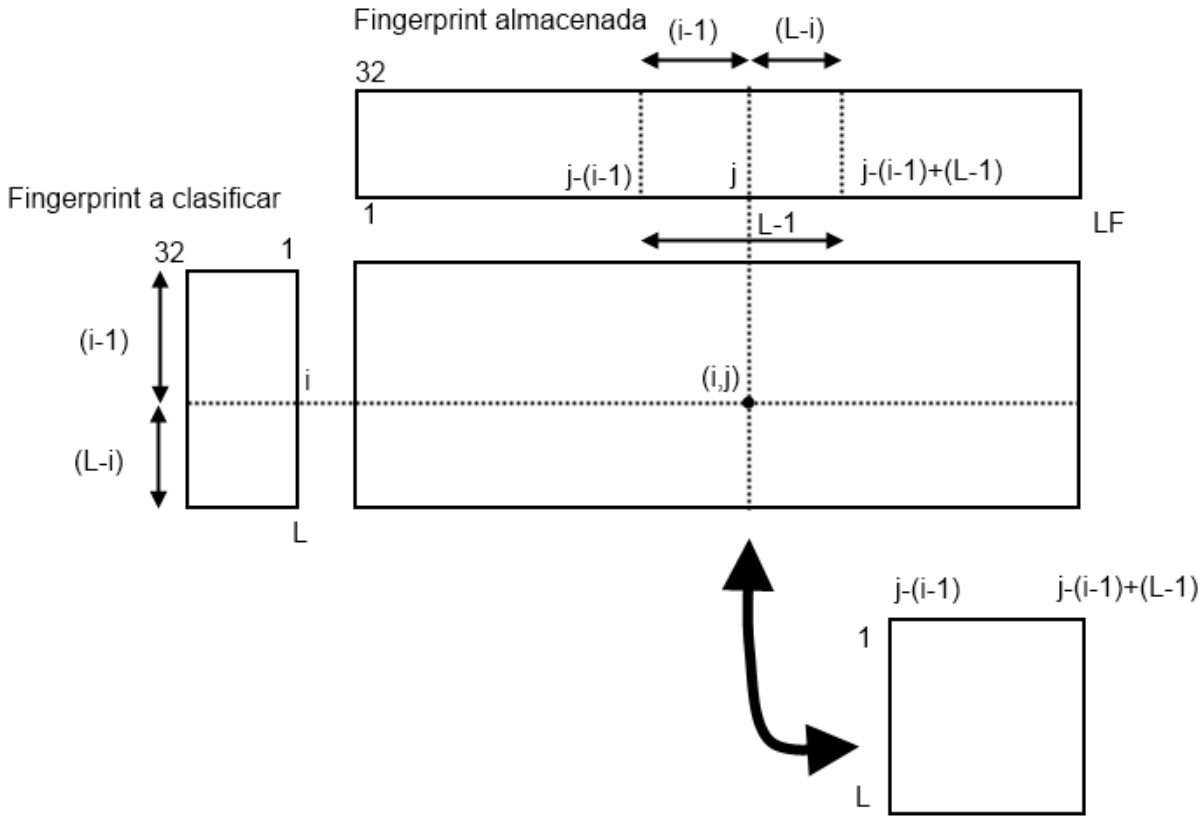
En todos los casos la probabilidad de falsos positivos en muy pequeña. Es claro que si en la base de datos se incluyen piezas musicales con fingerprints parecidas, la probabilidad de falsa alarma aumenta.

Búsqueda de la fingerprint en la base de datos

Para realizar la búsqueda de la fingerprint en la base de datos existen varios enfoques posibles. Uno de ellos es el de la fuerza bruta, es decir recorrer toda la base de datos buscando las fingerprints con menos errores de bit, si bien esto lleva a un resultado correcto es un proceso que insume muchos recursos y que puede demorar mucho tiempo.

El método que se implementó en este trabajo se basa en suponer que la probabilidad de encontrar al menos una de las subfingerprints en la posición óptima de la base de datos es alta. De esta manera buscaremos aquellas subfingerprints que tengan coincidencia exacta y las tomaremos como candidatos.

En nuestro caso contamos con una base de datos de audios de canciones enteras, mientras que la base de consultas consiste en audios de 3 o 5 segundos de esas canciones. El procedimiento que utilizamos es el de tomar la fingerprint del audio a clasificar y buscar coincidencia de alguna de las subfingerprints en toda la base de datos. Cuando se obtiene una coincidencia se guarda el número de subfingerprint del audio a clasificar y el de la base de datos donde existió esa coincidencia. Esto es obtengo el par (i, j), luego usando este par como anclaje, hallo el tramo de audio de la canción de la base que se alinee con la fingerprint del audio a clasificar. Este procedimiento se observa mejor en la siguiente figura:



Una vez que tengo dos fingerprints de igual tamaño calculo las distancias que hay entre las subfingerprints de los dos audios y los guardo en una matriz. Luego usando la técnica DTW estableceremos si las fingerprints son iguales o no.

Técnica

Dynamic time warping (DTW) es un algoritmo para medir la similitud entre dos secuencias que varían en el tiempo dadas ciertas restricciones. Se basa en el principio de optimalidad y en la programación dinámica.
La programación dinámica es una herramienta para resolver problemas que impliquen decisiones secuenciales. En particular se puede utilizar para resolver el problema de encontrar el camino de menor costo entre dos puntos.
Consideremos un conjunto de N puntos, consideremos que a cada par de puntos (i, j) se le asocia un costo C_{ij} que representa el costo de ir de i a j en un paso. El problema que resuelve programación dinámica es el de encontrar el menor costo y el mejor camino para ir de un punto i' a uno j' en la cantidad de pasos que sea necesario.
Llamaremos política a la regla de decisión que indica que punto se vistita luego del punto j. Las políticas son las que determinan las secuencias de puntos por las que se debe ir del punto i' al j', es por esto que el costo queda determinado por las políticas y por el punto de destino j'. La pregunta que queda por responder es cuál es la política que implica el menor costo.
El principio de optimalidad de Bellman dice que "dada una secuencia óptima de decisiones, toda subsecuencia de ella es, a su vez, óptima".

Es decir que si llamamos $\varphi(1,i)$ al mínimo costo entre el punto 1 y el i, usando el principio de optimalidad se debe cumplir que:

$$\varphi(1,i) = \min_j(\varphi(1,j) + \varsigma(j,i))$$

, siendo $\varsigma(j,i)$ el costo de ir de j a i en un paso.
Generalizando lo anterior al caso de ir de cualquier punto i a j

$$\varphi(i,j) = \min_l(\varphi(i,l) + \varphi(l,j))$$

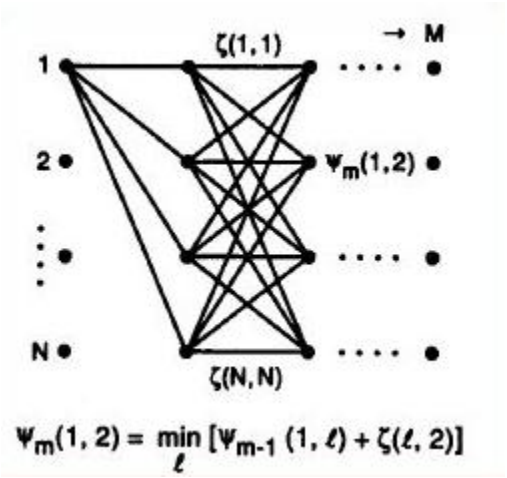
Donde $\varphi(i,l)$ es el mínimo costo de ir de i a l en cuantos pasos sea necesario.
Esto implica que cualquier secuencia parcial de la secuencia óptima entre i y j debe ser óptima.

Para resolver el problema debemos resolver el siguiente sistema de ecuaciones:

$$\begin{cases} \varphi_1(i,l) = \varsigma(i,l) \qquad l = 1,2,\dots N \\ \varphi_2(i,l) = \min_k(\varphi_1(i,k) + \varsigma(k,l))\ k = 1,2,\dots N\ \ l = 1,2,\dots N \\ \varphi_3(i,l) = \min_k(\varphi_2(i,k) + \varsigma(k,l))\ k = 1,2,\dots N\ \ l = 1,2,\dots N \\ \vdots \\ \varphi_s(i,l) = \min_k(\varphi_{s-1}(i,k) + \varsigma(k,l))\ k = 1,2,\dots N\ \ l = 1,2,\dots N \end{cases}$$

Siendo $\varphi_s(i,l)$ mejor camino de máximo S pasos entre i y l.

Otro problema que resuelve la programación dinámica es el de encontrar la secuencia óptima de un número fijo M de pasos entre el punto i y el j, y el costo mínimo asociado $\varphi_M(i,j)$. Consideremos N puntos ubicados verticalmente y horizontalmente tenemos las M transiciones como se ven en el siguiente dibujo:



Nuevamente aplicamos el principio de optimalidad, luego del m-ésimo movimiento el camino termina en punto l que puede ser cualquiera de los N puntos, con el costo mínimo costo asociado $\varphi_m(i,l)$. Si en el movimiento m+1 se termina en el punto n entonces

$$\varphi_{m+1}(i,n) = \min_l(\varphi_m(i,l) + \varsigma(l,n))$$

De esta manera encontramos una ecuación recursiva que nos permite hallar el camino óptimo de manera incremental. Ya que aunque hay N movimientos que terminan en l, el principio de optimalidad indica que solo hay que considerar el mejor de ellos.

A continuación se describen los pasos del algoritmo:

1. Inicialización:

$$\varphi_1(i,n) = \zeta(i,n)$$
$$\xi_1(n) = i$$
$$\text{for } n = 1,2,\ldots,N.$$

2. Recursión

$$\varphi_{m+1}(i,n) = \min_{1\le \ell \le N} [\varphi_m(i,\ell) + \zeta(\ell,n)]$$
$$\xi_{m+1}(n) = \arg \min_{1\le \ell \le N} [\varphi_m(i,\ell) + \zeta(\ell,n)]$$
$$\text{for } n = 1,2,\ldots,N \text{ and } m = 1,2,\ldots,M-2$$

3. Terminación

$$\varphi_M(i,j) = \min_{1\le \ell \le N} [\varphi_{M-1}(i,\ell) + \zeta(\ell,j)]$$
$$\xi_M(j) = \arg \min_{1\le \ell \le N} [\varphi_{M-1}(i,\ell) + \zeta(\ell,j)]$$

4. Backtracking : el camino óptimo es $(i,i_1,i_2, \ldots, i_{m-1},j)$ donde

$$i_m = \xi_{m+1}(i_{m+1}), \qquad m = M-1, M-2, \ldots, 1$$

, con $i_M=j$

Este algoritmo solo tiene que calcular solo los N caminos que finalizan en los N puntos al término de un potencial movimiento. Cuando se llega a un destino. el camino óptimo y su costo mínimo asociado, son el resultado del algoritmo sin tener que revisar costos parciales previos. Esto repercute en que solo se realizan un orden de NM operaciones.

La técnica de DTW utiliza programación dinámica pero restringido a que solo se pueden realizar ciertos movimientos.

En nuestro caso estamos tratando de hallar el mejor camino entre dos fingerprints, es decir, que buscamos cual es el camino de subfingerprints que conduce a la menor distancia entre las fingerprints.

Se tomaron las siguientes restricciones: solo son validos los movimientos en horizontal, vertical o diagonal. Es decir, que si estoy comparando las subfingerprints i y j, el siguiente paso será el que tenga menor costo entre las subfingerprints i con j+1, i+1 con j e i+1 con j+1.

Debido a que nos interesan principalmente movimientos en diagonal, porque esto implicaría que las piezas avanzan a un tempo, agregamos un costo adicional a los movimientos en horizontal y vertical en este caso se utilizo como pena un bit para los movimientos horizontales y verticlaes.

En la práctica procedimos de la siguiente manera. Una vez que se tienen las fingerprints a comparar se calcula la distancia de cada una de las subfingerprints de una fingerprint con todas las de la otra fingerprint, esto se almacena en una matriz.

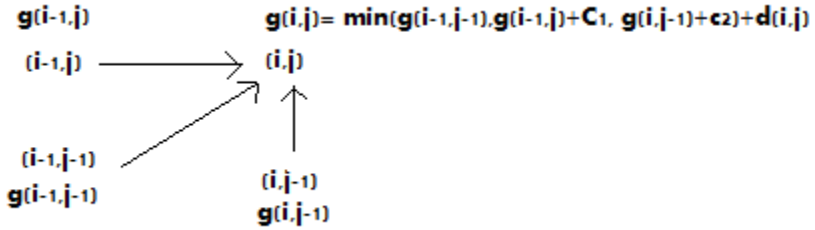
Luego consideraremos la función costo acumulado entre las subfingerprints (i, j) como:

$$g(i,j) = \min\big(g(i-1,j) + C1, g(i,j-1) + C2, g(i-1,j-1)\big) + d(i,j)$$

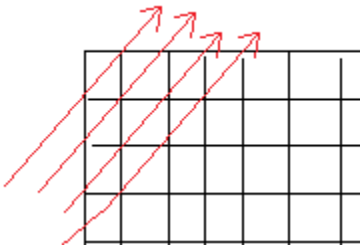
, siendo d(i,j) la distancia de Hamming de las subfingerprint i con la j.
Es decir que, el costo acumulado en (i,j) se calcula eligiendo entre los 3 caminos posibles por los cuales se puedo llegar hasta (i,j) el de menor costo.
Si elegimos que g(0,0)=0 como semilla de esta función recursiva, entonces podemos calcular el mínimo costo acumulado para cualquier par de i,j.
Por ejemplo:
g (0,1)=min (g (-1,1)+C1, g(0,0)+C2, g(-1,0)) +d(0,1), dado que el primer y tercer término no existen ya que no hay subfingerprint con índice negativo entonces

$$g(0,1)=g(0,0)+C2 +d(0,1)=C2+d(0,1)$$

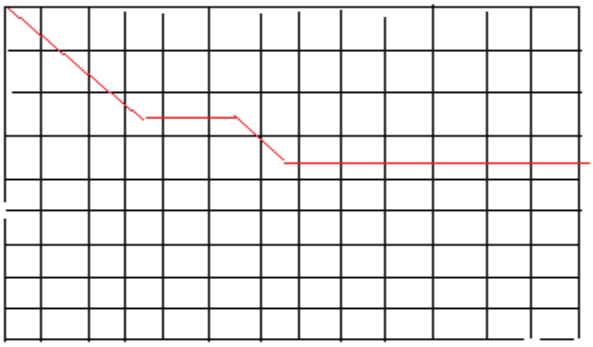
El siguiente dibujo muestra de forma más clara el proceso de obtención de los costos acumulados.



Esto sugiere el armado de la matriz de costos acumulados de la manera siguiente:



Luego de generada la matriz, se busca en la última fila y en la última columna menor valor acumulado, esto es porque si una de los audios es más rápido que el otro entonces el camino óptimo puede llevar a resultados como los siguientes



En la figura la línea roja representa el camino acumulado óptimo, en este caso el costo acumulado menor se encontrara en el cuarto elemento de la última columna.

Una vez hallado el menor costo acumulado se procede a compararlo con el valor umbral que como anteriormente se había mencionado será el 25% o el 35% del total de bits del audio a clasificar.

Resultados

Procedimiento

Se trabajo sobre una base de 25 canciones, de varios géneros, y varias duraciones (entre 2 y 8 minutos),

1	Vilma Palma E Vampiros - Auto Rojo.wav
2	The Beatles - Love Me Do.wav
3	Red Hot Chili Peppers - Under the Bridge.wav
4	Michael Jackson - Billie Jean.wav
5	La Vela Puerca - El Huracán.wav
6	Kiss - I Was Made for Loving You.wav
7	Jaime Roos - Brindis por Pierrot.wav
8	Frank Sinatra - Bad Bad Leroy Brown.wav
9	Eric Clapton - Cocaine.wav
10	Creedence Clearwater Revival - Have You Ever Seen the Rain.wav
11	Buitres - Besos.wav
12	Queen - Bohemian Rhapsody.wav
13	AC-DC - Back In Black.wav
14	Daniel Agostini - La Ventanita.wav
15	Gilda - Paisaje.wav
16	Karibe con K - Amores Como El Nuestro.wav
17	Queen - I Want To Break Free.wav
18	Toto - Africa.wav
19	Lou Bega - Mambo Mambo.wav
20	The Clash - Should I Stay Or Should I Go.wav
21	Rolling Stones - (I Can't Get No) Satisfaction.wav
22	Ramones - Blitzkrieg bop.wav
23	Led Zeppelin - Stairway To heaven.wav
24	Dire Straits - Sultans Of Swing.wav
25	AC-DC - Highway To hell.wav

Con las herramientas previamente presentadas, se procedió a ensayar con varias muestras, en diversas condiciones. El procedimiento general fue:

-Recibir audio a clasificar, y generar su fingerprint, tomando 3 segundos para audio digital alterado, y 5 segundos para audio capturado de forma externa (grabación con dispositivos electrónicos)

-Comparar la fingerprint a clasificar, con las de la base de datos, buscando coincidencias, para poder usar como puntos de anclaje. Se comienza buscando coincidencias exactas (subfingerprints 100% iguales), y en caso de no existir coincidencias, o que los datos obtenidos no fueran concluyentes, se incrementa gradualmente los bits de diferencia admitidos. A cada incremento gradual, aparecen más coincidencias, de forma exponencial, por lo que solo se buscan subfingerprints, con hasta 2 bits de diferencia. Con los índices hallados, se alinean las fingerprints de la base de datos, con la que se quiere clasificar.

-Una vez obtenidas las fingerprints alineadas, buscar el mejor camino mediante DTW, para cada pieza musical de la base datos y comparar el costo acumulado, contra el umbral establecido. Si no se encuentran resultados concluyentes, se incrementa la cantidad de bits de diferencia admitidos en 1, hasta llegar límite establecido.

-Presentar los resultados, indicando si se encontró alguna coincidencia, con costo acumulado por debajo del umbral establecido, o en caso contrario, cual fue el mejor costo acumulado calculado, al final del procedimiento.

Usamos indicadores de “precision” y “recall”, para evaluar el rendimiento y relevancia de los resultados obtenidos, los mismos definidos de la siguiente manera:

$$precision = \frac{verdaderos\ positivos}{verdaderos\ positivos + falsos\ positivos},$$

$$recall = \frac{verdaderos\ positivos}{verdaderos\ positivos + falsos\ negativos}$$

Compresión MP3

Se eligieron 5 canciones aleatorias de la base de datos, y se tomaron 3 segundos cualesquiera de cada una. Cada uno de estos segmentos, fueron comprimidos en MP3, a 128,64, y 32 Kbps, y se relevaron las sumas cumulativas en cada caso.

En todos los casos, la detección fue correcta, para cualquiera de los umbrales elegidos (1164 bits de diferencia, para BER=0.25, y 2329 bits de diferencia, para BER=0.35).

Ruido Gaussiano

Se toman 15 extractos de 3 segundos cada uno, y se le agrega ruido gaussiano a distintos niveles de SNR (Sound To Noise Ratio), entre 20 y 0 db

Grabaciones Externas

Se graban mediante celulares, y reproductores musicales, 10 extractos de 5 segundos de canciones reproducidas desde una PC. Estas grabaciones se obtuvieron en diferentes ambientes, por ejemplo en algunas había un ventilador encendido, en otras había una radio encendida de fondo, otras se realizaron con el menor sonido ambiente posible.

Se obtuvieron los siguientes resultados:

		"BER=0.25"						"BER=0.35"					
	ANALIZADAS	VP	FP	VN	FN	Precision	Recall	VP	FP	VN	FN	Precision	Recall
SNR	15	9	0	0	6	1	0.6	12	0	0	3	1	0.8
MP3	20	20	0	0	0	1	1	20	0	0	0	1	1
GRABACIONES	10	4	0	0	6	1	0.4	4	0	0	6	1	0.4

Modificación de la escala temporal

Se realizaron pruebas con segmentos de 3 segundos de canciones modificadas en su escala temporal de +1% ,-1% +3% y -3%. En todos los casos el sistema no pudo identificar ninguna de las canciones.

Dado que la modificación temporal se traduce en una homotecia en las frecuencias se probó comparar las fingerprints pero realizándoles un shift en las mismas para poder compensar el cambio de banda de frecuencias. Nuevamente los resultados obtenidos fueron malos, ya que no el sistema no pudo identificar ninguno de los audios.

Prueba de audios que no se encuentran en la base de datos

Se tomaron 3 segundos de audio de 5 canciones que no se encontraban en la base de datos.

Se obtuvieron 5 verdaderos negativos en el total de 5 pruebas con un umbral BER=0.25. Al aumentar el umbral a BER=0.35 se obtuvo un falso positivo y 4 verdaderos negativos.

Conclusiones

- El sistema de reconocimiento de canciones tuvo buenos resultados cuando se tomaron compresiones mp3 a distintas tasas, por lo tanto es robusto frente a la compresión
- Se obtuvieron buenos resultados al agregarle ruido gaussiano aun con SNR=0 db, por lo que también es robusto frente al ruido
- Al probar con grabaciones hechas con aparatos electrónicos los resultados ya no son tan buenos , esto se debe a que se introducen ecos y ruidos externos que modifican las fingerprints obtenidas
- Al modificar la línea temporal de la canción, se realiza una homotecia del espectro, esto implica que energía que se encontraba en cierta banda de frecuencias se pase para otras, modificando la fingerprint obtenida y por esto es que se obtiene malos resultados en la búsqueda
- Para intentar revertir el efecto del cambio de banda de la energía al modificar la escala temporal, se implemento el shift en la fingerprint, , sin embargo no se obtuvieron buenos resultados.
- Al probar con audios de canciones que no estaban en la base de datos se obtuvieron buenos resultados dando en la mayo ría de los casos verdaderos negativos

Posibles mejoras a implementar

- Se puede acelerar la búsqueda de la fingerprint en la base de datos implementando una tabla hash. La misma se arma de la siguiente manera, para cada secuencia de bits, es decir para cada subfingerprint posible, se hace un listado de las canciones de la base de datos en la que dicha subfingerprint aparece y la posición en la que lo hace.
  De esta manera cuando busco una subfingerprint determinada, debo buscar en la tabla y esta me devolverá en que canciones se encuentra y en qué lugar de las mismas.
  Dado que la cantidad de subfingerprints posibles es de 2^{32} , es impracticable realizar la tabla hash de ese tamaño. Lo correcto sería tomar los primeros o los últimos 16 bits de cada subfingerprint y realizar la tabla y la búsqueda solo con esos bits.

- Para mejorar la robustez en cuanto a modificaciones en las escalas temporales, una posible solución es la de generar fingerprints tomando bandas de frecuencia que se solapen.
  Las modificaciones en escalas temporales se traducen en homotecias en frecuencia lo que puede provocar que energía que se encontraba en una cierta banda se pase a otra de las bandas.
  El tomar bandas tan rígidas y sin solapar provoca que estos cambios de energía se traduzcan en cambios en los bits de las fingerprints provocando falsos negativos. Si consideramos las bandas de frecuencias un poco solapadas podemos amortiguar este pasaje de energías de una banda a otra.

Referencias

- R.O. Duda et al, Pattern Classification, 8.5 "Recognition with strings"
- L.R. Rabiner, B.H. Juang, "Fundamentals of Speech Recognition", 1993, Chp. 4.7, "Time allignment and Normalization"

- Ground-Truth Transcriptions of Real Music from Force-Aligned MIDI Syntheses, Turetsky and Ellis, ISMIR 200
- “A Highly Robust Audio Fingerprinting System” de Jaap Haitsma y Ton Kalker
- “Robust Audio Hashing for Content Identification” de Jaap Haitsma, Ton Kalker y Job Oostveen