UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Sistema de identificación de tarareos

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jose Alemany Bordera

Tutor: Carlos David Martínez Hinarejos

Septiembre de 2014

El presente trabajo fin de grado lo dedico...

a mi pareja, por su paciencia y comprensión,
a mi familia, por su apoyo y ánimo,
a mis amigos y compañeros,
y a mi tutor.

Gracias a todos.

Resumen

Actualmente, toda la información multimedia es almacenada en la web y los sistemas de recuperación evolucionan con el objetivo de facilitar dicha información. La búsqueda e identificación de melodías mediante el tarareo ha surgido de la necesidad de reconocer melodías que han quedado grabadas en los individuos pero los cuales no son capaces de identificar.

En el presente trabajo abordaremos el diseño e implementación de un sistema de identificación de melodías mediante consultas realizadas por tarareo, centrándonos exclusivamente en las tareas de extracción automática de características y en la función de similitud entre melodías.

Además, como esta necesidad de identificar la melodía surge de forma espontánea o al escucharla en cualquier momento del día, se ha decidido implementar una aplicación para dispositivos Android que permite acceder al sistema de identificación de melodías en cualquier momento. De esta forma, se cubre una necesidad puntual del usuario.

Palabras clave: recuperación de información, extracción de características, función similitud, DTW, Android, *socket*

Abstract

Currently, all media information is stored on the web and the recovery systems evolve in order to provide that information. The search and identification of melodies by humming arises from the need to recognize tunes that have been recorded in individuals but which are unable to identify.

In this project we will address the design and implementation of a system of identification of melodies by humming through consultations, focusing solely on the task of automatic extraction of features and function of similarity between melodies.

Moreover, as this need to identify the melody arises spontaneously or listening it at any time of the day, it was decided to deploy an application for Android devices that allows you to access tunes identification system at any time. Thus, a specific need of the user is covered.

Keywords: information retrieval, feature extraction, similarity function, DTW, Android, *socket*

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

Una gran cantidad de información fluye continuamente a través de Internet gracias a su accesibilidad mediante el uso de computadoras personales, *smartphones* y Smart Tv. Todo tipo de contenido es enviado, compartido y descargado. La forma más común de acceder a la información es utilizando motores de búsqueda como Google o Yahoo. El volumen de los contenidos multimedia aumenta y la necesidad de almacenarlos en bases de datos especializadas surge, así como nuevos métodos de recuperación.

En los sistemas de recuperación tradicionales, la información se recupera mediante consultas de palabras clave. Los usuarios expertos no tienen dificultad para encontrar la información que buscan en la mayoría de los casos, pero los usuarios más comunes (inexpertos o principiantes) en muchas ocasiones no logran encontrarla. El objetivo de los sistemas de recuperación de información es lograr que cualquier tipo de usuario encuentre la información que busca de la forma más fácil e intuitiva.

En los sistemas de recuperación de información multimedia se han producido grandes avances que permiten recuperar la información con nuevas técnicas más intuitivas. Hay sistemas que permiten encontrar información basándose en el contenido de una imagen, como QBIC [8] o Google [9]. También existen otros sistemas que permiten identificar canciones mediante contenido musical, como Shazam [21] o Midomi [14], e incluso identificar similitudes entre contenidos para ofrecer al usuario información que se adecua a sus gustos.

El caso de nuestro estudio está centrado en la recuperación de información multimedia mediante contenido musical, en particular, melodía cantada, tarareada o silbada. Probablemante todo el mundo ha tenido la experiencia de escuchar una melodía fascinante pero de la cual desconoce o no recuerda su autor o título. La identificación o consulta por tarareo es una solución natural al problema de reconocimento de piezas desconocidas.

1.2. Objetivos

El propósito del siguiente proyecto consiste en el desarrollo de un sistema de identificación y recuperación de piezas de música basándose en tarareos, realizados por los usuarios, como consultas.

Uno de los objetivos principales es que la base de datos, donde estarán las canciones codificadas y la cual tendrá un tamaño de 50 canciones, pueda construirse mediante el mismo analizador que el utilizado para codificar los tarareos. Con ello se intenta automatizar el proceso de contrucción de la misma y ofrecer flexibilidad ante nuevas entradas en la base de datos.

En ambos casos, la codificación obtenida debe ser lo más representativa posible y que ofrezca el mayor poder discriminativo para que al realizar las comparaciones con las canciones se obtengan los resultados deseados. Además, la función de similitud utilizada debe poder identificar las canciones teniendo en cuenta las imprecisiones del canto del usuario.

Con el fin de lograr dichos objetivos, se realizará un estudio donde se analizarán diversas técnicas de representación y comparación (función de similitud). Para ello se recogerán muestras de tarareos, realizadas por varios individuos, de las canciones que componen la base de datos.

Finalmente, mediante una aplicación para dispositivos Android, se permitirá al usuario realizar consultas y obtener los resultados en un listado ordenado por grado de similitud. La aplicación, de forma transparente, será la encargada de enviar la consulta al servidor, donde se procesará la consulta, se aplicará la función de similitud y se obtendrán los resultados.

En resumen, los objetivos de este trabajo son:

- Implementar un sistema de reconocimiento de melodías que permita identificarlas mediante consultas cantadas, tarareadas o silbadas. Para ello, debemos:
  - Diseñar un método automático de extracción de características para melodías, tanto polifónicas como monofónicas, que ofrezca un alto poder discriminativo.
  - Diseñar una función que nos permita comparar dos melodías.
- Desarrollar una aplicación para dispositivos Android para que los usuarios puedan realizar las consultas y obtener los resultados.

1.3. Problemas básicos y enfoques

La tarea de identificación de melodías por tarareo se enfrenta a problemas relacionados con el tratamiento de los datos de música y de tarareo, además de los problemas habituales de recuperación de información.

Los usuarios pueden tararear o cantar cualquier segmento de una melodía que ellos recuerden y ésta puede tener diversas longitudes. Por ello, es necesario que la función de correlación entre el tarareo del usuario y las canciones de la base de datos pueda empezar en varios puntos de una canción. La función de similitud utilizada en este proyecto permite abordar dicho problema.

Muy pocas personas pueden tararear una pieza de música exactamente de la misma forma como está escrita en la partitura. Por tanto, hay que tener en cuenta que el canto del usuario

contendrá imprecisiones respecto a la pieza de música original y estas no deben afectar, en lo posible, a su identificación.

Una pieza de música debe poder ser reconocida independientemente del tempo o entonación con la que ésta sea tarareada. En este proyecto explicamos las técnicas y decisiones tomadas para permitir a los usuarios tararear su canción en cualquier tempo y entonación, y que esto no afecte al resultado de su búsqueda.

Al tratarse de una consulta de usuario, los tiempos de respuesta deben ser bajos. Por ello, la función de similitud realizará la correspondencia de forma eficiente para que el tamaño de la base de datos afecte lo mínimo posible y poder obtener los resultados en los tiempos deseados.

1.4. Planificación

Este trabajo se ha organizado de la siguiente forma. En el Capítulo 2, se describen los principios de la identificación por tarareo: conceptos musicales, arquitectura básica de un sistema, requisitos para la identificación y problemas técnicos presentes. Para lograr alcanzar dichos requisitos se explicarán, en el Capítulo 3, varias técnicas de extracción y representación de características, funciones de comparación y estructuración de bases de datos; también analizaremos trabajos previos. A continuación, el Capítulo 4 contendrá todos los aspectos relacionados con el desarrollo del sistema de identificación por tarareo implementado y la aplicación StartHum. Analizaremos, en el Capítulo 5, los resultados obtenidos tras probar, con una colección de consultas, algunas técnicas explicadas en los Capítulos 3 y 4. Para finalizar, el Capítulo 6 cerrará el trabajo y se discutirán las posibles investigaciones futuras.

Capítulo 2

Principios de la identificación por tarareo

2.1. Conceptos musicales

Los principales conceptos musicales utilizados a lo largo del proyecto son definidos a continuación.

- **Melodía:** Sucesión organizada de notas de tono y duración específicas, enlazadas juntas en el tiempo para producir una expresión musical coherente.
- **Nota:** Una nota es el elemento más básico y primordial del sonido y de la música. Es representada con un valor de tono y una duración (ver Figura 2.1). La altura en el pentagrama y la clave definen el valor de tono; la duración es definida por la figura musical con la que se representa la nota.

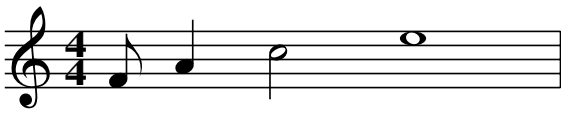


Figura 2.1: Ejemplo de notas musicales.

- **Tono:** El tono o altura es la impresión que nos produce la frecuencia de vibración a la que se manifiesta una determinada onda sonora. La codificación MIDI será utilizada para representar los valores de tono en nuestro proyecto[1].
- **Duración:** La duración es representada con figuras musicales y expresan la duración de una nota. Éste es un valor relativo que depende del tempo.
- **Intervalo:** El intervalo es la diferencia de altura entre dos tonos musicales oídos sucesiva o simultáneamente.
- **Silencio:** Periodo de silencio que es representado únicamente por la duración.
- **Tempo:** Hace referencia a la velocidad con la que debe ejecutarse una pieza musical.
- **Ritmo:** Marca la sucesión regular de los elementos débiles y fuertes del sonido. El ritmo tiene independencia, a diferencia de la melodía que está ligada a un ritmo.

[1] `http://subsynth.sourceforge.net/midinote2freq.html`

- **Tonalidad:** La tonalidad o clave de una pieza musical es un concepto complejo y define la nota sobre la que descansa una melodía.
- **Intensidad:** La intensidad es la cualidad que diferencia un sonido suave de un sonido fuerte. Depende de la fuerza con la que sea ejecutado y de la distancia del receptor que lo percibe.
- **Timbre:** El timbre se trata del matiz característico de un sonido. El timbre es utilizado en muchas ocasiones como criterio de identidad. Es esta propiedad del sonido que nos permite distinguir una guitarra de una trompeta.

2.2. Arquitectura básica

Los sistemas de identificación por tarareo son los encargados de reconocer una melodía por su contenido mediante el canto o tarareo realizado por un usuario. Un sistema general de consulta por tarareo está formado por tres componentes (ver Figura 2.2).

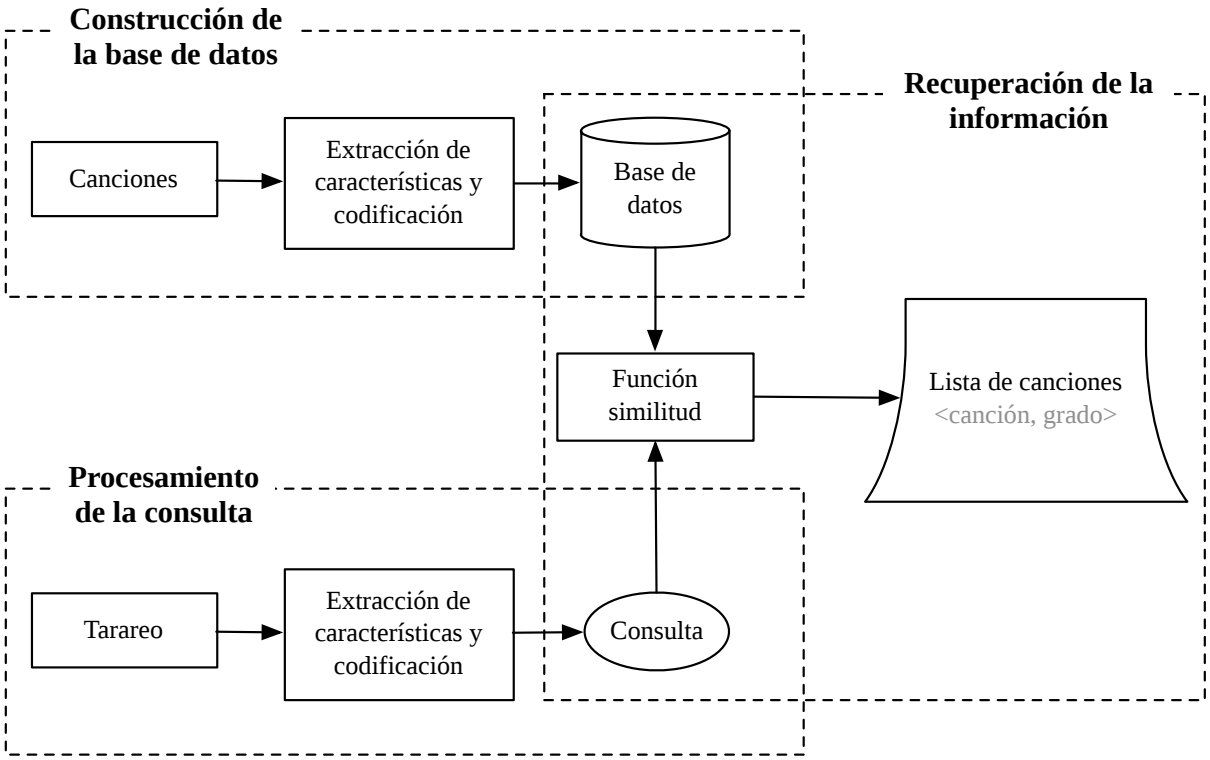


Figura 2.2: Arquitectura básica de un sistema de consulta mediante tarareo.

La construcción de la base de datos se realiza mediante la codificación de las características más discriminatorias y representativas de las canciones que la componen. Esta información será posteriormente utilizada para computar la función de similitud con la consulta realizada. También debe incluirse en la base de datos toda la información relativa a las canciones (título de la canción, artista, etc.) que el usuario recuperará como resultado de su consulta. Opcionalmente pueden incluirse las canciones originales para que el usuario pueda escuchar y verificar que los resultados de su consulta son correctos.

En el procesamiento de la consulta, las características deben ser extraidas y codificadas de forma coherente con la información de la base de datos. La técnica de extracción y codificación de características puede ser la misma o puede relajarse debido a que la consulta debe tener un tiempo de respuesta bajo.

Para recuperar la información referente a la consulta realizada, el sistema aplica la función de similitud a la consulta contra las canciones de la base de datos y obtiene la canción identificada o un listado, ordenado por mayor grado de similitud, de las posibles canciones.

2.3. Requisitos

Los principales requisitos básicos que debe cumplir un sistema de identificación de melodías por tarareos son los siguientes.

1. Los resultados deben ser precisos.
2. El sistema de identificación de melodías debe poder trabajar con grandes volúmenes de datos.
3. El usuario debe poder tararear cualquier fragmento de una melodía.
4. La tonalidad empleada en el tarareo no debe afectar al resultado de la consulta.
5. El usuario podrá tararear a cualquier tempo o velocidad.
6. La función de similitud debe permitir errores en el tarareo.
7. El tiempo de respuesta debe ser reducido.

En los siguientes apartados explicamos en detalle en qué consisten los requisitos mencionados y las dificultades que presentan.

2.3.1. Precisión en los resultados

Cuando los usuarios realizan consultas a un motor de búsqueda como Google, la información que buscan puede encontrarse entre los 5 o 7 primeros resultados devueltos. Esto es debido a que las palabras clave utilizadas pueden ser las correctas o una aproximación a estas. En el caso de la identificación de melodías por tarareo, el resultado se espera encontrar en la primera opción, es decir, la precisión de los resultados debe ser mayor. Que los resultados sean precisos es el requisito más importante.

Para obtener la precisión deseada es necesario analizar en detalle qué aspectos de una melodía, en relación al canto o tarareo de un usuario, son más parecidos, es decir, que características son las que se mantienen durante el tarareo. Estas características deben tener el mayor poder discriminativo posible.

Las métricas utilizadas para analizar los resultados obtenidos del proceso de identificación de las melodías mediante el tarareo serán distintas de las utilizadas en otros sistemas de recuperación de información. La medida de tasa de éxito superior a una posición x es la más utilizada y se define como el porcentaje de consultas que se encuentran entre las primeras x posiciones de los resultados obtenidos.

2.3.2. Tamaño de la base de datos

El tamaño de la base de datos afecta al tiempo de respuesta, ya que al aumentar el número de canciones se aumenta el número de veces que la función de similitud se aplica. Además, con el aumento de la base de datos también se ve mermado el poder discriminativo. Aún así, un sistema de identificación de melodías por tarareo debe ser capaz de trabajar con grandes bases de datos. Decisiones relacionadas con la forma de estructurar la base de datos deben ser tratadas para solucionar estos problemas.

2.3.3. Fragmentos de melodía

Cuando los usuarios realizan una consulta a la base de datos, el fragmento tarareado puede pertenecer a cualquier instante de una pieza de música. El primer problema es incluir esta característica en la función de similitud. Dependiendo de la longitud del fragmento tarareado, la consulta tendrá mayor o menor poder discriminativo. El segundo problema será definir la longitud mínima que deberá tener una consulta y que ésta tenga suficiente poder discriminativo.

2.3.4. Tempo y tonalidad del tarareo

El tempo y la tonalidad son dos características independientes de una melodía, es decir, una pieza de música puede ser interpretada y reconocida aun si varía alguna o ambas de estas características (ver Figuras 2.3 y 2.4). Es por ello que cuando los usuarios realicen una consulta mediante tarareo será necesario normalizar estos aspectos para poder identificar correctamente la canción deseada.

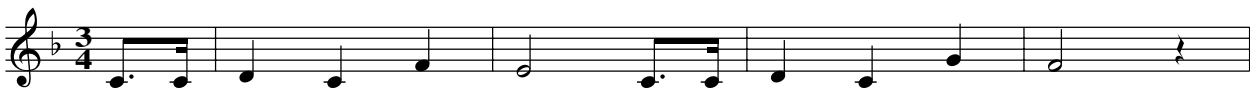


Figura 2.3: Principio de la melodía “Cumpleaños Feliz”.

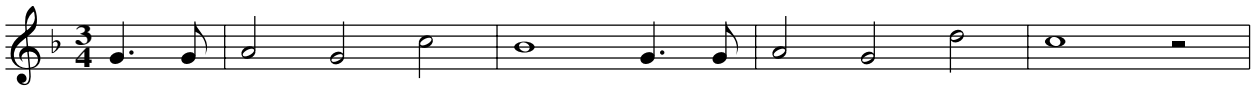


Figura 2.4: Melodía “Cumpleaños Feliz” con distintos tonos y tempo.

2.3.5. Imprecisiones de la consulta

Para la mayoría de usuarios que tararean o cantan una melodía es extremadamente complicado reproducirla con total exactitud (incluso los profesionales tienen esta dificultad). Por lo tanto, habrá una alta probabilidad de que la consulta contenga errores. Los errores más comunes cometidos al tararear una canción son la inserción, la omisión o el reemplazo de notas musicales.

El problema será cómo identificar una pieza musical con consultas erróneas. El sistema de identificación de melodías por tarareo debe poder identificar las canciones aunque contenga imprecisiones.

2.3.6. Tiempo de respuesta

El tiempo de respuesta de un sistema de recuperación de la información debe ser reducido. Los usuarios sienten que el sistema está reaccionándo de forma instantánea cuando los resultados son mostrados en un tiempo inferior o igual a 0,1 segundos; entre 0,1 y 1 segundo los pensamientos de los usuarios permanecen sin interrupciones; y 10 segundos es el límite para mantener la atención del usuario centrada [16].

La tarea de identificación de la melodía mediante el tarareo es compleja debido a los requisitos que deben cumplirse vistos en los apartados 2.3.3, 2.3.4 y 2.3.5. A diferencia de la recuperación por equivalencia, el tiempo empleado para la recuperación por similitud es mucho mayor.

Capítulo 3

Técnicas y trabajos relacionados

3.1. Características más discriminatorias

Conocer qué características son las que nos permiten identificar una pieza musical de otra y cómo representarlas es la tarea más importante para su correcta identificación.

Los atributos fundamentales de la música son la melodía, el ritmo, el tempo (lento o rápido), la intensidad (suave o fuerte), el timbre y, en algunos casos, la letra. Es en estas dimensiones en las que somos capaces de distinguir una pieza musical de otra. La melodía y el ritmo son las características más distintivas [10]. La melodía de una pieza musical es la succesión de notas de tono con una duración específica distribuidas u organizadas de forma que siguen un patrón en el tiempo (ritmo).

La extracción de características se centrará en obtener el valor de tono y su duración en una melodía. Ésta es una tarea relativamente sencilla para melodías monofónicas, es decir, melodías con un único timbre. La mayoría de consultas serán melodías de este tipo. Con los algoritmos [5, 7] se obtiene el valor de tono y un valor de confianza para cada instante de tiempo de la melodía. En cambio, para melodías polifónicas, con más de un timbre, la obtención de la melodía producida por cada timbre [18] o la obtención de la melodía predominante [20] son tareas más complejas. Algunas consultas y la mayor parte de las canciones de la base de datos son melodías polifónicas.

3.2. Representación de la melodía

Cómo representar las características de la música para realizar las comparaciones es un problema que afecta al poder discriminativo de los datos. A continuación, se describen dos sistemas de codificación usuales para representar características musicales.

3.2.1. Codificación UDS

Dado que las personas pueden identificar una melodía incluso variando la tonalidad de la misma o variando el tempo es necesario aplicar una codificación que permita abstraer estas características. Por esta razón, más que el valor del tono se utiliza el intervalo relativo entre las notas seguidas de la melodía. Esta variación se conoce como contorno melódico.

La representación del contorno melódico ha sido utilizada en muchas investigaciones debido a la ventaja que presenta al no ser necesario preocuparse por las diferencias de tonalidad

entre la consulta y las canciones de la base de datos [2]. En la representación del contorno melódico son utilizadas tres letras o símbolos; U, D y S que representan las diferencias de altura entre dos tonos consecutivos. El símbolo U representa la subida de tono respecto al valor de tono anterior; el símbolo D, la bajada de tono; y el símbolo S, el mantenimiento del valor de tono (ver Figura 3.1).

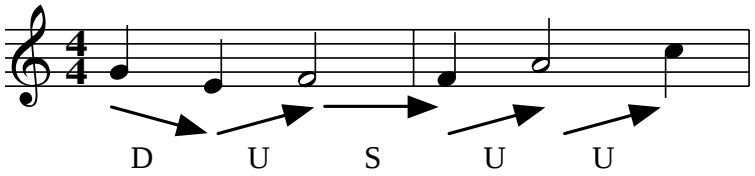


Figura 3.1: Representación UDS de una melodía.

Sin embargo, el utilizar únicamente la característica del valor de tono es insuficiente para lograr un alto poder discriminatorio ante grandes bases de datos. Una forma de ampliar la codificación para tener en cuenta también la característica de la duración de las notas es añadir tres símbolos más a la codificación UDS. La diferencia de duración entre dos notas consecutivas es representada por los símbolos L, S y E, que representan respectivamente que la duración respecto a la nota anterior es mayor, menor o la misma.

3.2.2. Codificación MIDI

Otra forma de representar la información musical es utilizando directamente los valores de tono obtenidos tras la aplicación de los algoritmos de extracción de características del audio nombrados en la Sección 3.1 [5, 7]. Esta representación se considera que tiene mayor poder discriminativo que la codificación UDS debido a que mantiene más las características propias de la melodía.

Aunque esta codificación de las piezas de música represente mejor las características de la melodía, no tiene en cuenta las principales diferencias nombradas en los apartados 2.3.4 y 2.3.5 entre los datos de música y tarareo. Existen dos alternativas que permiten procesar la codificación MIDI y obtener una nueva codificación en la cual la característica de la tonalidad ha sido normalizada (ver Figura 3.2):

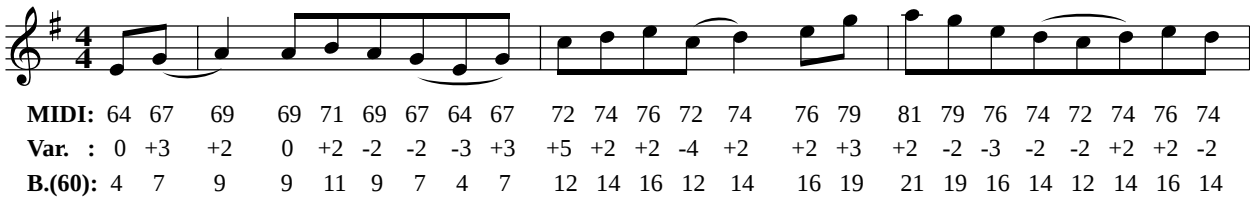


Figura 3.2: Representación MIDI y variantes.

- Variación del tono (Var. en la Figura 3.2): La representación de la secuencia de valores MIDI es convertida en la diferencia de valores con la anterior nota. Ésta representación

no es tan discriminatoria, debido a que solo se está utilizando la diferencia entre dos notas como característica principal de la comparación.

- Tono base (B.(60) en la Figura 3.2): Se establece un tono base el cual se resta a todos los valores MIDI de la representación de la melodía [23]. Esta codificación proporciona más poder discriminativo que la variación del tono, pero la elección del tono base es una tarea crítica.

Si utilizamos directamente los valores de tono como representación, las características de tempo y tonalidad tendrán que ser tratadas durante la aplicación de la función de similitud para que no afecten al resultado.

3.3. Identificación por similitud

La función de recuperación por similitud es la encargada de comparar las características del tarareo con cada canción de la base de datos y proporcionar una lista clasificada de acuerdo con las distancias obtenidas. Además, debe tener en cuenta las propiedades de los tarareos que dificultan la identificación y que aún no han sido tratadas en la representación de la misma.

3.3.1. Algoritmo distancia de Levenshtein

El algoritmo distancia de edición o distancia de Levenshtein se define como el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Es un algoritmo basado en programación dinámica[1]. Este algoritmo es muy utilizado en trabajos relacionados con la teoría de la información y con las ciencias de la computación [2, 11, 19, 12].

La distancia es interpretada como "coste" y las operaciones permitidas son: la inserción, el borrado y la sustitución de un carácter en la cadena. Cuanto menor es el coste, más similares se considera que son ambas cadenas. El coste asociado a las operaciones es, por definición, de 1 para la inserción y el borrado y 2 para la sustitución, pero estos valores pueden configurarse libremente dependiendo del tipo de problema a resolver. Las Ecuaciones (3.2) y (3.3) definen el algoritmo.

$$d(i,j) = |h(i) - r(j)| \tag{3.1}$$

$$D(0,0) = 0 \tag{3.2}$$

$$D(i,j) = min \begin{cases} D(i-1,j) + d(i,j) & & \text{(inserción)} \\ D(i-1,j-1) + 2d(i,j) & 1 \leq i \leq m & \text{(sustitución)} \\ D(i,j-1) + d(i,j) & 1 \leq j \leq n & \text{(borrado)} \end{cases} \tag{3.3}$$

La Ecuación (3.1) hace referencia a la función distancia (d) entre caracteres, que puede ser configurada y es independiente del algoritmo, donde h hace referencia a la cadena de caracteres de entrada y r la cadena de referencia.

[1]Técnica de la informática que permite abordar problemas complejos los cuales contienen subproblemas del problema original reduciendo su tiempo de ejecución a cambio de un mayor uso de la memoria.

Esta técnica es utilizada con la codificación UDS, explicada en el apartado 3.2.1. Para permitir, además, que el tarareo pueda pertenecer a cualquier segmento de la melodía es necesario incluir la Ecuación (3.4).

$$D(i,0) = 0 \tag{3.4}$$

De esta forma se obtiene como resultado un conjunto de distancias, en lugar de un solo valor, con las distancias de edición tomadas desde todos los puntos de la melodía.

3.3.2. Algoritmo DTW

El algoritmo DTW (*Dynamic Time Warping*) o distancia de alineamiento temporal no lineal permite alinear, en el análisis de series de tiempo, las partes similares entre dos secuencias temporales que pueden variar en el tiempo o la velocidad. Este algoritmo presenta muchas ventajas ya que permite las variaciones temporales producidas al tararear una canción y las producidas con el habla. Por lo tanto, este método es utilizado de forma usual para la coincidencia de los datos de música y del reconocimiento del habla [1, 12, 17, 19, 15].

Éste también es un algoritmo basado en la programación dinámica y consiste en las Ecuaciones recursivas (3.6), (3.7), (3.8) y (3.9) que definen el algoritmo DTW.

$$d(i,j) = |h(i) - r(j)| \tag{3.5}$$
$$DTW(0,0) = 0 \tag{3.6}$$
$$DTW(0,j) = DTW(0,j-1) + d(0,j) \tag{3.7}$$
$$DTW(i,0) = DTW(i-1,0) + d(i,0) \tag{3.8}$$
$$DTW(i,j) = d(i,j) + min \begin{cases} DTW(i-1,j) \\ DTW(i-1,j-1) \\ DTW(i,j-1) \end{cases} \tag{3.9}$$

La Ecuación (3.5) hace referencia a la función distancia (d) entre los puntos de las secuencias temporales, que puede ser configurada y es independiente del algoritmo, donde h hace referencia a la secuencia de entrada y r a la secuencia de referencia.

Para visualizar mejor el recorrido del algoritmo puede verse en la Figura 3.3 el grafo de dependencias correspondiente a la ecuación recursiva del algoritmo DTW. Como sucedía en el algoritmo distancia de edición, éste debe ser modificado para que pueda iniciarse en cualquier punto de la melodía. Por tanto, la Ecuación (3.7) debe ser sustituida por la Ecuación (3.10).

$$DTW(0,j) = 0 \tag{3.10}$$

De esta forma, el algoritmo obtiene las distancias menores tomadas desde todos los puntos de la melodía. El grafo de dependencias que representa dicho cambio puede verse en la Figura 3.4. Aun así, se considera que el algoritmo DTW tiene bajo poder discriminativo porque la duración de las notas no se tiene en cuenta al calcular la similitud.

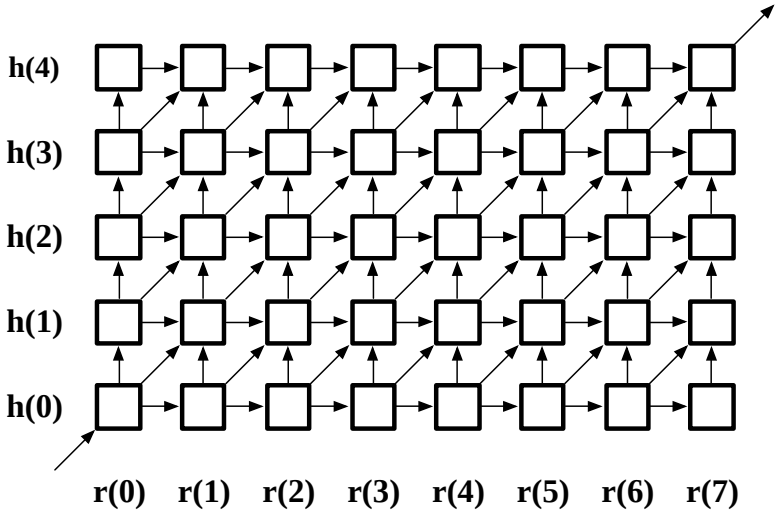


Figura 3.3: Grafo de dependencias del algoritmo DTW.

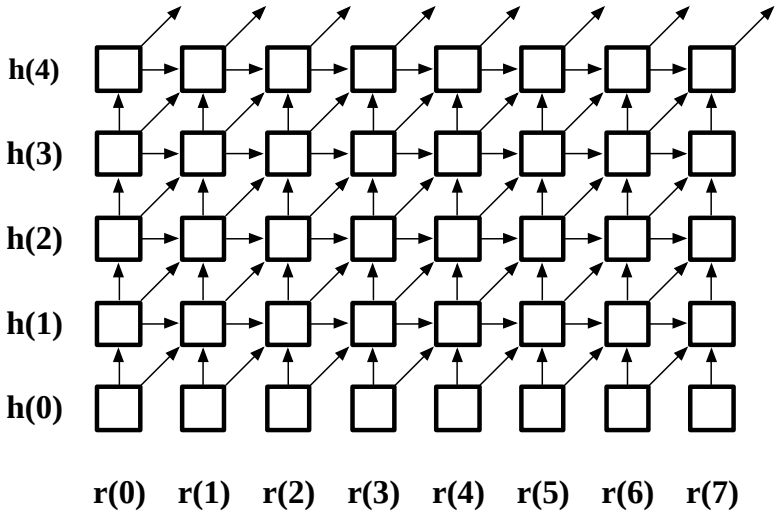


Figura 3.4: Grafo de dependencias del algoritmo DTW modificado.

3.4. Organización de la base de datos

Cómo estructurar la base de datos es un aspecto importante a tratar, como se describe en el Capítulo 2, ya que si tenemos una gran base de datos ésta puede afectar considerablemente al tiempo de recuperación de los resultados y disminuye el poder discriminativo de los mismos. A continuación, presentamos dos tipos de estructuras para bases de datos.

3.4.1. Base de datos no estructurada

No estructurar la base de datos es una opción válida durante la construcción de la misma. Esto presenta la ventaja de que el coste de la construcción es bajo respecto a una base de datos estructurada.

Sin embargo, al no estar estructurada, los incrementos de los cálculos en la función de similitud repercuten con mayor importancia. Además, si la base de datos aumenta, el tiempo de respuesta también aumenta de forma proporcional.

3.4.2. Base de datos estructurada

Una buena estructuración de la base de datos es necesaria para obtener los resultados de forma más eficiente. Que la base de datos sea estructurada es un requisito imprescindible si ésta contiene un gran volumen de entradas.

Para ello, se aplican técnicas de indexación y *clustering* que clasifican los datos y permiten hacer que la recuperación de la información sea más selectiva [12]. De esta forma, el tamaño de la base de datos e incluso el coste de la función de similitud no penalizan tan gravemente los tiempos de recuperación y respuesta [16].

3.5. Trabajos previos

Algunos de los sistemas de consulta por tarareo actuales más interesantes son presentados a continuación.

- TANSEN [2]

  Es un sistema de recuperación de música basado en el tarareo como consulta. Fue desarrollado en 2003 por la compañía IIT Bombay. Su base de datos esta formada por canciones de películas de origen indio. Describe un sistema el cual representa las melodías mediante la técnica del contorno melódico y que utiliza como función de similitud la distancia de Levenshtein. En un pequeño estudio que se realizó con 20 canciones en la base de datos y 100 consultas, se consiguió obtener un porcentaje de acierto en la identificación de un 95 %.

- Tararira [13]

  Creado en 2004 como proyecto fin de carrera de Ernesto López, Martín Rocamora y Gonzalo Sosa, es un sistema de recuperación de melodías por tarareo. Fue diseñado para Linux y está disponible como software libre (licencia GNU/GPL). Su base de datos esta formada por 205 canciones de los Beatles codificadas en formato MIDI. La función implementada para la identificación de los tarareos está basada en el algoritmo LDTW (*Local Dynamic Time Warping*), modificación del algoritmo DTW que restringe a una banda de cierta anchura la expansión de la dependencia. La evaluación de los resultados obtenidos por este sistema son superiores al 70 %. Hay que tener en cuenta que los resultados varían dependiendo del volumen de la base de datos.

- SoundCompass [12]

  El sistema de recuperación de melodías por tarareo SoundCompass es un sistema más completo. Tiene una base de datos de 21804 canciones estructurada por índices que permite una recuperación de la información más eficiente. Permite tararear cualquier segmento de una canción con la restricción de realizar el tarareo con la sílaba “ta” y aplica una función de similitud basada en comparación lineal con distancias Manhattan. Los resultados de precisión obtenidos están sobre el 70 %.

- Midomi [14]

  Es un proyecto desarrollado por Melodis Corporation en 2006. Midomi trabaja con consultas realizadas mediante tarareos, canto o silbadas. Su base de datos supera los 2 millones de archivos de música. Actualmente es el mejor sistema de reconocimiento de melodías por tarareo. Como es comercial, los aspectos técnicos de implementación no están disponibles. También admite una búsqueda avanzada de voz que permite aplicar filtros de idioma, género, artista y títulos de canciones.

Capítulo 4

Aplicación StartHum

4.1. Descripción del sistema

El sistema que se ha desarrollado a lo largo de este trabajo de fin de grado tiene como objetivo principal la identificación de melodías mediante consultas de tarareos realizadas por usuarios. Para que el sistema sea más accesible, se ha implementado también una aplicación para dispositivos Android que será la puerta de entrada al sistema.

El sistema de identificación de melodías mediante tarareo implementado está basado en una arquitectura cliente-servidor. Los usuarios (los clientes), desde la aplicación, solicitan el servicio de identificación de melodías al sistema identificador (el servidor) y obtienen en sus dispositivos una lista de canciones ordenada de mayor a menor similitud. Para poder realizar las consultas al sistema de identificación de melodías, el usuario deberá tener instalada, en su dispositivo Android, la aplicación StartHum y acceso a Internet para poder realizar las consultas.

Nuestro sistema cumplirá los requisitos mencionados en el Capítulo 2 y aplicará las técnicas descritas en el Capítulo 3, adecuándose al sistema de identificación de melodías implementado. Para ello, nos hemos centrado en los procesos de extracción de características, su representación y en la función de similitud encargada de comparar los tarareos con las canciones de la base de datos. Sin embargo, el tema de la estructuración de la base de datos no ha sido tratado debido a que trabajaremos con una base de datos relativamente pequeña (50 canciones, ver Apéndice A).

Se espera que las técnicas utilizadas para codificar las melodías (tanto los tarareos como las canciones de la base de datos) permitan obtener un mayor poder discriminativo de los datos y que la función de similitud diseñada mejore el porcentaje de acierto del sistema.

4.2. Arquitectura del sistema

Como se ha mencionado en la Sección 4.1, el sistema estará basado en una arquitectura cliente-servidor. Por ello, la arquitectura de nuestro sistema seguirá la estructura de la Figura 4.1.

La aplicación solo tendrá dos funciones: (1) capturar el tarareo y enviarlo al servidor donde se encuentra el sistema de identificación por tarareo; (2) recibir los resultados y mostrarlos

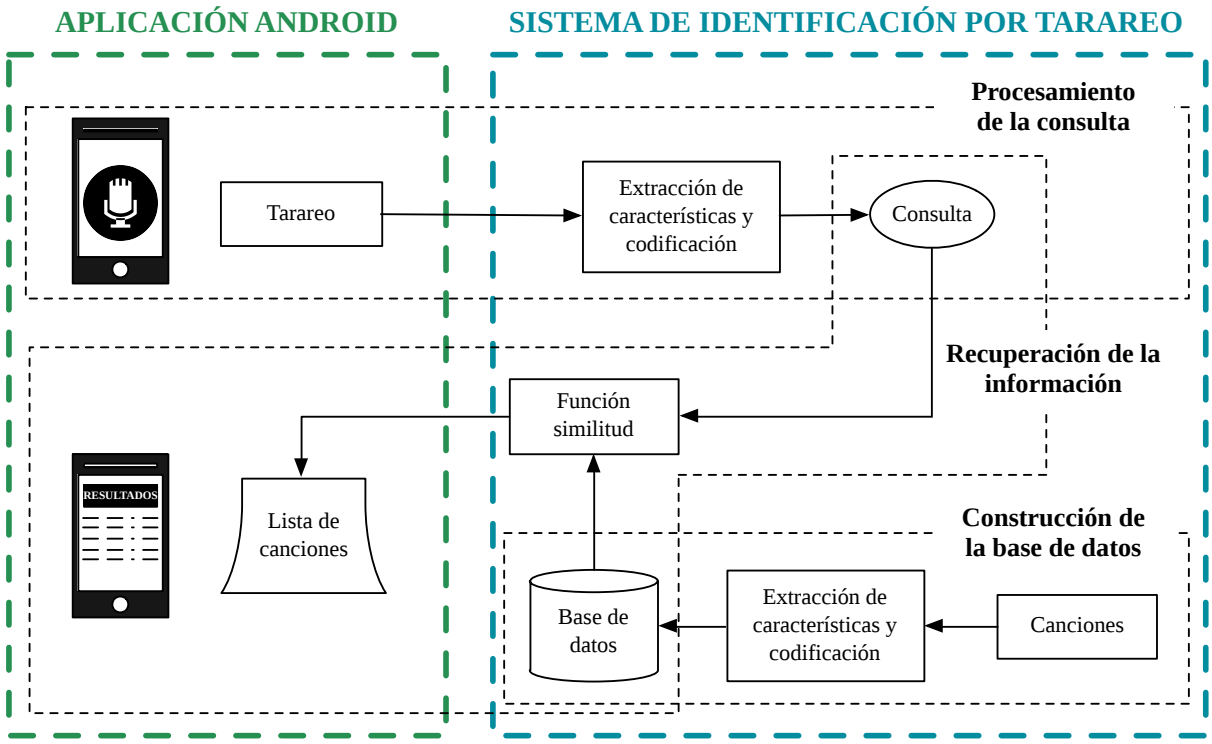


Figura 4.1: Arquitectura de nuestro sistema de consulta mediante tarareo.

al usuario. Mientras, el sistema de identificación de melodías por tarareo estará escuchando por peticiones y será el encargado de procesarlas y devolver los resultados.

4.3. Tecnologías empleadas

A continuación, se explica que tecnologías o herramientas se han utilizado para implementar el sistema de identificación de melodías por tarareo y la aplicación Android. Destacaremos las razones por las cuales se han elegido estas herramientas y haremos referencia a la parte de la arquitectura a la cual pertenecen (ver Figura 4.1).

4.3.1. Lenguaje de programación C++

Debido a que el sistema de identificación de melodías por tarareo debe realizar un trabajo tan costoso, y es imprescindible que el tiempo de cómputo sea bajo para obtener los resultados de forma que el usuario los perciba como instantáneos, se ha utilizado el lenguaje de programación C++ para implementarlo. Este es un lenguaje orientado a objetos muy potente en lo que se refiere a creación de sistemas complejos, robusto y eficientes.

Además, las librerías utilizadas para la extracción de características que se utilizarán durante la labor de la identificación de melodías por tarareo han sido desarrolladas con este lenguaje de programación.

4.3.2. Lenguaje de programación Java

Se ha utilizado el lenguaje de programación Java para implementar la aplicación para dispositivos Android que será la interfaz donde se recogerán las consultas y se mostrarán los resultados. Se podría haber elegido cualquier otra plataforma (Apple IOS, Windows Phone, etc.); sin embargo, se ha decidido utilizar Android debido a que en el último año la cuota de mercado que acapara es aproximadamente del 80 %.

La API de Android nos ha permitido mediante las clases *MediaRecoder* y *AudioRecord* poder realizar capturas de audio de alto nivel de las consultas realizadas por los usuarios. También hemos podido diseñar de forma cómoda una sencilla interfaz para que el usuario visualice el resultado de sus consultas con una *ListView*.

Otra de las razones por las que se ha utilizado el lenguaje de programación Java es por la facilidad de implementación de los *sockets* para la comunicación con el servidor donde se encuentra el sistema de identificación de melodías por tarareo. Se ha utilizado la clase *Runtime*, que permite ejecutar aplicaciones y obtener los resultados; de esta forma, se ha enlazado la aplicación Android con el sistema de consultas implementado en el lenguaje C++.

En resumen, se ha utilizado Java para implementar la aplicación y diseñar el sistema de comunicación con el sistema identificador de melodías.

4.3.3. Librería Aubio

Aubio [5, 6] es una herramienta diseñada para la extracción de características de las señales de audio. Es una librería multiplataforma escrita en el lenguaje de programación C y con una interfaz opcional en Python.

Aubio ofrece las características de: filtros digitales, detección de comienzos o picos (*onset*), extracción del tono, coeficientes para la representación del habla basados en la percepción auditiva humana (MFCC), detección de ritmo y tempo, etc. Además, ofrece unas aplicaciones como ejemplo de las cuales hemos utilizado parte del codigo para la extracción de las características.

Esta herramienta ha sido utilizada en nuestro proyecto para extraer las características del tarareo y para la construcción de la base de datos de canciones. Los datos se codificarán en formato MIDI y para ello se extraerán las características mediante la aplicación de los algoritmos de detección de las frecuencias de la señal y de detección de comienzos (*onsets*). Esta información nos permitirá definir la nota que se reproduce en un instante del tiempo y su duración.

La mayor ventaja que nos ofrece esta librería es la gran variedad de funciones de alto nivel dedicadas al tratamiento del audio y a la extracción del tono. Además, es una librería que sigue en evolución y que está muy bien documentada.

4.3.4. Librería Essentia

Essentia [4] es una librería para el análisis de audio y la recuperación de información musical. Ha sido escrita en el lenguaje de programación C++ y desarrollada por el grupo MTG (*Music Technology Group*) de la Universitat Pompeu Fabra de Barcelona, en la que ha colaborado también el creador de la librería Aubio.

La ventaja que esta librería presenta frente a la descrita en el apartado 4.3.3 es que tiene implementado el algoritmo MELODIA [20]. Este es especialmente adecuado para extraer la melodía predominante en la música polifónica, pero también funciona para señales monofónicas. El enfoque se basa en la creación y caracterización de los contornos melódicos y la eliminación de valores de tono de la señal. El algoritmo MELODIA también será evaluado en nuestro proyecto.

4.3.5. Librería TinyXML2

La información de las canciones que componen la base de datos se ha de almacenar para que el usuario pueda recuperarla tras realizar su consulta al sistema identificador de melodías. La política utilizada para almacenar esta información es mediante un archivo XML (`db.xml`).

TinyXML2 [22] es un simple y eficiente analizador sintáctico de ficheros XML para C++ que se utilizará para leer del fichero XML (`db.xml`) las rutas de las canciones de la base de datos para realizar las comparaciones, y creará un nuevo fichero XML donde se almacenarán los resultados para mostrarselos al usuario que ha realizado la consulta.

4.3.6. Librería PortAudio

PortAudio [3] es una librería multiplataforma de código abierto especializada en audio. Esta librería permite escribir simples programas de audio en C++ o C y proporciona una API muy sencilla para la grabación y reproducción de sonidos (ondas sinusoidales, etc.).

La librería PortAudio no tiene un papel relevante en este proyecto. Esta librería se ha utilizado únicamente para poder reproducir las melodías, representadas por duraciones y valores de nota MIDI, tras haberles aplicado las técnicas de extracción de características. De esta forma, se pudo verificar que se estaba realizando la extracción de forma correcta.

4.3.7. Sonic Visualicer

Sonic Visualicer es una aplicación que permite ver y analizar el contenido de los archivos de audio. Se ha utilizado por este motivo y debido a que desde este programa se permite aplicar los algoritmos de las librerías descritas en los apartados 4.3.3 y 4.3.4, y visualizar los resultados. Algunas imágenes que incluye este trabajo han sido exportadas de este programa.

4.4. Implementación del sistema de identificación

El sistema de identificación de melodías por tarareo se ha estructurado en las siguientes componentes (Figura 4.2):

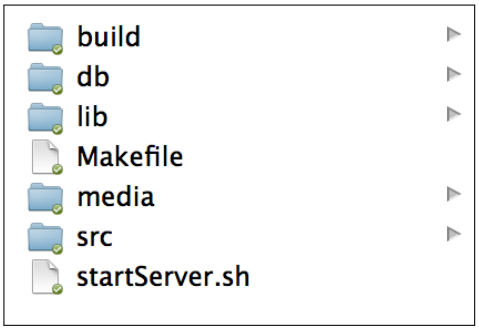


Figura 4.2: Estructura del directorio del sistema de identificación.

- build: donde se encuentran almacenadas las aplicaciones que se han implementado en este proyecto para realizar la tarea de identificación de melodías.
- db: es donde se encuentran las canciones codificadas de la base de datos con las que se aplicará la función de similitud o distancia. También se almacena aquí el fichero `db.xml` con toda la información referente a las canciones.
- lib: en esta carpeta se han almacenado las librerías externas que utiliza este sistema para que al exportar el servidor no sea necesario tener que instalarlas.
- media: aquí se encuentran todos los ficheros multimedia del sistema, es decir, las canciones de la base de datos y los tarareos de consulta realizados por los usuarios.
- src: todo el código implementado se ha almacenado en esta carpeta.

Los ficheros `Makefile` y `startServer.sh` corresponden al compilador de las aplicaciones del sistema de identificación de tarareos y al lanzador del servidor encargado de escuchar las consultas y procesarlas, respectivamente.

4.4.1. Extracción de características

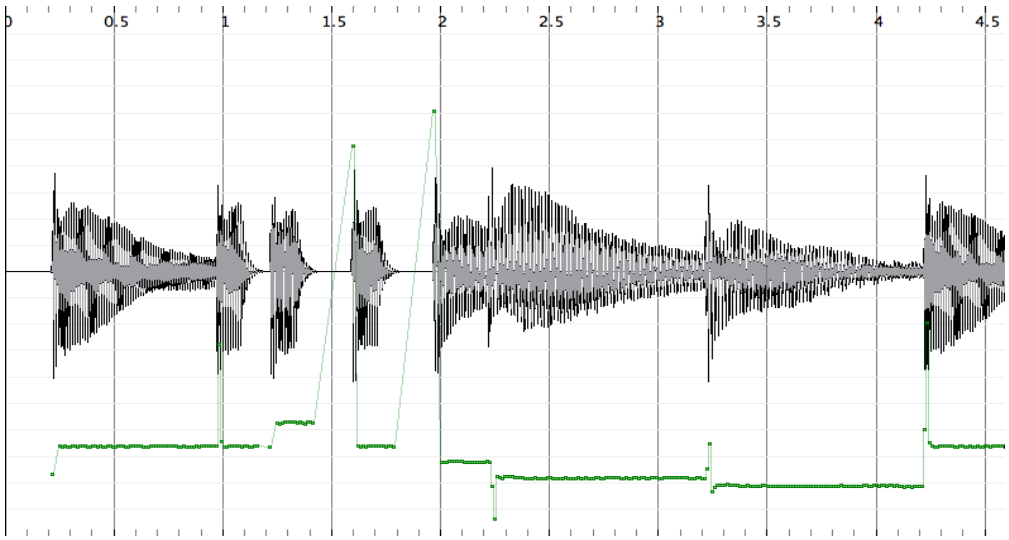
La extracción de características se ha realizado utilizando las librerías de audio mencionadas en la Sección 4.3, debido a que nos ofrecen funciones de alto nivel capaces de extraer las características necesarias para el sistema de identificación de melodías que estamos implementando.

Para realizar esta tarea vamos a implementar dos métodos de extracción de características:

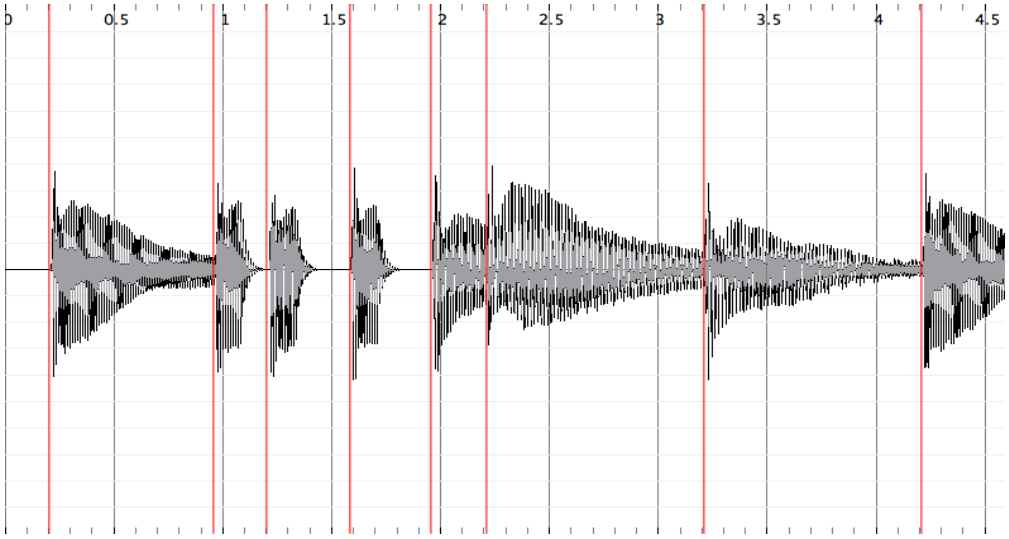
1. Mediante las técnicas que proporciona la librería Aubio para la detección de las frecuencias en cada instante del tiempo de una señal de audio y la detección de comienzos de la misma. Tratando adecuadamente la información obtenida, se consigue la representación de la melodía.
2. Mediante la utilización de la librería Essentia y de su algoritmo MELODIA que obtiene una secuencia de frecuencias con la melodía predominante.

Utilizando la librería Aubio

La librería Aubio proporciona herramientas para la extracción de las características propias de una señal de audio. Entre los algoritmos que nos proporciona la librería se ha hecho uso de: el algoritmo de extracción de las frecuencias, para obtener los valores de la melodía en cada instante del tiempo; y del algoritmo de detección de comienzos, para obtener los intervalos de tiempo entre los que se reproduce una determinada nota. Un ejemplo de los resultados obtenidos por dichos algoritmos pueden visualizarse en la Figura 4.3.



(a) Extracción de frecuencias



(b) Detección de comienzos

Figura 4.3: Extracción de características de la canción “Seven Nation Army”.

Segmentada la señal de audio en intervalos mediante la detección de comienzos y obtenidas las frecuencias en cada instante del tiempo, se ha analizado cada segmento separadamente para elegir qué nota corresponde a dicho intervalo de tiempo. Para realizar la elección de la nota MIDI se han aplicado diversas funciones estadísticas como la moda, la media y la mediana. También se ha utilizado la varianza para decidir en qué intervalo no es posible

detectar una nota.

Tras realizar pruebas con diversas piezas musicales, se comprobó que aplicando la varianza y la moda se obtiene una mejor representación de las mismas que con las otras funciones (media y mediana). La aplicación de la función varianza nos ha permitido detectar los intervalos en los que no es posible realizar una correcta detección de la nota, y la moda elegir la nota MIDI que se reproduce en dicho intervalo. El algoritmo 1 corresponde al pseudocódigo para la elección de la nota en dicho intervalo.

Algoritmo 1: Elección de la nota en los intervalos

Entrada: Frecuencias del intervalo.
Salidas : Nota MIDI correspondiente a dicho intervalo.

Se calcula la varianza de las frecuencias del intervalo
Si $varianza < umbral\_varianza$ **:**
    Se calcula la moda del conjunto de valores MIDI
    Se devuelve la moda
Sino:
    No ha sido posible determinar una nota válida
    Se devuelve 0, que representa el silencio
FinSi

Los resultados obtenidos de la extracción de características, utilizando el método descrito en este apartado, pueden visualizarse en la Figura 4.4.

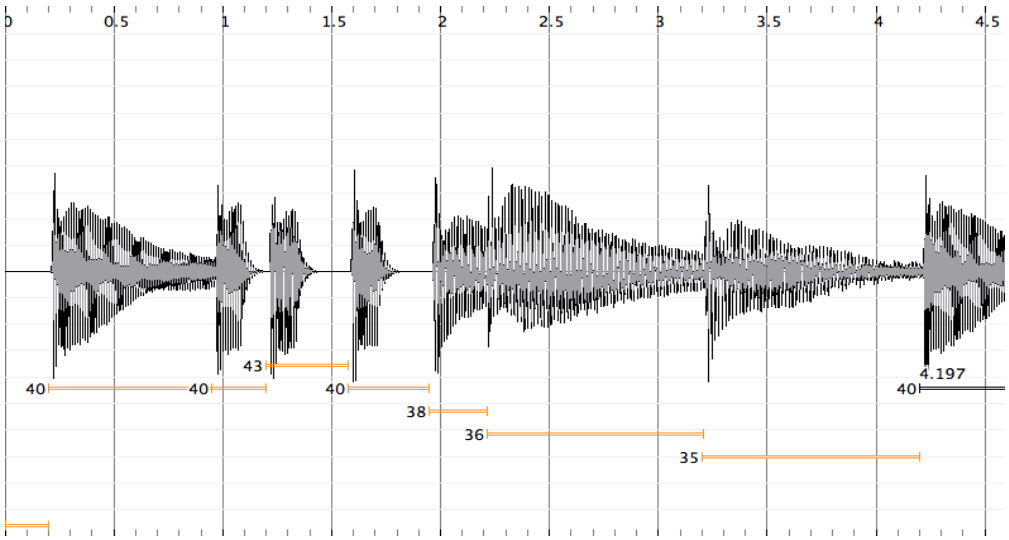


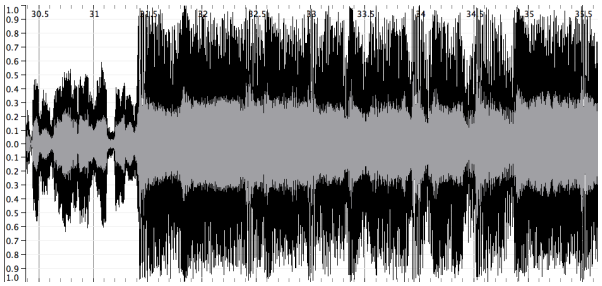
Figura 4.4: Resultado MIDI realizado mediante moda.

La extracción de características que se ha implementado en este apartado constituye la aplicación melody que se ha almacenado en la carpeta de aplicaciones (Figura 4.2).

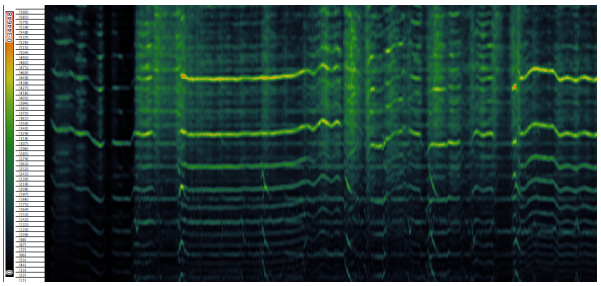
Utilizando la librería Essentia

La librería Essentia proporciona el acceso a la utilización del algoritmo MELODIA para la extracción de las características tonales de la melodía predominante de una pieza musical polifónica.

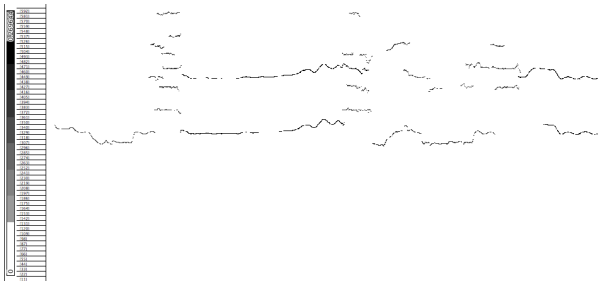
El algoritmo MELODIA consiste en la aplicación de 4 procesos. Dada una señal de audio polifónica o monofónica (Figura 4.5a), se extraen las frecuencias presentes en la señal de audio en todos los puntos del tiempo. Dadas estas frecuencias, se aplica una función de relevancia que acumula la energía de estas frecuencias armónicas para destacar las notas más importantes (Figura 4.5b). De los resultados del proceso anterior, se realiza un seguimiento de los contornos de tono (Figura 4.5c) de los cuales se determina los que pertenecen a la melodía y los que no y se obtiene el resultado de la Figura 4.5d.



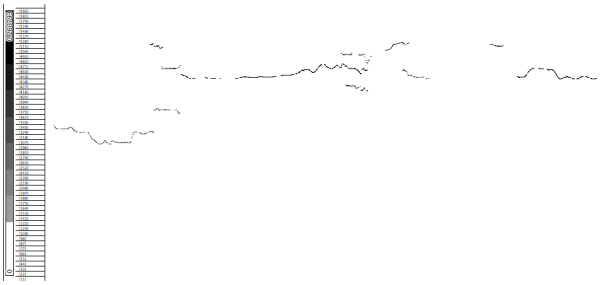
(a) Señal de audio “Everybody Talks”



(b) Representación de la energía de las frecuencias



(c) Contorno de tonos



(d) Melodía predominante resultante

Figura 4.5: Ejemplo de funcionamiento del algoritmo MELODIA.

Essentia está basado en una colección de algoritmos. Esta librería no proporciona la lógica común de alto nivel de programación. Para utilizar los algoritmos de Essentia es necesario instanciar la factoría de algoritmos, indicar los algoritmos que se var a utilizar con sus parámetros, indicar las entradas y salidas de los mismos y aplicar una orden para que se ejecute. Se proporciona un ejemplo de su uso en la Figura 4.6, donde pueden verse comentados los pasos necesarios para utilizar los algoritmos de la librería.

La extracción de la frecuencia en cada instante del tiempo de una pieza de música la realizamos aplicando el algoritmo MELODIA. En la conversión de esta sucesión de frecuencias a notas MIDI con duración aplicamos el algoritmo de detección de comienzos que también se proporciona en esta librería. De esta forma, obtenemos de una pieza de música su melodía en formato MIDI (Figura 4.7).

```
1  // inicia essentia
2  essentia::init();
3
4  ...
5
6  // inicia la fábrica y añade los algoritmos
7  streaming::AlgorithmFactory& factory = streaming::AlgorithmFactory::
       instance();
8
9  Algorithm* audioload = factory.create("MonoLoader",
10                                       "filename", source,
11                                       "sampleRate", sample_rate,
12                                       "downmix", "mix");
13
14 Algorithm* equalLoudness = factory.create("EqualLoudness");
15 Algorithm* predominantMelody = factory.create("PredominantMelody",
16                                               "frameSize", frame_size,
17                                               "hopSize", hop_size,
18                                               "sampleRate", sample_rate,
19                                               "voicingTolerance",
                                                     voice_tolerance);
20
21 ...
22
23 // conecta los algoritmos
24 audioload->output("audio") >> equalLoudness->input("signal");
25 audioload->output("audio") >> onsetrate->input("signal");
26
27 equalLoudness->output("signal") >> predominantMelody->input("signal");
28
29 predominantMelody->output("pitch") >> PC(pool, "tonal.predominant_melody.
       pitch");
30 predominantMelody->output("pitchConfidence") >> PC(pool, "tonal.
       predominant_melody.pitchConfidence");
31
32 ...
33
34 // lanza los algoritmos
35 Network network(audioload);
36 network.run();
37
38 ...
39
40 // cierra essentia
41 essentia::shutdown();
```

Figura 4.6: Aplicación de algoritmos Essentia.

La extracción de características que se ha implementado en este apartado constituye la aplicación predominant_melody que se ha almacenado en la carpeta de aplicaciones (Figura 4.2).

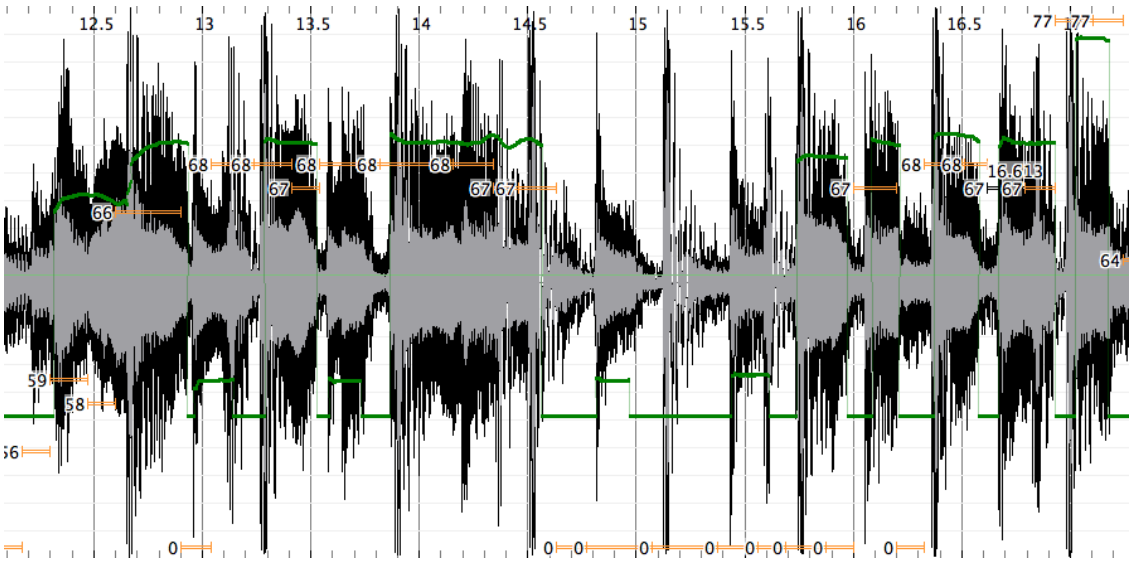


Figura 4.7: Señal de audio “Hey, Soul Sister” y resultado de la extracción.

4.4.2. Construcción de la base de datos

Para la construcción de la base de datos se eligieron previamente un total de 50 canciones de diversos géneros (ver Apéndice A). La extracción y codificación de las características más discriminativas de estas canciones se extraen mediante las técnica descritas en el apartado 4.4.1. Únicamente se han escogido las canciones originales para construir la base de datos para que de esta forma el sistema sea más flexible frente a nuevas entradas.

Las características extraídas se almacenan en ficheros de texto plano siguiendo el esquema de la Figura 4.8, donde cada fila representa el instante de inicio de la nota, su duración y el valor en formato MIDI de la nota.

3.366894	0.098685	36
3.465578	0.754649	37
4.220227	0.116100	38
4.336327	0.081270	35
4.417596	0.423764	36
⋮	⋮	⋮

Figura 4.8: Aspecto de un elemento de la base de datos

Además de la información que se almacena en la base de datos para realizar la identificación, también es necesario almacenar los metadatos de las canciones, es decir, la información relativa al título de las canciones, nombre del artista, etc. Esta información es almacenada en el fichero XML `db.xml`, la estructura del cual está formada por entradas de canciones con los metadatos de las mismas (Figura 4.9).

La información del fichero XML es utilizada para acceder de forma directa a la ruta donde se encuentran los ficheros codificados de las canciones, a las cuales representan, para

```
<?xml version="1.0"?>

<!DOCTYPE db_schema [
<!ELEMENT repertory (song+)>
<!ELEMENT song (author,title,genre,thumb_url,samples)>
<!ELEMENT samples (sample+)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT genre (#PCDATA)>
<!ELEMENT thumb_url (#PCDATA)>
<!ELEMENT sample EMPTY>

<!ATTLIST song id ID #IMPLIED>
<!ATTLIST song path CDATA #IMPLIED>
<!ATTLIST sample path CDATA #IMPLIED>
<!ATTLIST sample weigth CDATA #IMPLIED>
]>
```

Figura 4.9: Fichero DTD que define la estructura del fichero XML `db.xml`.

aplicar la función distancia y para responder a la consulta del usuario con la información de las mismas.

4.4.3. Representación de la melodía

Las melodías se representarán de dos formas para compararlas entre ellas: mediante la codificación UDS y mediante la codificación MIDI sin alteraciones. Debido a que en la base de datos se almacenará en formato MIDI, se ha diseñado una función que, dados los valores MIDI de las notas y la duración, convierta esta codificación al formato UDS (Figura 4.10).

Como puede apreciarse en el código, la codificación UDS utilizada es la que utiliza 6 símbolos para representar tanto el contorno melódico como el temporal. De esta forma, se obtendrá un mayor poder discriminativo de los datos.

4.4.4. Función similitud

La función de similitud será la encargada de identificar la melodía tarareada por el usuario. La combinación de esta función con la técnica de extracción de características es la clave que permitirá obtener una correcta identificación de la melodía.

Para realizar esta tarea vamos a implementar dos funciones de similitud o identificación de la melodía que corresponden con variaciones de las funciones descritas en el Capítulo 2.

1. La función distancia de edición (o Levenshtein) con pesos adaptados a nuestro problema de la identificación.

2. La función distancia de alineamiento temporal no lineal (o DTW) con un sistema más complejo de penalización que permitirá obtener mejores resultados.

Debido a que los algoritmos de ambas funciones permiten iniciarse en todos los puntos de la melodía es necesario tener en cuenta varios aspectos: es necesario tener definido un

```
1  /**
2   * @brief Codificación a formato UDS.
3   *
4   * @param stream Puntero a un objeto FILE de la base de datos.
5   * @param seq Donde se almacenará la secuencia UDS.
6   */
7  void convert_to_UDS (FILE *stream, vector<int> &seq)
8  {
9    double onset, duration, note;
10   double last_duration = 0, last_note;
11
12   while (fscanf (stream, "%lf %lf %lf", &onset, &duration, &note) == 3) {
13     if (note) {
14       if (last_duration) {
15         seq.push_back ((last_note > note) ? 'D' : ((last_note < note) ? 'U
               ' : 'S'));
16         seq.push_back ((last_duration > duration) ? 'S' : ((last_duration
               < duration)? 'L' : 'E'));
17       }
18       last_duration = duration;
19       last_note = note;
20     }
21   }
22 }
```

Figura 4.10: Función de MIDI a representación UDS.

coste máximo o umbral para descartar todo aquello que sobrepasa dicho umbral (los valores estarán normalizados), elegir un número máximo de hipótesis y no permitir solapamientos entre hipótesis. Para ello es necesario almacenar en cada punto, a parte de coste, el inicio y fin de dicha hipótesis.

La tarea de identificación de melodías dada una consulta constituye la aplicación matching, que se ha almacenado en la carpeta de aplicaciones (Figura 4.2), la cual tiene implementadas ambas funciones de identificación.

Función distancia Levenshtein implementada

Para la identificación de las melodías, se ha implementado el algoritmo distancia de edición descrito en el apartado 3.3.1 del Capítulo 3. Este algoritmo se ha utilizado únicamente para realizar las comparaciones con las melodías representadas en formato UDS.

Para adaptar dicho algoritmo a la tarea de la identificación que se pretende lograr en este trabajo, se ha personalizado la función distancia. Debido a que la base de datos ha sido construida únicamente con las canciones originales, es posible que su representación tenga notas que no pertenecen a la melodía principal. Por ello, la distancia o coste asignado a las acciones de reemplazo e inserción son mayores que la acción de borrado. De esta forma, se suaviza que la melodía de la base de datos contenga ruido proveniente de la extracción de las características de las melodías polifónicas. Los pesos asociados a las acciones de borrado, inserción y reemplazo son de 1, 2 y 3, respectivamente.

En cuanto a los aspectos relacionados con la normalización de las puntuaciones y la elección de las hipótesis, se ha realizado, en primer lugar, la normalización de los costes con la longitud de las hipótesis; seguidamente, la ordenación de las hipótesis de menor a mayor coste; en tercer lugar, se han eliminado las hipótesis solapadas de mayor coste; para finalizar, se hace la elección de las hipótesis de acuerdo a un máximo de 3 hipotesis por canción. También comprobamos que la hipótesis no sea excesivamente larga como resultado de aplicarle un peso bajo a la acción de borrado.

De esta forma, obtenemos los resultados de la comparación realizada que se acumularán junto con los de las otras canciones y se devolverán al usuario los de menor coste.

Función DTW implementada

Para la identificación de las melodías codificadas en formato MIDI, se ha implementado el algoritmo DTW modificado para permitir el inicio en cualquier punto de la melodía, y que ha sido explicado en el apartado 3.3.2 del Capítulo 3.

La asignación de los pesos para establecer la distancia no se ha realizado tan solo teniendo en cuenta la distancia entre dos valores MIDI, sino que se ha tenido en cuenta también la distancia de las notas de la comparación anterior. Es por ello que cada punto de la comparación almacena el inicio, el fin, el coste actual acumulado y, además, la distancia de las notas MIDI. El objetivo de este algoritmo es alinear dos funciones en el tiempo, sin normalizar sus valores, y que obtengamos la hipótesis en la cual su altura respecto a las notas de la melodía se mantenga. Un ejemplo que ilustra el comportamiento del algoritmo puede verse en la Figura 4.11.

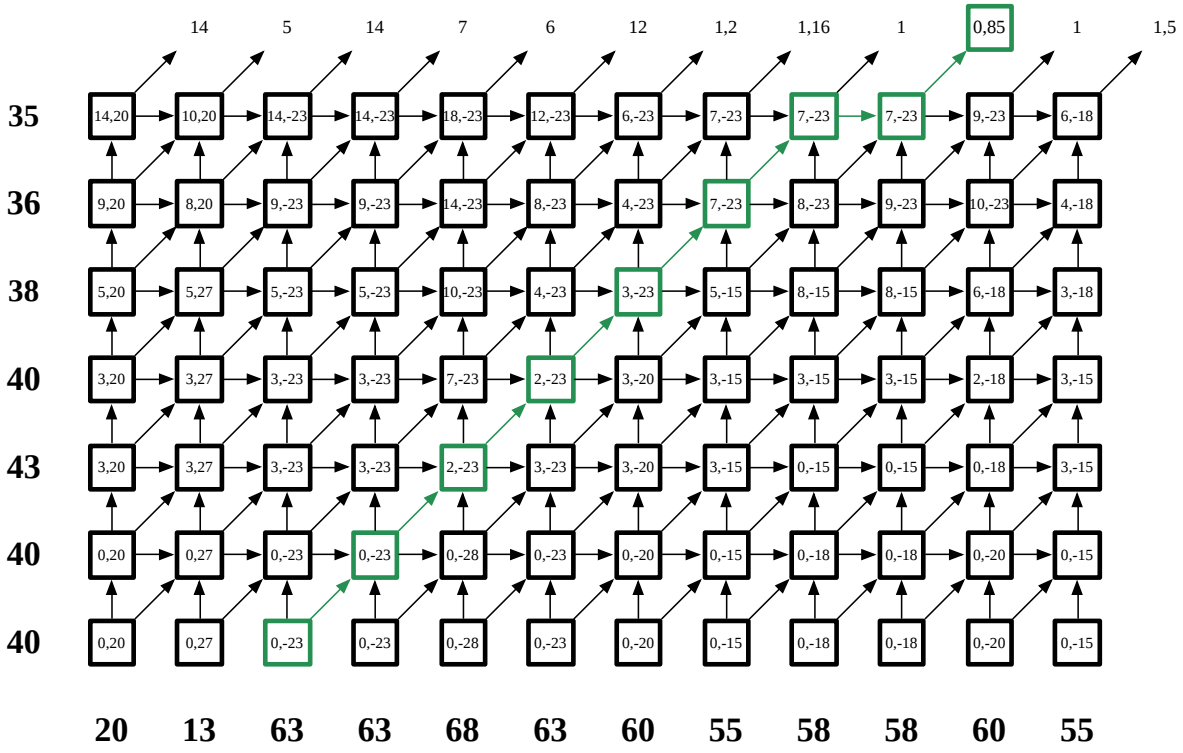


Figura 4.11: Ejemplo de comparación entre dos secuencias con DTW.

Los valores que contiene cada nodo corresponden al coste acumulado, y la altura de la secuencia respecto a la consulta. En estos también se acumula el inicio y el fin que nos permitirá normalizar los costes (última fila). El algoritmo funciona de acuerdo a las Ecuaciones (4.1), (4.2), (4.3) y (4.4).

$$d(i,j) = h(i) - r(j) \tag{4.1}$$

$$DTW(0,j) = 0 \tag{4.2}$$

$$DTW(i,0) = DTW(i-1,0) + |altura_{i-1,0} - d(i,0)| \tag{4.3}$$

$$DTW(i,j) = min \begin{cases} DTW(i-1,j) + |altura_{i-1,j} - d(i,j)| \\ DTW(i-1,j-1) + |altura_{i-1,j-1} - d(i,j)| \\ DTW(i,j-1) + |altura_{i,j-1} - d(i,j)| \end{cases} \tag{4.4}$$

h y r hacen referencia a una consulta y a una determinada canción de la base de datos contra la que se está comparando, respectivamente. Al igual que la función de comparación anterior se normaliza el resultado, se eliminan los posibles solapamientos de las hipótesis y se elige un conjunto máximo de 3 hipótesis.

De esta forma realizamos la identificación sin tener que preocuparnos de las diferencias en cuanto a tonalidad, tempo o instante de inicio del tarareo.

4.4.5. Sistema de comunicación

El sistema de identificación de melodías por tarareo que se ha descrito proporciona la funcionalidad para comparar e identificar la melodía correspondiente a los tarareos realizados por los usuarios. Debido a que se ha decidido utilizar la plataforma Android como entrada al sistema, es necesario implementar la conexión entre ambos.

La aplicación Android enviará las consultas realizadas por los usuarios y recibirá los resultados de la identificación. Debido a que la aplicación que envía las consultas está implementada en Java, se ha implementado en Java la conexión. Para ello se ha hecho uso de los *sockets* que permiten la comunicación entre procesos de diferentes máquinas de una red de forma sencilla. El protocolo de envío que se ha decidido utilizar es TCP (*Transfer Control Protocol*).

Además, como es posible que diversos usuarios realicen, desde dispositivos diferentes, consultas al sistema de identificación al mismo tiempo; se ha decidido hacer uso de hilos de ejecución. De esta forma se podrán atender las consultas de forma concurrente (ver Figura 4.12).

La clase *WorkerRunnable* es la encarregada de recibir la información relacionada con la consulta (identificador del dispositivo Android y señal de audio que representa la consulta), lanzar los procesos de identificación de la melodía y enviar el resultado XML al dispositivo Android.

```
1  ServerSocket server;
2  Socket connection;
3
4  try {
5    server = new ServerSocket(7000);
6    while (true) {
7      System.out.println("Esperando nuevo cliente...");
8      connection = server.accept();
9      // Llega un cliente.
10     System.out.println("Nuevo cliente aceptado");
11
12     new Thread(new WorkerRunnable(connection, "Multithreaded Server")).
           start();
13   }
14 } catch (Exception e) {
15   System.err.println(e);
16 }
```

Figura 4.12: Servidor que recibe las conexiones al *socket* y lanza el hilo de ejecución que las atiende.

4.5. Implementación de la aplicación Android

La aplicación Android es el medio mediante el cual los usuarios realizan consultas al sistema identificador de melodías y visualizan los resultados. La aplicación tiene como nombre StartHum, que simboliza la acción de empezar a tararear. Esta aún no se encuentra disponible en la tienda oficial de Android (*Play Store*), pero se pretende publicarla cuando el sistema de identificación sea más estable y se aumente el número de canciones. El icono diseñado para la aplicación puede verse en la Figura 4.13.

Figura 4.13: Icono de la aplicación StartHum.

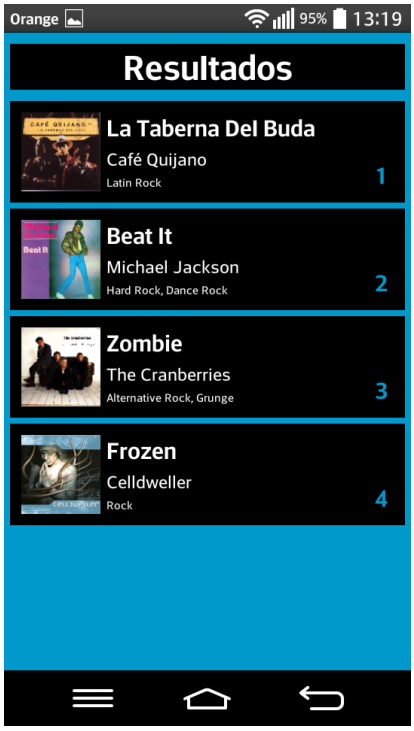
Al lanzarse la aplicación en los dispositivos Android se crea una carpeta, si no existe previamente, con el nombre de la aplicación donde se almacenan los archivos necesarios de la aplicación.

4.5.1. Interfaz de usuario

La aplicación diseñada para Android es bastante sencilla y solo tiene 2 ventanas o actividades (Figura 4.14).

La interfaz de captura de la consulta, que es la interfaz principal de inicio de la aplicación, solo contiene un botón (la imagen del micrófono) que al pulsarlo captura 15 segundos de audio. Una vez realizada la captura y obtenida la respuesta, se muestra la interfaz de

(a) Interfaz de captura de la consulta



(b) Interfaz de visualización de resultados

Figura 4.14: Interfaces de la aplicación StartHum.

visualización de los resultados.

Tras pulsar el botón de captura del audio desaparece el micrófono y se muestra un mensaje que indica que se está realizando la captura de la consulta. Además, se inicia una animación que escala el botón circular central, durante dichos segundos, para ofrecer al usuario la sensación de continuidad en la aplicación.

Los resultados obtenidos son mostrados de forma ordenada según su proximidad a la consulta realizada. Para realizar nuevas consultas solo es necesario volver a la interfaz principal.

4.5.2. Captura de la consulta

El audio que se captura desde los dispositivos Android debe tener la mejor calidad posible para que el sistema de identificación de la consulta funcione adecuadamente. Éste trabaja mejor con ficheros de audio en formato ".wav" (o equivalente). La captura de la señal de audio que se ha implementado para la aplicación realiza la grabación del audio en formato ".wav" (alta calidad).

Según qué dispositivos se utilizan, la calidad del micrófono es mejor o peor. El grabador de audio implementado se configura de forma óptima eligiendo los parámetros adecuados para obtener el mejor resultado durante la grabación. Las capturas realizadas se almacenan en la carpeta de la aplicación y se eliminan cuando no son necesarias. Además, se decidió que la captura de la consulta se fijaría a 15 segundos porque es el tiempo suficiente para tener un buen poder discriminativo.

4.5.3. Visualización de los resultados

Una vez recibido el resultado en formato XML, por parte del sistema de identificación de melodías por tarareo, se procesa dicho fichero extrayendo sus características. El título de la canción, el artista que la interpreta y los géneros a los cuales pertenece dicha canción son mostrados al usuario, por este orden de importancia, mediante una vista de Android llamada *ListView* que permite visualizar los resultados por filas. Además, los resultados se muestran ordenados de mayor a menor grado de similitud.

Para que el usuario obtenga una mayor experiencia visual de los resultados obtenidos, se muestra una imagen representativa del disco de la canción a la cual pertenecen. Esta información, en lugar de ser obtenida desde el servidor (lo que retrasaría los tiempos de obtención de los resultados debido a que sería necesario esperar a obtener la imagen para procesarse), se obtiene directamente de un enlace Web. Esta información también se almacena en la carpeta de la aplicación y puede ser utilizada para otras consultas.

Capítulo 5

Evaluación de resultados

5.1. Descripción del escenario

Se ha llevado a cabo una serie de evaluaciones, en condiciones reales, del sistema de identificación de melodías implementado. De esta forma, se estudia el alcance del sistema con los diferentes métodos de extracción de características y funciones de identificación implementados, así como la media de los tiempos de respuesta. Además, se muestra una idea aproximada de su funcionamiento.

Para realizar la evaluación del sistema de identificación de melodías mediante tarareo se ha tomado un total de 80 muestras de tarareos realizados por 10 individuos diferentes. Las acciones permitidas a los usuarios para realizar las consultas han sido las de canto, silbido y tarareo, como se especificó en la descripción del sistema (Capítulo 4). No se estableció ninguna restricción sobre el tramo de melodía interpretado. Además, las muestras tomadas tienen una duración de 15 segundos, que equivale a la captura realizada por la aplicación Android.

5.2. Resultados de la identificación

Para analizar los resultados obtenidos tras realizar las identificaciones de las muestras se ha utilizado la métrica tasa de acierto superior a una posición x. En nuestro caso, evaluamos el sistema teniendo en cuenta el porcentaje de canciones que se han identificado en primera posición, los obtenidos entre las tres primeras posiciones y los obtenidos entre las cinco primeras. La Tabla 5.1 muestra los resultados obtenidos.

	Extracción mediante Aubio		Extracción mediante Essentia	
	LD	DTW	LD	DTW
Primeros (%)	22,50	28,75	15,00	18,75
Primeros 3 (%)	36,25	42,50	31,25	22,25
Primeros 5 (%)	48,75	63,75	33,75	37,50

Tabla 5.1: Tasas de acierto del sistema de identificación de melodías.

La evaluación se ha llevado a cabo probando con las dos técnicas de extracción automática de características implementadas (apartado 4.4.1), tanto para la construcción de la base de datos como para las consultas realizadas, y las funciones de similaridad LD (*Levenshtein*

Distance) y DTW (*Dynamic Time Warping*) personalizadas para el problema de identificación de melodías (apartado 4.4.4).

Como se puede observar, los resultados obtenidos de la identificación no son todo lo buenos que se esperaba obtener. Aún así, se ha podido observar que la función similitud DTW implementada ha tenido más tasa de acierto que la distancia de edición y que la extracción de características mediante Aubio ha proporcionado mejores resultados. Tras realizar la evaluación, se ha detectado que las canciones que han facilitado su identificación son aquellas con menores acompañamientos musicales.

También se ha identificado una pequeña dependencia respecto a la calidad del tarareo realizada por el usuario, al dispositivo de captura utilizado y a la forma de realizar la consulta al sistema (voz cantada, tarareada o silbada).

5.3. Resultados temporales

En cuantos a los tiempos de respuesta del sistema de identificación, se han realizado varias pruebas desde diversos dispositivos Android y se ha detectado que el tiempo necesario para que el usuario visualice los resultados oscila entre 1 y 2 segundos.

La mayor parte de ese tiempo está dedicada al envío de la consulta y a la recepción de la respuesta. Por tanto, un aspecto a mejorar para próximas versiones sería enviar la captura mediante *streaming* o extraer las características en el propio dispositivo y enviar un fichero de texto plano, que es más reducido.

Los tiempos necesarios para la extracción de características y la función de similitud son insignificantes. Cabría probar a aumentar la base de datos y comprobar cuánto afecta esto a los tiempos de respuesta del sistema de identificación.

Capítulo 6

Conclusiones

6.1. Conclusiones

Dada la amplitud y complejidad del problema de identificación de melodías mediante tarareo, el presente trabajo es apenas una aproximación al tema. Aún así, se han abordado aspectos importantes de la identificación, como la extracción de las características, su representación y la función de comparación, obteniendo de esta forma un sistema completo que dispone de una aplicación Android que permite acceder a dicho sistema de identificación, cumpliendo así con los objetivos marcados en la Sección 1.2.

Al principio de realizar el proyecto se desconocía el funcionamiento de los sistemas de identificación de melodías por tarareo, pero el estudio realizado de los mismos nos ha permitido adquirir los conocimientos necesarios e implementar nuestro propio sistema de identificación.

Los principales logros alcanzados en este proyecto han sido: la elaboración de las técnicas que han permitido, dada la secuencia de frecuencias obtenidas por las librerías de audio, poder extraer las notas correctas de las melodías; el diseño de la función DTW implementada, la cual no necesita normalizar los valores de las notas para realizar la correcta comparación; y la implementación de una aplicación sencilla que permite a los usuarios realizar las consultas.

Las principales dificultades encontradas se han debido a la dependencia de las librerías de audio para la extracción de características y la generalización de los parámetros de éstas para obtener en todo momento una buena representación de las melodías. Las canciones utilizadas para la creación de la base de datos, al ser polifónicas, han incluido ruido que no se ha podido eliminar completamente durante la extracción automática de las características.

Como resultado final del trabajo se ha obtenido un sistema de identificación de las melodías, el cual identifica las melodías mediante consultas de canto, tarareo o silbido con un resultado de acierto del 28,75 % en la primera posición, 42,50 % en las tres primeras posiciones y 63,75 % en las cinco primeras. Además, se ha implementado una aplicación Android para realizar las consultas al sistema de identificación.

Aunque los resultados no han sido demasiado buenos, hay que tener en cuenta que esta ha sido una primera aproximación que irá mejorando con el tiempo y la experiencia. El objetivo final será comercializar la aplicación cuando ésta haya sido mejorada y su base de datos ampliada.

6.2. Trabajo futuro

La transcripción automática de las melodías ha sido una tarea crítica en este proyecto que nos ha impedido obtener mejores resultados. Sería necesario profundizar más en este tema y obtener las características de las melodías a más bajo nivel para tener toda la información para realizar la extracción y transcripción de dichas características de forma que obtengamos un mayor poder discriminativo. También resultaría interesante probar otras técnicas de comparación más complejas y de fragmentación de la melodía en submelodías para realizar la identificación.

Una vez mejorados estos aspectos, se debería volver a evaluar el sistema de identificación de melodías aumentando la base de datos de forma exponencial y comprobarse en cuánto se reduce la capacidad discriminativa del conjunto de canciones y en cuánto aumentan los tiempos de recuperación de la información. En el caso de que los tiempos de recuperación fueran demasiado elevados, se debería decidir si es necesario estructurar la base de datos.

Otro aspecto interesante que podría abordarse es el de realimentar la base de datos con tarareos que fueran lo suficientemente representativos o, incluso, construir la base de datos solamente con tarareos. Estos dos puntos deberían estudiarse para ver qué ventajas y desventajas ofrecen al sistema de identificación de melodías.

En cuanto a la aplicación Android, sería interesante proporcionar al usuario la posibilidad de reproducir las canciones que ha obtenido como resultado de la tarea de identificación realizada por el sistema implementado, así como dar la posibilidad también de realizar la compra de dichas canciones a través de la tienda de música del *Play Store*.

Todas estas posibles mejoras no han podido ser abordadas durante el transcurso de la realización del trabajo por falta de tiempo o por exceder sus objetivos. Se pretende alcanzarlas en un futuro y así poder comercializar la aplicación Android en el *Play Store*.

Apéndice A

Canciones que componen la base de datos

#	Título	Artista	Género
01	Seven Nation Army	The White Stripes	Alternative
02	Smells Like Teen Spirit	Nirvana	Grunge
03	Somebody That I Used To Know	Gotye	Alternative
04	Hey Soul Sister	Train	Pop Rock
05	Titanium	David Guetta	Pop, Soul
06	Everybody Talks	Neon Trees	Alternative
07	Call Me Maybe	Carly Rae Jepsen	Pop
08	I Want To Break Free	Queen	Rock
09	Zombie	The Cranberries	Alternative, Grunge
10	The Show Must Go On	Queen	Symphonic, Hard Rock
11	We Will Rock You	Queen	Arena Rock
12	Beat It	Michael Jackson	Hard Rock, Dance Rock
13	Thriller	Michael Jackson	Pop, Funk
14	Yesterday	Beatles	Baroque Pop
15	Welcome To The Jungle	Guns N' Roses	Heavy Metal, Hard Rock
16	Dirty Paws	Of Monsters And Men	Indie Folk, Indie Pop
17	Imagine	John Lennon	Rock
18	Californication	Red Hot Chilli Peppers	Alternative, Funk Rock
19	I Am Yours	Jason Mraz	Pop Rock, Reggae
20	Por La Boca Vive El Pez	Fito y Fitipaldi	Rock
21	Muñeca De Trapo	La Oreja De Van Gogh	Pop Rock
22	Danza Húngara nº5	Johann Brahms	Classical Music
23	Hoy No Me Puedo Levantar	Mecano	Tecno Pop
24	Me Colé En Una Fiesta	Mecano	Pop Rock
25	Sin Ti No Soy Nada	Amaral	Blues
26	Princesas	Pereza	Pop Rock
27	La Taberna Del Buda	Café Quijano	Latin Rock
28	Tequila	Café Quijano	Latin Rock
29	Frozen	Celldweller	Rock
30	It's Raining Men	Geri Halliwell	Disco, Dance
31	Por Si Te Vas	La Musicalité	Pop Rock
32	Love The Way You Lie	Eminem	Hip Hop, Pop

33	Boadicea	Enya	New Age, Celta
34	Rehab	Amy Winehouse	Soul, Jazz
35	It's My Life	Bon Jovi	Hard Rock, Glam Metal
36	Cero	Dani Martin	Pop Rock
37	El Porompompero	Manolo Escobar	Rumba
38	Eva Maria	Formula V	Pop, Bubblegum Pop
39	We Are Young	Fun	Indie Pop, Pop
40	Terriblemente Cruel	Leiva	Pop, Pop Rock
41	Costa Del Silencio	Mägo De Oz	Heavy, Metal
42	Waves	Mr Probz	Acoustic
43	Whatever	Oasis	Britpop
44	One	Order Of Era	Pop, Alternative
45	Si Yo Tuviera Una Escoba	Los Sírex	Rock
46	I'm A Terrible Person	Rooney	Rock, Power Pop
47	El Toro Y La Luna	El Matador	Flamenco, Rumba
48	El Ciclo De La Vida	El Rey Leon	Pop, African Choir
49	La Flaca	Jarabe De Palo	Latin Rock
50	La Luna Me Sabe A Poco	Marea	Rock

Bibliografía

[1] Adams, N. H. Time series representations for music information retrieval, 2004.

[2] Anand, Sundaram, B., and Rao, P. Tansen: A query-by-humming based music retrieval system. *Proc. National Conference on Communications (NCC)* (2003), 75–79.

[3] Bencina, R. Portaudio. `http://www.portaudio.com`, 2007.

[4] Bogdanov, D., Wack, N., Gómez, E., Gulati, S., Herrera, P., Mayor, O., Roma, G., Salamon, J., Zapata, J., and Serra, X. Essentia: an audio analysis library for music information retrieval. In *International Society for Music Information Retrieval Conference (ISMIR'13)* (Curitiba, Brazil, 04/11/2013 2013), pp. 493–498.

[5] Brossier, P. *Automatic Annotation of Musical Audio for Interactive Applications*. PhD thesis, Queen Mary University of London, UK, August 2006.

[6] Brossier, P. M. Aubio, a library for audio labelling. `http://aubio.piem.org`, 2003.

[7] de Cheveigné, A., and Kawahara, H. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America 111*, 4 (2002), 1917–1930.

[8] Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., and Yanker, P. Query by image and video content: The qbic system. *Computer 28*, 9 (1995), 23–32.

[9] Google. Image search on google. `http://www.google.com/insidesearch/features/images/searchbyimage.html`, 2010.

[10] Jin, M., Kim, J., and Yoo, C. D. Humming-based human verification and identification. *Acoustics, Speech, and Signal Processing, IEEE International Conference* (2009), 1453–1456.

[11] Kline, R. L., and Glinert, E. P. Approximate matching algorithms for music information retrieval using vocal input. In *Proceedings of the Eleventh ACM International Conference on Multimedia* (New York, NY, USA, 2003), MULTIMEDIA '03, ACM, pp. 130–139.

[12] Kosugi, N. *A study on content-based music retrieval with humming*. PhD thesis, 2005.

[13] López, E., and Rocamora, M. Tararira: Sistema de búsqueda de música por melodía cantada. In *Proc. of the X Brazilian Symposium on Computer Music* (2005), pp. 142–153.

[14] Midomi. `http://www.midomi.com`, 2006.

[15] Muda, L., Begam, M., and Elamvazuthi, I. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *CoRR abs/1003.4083* (2010).

[16] Nielsen, J. Response times: The 3 important limits. `http://whichww.nngroup.com/articles/response-times-3-important-limits/`, 1993.

[17] Papiotis, P. Real-time accompaniment usinging lyrics-matching query-by-humming (qbh). Master's thesis, 2010.

[18] Pertusa, A. Transcripción de melodías polifónicas mediante redes neuronales dinámicas, 2003. Master Thesis, Universidad de Alicante, Departamento de Lenguajes y Sistemas Informáticos.

[19] Rocamora, M., Cancela, P., and Pardo, A. Query by humming: Automatically building the database from music recordings. *Pattern Recognition Letters 36* (2014), 272–280.

[20] Salamon, J., Gómez, E., Ellis, D., and Richard, G. Melody extraction from polyphonic music signals: Approaches, applications and challenges. *IEEE Signal Processing Magazine 31* (year02/2014 2014), 118–134.

[21] Shazam. `http://www.shazam.com`, 2008.

[22] Thomason, L. Tinyxml-2. `https://github.com/leethomason/tinyxml2`, 2010.

[23] Zhu, Y., and Shasha, D. Warping indexes with envelope transforms for query by humming. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2003), SIGMOD '03, ACM, pp. 181–192.