UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Reconocedor Automático de Melodías de Música Clásica

Proyecto Final de Carrera

Ingeniería Informática

Autor: Miguel Domingo Ballester

Director: Carlos David Martínez Hinarejos

10 de diciembre de 2014

Resumen

Todos hemos escuchado alguna vez una melodía y nos hemos quedado con la duda de saber el título de la canción a la que pertenece. Existen aplicaciones como Shazam capaces de solucionar este problema. Sin embargo, estas no funcionan tan bien cuando nos encontramos dentro del ámbito de la música clásica.

El objetivo de este proyecto es, partiendo de una base de datos de tamaño moderado compuesta por canciones pertenecientes a la música clásica, estudiar las técnicas de análisis, formas de proceso y modelos de señal más apropiadas para ser capaz de reconocer una canción de la base de datos a partir de: un fragmento de la canción, parte de su melodía, un tarareo de la misma, un silbido de su melodía, o cantando (en caso de que la canción posea letra).

Palabras clave: reconocimiento de audio, *query by humming*, extracción de características, DTW, k-nn, música clásica

Agradecimientos

A mi familia. En especial a mi madre, por todo su apoyo y amor incondicional, y sus amplios conocimientos de música clásica.

A mis amigos, por toda su ayuda y paciencia para grabar tantísimas canciones.

A mi director, por todos sus consejos y tutela a lo largo del proyecto.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

La motivación para la realización de este proyecto viene tras todas aquellas veces en las que mi madre y mi tía han recurrido a mí para que les ayudase a recordar el título de una determinada canción.

En ocasiones teníamos acceso a un fragmento de la canción; otras veces nuestra única ayuda era tararear la melodía y, en muy raras ocasiones, contábamos con parte de la letra. Sin embargo, independientemente de la situación en la que nos encontráramos, el procedimiento a realizar era el mismo: usar un programa de reconocimiento de audio para tratar de obtener el título de la canción.

Este procedimiento no era siempre igual de satisfactorio. Algunas veces en apenas unos segundos teníamos el título; otras terminábamos desistiendo tras probar de todas las formas posibles. Gracias a esto, observé que las veces en las que teníamos acceso a un fragmento de la canción o conocíamos parte de la letra, encontrábamos el título con mayor facilidad que cuando debíamos tararear la melodía. Además, si se trataba de una canción con letra (independientemente de que la conociésemos o no) era más sencillo conocer el título que si no tenía letra. Por último, las canciones más "clásicas" (entendiendo por esto tanto las canciones consideradas dentro del género de la música clásica así como canciones de características similares) eran las más difíciles de localizar y las que más veces terminábamos desistiendo de conocer su título.

Todo esto ha despertado en mi las ganas de saber más sobre el reconocimiento de canciones (en especial de música clásica), y me ha llevado a la realización de este proyecto.

1.2. Objetivo

El objetivo del presente trabajo es realizar un pequeño estudio sobre diversas técnicas usadas en el reconocimiento de canciones. Las canciones a reconocer estarán contenidas en una base de datos de tamaño moderado (ver Capítulo 4) y o bien pertenecerán al género de la música clásica o serán canciones de características muy similares. A partir de ahora, se hará referencia a estas canciones como *canciones clásicas*.

Para cada una de la técnicas a estudiar se verá la forma de representación, análisis, proceso y clasificación de las mismas. Dichas técnicas son las siguientes:

- **Fragmento de la canción**: la canción a reconocer se obtendrá a partir de un fragmento de la misma. A fin de crear mayor diversidad en los archivos de audio, cada uno de estos fragmentos pertenecerá a un archivo de audio distinto.
- **Parte de la melodía**: la canción a reconocer se obtendrá a partir de un fragmento de la melodía de la misma. A fin de lograr una mayor diversidad melódica, cada uno de estos fragmentos pertenecerá a una versión distinta de la canción, entendiendo como tal: la misma canción tocada por diversos instrumentos (un único instrumento cada vez), versiones instrumentales de las canciones con letra, etc.
- **Tarareo**: la canción a reconocer se obtendrá a partir de un tarareo de la melodía de la misma.
- **Silbido**: la canción a reconocer se obtendrá a partir de un fragmento silbado de la melodía de la misma.
- **Canto**: la canción a reconocer se obtendrá a partir de un fragmento cantado de la misma. Esta técnica sólo podrá ser aplicada para aquellas canciones que posean letra.

Para la realización de las pruebas de clasificación se contará con unas muestras grabadas a partir de voluntarios. Dichas muestras tendrán una duración aproximada de 10 segundos y estarán en formato MP3.

1.3. Conceptos previos

En este apartado se recogen algunos de los conceptos necesarios a la hora de abordar el proyecto.

1.3.1. Muestra

Dentro del ámbito del reconocimiento de audio, se conoce como *muestra* al fragmento de audio del cual se quiere obtener la información. En este caso, una *muestra* será aquel fragmento musical (compuesto a partir de alguna de las técnicas expuestas en la Sección 1.2) del cual se quiere conocer el título de la canción a la que pertenece.

1.3.2. Reconocimiento de música

A grandes rasgos, el reconocimiento de música se compone principalmente de tres pasos:

- Construcción de la base de datos.
- Procesado de la muestra.
- Clasificación.

Así pues, el objetivo es extraer información de la muestra mediante el uso de tecnologías de procesado de señales de audio. Esta información se comparará con cada una de las canciones de la base de datos, calculando una medida de semejanza o disimilitud con cada canción. Este mismo proceso, pero en un contexto en el que las muestras están compuestas únicamente por tarareos, es observable en la Figura 1.1. [1]

1.3.3. *Pitch*

Musicalmente hablando, el concepto de *pitch* hace referencia a la escala musical, denotada por la nota y la octava a la que esta se encuentra. La forma de representación viene denotada por la nota musical y el valor numérico de la octava. Ejemplo: C4, C#4, D4 [2].

1.3.4. *Onset*

En términos musicales, el *onset* determina el instante de tiempo en el que comienza a sonar una determinada nota.

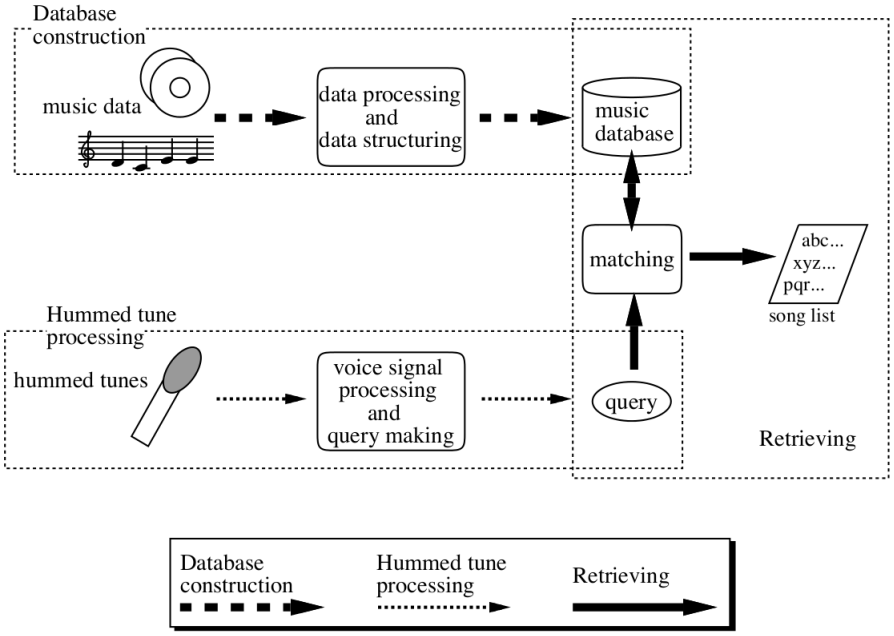

Figura 1.1: *Proceso general del reconocimiento de música mediante tarareos.*

1.3.5. Música polifónica

Se considera música polifónica a aquella pieza musical en la que dos o más notas pueden sonar simultáneamente, ya sea haciendo uso de diversos instrumentos (p. ej. la voz, la guitarra y el bajo) o un único instrumento capaz de tocar más de una nota a la vez (p. ej. el piano) [3].

1.3.6. Melodía

El concepto de *melodía* puede llegar a ser un tanto confuso. A fin de tener un marco claro en el que trabajar, se ha decidido adoptar la definición dada en [4]: *"la melodía es la secuencia de* pitches *(monofónica) única que un oyente puede reproducir si se le pide que silbe o tararee una pieza musical polifónica, y que un oyente reconocería como la 'esencia' de esa música cuando la escucha en comparación."*

1.4. Organización del documento

Este documento está organizado de la siguiente manera:

- El *Capítulo 2* ilustra brevemente el estado actual del arte.
- En el *Capítulo 3* se habla sobre cómo extraer la información de la canción.
- El *Capítulo 4* contiene una descripción de la base de datos.
- En el *Capítulo 5* se explica cómo se clasifican cada una de las muestras.
- Los experimentos realizados y resultados del estudio se recogen en el *Capítulo 6*.
- Por último, en el *Capítulo 7* se detallan las conclusiones del proyecto y se habla de las posibles mejoras a realizar en un futuro.

Capítulo 2

Estado del arte

Este capítulo pretende hacer un breve repaso al estado actual del arte de las formas de resolver el problema de la identificación de canciones.

2.1. *Query by humming*

Query by humming (consulta mediante tarareos) engloba las técnicas de clasificación mediante *tarareo*, *canto* y *silbido* (en menor medida). En este proyecto también se contempla la clasificación por *fragmento* y la clasificación por *melodía* (ver Sección 4.3). Sin embargo, estas técnicas no son tan comunes, con lo que no serán mencionadas en este capítulo.

2.2. Formato MIDI

Este formato es el más usado hoy día a la hora de construir la base de datos [5] y es el que mejores resultados está dando, puesto que la precisión a la hora de obtener la codificación de las canciones es mayor (no es necesario realizar un proceso de extracción de características ya que se dispone de ellas desde un principio).

El principal problema de este formato es que es necesario crear y revisar los archivos MIDI manualmente, lo que dificulta una futura incorporación de nuevas canciones a la base de datos.

2.3. Formato de audio

Este formato implica la construcción de la base de datos a partir de archivos de audio. El uso de este formato está menos extendido, ya que obliga a realizar un proceso de extracción de características, con las consecuentes pérdidas de precisión que esto conlleva.

Por lo general, los resultados obtenidos por este medio son menos satisfactorios que creando la base de datos a partir de archivos MIDI. Sin embargo, una de sus principales ventajas es que permite automatizar el proceso, facilitando la futura inclusión de nuevas canciones en la base de datos. En [6–9] se recogen algunos estudios con el uso de este formato.

2.4. Géneros musicales

Un amplio rango de géneros musicales es usado en los diversos estudios sobre *query by humming*. Sin embargo, la mayoría de estudios coinciden en que las canciones compuestas únicamente por música instrumental son más difíciles de evaluar y aún queda bastante por mejorar [3,10]. Esto es debido a que tener múltiples instrumentos emitiendo distintas notas simultáneamente dificulta tanto la *codificación de las canciones* (ya sea a partir de un proceso de *extracción de características*, como manualmente a partir de *archivos MIDI*), como la *creación de muestras* por parte de los usuarios, quienes deberán centrarse en el instrumento (o instrumentos, en la caso de que haya más de uno emitiendo las mismas notas a la vez) que crean más característicos o más sencillo de imitar[1]. Además, el rango de la voz humana difiere con el emitido por los instrumentos. Por tanto, es más fácil que se produzcan diferencias considerables al tratar de imitar un instrumento que al tratar de imitar a un cantante.

2.5. MIREX

MIREX [11, 12] es un proyecto comunitario para la evaluación formal de algoritmos y técnicas relacionadas con la *recuperación de información a partir de música* (MIR[2]). Esta evaluación se lleva a cabo una vez al año

[1] Durante la grabación de las muestras usadas en este proyecto, muchos de los usuarios tenían problema al tratar de imitar determinados instrumentos (como el arpa), por lo que o bien optaban por imitar un instrumento que consideraban menos característico, o bien por hacer breves pausas.

[2] Del inglés *Musical Information Retrieval*.

y está compuesta por distintas tareas, entre las que se encuentra *query by singing/humming*.

La tarea de *query by singing/humming* se realiza sobre los siguientes *corpus*:

- ***Roger Jang's corpus*** [13]: Compuesto por 4431 *queries* y 48 archivos MIDI de canciones populares chinas e inglesas. Todas las *queries* comienzan al inicio de la canción. Se dispone de un etiquetado manual de los *pitches*.
- **IOACAS *corpus*** [14]: Compuesto por 759 *queries* y 298 archivos MIDI monofónicos de canciones pop chinas. No se garantiza que las *queries* se correspondan con el inicio de las canción.

Esta tarea está compuesta a su vez por varias subtareas. A continuación se muestran varias de estas subtareas y los resultados obtenidos para ellas en MIREX 2014 [15]:

- ***Hidden Jang***: Tarea realizada sobre *Roger Jang's corpus* para un conjunto de *queries* oculto. La tasa de acierto obtenida por los distintos participantes se encuentra entre 0.47 y 0.93 MRR[3].
- ***Jang***: Tarea realizada sobre *Roger Jang's corpus*. La tasa de acierto obtenida por los distintos participantes se encuentra entre 0.43 y 0.96 MRR.
- ***IOACAS***: Tarea realizada sobre el *corpus* de IOACAS. La tasa de acierto obtenida por los distintos participantes se encuentra entre 0 y 0.80 MRR.

[3]Del inglés *Mean Reciprocal Rank*. Es una medida estadística para la evaluación de un proceso que produce una lista de posibles respuestas ante determinadas entradas, ordenada en función de la probabilidad de su exactitud. El *reciprocal rank* de una determinada entrada es la inversa multiplicativa del rango de la primera respuesta correcta. El *mean reciprocal rank* es la media de los *reciprocal ranks* de los resultados de todas las entradas [16].

Capítulo 3

Extracción de características

3.1. Objetivo

Este proceso tiene como objetivo extraer la información de las canciones de la base de datos y las muestras a clasificar, así como representar dicha información en un formato adecuado para poder realizar la clasificación.

3.2. Representación

A continuación, se verán algunas de las codificaciones más usadas para la representación de las características.

3.2.1. Codificación MIDI

Esta codificación tiene su origen en el formato MIDI (*Interfaz Digital de Instrumentos Musicales*[1]) y es la utilizada con mayor frecuencia. En [2] se sugiere que esté compuesta por los distintos *pitches* contenidos en la muestra, representados por su *número de tono*, *tiempo de inicio*, *duración* y *ruido relativo*. Sin embargo, tal y como se verá en el Capítulo 5, será suficiente con que se representen usando el *tiempo de inicio* y el *número de tono*. En la Tabla 3.1 se puede ver más detalladamente el valor de cada nota.

Esta codificación será la usada por defecto en el desarrollo del proyecto. En la Figura 3.1 se puede observar un ejemplo de esta codificación.

[1]Del inglés *Musical Instrument Digital Interface*.

Tabla 3.1: *Número de tono, nombre del* pitch *y frecuencia* [1].

tone number	pitch name	frequenzy (Hz)	tone number	pitch name	frequenzy (Hz)	tone number	pitch name	frequenzy (Hz)
0	C–1	8.175799	43	G2	97.998859	86	D6	1174.659072
1	C#–1	8.661957	44	G#2	103.826174	87	D#6	1244.507935
2	D–1	9.177024	45	A2	110.000000	88	E6	1318.510228
3	D#–1	9.722718	46	A#2	116.540940	89	F6	1396.912926
4	E–1	10.300861	47	B2	123.470825	90	F#6	1479.977691
5	F–1	10.913382	48	C3	130.812783	91	G6	1567.981744
6	F#–1	11.562326	49	C#3	138.591315	92	G#6	1661.218790
7	G–1	12.249857	50	D3	146.832384	93	A6	1760.000000
8	G#–1	12.978272	51	D#3	155.563492	94	A#6	1864.655046
9	A–1	13.750000	52	E3	164.813778	95	B6	1975.533205
10	A#–1	14.567618	53	F3	174.614116	96	C7	2093.004522
11	B–1	15.433853	54	F#3	184.997211	97	C#7	2217.461048
12	C0	16.351598	55	G3	195.997718	98	D7	2349.318143
13	C#0	17.323914	56	G#3	207.652349	99	D#7	2489.015870
14	D0	18.354048	57	A3	220.000000	100	E7	2637.020455
15	D#0	19.445436	58	A#3	233.081881	101	F7	2793.825851
16	E0	20.601722	59	B3	246.941651	102	F#7	2959.955382
17	F0	21.826764	60	C4	261.625565	103	G7	3135.963488
18	F#0	23.124651	61	C#4	277.182631	104	G#7	3322.437581
19	G0	24.499715	62	D4	293.664768	105	A7	3520.000000
20	G#0	25.956544	63	D#4	311.126984	106	A#7	3729.310092
21	A0	27.500000	64	E4	329.627557	107	B7	3951.066410
22	A#0	29.135235	65	F4	349.228231	108	C8	4186.009045
23	B0	30.867706	66	F#4	369.994423	109	C#8	4434.922096
24	C1	32.703196	67	G4	391.995436	110	D8	4698.636287
25	C#1	34.647829	68	G#4	415.304698	111	D#8	4978.031740
26	D1	36.708096	69	A4	440.000000	112	E8	5274.040911
27	D#1	38.890873	70	A#4	466.163762	113	F8	5587.651703
28	E1	41.203445	71	B4	493.883301	114	F#8	5919.910763
29	F1	43.653529	72	C5	523.251131	115	G8	6271.926976
30	F#1	46.249303	73	C#5	554.365262	116	G#8	6644.875161
31	G1	48.999429	74	D5	587.329536	117	A8	7040.000000
32	G#1	51.913087	75	D#5	622.253967	118	A#8	7458.620184
33	A1	55.000000	76	E5	659.255114	119	B8	7902.132820
34	A#1	58.270470	77	F5	698.456463	120	C9	8372.018090
35	B1	61.735413	78	F#5	739.988845	121	C#9	8869.844191
36	C2	65.406391	79	G5	783.990872	122	D9	9397.272573
37	C#2	69.295658	80	G#5	830.609395	123	D#9	9956.063479
38	D2	73.416192	81	A5	880.000000	124	E9	10548.081821
39	D#2	77.781746	82	A#5	932.327523	125	F9	11175.303406
40	E2	82.406889	83	B5	987.766603	126	F#9	11839.821527
41	F2	87.307058	84	C6	1046.502261	127	G9	12543.853951
42	F#2	92.498606	85	C#6	1108.730524			

3.2.2. Codificación UDS

Esta codificación es también conocida como *códigos de Parson* y fue introducida por primera vez en [17]. Partiendo de la codificación vista en el apartado 3.2.1, por cada uno de los *pitches* que la componen se compara su *número de tono* con el del *pitch* anterior, denotando con una U si el tono ha aumentado, una D si ha disminuido, y una S si se mantiene igual. De esta forma, esta codificación estará compuesta por una secuencia de Us, Ds

0 40
1 43
2 50
3 50
4 55
5 45

Figura 3.1: *Ejemplo de codificación MIDI.*

y *S*s. En la Figura 3.2 se puede ver la codificación UDS para el ejemplo de la Figura 3.1.

UUSUD

Figura 3.2: *Ejemplo de codificación UDS.*

3.3. Extracción de los *pitches*

3.3.1. `Aubio`

`Aubio` [18, 19] es una herramienta diseñada para la extracción de anotaciones en señales de audio. Dentro de sus característica se incluye la *segmentación de un archivo de audio*, *detección de* pitches, *detección de ritmos*, y *producción de secuencias MIDI a partir de audio en vivo.*

De entre todas estas características cabe resaltar la *detección de* pitches y la *detección de* onsets. Estas detecciones puede ser llevadas a cabo mediante diversos algoritmos, siendo los más destacados:

- ***Yin*** [20]: Algoritmo para la estimación de la frecuencia fundamental en el habla y en los sonidos musicales. Está basado en el método de autocorrelación y es apropiado para voces y música con altos valores de *pitch.*
- ***Yinfft*** [19]: Este algoritmo deriva del algoritmo *Yin* y utiliza una transformada de Fourier para computar una función diferencial cuadrática ajustada, permitiendo el cálculo del peso espectral y simplificando la selección del periodo.
- ***Energy***: Algoritmo para la detección de *onsets* basado en la distancia energética del espectro de la señal de entrada.

- ***Specflux*** [21]: Algoritmo para la detección de *onsets* basado en el cambio de magnitud de la frecuencia.

3.3.2. Proceso de extracción

El proceso de extracción consiste en, por cada una de las canciones de la base de datos y por cada una de las muestras a clasificar (ver Sección 4.3), invocar a `aubionotes` (aplicación de la línea de órdenes incluida en la suite `aubio`, vista en el apartado 3.3.1).

`Aubionotes` toma como argumento, entre otros parámetros:

- **Fichero de audio a analizar**: Canción de la base de datos o muestra a clasificar.
- **Unidades del *pitch***: Unidades con las que representar el valor del *pitch*. Este parámetro se ha dejado a su valor por defecto para obtener una representación en función del *número de tono* (tal y como se vio en el apartado 3.2.1).
- **Algoritmo de detección de *pitches***: Algoritmo usado en la detección de *pitches* (ver apartado 3.3.1). Se ha decidido dejar el algoritmo por defecto (*Yinfft*) ya que en [19] se demuestra que es un poco más eficiente para este tipo de tarea.
- **Algoritmo de detección de *onsets***: Algoritmo usado en la detección de onsets (ver apartado 3.3.1). Tras unas pruebas iniciales, se ha decidido utilizar *Energy* para las canciones de la base de datos y para las muestras de tipo *fragmento*, y *Specflux* para el resto de muestras.

En la Figura 3.3 se puede ver un ejemplo de ejecución de `aubionotes`.

```
aubionotes -i query.mp3 -v -O specflux
```

Figura 3.3: *Ejemplo de ejecución de* ***`aubionotes`****.*

Capítulo 4

Base de datos

4.1. Introducción

La base de datos está compuesta por 50 canciones en formato MP3, de las cuales tan sólo 20 poseen letra. En la medida de lo posible, se ha tratado de obtener versiones libres de derechos de autor, extrayendo de *Youtube* [22] todas aquellas canciones que no han podido ser encontradas en su versión libre. Las principales fuentes para la obtención de dichas versiones han sido:

- **Classical Cat**: Biblioteca compuesta de índices a representaciones de piezas de música clásica de libre distribución. [23]
- **Archivos Europeos**: Biblioteca de contenido digital europeo de libre acceso a *investigadores*, *historiadores* y *público general*. [24]
- **Biblioteca de audio de Youtube**: Biblioteca compuesta por pistas musicales libres de derechos de autor. [25]

Para más información sobre las canciones que componen la base de datos, consultar los Apéndices A y B.

4.2. Contenido de la base de datos

Tal y como se vio en el apartado 3.2.1, las características se representan mediante el *tiempo de inicio* y el *número de tono* de cada uno de los *pitches* que componen el archivo de audio. Por tanto, tanto las canciones de la base de datos como las muestras a clasificar (ver Sección 4.3) estarán compuestos por un archivo de texto que contendrá una línea por cada uno de los *pitches*

que posea, estando a su vez cada línea compuesta por las características anteriormente citadas.

En la Figura 4.1 puede verse un ejemplo del contenido de un fichero de la base de datos.

```
3.651338 46
4.864581 58
4.957460 46
5.224490 58
5.230295 46
5.241905 58
...
```

Figura 4.1: *Contenido de un fichero de la base de datos.*

4.3. Muestras a clasificar

En este apartado se verá en qué consisten las muestras a clasificar y cómo han sido obtenidas.

4.3.1. Tarareo, canto y silbido

Estas muestras consisten en grabaciones, de unos diez segundos de duración, de tarareos, cantos y silbidos de las melodías de las canciones de la base de datos, obtenidas gracias a la colaboración de voluntarios. Estas grabaciones están en formato MP3 y han sido realizadas mediante una grabadora de voz *Sony ICDPX333.CE7*.

La distribución de estas muestras es la siguiente:

- **Tarareo**: Se han obtenido un total de 300 muestras de tarareo. Cada una de las canciones de la base de datos se ve representada en varias de estas muestras.
- **Canto**: Se han obtenido un total de 150 muestras de canto. Estas muestras recogen únicamente las canciones que poseen letra. Cada una de estas canciones se ve representada en varias de estas muestras.
- **Silbido**: Se han obtenido un total de 150 muestras de silbido. Cada una de las canciones de la base de datos se ve representada en una o varias de estas muestras.

4.3.2. Fragmento de la canción

Estas muestras consisten en fragmentos aleatorios, de unos diez segundos de duración, de las canciones de la bases de datos, y han sido obtenidas de *Youtube* [22] asegurándose de que cada muestra pertenece a un vídeo distinto (a fin de crear mayor diversidad en los archivos de audio).

Por cada canción de la base de datos se han obtenido seis muestras distintas, sumando un total de 300 muestras de este tipo.

4.3.3. Fragmento de la melodía

Estas muestras consisten en fragmentos aleatorios, de unos diez segundos de duración, de la melodía de las canciones de la bases de datos, y han sido obtenidas de *Youtube* [22]. A fin de lograr una mayor diversidad melódica se ha procurado buscar diversas versiones de cada canción, entendiendo como tal: la misma canción tocada por diversos instrumentos, versiones instrumentales de las canciones con letra, etc.

Por cada canción de la base de datos se han obtenido seis muestras distintas, sumando un total de 300 muestras de este tipo.

Capítulo 5

Clasificación

5.1. Objetivo

El objetivo del proceso de clasificación es ser capaz de conocer a qué canción pertenece cada una de las muestras a clasificar (ver Sección 4.3). Para ello, se cogerán cada una de estas muestras una a una y se compararán con cada una de las canciones de la base de datos (ver Sección 5.2). Esta comparación obtendrá como resultado un valor correspondiente a cuan diferente son muestra y canción, de tal manera que una vez computadas todas las comparaciones pueda concluirse a qué canción pertenece dicha muestra (ver Sección 5.4).

5.2. Cálculo de distancias

5.2.1. DTW

DTW (*Alineamiento Temporal Dinámico*[1]) es un algoritmo basado en programación dinámica para el cálculo de distancias entre dos secuencias temporales variables en el tiempo o espacio. Actualmente es el algoritmo más usado para este tipo de tarea [26–30].

DTW funciona de acuerdo a las Ecuaciones (5.1), (5.2), (5.3), (5.4) y (5.5).

$$d(i,j) = |t(i) - r(j)| \tag{5.1}$$

[1]Del inglés *Dynamic Time Warping*

$$D(1,1) = |t(1) - r(1)| \tag{5.2}$$

$$D(i,1) = \infty, \quad i = 2, \ldots, m \tag{5.3}$$

$$D(1,j) = \infty, \quad j = 2, \ldots, n \tag{5.4}$$

$$D(i,j) = d(i,j) + \text{mín} \begin{cases} D(i-1,j) \\ D(i-1,j-1) \\ D(i,j-1) \end{cases} \tag{5.5}$$

La Ecuación (5.1) hace referencia a la distancia entre dos *pitches*, en dónde *t(i)* es un *pitch* de la muestra (siendo $i = 1, \ldots, m$) y *r(j)* es un *pitch* de la canción de referencia (siendo $j = 1, \ldots, n$).

En la Ecuación (5.2) se lleva a cabo la inicialización del algoritmo. La Ecuación (5.3) asegura que el camino óptimo que toma DTW comience al inicio de la muestra.

De forma similar, la Ecuación (5.4) asegura que el camino óptimo que toma DTW comience al inicio de la canción de referencia. Sin embargo, la muestra no tiene por qué corresponderse con el inicio de la canción, por lo que esta ecuación deberá sustituirse por la Ecuación (5.6), la cual hace que el camino óptimo pueda comenzar en cualquier instante de la canción.

$$D(1,j) = |t(1) - r(j)|, \quad j = 2, \ldots, n \tag{5.6}$$

Respecto a la Ecuación (5.5), en [26] se demuestra que la Ecuación (5.7) es un poco más eficiente para este tipo de tarea, por lo que se ha decidido utilizar la Ecuación (5.7) en lugar de la Ecuación (5.5).

$$D(i,j) = d(i,j) + \text{mín} \begin{cases} D(i-2,j-1) \\ D(i-1,j-1) \\ D(i-1,j-2) \end{cases} \tag{5.7}$$

De este modo, el algoritmo DTW a utilizar es el definido por las Ecuaciones (5.1), (5.2), (5.3), (5.6) y (5.7)

5.2.2. Distancia de edición

Conocido también como *distancia de Levenshtein*, este algoritmo calcula la diferencia entre dos secuencias de cadenas, estando esta diferencia com-

puesta, *a groso modo*, por la cantidad de operaciones de *inserción*, *sustitución* ó *borrado* necesarias para pasar de la cadena origen a la cadena destino. Al igual que DTW (ver apartado 5.2.1), es un algoritmo basado en programación dinámica.

Existen múltiples variantes de este algoritmo, siendo una de la más usuales la definida por las Ecuaciones (5.8), (5.9), (5.10) y (5.11).

$$d(0,0) = 0 \tag{5.8}$$

$$d(i,0) = d(i-1,0) + 1, \quad i = 1,\ldots,m \tag{5.9}$$

$$d(0,j) = d(0,j-1) + 1, \quad j = 1,\ldots,n \tag{5.10}$$

$$d(i,j) = \text{mín} \begin{cases} d(i-1,j) + 1 & (borrado) \\ d(i-1,j-1) + 1 & (sustitución) \\ d(i-1,j-1) & (coincidencia) \\ d(i,j-1) + 1 & (inserción) \end{cases} \tag{5.11}$$

La Ecuación (5.8) realiza la inicialización del algoritmo. La Ecuación (5.9) asegura que se comience a comparar al inicio de la muestra (siendo m la talla de la misma).

De forma similar, la Ecuación (5.10) asegura que la comparación comience al inicio de la canción de referencia (siendo n la talla de la misma). Sin embargo, la muestra no tiene por qué corresponderse con el inicio de la canción, por lo que esta ecuación deberá sustituirse por la Ecuación (5.12), la cual hace que el coste de *insertar* al inicio sea 0, dando así libertad a la muestra de comenzar en cualquier punto de la canción de referencia.

$$d(0,j) = 0, \quad j = 1,\ldots,n \tag{5.12}$$

Además, será necesario permitir que la muestra pueda terminar en cualquier punto de la canción de referencia y no necesariamente al final. Para ello, será necesario sustituir la Ecuación (5.11) por la Ecuación (5.13), la cual reduce el coste de *borrar* en la última posición a 0.

$$d(i,j) = \text{mín} \begin{cases} d(i,j-1) + 1, & \text{si } i < m & (borrado) \\ d(i,j-1), & \text{si } i = m & (borrado\ al\ final) \\ d(i-1,j-1) + 1 & & (sustitución) \\ d(i-1,j-1) & & (coincidencia) \\ d(i-1,j) + 1 & & (inserción) \end{cases} \tag{5.13}$$

Por tanto, el algoritmo de distancias de edición a utilizar es el definido por las Ecuaciones (5.8), (5.9), (5.12) y (5.13).

En la Figura 5.1 se muestra un ejemplo del alineamiento realizado por este algoritmo.

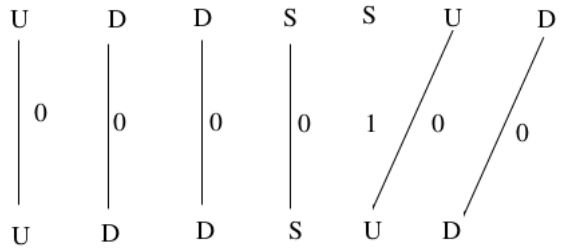

Figura 5.1: *Alineamiento óptimo entre cadenas* [31].

Las Ecuaciones (5.12) y (5.13) han sido obtenidas de [32].

5.3. Suavizado de los tonos

Cada usuario tiene un tono distinto a la hora de cantar. Por tanto, antes de aplicar el algoritmo DTW (ver apartado 5.2.1), será necesario realizar un pequeño suavizado de los datos, el cual viene dado por las Ecuaciones (5.14), (5.15), (5.16), (5.17) y (5.18) [28].

$$\begin{cases} t(i) = t(i) - \text{media}(t), & i = 1, \ldots, m \\ r(j) = r(j) - \text{media}(r), & j = 1, \ldots, n \end{cases} \tag{5.14}$$

$$\begin{cases} span = 4 \\ center = 0 \end{cases} \tag{5.15}$$

$$distancia = \text{mín} \begin{cases} s_{-1} &= DTW(r, t - center - span) \\ s_0 &= DTW(r, t - center) \\ s_1 &= DTW(r, t - center + span) \end{cases} \tag{5.16}$$

$$center = \begin{cases} center - span, & \text{si mín}(s_{-1}, s_0, s_1) = s_{-1} \\ center + span, & \text{si mín}(s_{-1}, s_0, s_1) = s_1 \end{cases} \tag{5.17}$$

$$span = span/2, \quad \text{si } span > 2 \tag{5.18}$$

La Ecuación (5.14) resta a cada uno de los *pitches* de la muestra y la canción de referencia, la media del valor de todos sus *pitches*.

La Ecuación (5.15) define los valores de *centro* y *alcance* necesarios para aplicar el suavizado.

La Ecuación (5.16) calcula la distancia entre la muestra y la canción de referencia, haciendo uso de el algoritmo DTW y los valores anteriormente definidos.

Por último, las Ecuaciones (5.17) y (5.18) actualizan los valores de *centro* y *alcance* respectivamente. Esta actualización es necesaria porque una vez llevada a cabo se deberá volver a calcular la Ecuación (5.16), siendo la distancia resultante de este último cálculo el resultado total de la distancia entre la muestra y la canción de referencia.

La Figura 5.2 ilustra el proceso de búsqueda llevado a cabo por este suavizado.

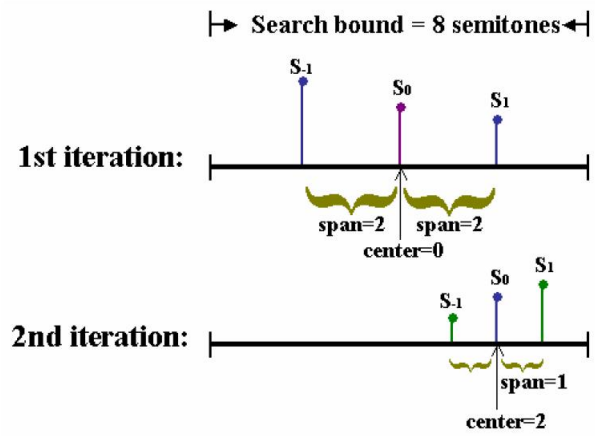

Figura 5.2: *Suavizado de los tonos* [28].

5.4. Clasificación

5.4.1. Distancia más cercana

Este método de clasificación consiste en, una vez calculadas las distancias entre una determinada muestra y cada una de las canciones de la base de datos (ver Sección 5.2), seleccionar la distancia de menor valor. De este modo, dicha muestra pertenecerá a aquella canción cuya distancia sea la anteriormente citada.

A fin de otorgar un pequeño margen de error, se plantea una variante de este método de forma que una determinada muestra pueda clasificarse como cualquiera de las n primeras canciones más cercanas en lugar de únicamente la canción más cercana. De esta forma, en lugar de clasificar la muestra como perteneciente a la canción correspondiente a la distancia de menor valor, esta podrá ser clasificada como cualquiera de las n canciones correspondientes a las n distancias de menor valor, siendo n típicamente 1, 3 ó 5.

5.4.2. K-NN

La clasificación por el k vecino más cercano (k-NN[2]) consiste en, dada una secuencia de distancias y la canción a la que pertenece cada distancia (correspondiéndose todas a ellas a distancias obtenidas a partir de una misma muestra), se seleccionan los k elementos de menor distancia (siendo k un valor a determinar). Una vez seleccionados, se agrupan los conjuntos por canciones, de forma que la muestra pertenecerá a aquella canción cuyo grupo tenga el mayor número de elementos.

5.5. Validación cruzada

La *validación cruzada* es un método estadístico para la evaluación de resultados en un conjunto de datos independiente. Esta técnica puede aplicarse sobre el conjunto de muestras a clasificar, de manera que sean las propias muestras las usadas para llevar a cabo la clasificación.

Para ello, es necesario dividir el conjunto de muestras en dos subconjuntos conocidos como *training* y *test*. El subconjunto de *training* será el usado para la clasificación de la muestras (prototipos), y el de *test* será el que contenga las muestras a clasificar.

Una vez hecha esta división, el proceso de clasificación se realizará con los algoritmos vistos en las Secciones 5.2 y 5.4, teniendo en cuenta que las muestras a clasificar serán únicamente las pertenecientes al conjunto de *test*, y que en lugar de usar las canciones de la base de datos se usarán las muestras contenidas en el conjunto de *training* tanto para el cálculo de distancias como para la posterior clasificación (teniendo en cuenta que las muestras de *training* son representaciones de las canciones de la base de datos, con lo que una muestra de *test* no se clasificará como una de las muestras del conjunto de *training* sino como la canción de la base de datos representada por la muestra de *training* resultante del proceso de clasificación).

[2]Del inglés *K-Nearest Neighbour*.

A fin de que los resultados obtenidos tengan significancia estadística, este proceso se deberá realizar varias veces con nuevos conjuntos de *training* y *test*, de forma que todas las muestras pasen a formar parte de ambos conjuntos en algún momento. Otra manera de hacer esto es dividir el conjunto original en p subconjuntos e ir haciendo combinaciones tomando cada vez uno de los p subconjuntos como el conjunto de *test* y el resto como conjunto de *training*. Este último método es menos exhaustivo que el primero, pero continua siendo estadísticamente significante y su práctica es bastante habitual, por lo que se ha decidido utilizarlo.

Una vez computados los resultados para cada una de las combinaciones anteriormente mencionadas, el resultado total será la media de estos resultados.

Capítulo 6

Experimentos

6.1. Introducción

En este apartado se detallan todos los experimentos realizados a lo largo del desarrollo del proyecto a fin de obtener los mejores resultados posibles en la clasificación de cada uno de los distintos tipos de muestras (ver Sección 4.3).

6.2. DTW - Distancia más cercana

Este experimento tiene por objetivo usar el algoritmo *DTW* (ver apartado 5.2.1) para el cálculo de distancias, y la técnica de *distancia más cercana* (ver apartado 5.4.1) para clasificar.

Para ello, por cada una de las muestras a clasificar se aplicará *DTW* sobre la muestra actual y cada una de las canciones de la base de datos, obteniendo así la distancia entre cada muestra y las canciones de la base de datos. Para las muestras del tipo *tarareo*, *canto* y *silbido* será necesario aplicar el proceso de suavizado de tonos visto en la Sección 5.3 antes de aplicar el algoritmo *DTW*, siendo el resultado obtenido tras aplicar este suavizado sobre cada muestra el que se usará en el cálculo de *DTW*.

Una vez computadas las distancias, se aplicará *distancia más cercana* para clasificar la muestra.

En la Tabla 6.1 se muestra la tasa de acierto obtenida (%) como resultado de este experimento, agrupada en función de los distintos tipos de muestras y el valor de n usado a la hora de realizar la clasificación.

Tabla 6.1: *Resultados DTW - Distancia más cercana.*

n	Tarareo	Canto	Silbido	Fragmento	Melodía
1	2.3	4.6	4	50	6.3
3	8.6	11.3	14	61	13.6
5	11	17.3	18	67	18.3

6.3. Distancia de edición - Distancia más cercana

Este experimento es similar al descrito en la Sección 6.2, pero usando en esta ocasión el algoritmo de *distancia de edición* (ver apartado 5.2.2). Dado que este algoritmo se basa en el cálculo de distancia entre cadenas, en lugar de utilizar la *codificación MIDI* (ver apartado 3.2.1) se usará la *codificación UDS* (ver apartado 3.2.2). Además, puesto que únicamente se tiene en cuenta la diferencia de tono entre *pitches* y no su valor exacto, no será necesario aplicar el proceso de suavizado.

Así pues, el cálculo de la distancia consistirá en, para cada una de las muestras a clasificar, aplicar el algoritmo de *distancia de edición* sobre la muestra actual y cada una de las canciones de la base de datos, obteniendo así la distancia entre muestra y canción. Por último, se aplicará *distancia más cercana* para clasificar la muestra.

En la Tabla 6.2 se muestra la tasa de acierto obtenida (%) como resultado de este experimento, agrupada en función de los distintos tipos de muestras y el valor de n usado a la hora de realizar la clasificación.

Tabla 6.2: *Distancia de edición - Distancia más cercana.*

n	Tarareo	Canto	Silbido	Fragmento	Melodía
1	3.3	2.6	2	13	3
3	6.3	5.3	4.6	22.6	8
5	11.3	10.6	11.3	30	11.6

6.4. DTW - K-NN

Este experimento es similar al descrito en la Sección 6.2, pero combinado con la técnica de clasificación por el *k vecino más cercano* (ver apartado 5.4.2).

Así pues, se aplicará *DTW* sobre cada una de las muestras a clasificar y las canciones de la base de datos. Sin embargo, en lugar de que este devuelva la distancia mínima entre muestra y canción, se realizará una ligera modificación para que devuelva todas las distancias computadas[1]. Por tanto, el resultado de aplicar este algoritmo será, por cada muestra a clasificar, un conjunto de distancias y a la canción a la que pertenece cada distancia.

Tras esto, por cada uno de los conjuntos obtenidos, se ordenará el mismo en función de la distancia y se aplicará *K-NN*. De este modo, se tomarán las k primeras muestras del conjunto y se agruparán en función de la canción a la que pertenezca cada muestra. Una vez hecho esto, se ordenarán los conjuntos de mayor a menor (en función de su tamaño) y se clasificará a la muestra como una de las canciones a las que pertenecen los n primeros conjuntos.

En la Tabla 6.3 se muestra la tasa de acierto obtenida (%) como resultado de este experimento, agrupada en función de los distintos tipos de muestras y el valor de n usado a la hora de realizar la clasificación. Para todos los casos, el valor de k usado ha sido *110*[2].

Tabla 6.3: *DTW - K-NN.*

n	Tarareo	Canto	Silbido	Fragmento	Melodía
1	2.3	0	2.6	46.6	8
3	6.3	4.6	7.3	61	16
5	9	8	8.6	69	20.3

6.5. DTW - K-NN - Validación cruzada

Este experimento es similar al descrito en la Sección 6.4, pero usando además la técnica de *validación cruzada* (ver Sección 5.5).

Para realizar la *validación cruzada*, se han dividido las muestras a clasificar en cuatro conjuntos (generados aleatoriamente) de 300 muestras cada uno. Estos conjuntos serán combinados de forma que cada vez uno de ellos sea usado como conjunto de *test* y los otros tres restantes formen el conjunto de *training*.

[1]Tal y como se vio en el apartado 5.2.1, *DTW* contempla la posibilidad de que la muestra comience en cualquier instante de la canción. Para ello, calcula la distancia entre la muestra y cada uno de los posibles instantes en las que ésta puede comenzar, y devuelve el mínimo valor. Este último cálculo es el que se eliminará, devolviendo así todas las distancias calculadas.

[2]Valor resultante de una serie de pruebas previas.

Una vez obtenidos los conjuntos de *test* y *training*, se aplicará *DTW* sobre cada una de las muestras del conjunto de *test* y las muestras del conjunto de *training*. Nuevamente, en lugar de que éste devuelva la distancia mínima entre muestras, se realizará una ligera modificación para que devuelva todas las distancias computadas. Por tanto, el resultado de aplicar este algoritmo será, por cada muestra del conjunto de *test*, un conjunto de distancias y la canción a la que pertenece cada distancia (siendo esta la canción de la base de datos a la que pertenece la muestra del conjunto de *training* que ha devuelto esta distancia).

Tras esto, por cada uno de los conjuntos obtenidos, se ordenará el mismo en función de la distancia y se aplicará *K-NN*. De este modo, se tomarán las k primeras muestras del conjunto y se agruparán en función de la canción a la que pertenezca cada muestra. Una vez hecho esto, se ordenarán los conjuntos de mayor a menor (en función de su tamaño) y se clasificará a la muestra como una de las canciones a las que pertenecen los n primeros conjuntos.

Este proceso se repetirá para cada una de las posibles combinaciones de conjuntos de *test* y *training* anteriormente mencionados, siendo el resultado del experimento la media de los resultados obtenidos para cada una de estas combinaciones.

En la Tabla 6.4 se muestra la tasa de acierto obtenida (%) como resultado de este experimento, agrupada en función de los distintos tipos de muestras y el valor de n usado a la hora de realizar la clasificación. Para todos los casos, el valor de k usado ha sido *100*[3].

Tabla 6.4: *DTW - K-NN Validación cruzada.*

n	Tarareo	Canto	Silbido	Fragmento	Melodía
1	11.3	18.6	2.6	18.6	5
3	23.3	40.6	7.3	29.3	14.6
5	30	52.6	10.6	36.6	22.6

6.6. DTW - K-NN - Validación cruzada II

Este experimento es similar al descrito en la Sección 6.5, pero separando las muestras a clasificar por conjuntos antes de aplicar la técnica de *validación cruzada*.

[3]Valor resultante de una serie de pruebas previas.

En esta ocasión, para realizar la *validación cruzada*, las muestras a clasificar han sido divididas en función de su tipo. Estos han sido divididos a su vez en varios subconjuntos (generados aleatoriamente) de la siguiente manera:

- **Tarareo**: 6 subconjuntos de 50 muestras cada uno.
- **Canto**: 3 subconjuntos de 50 muestras cada uno.
- **Silbido**: 3 subconjuntos de 50 muestras cada uno.
- **Fragmento**: 6 subconjuntos de 50 muestras cada uno.
- **Melodía**: 6 subconjuntos de 50 muestras cada uno.

Estos subconjuntos serán combinados de forma que cada vez uno de ellos sea usado como conjunto de *test* y los subconjuntos restantes formen el conjunto de *training*.

Una vez obtenidos los conjuntos de *test* y *training*, se aplicará DTW sobre cada una de las muestras del conjunto de *test* y las muestras del conjunto de *training*. Nuevamente, en lugar de que éste devuelva la distancia mínima entre muestras, se realizará una ligera modificación para que devuelva todas las distancias computadas. Por tanto, el resultado de aplicar este algoritmo será, por cada muestra del conjunto de *test*, un conjunto de distancias y la canción a la que pertenece cada distancia (siendo esta la canción de la base de datos a la que pertenece la muestra del conjunto de *training* que ha devuelto esta distancia).

Tras esto, por cada uno de los conjuntos obtenidos, se ordenará el mismo en función de la distancia y se aplicará *K-NN*. De este modo, se tomarán las k primeras muestras del conjunto y se agruparán en función de la canción a la que pertenezca cada muestra. Una vez hecho esto, se ordenarán los conjuntos de mayor a menor (en función de su tamaño) y se clasificará a la muestra como una de las canciones a las que pertenecen los n primeros conjuntos.

Este proceso se repetirá para cada una de las posibles combinaciones de conjuntos de *test* y *training* anteriormente mencionados, y cada uno de los tipos de muestras existente. De este modo, el resultado del experimento será la media de los resultados obtenidos para cada una de estas combinaciones por cada uno de los tipos de muestras a clasificar.

En la Tabla 6.5 se muestra la tasa de acierto obtenida (%) como resultado de este experimento, agrupada en función de los distintos tipos de muestras y el valor de n usado a la hora de realizar la clasificación. El valor de k usado ha

sido *100*[4] para todos los casos excepto para las muestras de tipo *fragmento*, en cuyo caso se ha usado un valor de $k = 150$[5].

Tabla 6.5: *DTW - K-NN - Validación cruzada II.*

n	Tarareo	Canto	Silbido	Fragmento	Melodía
1	12	21.3	3.3	17.3	6.6
3	20.3	40	9.3	35.67	13.6
5	28.3	54	14	46	21.3

6.7. Resumen de los resultados

6.7.1. Clasificación a partir de la base de datos

En la Tabla 6.6 se muestra la mejor tasa de acierto obtenida (%) para cada uno de los distintos tipos de muestra en los experimentos vistos en las Secciones 6.2 a 6.4, agrupada en función de los mismos y el valor de n usado a la hora de realizar la clasificación.

Tabla 6.6: *Clasificación a partir de la base de datos.*

n	Tarareo	Canto	Silbido	Fragmento	Melodía
1	3.3	4.6	4	50	8
3	6.3	11.3	14	61	16
5	11.3	17.3	18	67	20.3

Para *canto*, *silbido* y *fragmento* los mejores resultados han sido obtenidos en la Sección 6.2 (DTW - distancia más cercana), mientras *tarareo* ha obtenido mejores resultados en la Sección 6.3 (distancia de edición - distancia más cercana), y *melodía* lo ha hecho en la Sección 6.4 (DTW - K-NN).

6.7.2. Clasificación a partir de las propias muestras

En la Tabla 6.7 se muestra la mejor tasa de acierto obtenida (%) para cada uno de los distintos tipos de muestra en los experimentos vistos en las secciones 6.5 y 6.6, agrupada en función de los mismos y el valor de n usado a la hora de realizar la clasificación.

[4]Valor resultante de una serie de pruebas previas.
[5]Valor resultante de una serie de pruebas previas.

Tabla 6.7: *Clasificación a partir de las propias muestras.*

n	Tarareo	Canto	Silbido	Fragmento	Melodía
1	11.3	21.3	3.3	18.6	5
3	23.3	40	9.3	29.3	14.6
5	30	54	14	36.6	22.6

Para *fragmento* y *melodía* los mejores resultados han sido obtenidos en la Sección 6.5 (validación cruzada general), mientras que para *tarareo*, *canto* y *silbido* lo han sido en la Sección 6.6 (validación cruzada por modalidad).

Capítulo 7

Conclusiones

7.1. Conclusiones

En este proyecto se ha llevado a cabo una primera aproximación al reconocimiento de canciones a través de varias técnicas (ver Sección 1.2), haciendo uso de distintos algoritmos de extracción de características (ver Capítulo 3) y de clasificación (ver Capítulo 5).

Los resultados obtenidos no han sido todo lo bueno que se deseaba. Sin embargo, se debe tener en cuenta que el género musical elegido se encuentra dentro de las limitaciones del estado actual del arte. Se ha tratado de buscar en la literatura algún estudio de características similares a fin de tener una idea más contextualizada de la calidad de los resultados, mas no se ha podido encontrar nada a tal efecto.

En un principio se planteaba la posibilidad de hacer una comparativa sobre las distintas técnicas estudiadas. Sin embargo, los resultados obtenidos no han sido lo suficientemente significativos como para llevarla a cabo.

Pese a todo, cabe destacar los resultados obtenidos a partir de muestras de *melodía*. La gran diferencia entre estos resultados y los obtenidos mediante el restos de técnicas es debida a la naturaleza de las muestras, puesto que éstas han sido generadas haciendo uso de un instrumento diferente cada vez, y aunque musicalmente las notas sean las mismas, los tonos emitidos por distintos instrumentos difieren considerablemente, lo que produce que la extracción de características produzca resultados bastante dispares.

Respecto a las muestras de *fragmento*, la intuición lleva a pensar que dado que contienen la misma cantidad de instrumentos que en las canciones de la base de datos, a la hora de extraer las características estos deberían de afectar del mismo modo que al construir la base de datos, y por tanto

se debería poder reconocer más fácilmente. Si bien esto es cierto, hay que tener en cuenta que debido a las condiciones en las que se han obtenido las muestras (ver Sección 4.3), la frecuencia de muestreo y calidad de audio de las mismas es distinta para cada muestra, lo que provoca que las características extraídas difieran en mayor medida con las canciones a comparar.

Por último, se debe tener en cuenta que, aunque el problema de la instrumentalidad no está presente en las muestras de *canto*, la gran mayoría de las canciones de la base de datos que poseen letra tienen un considerable nivel de dificultad a la hora de ser cantadas por alguien que no haya sido entrenado para ello. Esto supuso un cierto reto a los voluntarios, produciendo como resultado que muchas de las muestras se asemejasen menos a las canciones originales que otras más sencillas y más conocidas por los voluntarios.

7.2. Trabajo futuro

A continuación se describen algunas posibles mejoras que sería interesante probar de cara a una mejora de este proyecto, y que no se han incluido en el mismo puesto que iban más allá de lo que se pretendía abordar en él.

En este proyecto se han visto algunos algoritmos para la detección de *pitches* (ver apartado 3.3.1). Sin embargo, en la literatura se proponen métodos más específicos para la detección de *pitches* en música polifónica tales como: el uso de modelos psicoacústicos [33]; la integración de reglas musicológicas para modelar la probabilidad de transición entre notas [34]; o el uso de conocimiento musical a la hora de realizar la estimación de los *pitches* [34, 35].

Otra mejora a tener en cuenta es aplicar un filtro para eliminar las frecuencias no audibles por el oído humano [3] a las canciones de la base de datos antes de proceder a la extracción de características, puesto que si el oído humano no es capaz de percibirlas, éstas no van a estar reproducidas dentro de las muestras. Esta mejora debería tenerse en cuenta únicamente a la hora de clasificar muestras del tipo *tarareo*, *canto* y *silbido* (ver Sección 4.3).

También sería interesante probar a extraer la melodía de las canciones y usar ésta a la hora de clasificar las muestras. En la literatura se proponen algunos algoritmos a tal efecto [3, 36–42]. Sin embargo, al contrario de lo que ocurre con la música vocal, los instrumentos tienen un mayor rango de *pitches*, y son capaces de producir rápidas secuencias de *pitch* con una alta variación entre cada uno de ellos. Además, un instrumento que esté tocando la melodía puede estar muy cercano, tanto en el timbre como en el contorno de *pitch*, a uno que esté tocando la música de acompañamiento, lo que dificulta

aún más la tarea de separar la melodía del acompañamiento. Esto se complica cuantos más instrumentos haya tocando a la vez [43]. Por tanto, en lugar de aplicar estas técnicas a todas las canciones y muestras, otra posible mejora sería aplicarlas únicamente a aquellas pertenecientes a las versiones *con letra*.

Se podrían también probar otros algoritmos de clasificación como los propuestos en [44] (*alineamiento recursivo* y *escalado lineal*) y en [45] (*búsqueda en árbol*).

Otra idea a probar es combinar las técnicas de *tarareo*, *silbido* y *canto* (cuando este sea posible). De esta manera, cada vez que un usuario quiera conocer el título de una determinada canción, se le plantea la posibilidad de introducir una muestra de cada uno de los tipos anteriormente mencionados. Estas muestras serían procesadas y clasificadas individualmente, pero a la hora de determinar la canción buscada por el usuario se combinarían los resultados individuales de cada muestra para determinar de qué canción se trata.

Por último, otra mejora interesante sería probar a construir la base de datos a partir de archivos MIDI. Esto supondría introducir las limitaciones planteadas en la Sección 2.2, pero permitiría realizar una comparativa entre este formato y el actualmente utilizado (ver Capítulo 4).

Bibliografía

[1] N. Kosugi. *A Study on Content-based Music Retrieval with Humming.* PhD thesis, Japan, October 2004.

[2] R. B. Dannenberg, W. P. Birmingham, B. Pardo, N. Hu, C. Meek, and G. Tzanetakis. A comparative evaluation of search techniques for query-by-humming using the musart testbed. *Journal of the American Society for Information Science and Technology*, 58(3):687–701, 2007.

[3] J. Salamon and E. Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech and Language Processing*, 20(6):1759–1770, 2012.

[4] G. E. Poliner, D. P. W. Ellis, F. Ehmann, E. Gómez, S. Steich, and B. Ong. Melody transcription from music audio: Approaches and evaluation. *IEEE Transactions on Audio, Speech & Language Processing*, 15(4):1247–1256, 2007.

[5] H. Yu, W. Tsai, and H. Wang. A query-by-singing technique for retrieving polyphonic objects of popular music. In GaryGeunbae Lee, Akio Yamada, Helen Meng, and SungHyon Myaeng, editors, *Information Retrieval Technology*, volume 3689 of *Lecture Notes in Computer Science*, pages 439–453. Springer Berlin Heidelberg, 2005.

[6] L. Fu and X. Xues. A new spectral-based approach to query-by-humming for mp3 songs database. In *WEC(2)*, pages 117–120, 2005.

[7] T. C. Nagavi and N. U. Bhajantri. An extensive analysis of query by singing/humming system through query proportion. *CoRR*, 4(6):73 – 86, 2012.

[8] A. K. Tripathy, N. Chhatre, N. Surendranath, and M. Kalsi. Query by humming system. *International Journal of Recent Trends in Engineering*, 2(5):373 – 379, 2009.

[9] A. Duda, A. Nürnberger, and S. Stober. Towards query by singing/humming on audio databases. In *Austrian Computer Society (OCG)*, pages 331–334, 2007.

[10] K. Ishihara, F. Kimura, and A. Maeda. Music retrieval using onomatopoeic query. In *Proceedings of the World Congress Engineering and Computer Science*, pages 23 – 25, 2013.

[11] Mirex home. `http://www.music-ir.org/mirex/wiki/MIREX_HOME`.

[12] J. S. Downie. The music information retrieval evaluation exchange (2005-2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[13] J. R. Jang. Mir-qbsh corpus. `http://www.cs.nthu.edu.tw/~jang`. MIR Lab, CS Dept., Tsing Hua Univ.Taiwan.

[14] Ioacas corpus. `http://english.ioa.cas.cn/`. Institute of Acoustics, Chinese Academy of Sciences.

[15] Mirex 2014 results. `http://www.music-ir.org/mirex/wiki/2014:MIREX2014_Results`.

[16] E. M. Voorhees. The TREC-8 question answering track report. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, pages 77 – 82, 1999.

[17] D. Parsons. *The Directory of Tunes and Musical Themes*. Spencer Brown, Cambridge, 1975.

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

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

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

[21] S. Dixon. Onset detection revisited. In *Proceedings of the 9th international conference on digital audio effects*, pages 133–137, 2006.

[22] Youtube. `http://www.youtube.com/`.

[23] Classic cat. `http://www.classiccat.net/index.php`.

[24] European archive. `http://www.europarchive.org/index.php`.

[25] Youtube. Audio library. `https://www.youtube.com/audiolibrary/music`.

[26] J. R. Jang and L. Hong-Ru. A general framework of progressive filtering and its application to query by singing/humming. *IEEE Transactions on Audio, Speech & Language Processing*, 16(2):350–358, 2008.

[27] Y. Kim and C. H. Park. Query by humming by using scaled dynamic time warping. In *SITIS'13*, pages 1–5, 2013.

[28] J. R. Jang and M. Y. Gao. A query-by-singing system based on dynamic programming. In *Workshop on Intelligent Systems Resolution (the 8th Bellman Continuum)*, pages 85–89, 2000.

[29] D. Jang and S. Jang. Query by singing/humming system based on the combination of dtw distances for mirex 2012. In *MIREX Audio Melody Extraction Contest Abstracts*, 2012.

[30] A. W. Fu, E. Keogh, L. Y. Lau, C. A. Ratanamahatana, and R. C. Wong. Scaling and time warping in time series querying. *The VLDB Journal*, 17(4):899–921, 2008.

[31] M. A. Raju, B. Sundaram, and P. Rao. Tansen: A query-by-humming based music retrieval system. In *Proceedings of National Conference on Communications (NCC)*, 2003.

[32] L. Cao, P. Hao, and C. Zhou. Music radar: A web-based query by humming system.

[33] R.F. Lyon and L. Dyer. Experiments with a computational model of the cochlea. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86.*, volume 11, pages 1975–1978, 1986.

[34] A. Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *ISMIR*, pages 216–221, 2006.

[35] J. P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the 6th International Conference on Music Information Retrieval*, pages 304–311, 2005.

[36] R. P. Paiva, T. Mendes, and A. Cardoso. Melody detection in polyphonic musical signals: Exploiting perceptual rules, note salience, and melodic smoothness. *Comput. Music J.*, 30(4):80–98, 2006.

[37] M. Marolt. On finding melodic lines in audio recordings. In *7th Int. Conf. on Digital Audio Effects (DAFx'04)*, pages 217–221, 2004.

[38] M. Goto. A real-time music-scene-description system: Predominant-f0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication*, 43(4):311–329, 2004.

[39] P. Cancela and M. St. Tracking melody in polyphonic audio. mirex 2008. In *MIREX Audio Melody Extraction Contest Abstracts*, 2008.

[40] M. P. Ryynanen and A. P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Comput. Music J.*, 32(3):72–86, 2008.

[41] V. Rao and P. Rao. Vocal melody extraction in the presence of pitched accompaniment in polyphonic music. *IEEE Trans. on Audio Speech and Language Processing*, 18(8):2145 – 2154, 2010.

[42] G. E. Poliner and D. P. W. Ellis. Classification approach to melody transcription. In *Proceedings of Int. Conf. Music Information Retrieval*, pages 161–166, 2005.

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

[44] X. Wu, M. Li, J. Liu, J. Yang, and Y. Yan. A top-down approach to melody match in pitch contour for query by humming. In *Proceedings. of International Conference of Chinese Spoken Language Processing*, 2006.

[45] J. R. Jang, H. Lee, and M. Kao. Content-based music retrieval using linear scaling and branch-and-bound tree search. In *ICME*, pages 289 – 292. IEEE Computer Society, 2001.

Apéndice A

Canciones sin letra

Título	**Compositor**	**Intérprete**
Marcha Radetzky	*Johann Strauss I*	*Concilium Musicum Wien*
Para Elisa	*Ludwig van Beethoven*	*Mauro Bertoli*
Claro de Luna	*Ludwig van Beethoven*	*Andrys*
Canon in D	*Johann Pachelbel*	*Lincoln Center CM Society*
Danubio Azul	*Johann Strauss II*	*Radio Symphonie Orch. Berlin*
Rosas del Sur	*Johann Strauss II*	*National Theatre Munich*
Primavera	*Antonio Vivaldi*	*John Harrison*
Vals de las Flores	*Pyotr Ilyich Tchaikovsky*	*Parys Conservatorium Concertvereniging Orkest*
El Lago de los Cisnes	*Pyotr Ilyich Tchaikovsky*	*Londen Sy.Orkest*
Obertura a la Caballería Ligera	*Franz von Suppé*	*United States Marine Band*
Aprendiz de Brujo	*Paul Dukas*	*Quinn Mason*
Tocata y Fuga	*Johann Sebastian Bach*	*Cor Van Esch*
Marcha Turca	*Wolfgang Amadeus Mozart*	*Michael Griffin*
La Marcha de los Toreros	*Georges Bizet*	*Desconocido*
Marcha Nupcial	*Felix Mendelssohn-Bartholdy*	*Desconocido*
Marcha	*Pyotr Ilyich Tchaikovsky*	*Parys Conservatorium Concertvereniging Orkest*
Eine Kleine Nachtmusik	*Wolfgang Amadeus Mozart*	*New Trinity Baroque*

Carros de Fuego	*Vangelis*	*Chariots of Fire OST*
Matrimonio de Amor	*Richard Clayderman*	*Richard Clayderman*
La Vida es Bella	*Nicola Piovani*	*Life is Beautiful OST*
The Game is On	*David Arnold, Michael Price*	*BBC's Sherlock OST*
Hedwig's Theme	*John Williams*	*Harry Potter OST*
Indiana Jones Theme	*John Williams*	*Indiana Jones OST*
Home Alone Main Title	*John Williams*	*Home Alone OST*
He's a Pirate	*John Williams*	*Pirates of the Caribbean OST*
Imperial March	*John Williams*	*Star Wars OST*
Star Wars Theme	*John Williams*	*Star Wars OST*
Superman Theme	*John Williams*	*Superman OST*
Concerning Hobbits	*Howard Shore*	*The Lord of the Rings OST*
The Three Hunters	*Howard Shore*	*The Lord of the Rings OST*

Apéndice B

Canciones con letra

Título	**Compositor**	**Intérprete**
Libiamo ne' Lieti Calici	*Giuseppe Verdi*	*Daniela Stigliano*
La Donna è Mobile	*Giuseppe Verdi*	*Jacques Pottier*
Aleluya	*George Frideric Handel*	*Akademisk Orkester og Kor*
O Fortuna	*Carl Orff*	*Coral Arte Viva*
Largo al Factotum della Città	*Gioachino Rossini*	*Riccardo Stracciari*
Amazing Grace	*John Newton*	*Celtic Woman*
O Mio Babbino Caro	*Giacomo Puccini*	*Montserrat Caballé*
Funiculì Funiculà	*Luigi Denza*	*Luciano Pavarotti*
O Sole Mio	*Eduardo di Capua*	*Luciano Pavarotti*
Gloria in Excelsis Deo	*Antonio Vivaldi*	*Andrea Bocelli*
Oda a la Alegría	*Ludwig van Beethoven*	*Der Haggen Orchestra*
Habanera	*Georges Bizet*	*Royal Opera House*
Time to Say Goodbye	*Francesco Sartori*	*Andrea Bocelli, Sarah Brightman*
Amigos para Siempre	*Andrew Lloyd Webber*	*Sarah Brightman, José Carreras*
Don't Cry for me Argentina	*Andrew Lloyd Webber*	*Sarah Brightman*
Somewhere over the Rainbow	*Harold Arlen*	*Judy Garland*
Que Sera Sera	*Jay Livingston*	*Doris Day*
The Phantom of the Opera	*Andrew Lloyd Webber*	*Emmy Rossum, Gerard Butler*
My Heart Will Go On	*James Horner*	*Celine Dion*
Once Upon a December	*David Newman*	*Liz Callaway*