ARTICLE TEMPLATE

Reconocimiento automático de acordes basado en una clasificación de las clases de altura por su volumen y una estimación del bajo

Arturo Camacho[a]

[a]Escuela de Ciencias de la Computación e Informática, Universidad de Costa Rica, San José, Costa Rica

ARTICLE HISTORY
Compiled 1 de marzo de 2021

Resumen
El reconocimiento automático de acordes en audio es una tarea difícil. En los últimos 20 años se han realizado numerosas investigaciones para lograrlo, en las que destaca recientemente el uso de redes neuronales profundas. En contraste, en este trabajo se proponemos un algoritmo basado en conocimiento y experticia musical. El algoritmo toma como entrada el volumen de las notas durante el acorde. Luego clasifica las clases de altura según su volumen total y usa una técnica novedosa y robusta para estimar el bajo. Con la clasificación de las clases, algoritmo forma una "palabra" y la busca en un diccionario creado manualmente con base en el conocimiento musical de un experto. Por último, usa la estimación del bajo para seleccionar la acepción correcta. El algoritmo fue probado con las estimaciones de volumen producidas por un programa y con anotaciones manuales de las fronteras de los acordes. Los resultados muestran tasas de reconocimiento del 93 % para la fundamental, 90 % para triadas mayores y menores, 90 % con inversiones, 82 % con sétimas y 79 % con ambas.

KEYWORDS
Sections; lists; figures; tables; mathematics; fonts; references; appendices

1. Introducción

La identificación de acordes a partir de audio es sumamente difícil, incluso para expertos, si la complejidad de los acordes es alta. Más complejo aún es crear un algoritmo que los identifique. La dificultad del problema es tal que a veces los expertos difieren al clasificar un acorde. Un caso bastante conocido es el del acorde inicial del tema *A Hard Day's Night* de Los Beatles. Los expertos han dado opiniones divergentes acerca de él. Ni siquiera los miembros de la propia banda han dado una respuesta satisfactoria: el guitarrista ha afirmado en entrevistas haber tocado un acorde de fa mayor con novena y el bajista pareciera haber tocado un re, que en conjunto forman un `Dm11`. Sin embargo, al escuchar la grabación algunos músicos sugieren otros acordes. Eso como si la grabación se hubiera distorsionado de forma tal que se escuchan notas que no fueron tocadas. Esto es común con instrumentos como guitarras eléctricas, que agregan armónicas que se pueden reconocer como notas por sí mismas. Esta ambigüedad de los acordes dificulta la evaluación de los algoritmos, ya que introduce variabilidad en las anotaciones con las que se comparan los resultados (Koops et al., 2019).

El reconocimiento automático de acordes tiene numerosas aplicaciones, entre ellas la

transcripción automática de música y la generación de metadatos para recuperar información en colecciones grandes de música. Ha sido un campo de investigación bastante activo en los últimos 20 años (Pauwels et al., 2019). El artículo de Fujishima (1999) fue seminal en el área. Su principal contribución fue la introducción de los *resúmenes de clases de altura* (PCP, por sus siglas en inglés), más recientemente llamados *cromagramas*. Un cromagrama pretende resumir el volumen de cada una de las doce notas de la escala cromática, sumadas sobre todas las octavas. Decimos *pretende* porque en realidad lo que hace es sumar el volumen de las *armónicas* de las notas. Cuando se toca una nota musical, normalmente se produce no solo la frecuencia fundamental sino también armónicas (múltiplos) de ella, que suelen corresponder a otras notas. De hecho, solo las armónicas cuyo orden es una potencias de dos corresponden a la nota original (1, 2, 4, 8, etc.); las otras corresponden a notas distintas (3, 5, 6, 7, etc.). La estrategia de Fujishima fue calcular un cromagrama en un lapso dentro del acorde y compararlo con un conjunto de plantillas diseñadas manualmente para reconocer cada uno de los posibles acordes. La comparación usa distancia euclidiana o correlación.

En los quince años siguientes los esfuerzos se concentraron, por una parte, en depurar los cromagramas, agregándoles prefiltrado (Cho and Bello, 2014), considerando altura en vez de armónicas (Lee, 2006), resolviendo el problema de armónicas desafinadas (Khadkevich and Omologo, 2013), usando coeficientes cepstrales en escala melódica (MFCC, Muller and Ewert, 2010) o considerando múltiples variaciones (Gómez, 2006). Por otra parte, se trabajó en mejorar las técnicas de clasificación (distancia y correlación), reemplazándolas por otras más complejas, como los modelos ocultos de Markov, en algunos casos genéricos (Sheh and Ellis, 2003) y en otros separados por género musical (Lee, 2007), n-gramas (Cheng et al., 2008) y posfiltrado (Cho and Bello, 2014). Una ventaja de los HMM es que infieren las fronteras de los acordes, es decir, los instantes en que se transiciona de un acorde a otro.

La principal dificultad que han enfrentan los sistemas de reconocimiento de acordes es la extracción de información acústica relevante para el reconocimiento de acordes, papel que han desempeñan los cromagramas. La revolución tecnológica que causaron las redes neuronales profundas, convolucionales y recurrentes (DNN, CNN y RNN, respectivamente, por sus siglas en inglés) en la segunda década del siglo, ha tenido impacto también en el área de reconocimiento de acordes. Los estudios más recientes se han enfocado principalmente en introducir y explotar estas redes (Humphrey and Bello, 2012). Uno de sus principales aportes es que han sido capaces de descifrar, por sus propios medios, información acústica relevante para el reconocimientos de acordes (Boulanger-Lewandowski et al., 2013; Deng and Kwok, 2016; Sigtia et al., 2015). Esto ha permitido incrementar el vocabulario, es decir, la cantidad de tipos de acordes reconocidos, de cinco (mayor y menor con y sin sétimas, y dominante con sétima) a quince (McFee and Bello, 2017).

Nuestro enfoque retoma el enfoque existente antes de Fujishima (1999): identificamos los acordes a partir de un reconocimiento previo de las notas que los conforman. En aquel tiempo este enfoque era inapropiado, debido a la alta tasa de errores producidos por los algoritmos de reconocimiento de notas. Sin embargo, el avance que se ha tenido en esta área permite retomar el enfoque. El algoritmo que proponemos toma como entrada el volumen instantáneo de cada una de las 88 notas del rango estándar del piano a lo largo del acorde las usa para inferir este. Para probarlo usamos un programa, escrito por el autor, que calcula el volumen de las notas a lo largo del tiempo. Por otra parte, los tiempos de inicio y finalización de cada uno de los acordes son introducidos manualmente mediante una aplicación para dispositivo móvil. Los usuarios tocan la pantalla cada vez que escuchan un nuevo acorde. El tiempo de toque

se registra como tiempo de inicio, y el tiempo en que se soltó como tiempo de finalización. Estos tiempos se pueden ajustar luego manualmente moviendo unas líneas que se dibujan en la pantalla para marcar los tiempos.

Se puede argumentar que requerir que los tiempos de inicio y finalización de los acordes se ingresen manualmente hace el sistema poco útil, pero opinamos lo contrario, por dos razones. En primer lugar, en la música es común que se toquen acordes de paso para transicionar entre dos acordes principales o como adorno dentro de un acorde de más larga duración. Permitirle a la persona establecer las fronteras de los acordes le da más libertad, ya que le permite tratar los acordes de paso por aparte, incluirlos dentro del principal o ignorarlos. En segundo lugar, identificar auditivamente cuándo empieza un nuevo acorde es mucho más fácil que identificar los acordes propiamente. Por tanto, una persona que tenga la capacidad de reconocer cambios de acorde pero no los acordes en sí verían gran utilidad en el sistema. Además, los cambios de acorde suelen ocurrir de forma regular en la música, típicamente cada 2 o 4 tiempos en música de ritmo binario o 4/4, o cada 3 tiempos en música ternaria. Por tanto, aún si la persona tiene dificultad para reconocer cambios de acorde, si toca la pantalla cada 2 o 4 pulsos (o 3 en música ternaria) posiblemente va a obtener buenos resultados. En el peor de los casos, si un acorde dura más pulsos que eso, lo que ocurre es que el acorde va aparece varias veces consecutivas. La persona lo puede dejar así o puede pedirle a la aplicación que combine los acordes, en cuyo caso se forma uno solo cuyo inicio es el inicio del primero y el final es el final del último. Eso suele incrementar la robustez del acorde, ya que el algoritmo cuenta con más datos para hacer la estimación. Si la persona es incapaz de identificar los pulsos en la música, debería empezar por desarrollar esa destreza antes de adentrarse en el mundo de la armonía. Por último, hacemos notar que existen algoritmos para identificar el pulso, los que podrían complementar nuestro sistema (Dixon, 2001; Ellis, 2007; Goto, 2001). Sin embargo, hay tipos de música en los que el pulso es irregular, por ejemplo muchas piezas de música clásica, las cuales requieren de un director que coordine los músicos, por lo que quizá estos algoritmos tendrían dificultades con estos tipos de música.

El resto del artículo se distribuye como sigue: la sección 2 describe el algoritmo, la sección 3 la evaluación, la sección 4 los resultados y la sección 5 las conclusiones.

2. Algoritmo

El algoritmo tiene dos partes principales que se ejecutan de forma independiente. La primera consiste en ordenar las doce clases de altura (do, do sostenido, re, re sostenido, etc.) por volumen, de mayor a menor. El volumen de las clases se obtiene sumando los volúmenes de las notas correspondientes a lo largo de las octavas y de la duración del acorde. A manera de comentario, en el análisis basado en partituras el volumen es ignorado, sea porque no está especificado en estas o porque se asume que no juega un papel importante en la armonía. Sin embargo, en un análisis automático basado en audio, el volumen es importante, no solo porque pueden haber errores al identificar las notas, sino también porque es el volumen es útil para discernir entre notas del acorde y notas de adorno, las cuales suelen tener menor duración, por lo que su volumen total a lo largo del acorde suele ser menor. La otra parte del algoritmo tiene como propósito identificar el bajo del acorde. Esto es es importante porque el bajo puede ser determinante para distinguir acordes. Por ejemplo, las notas do, fa y sol pueden conformar tres acordes distintos, dependiendo de qué nota toque el bajo (los acordes son `Csus4`, `Fsus2` y `G7sus4 no5`, dependiendo de si el bajo toca do, fa o

sol, respectivamente). El bajo también sirve para reconocer *inversiones*, que ocurren cuando el bajo no toca la fundamental del acorde sino otra nota. Por ejemplo, si en el acorde de do mayor (`C`), conformado por las notas do, mi y sol, el bajo toca la nota mi, el acorde se denota `C/E` (do con bajo en mi), y si toca sol, `C/G` (do con bajo en sol).

2.1. Cálculo del volumen de las clases de altura

Al algoritmo se le da como entrada el volumen de las notas en forma de una matriz $M \times 88$, donde la i-ésima fila corresponde al volumen de las 88 notas en el intante i y M es proporcional a la duración de la canción. El programa que usamos para estimar el volumen de las notas muestrea los volúmenes cada $\Delta t = 256/11025 \approx 0{,}023$ segundos, es decir, produce unas 43 estimaciones por segundo. Se asume que los tiempos de inicio y finalización de los acordes también son dados, en nuestro caso porque fueron introducidos manualmente. Inicialmente, el algoritmo analiza cada acorde por separado y luego los depura considerando pares consecutivos. El primer paso para identificar el acorde es hallar la submatriz V, de tamaño $m \times 88$, que lo contiene. Esto es trivial, ya tanto Δt como los tiempos de inicio y finalización del acorde son dados. El volumen total de la j-ésima nota ($j = 0, 1, \ldots, 87$)[1] a lo largo del acorde es

$$v_j = \sum_{i=0}^{m-1} V_{i,j}. \tag{1}$$

Los volúmenes de las notas que pertenecen a la misma clase de altura se suman para definir el volumen total de la k-ésima clase ($k = 0, 1, \ldots, 11$):

$$w_k = v_k + v_{k+12} + v_{k+24}, \ldots \tag{2}$$

2.2. Estimación del bajo

El siguiente paso consiste en estimar la clase de altura del bajo. Un enfoque ingenuo sería buscar la nota más grave con volumen no nulo, pero esta estrategia es altamente propensa a errores, ya que podría haber errores en la estimación de las notas o el bajo podría tocar adornos por lapsos breves. Una técnica más robusta es estimar el bajo a partir de las *notas características* de las clases. Si tomamos los volúmenes de las notas como una distribución de probabilidad, podemos calcular el valor medio de las clases ($k = 0, 1, \ldots, 11$) como

$$\mu_k = \frac{v_k k + v_{k+12}(k+12) + v_{k+24}(k+24) + \cdots}{w_k}. \tag{3}$$

La *nota característica* n_k de la clase k se obtiene redondeando la media respectiva a la nota más cercana de la clase. Por ejemplo, si la media de la clase 0 (nota la) es $\mu_1 = 22{,}7$, esta se redondea a 25, que es la nota más cercana de la clase (las notas de la clase son $1, 13, 26, \ldots, 85$), y se asigna la nota característica $n_0 = 25$. (Este paso de redondeo ha mostrado ser beneficioso en algunos casos, pero no ha sido probado de forma exhaustiva, por lo que tal vez sea innecesario y se pueda usar $n_k = \mu_k$).

[1] Usamos índices que empiezan en cero en vez de uno porque esto simplifica algunas fórmulas en nuestra exposición.

2.3. Ordenamiento de clases por volumen

El paso siguiente es ordenar las clases por volumen, de mayor a menor, y anular las que tengan un volumen menor que 5 % del total (reasignándoles un volumen $w_k = 0$). Esto se hace para evitar que el ruido, notas ornamentales o errores en la estimación de las notas afecten el resto del algoritmo.

Otro aspecto de la música que puede afectar negativamente la identificación de las notas es el vibrato. El programa que usamos para identificar las notas no reconoce vibrato, el cual se refleja en oscilaciones periódicas de extensión ± 1 o ± 2 semitonos en el volumen de las notas. Para lidiar con el vibrato ideamos una heurística que consiste en lo siguiente: empezando con la clase de mayor volumen, buscamos entre las clases con volumen menor aquellas con nota característica a distancia de ± 1 semitono y las anulamos (les asignamos $w_k = 0$). Luego repetimos esto con clases a una distancia de ± 2 semitonos, pero en este caso las eliminamos solo si su volumen está por debajo de dos tercios del volumen de la clase de referencia (valor empírico). El proceso se repite con el resto de las clases, tomadas en orden decreciente de volumen. La heurística descrita parte de la observación de que, en el vibrato, la nota central tiene más volumen que sus vecinas. Se ha observado que esta heurística sirve también para eliminar notas de apoyo u otros adornos irrelevantes para el acorde.

Las notas musicales tienen un alto factor de relatividad. Una melodía no se define en términos de las notas absolutas sino en términos de las relaciones entre ellas. Lo mismo ocurre con los acordes. Por ejemplo, la esencia de un acorde mayor, en posición fundamental, es que las notas agudas guardan con la fundamental una distancia de 4 y 7 semitonos, más algún múltiplo de 12. Por tanto, para simplificar el algoritmo, es útil trabajar en términos relativos y solo al final especificar la nota o clase absoluta.

El algoritmo considera las clases con base en la de mayor volumen. Por tanto, el paso que sigue es calcular las distancias de las otras clases con respecto a la más sonora. Estas distancias se especifican módulo 12, es decir, si son negativas se les suma 12. Por ejemplo, si las clases no nulas, ordenadas de mayor a menor volumen, son 7, 11, y 5 (notas sol, si y fa, respectivamente), las distancias son 4 $(11 - 7)$ y 10 $(5 - 7 + 12)$. Estas distancias se concatenan, en ese orden, para formar un vector de intervalos. Por conveniencia, se reemplazan los valores 10 y 11 por las letras A y B, respectivamente, para escribir el vector como una *palabra* en duodecimal, es decir, con los símbolos 0, 1, 2, ..., A, B.

2.4. Diccionario

Para determinar el tipo de acorde, se busca la palabra en un diccionario. En nuestro caso, el diccionario fue creado por el autor, con base en su conocimiento de la música. El diccionario suele contener varias *acepciones* para cada palabra. A diferencia de un diccionario de un lenguaje natural (como el español), en el que se buscan palabras completas y si no están la búsqueda es fallida, en nuestro caso si la palabra no se encuentra se usa entonces el prefijo más largo. Esto siempre es posible, ya que el diccionario contiene la palabra vacía, cuya acepción es vacía y se interpreta como ausencia de acorde (*no chord*), y los símbolos individuales (1, 2, ..., A, B), cuyas acepciones devuelven `1`, que se interpreta como unísono (una sola nota).

2.4.1. Forma de uso

Cada acepción del diccionario tiene tres partes separadas con coma: *bajo*, *fundamental* y *tipo*, y termina con punto y coma (`;`). El bajo y la fundamental son símbolos duodecimales y el tipo es una hilera, que puede estar vacía, en cuyo caso significa un acorde mayor. Un ejemplo de definición es el siguiente:

```
A5. 0,0,7 sus4 no5; A,A, sus2; 5,5, sus4;
```

Aquí se define la palabra `A5`, usada cuando las clases no nulas, ordenadas de mayor a menor volumen, guardan intervalos de 10 y 5 semitonos, respectivamente, con respecto a la más sonora. La primera acepción tiene bajo `0`, fundamental `0` y tipo `7 sus4 no5`. Esto significa que si el bajo estimado, es decir, la clase con nota característica más grave, corresponde a la nota más sonora (que es la que tiene intervalo de cero con respecto a sí misma), el acorde tiene como fundamental esa nota y es del tipo `7 sus4 no5`. Por ejemplo, si la clase más sonora es la clase 7 (nota sol), este acorde es un `G7 sus4 no5`. La segunda acepción tiene bajo `A`, fundamental `A` y tipo `sus2`. Esta acepción se usa si la clase con nota característica más grave está 10 semitonos por encima de la más sonora. Al igual que antes, la fundamental coincide con el bajo (ambas tienen `A`). Si la clase más sonora es sol, el acorde es `Fsus2`. Por último, la tercera acepción tiene bajo `5`, fundamental `5` y tipo `sus4`. Se usa si la clase con nota característica más grave es la clase localizada 5 semitonos por encima de la más sonora. Si esta última es sol, el acorde es `Csus4`.

En vez de que el algoritmo de una única respuesta, consideramos útil que brinde varias, ordenadas de mayor a menor idoneidad. El programa que usamos para estimar el volumen de las notas es particularmente bueno en identificar el bajo, por lo que optamos por ordenar las respuestas según el bajo. Si en el ejemplo del párrafo anterior las notas características de las clases, ordenadas de más grave a más aguda, correspondieran a las clases 5, A y 0, respecto a la clase de mayor volumen, y esta fuera la clase 7 (nota sol), el orden de las respuestas sería `C7 sus4`, `Fsus2` y `G7 sus4 no5`.

2.4.2. Dificultades del diseño

En la sección anterior explicamos cómo usar el diccionario. En esta sección explicamos parcialmente cómo diseñarlo. Más precisamente, nos enfocamos en las principales dificultades con las que nos enfrentamos al hacerlo. Recomendamos tomarlas en cuetna al crear un diccionario propio.

Rechazo de sufijos Una de las principales dificultades que encontramos al crear el diccionario fue decidir a partir de qué símbolo de una palabra rechazar las clases restantes. Suponga el caso extremo en que la palabra es muy larga, quizá de longitud 11, que corresponde al caso en que aparecen todas las notas de la escala cromática en un acorde. Esto puede ocurrir si durante un acorde se toca tal escala, aunque sea brevemente. Probablemente algunas clases se anulen porque su volumen contribuya menos de el 5 % del total, pero algunas podrían sobrevivir. Un segundo caso en que podrían aparecer más notas de la cuenta es que una de las notas del acorde tenga vibrato y la heurística diseñana para cancelar el vibrato no lo logre (esto podría ocurrir si una de las notas vecinas de la principalaparece con bastante volumen en una octava distinta, causando un salto de una o más octavas en la nota característica de la clase correspondiente). En tal caso, podría aparecer una palabra como `A5B1`, donde los símbolos `B` y `1` son causados por el vibrato. (Si la clase de mayor volumen fuera sol, la

palabra correspondería a las notas sol, fa, do, si y do sostenido). Esto no es un acorde usual, y el diccionario debería crearse de forma cuidadosa para evitar incluir palabras o acepciones que correspondan a acordes poco probables, que induzcan a falsos positivos. En nuestro caso, decidimos no incluir esta palabra en nuestro diccionario, así como tampoco la palabra `A5B`. Por tanto, en este caso se usaría la palabra `A5` que es el prefijo más largo de `A5B1` presente en nuestro diccionario. En tal caso, se ignorarían en la estimación del bajo las notas características de las clases correspondientes a `B` y `1`.

Reconocimiento de acordes en prefijos Un caso similar al anterior, pero de cierta forma opuesto, ocurre cuando un prefijo de una palabra forma un acorde común. Tomemos por ejemplo la palabra `47A2`. Si la clase más sonora es la clase 0 (nota do), las restantes, en orden decreciente de volumen, son mi, sol, si bemol y re. Estas notas conforman el acorde `C9`. Sin embargo, existe la posibilidad de que las últimas dos notas tengan mucho menos volumen que las primeras, sea porque eran notas de paso o porque eran débiles de por sí. En tal caso, el acorde sonaría predominantemente como un acorde mayor simple, `C`, y denotarlo como `C9` sería excesivo. Para solventar esta situación, usamos un estrategia ávida que consiste en, al recorrer la palabra de izquierda a derecha (de la clase más fuerte a la más débil), reportar un acorde común cada vez que se halle. Por ejemplo, definimos la palabra `47A2` como siguiente:

```
47A2. 0,0,; 0,0,7; 0,0,9; 4,0,|; 4,0,7|; 7,0,; 7,0,7|; A,0,7|;
```

Si la clase 0 es la más sonora y también es la que tiene la nota característica más grave, la respuesta empezaría con las acepciones que empiezan con cero (`0,0,; 0,0,7; 0,0,9;`), por lo que el acorde simple `C` aparecería como primera respuesta, luego `C7`, y en tercer lugar `C9`. Aprovechamos este ejemplo para introducir la barra vertical '`|`' que usamos para denotar una inversión. Por ejemplo, "`4,0,|`" indica un acorde mayor en primera inversión y "`4,0,7|`" un acorde dominante con sétima también en primera inversión. No incluimos la acepción "`4,0,9|`" porque consideramos que un acorde dominante con novena y en primera inversión es poco común, pero se puede agregar al diccionario si se quiere. Esta flexibilidad es una de las principales ventajas de nuestro algoritmo con respecto a enfoques de aprendizaje de máquina, como las redes neuronales: para introducir un nuevo acorde basta con crear o modificar unas cuantas entradas en el diccionario (aunque pueden ser muchas si la palabra es larga, ya que normalmente hay que considerar las permitaciones de sus símbolos), mientras que con redes neuronales se requiere crear una nueva red, generar una gran cantidad de datos de entrenamiento y entrenar la red.

2.5. Posprocesamiento

A continuación explicamos unas heurísticas que hemos creado para mejorar el reconocimiento de los acordes después de finalizada la etapa de reconocimiento inicial descrita anteriormente. Las primeras dos heurísticas combinan información de acordes consecutivos para mejorarlos y la última se aplica a nivel local (del acorde individual).

2.5.1. Acordes con tercera ausente

Una particularidad del programa que usamos para estimar el volumen de las notas es que la tercera de los acordes a veces aparece muy débil. Al principio pensamos que era un error del programa, pero después de analizar la situación nos dimos que

tiene sentido. En la música de occidente se acostumbra evitar dar volumen excesivo a la tercera del acorde. Por ejemplo, las posiciones estándar de guitarra, que pueden contener hasta seis notas, suelen duplicar la tónica y la quinta de los acordes, pero casi nunca la tercera. Lo mismo ocurre con el piano y los arreglo multiinstrumentales. Por tanto, es normal que la tercera del acorde suela aparecer con poca energía acústica. Esto es un problema serio, porque en la teoría de la música occidental la tercera de un acorde juega un papel preponderante (define si un acorde es mayor o menor, por ejemplo). Para aliviar este problema lo que hacemos es que, si la tercera está ausente (sea porque su volumen no alcanzó el 5 % o porque fue anulada por la heurística de eliminación del vibrato), partimos del supuesto de que nuestra mente sigue el principio de que si no se detectan cambios en las cosas es porque siguen igual. Entonces, si no se detecta la tercera de un acorde, se revisa la lista de volúmenes del acorde anterior. Si la tercera del acorde actual aparece ahí, se *toma prestada* de él. Si aparecen tanto la tercera mayor como la menor, se toma la de mayor volúmen. Nuestros experimentos muestran que esta heurística suele funcionar la mayor parte del tiempo. Sin embargo, dependiendo de la secuencia de acordes, podría más bien empeorar las cosas.

2.5.2. Acordes suspendidos

Un caso similar al anterior es el de los acordes suspendidos. Una suspensión ocurre cuando la la tercera de un acorde se retarda y, en vez de aparecer con el resto de las notas, aparece después, a veces en el acorde siguiente. Normalmente la tercera se queda en una nota del acorde anterior, que suele ser la segunda del actual o, más comúnmente, la cuarta. En ciertas situaciones, por ejemplo, para simplificar los acordes, uno querría eliminar tales suspensiones. Para eso ideamos una heurística similar a la de la sección anterior, solo que esta vez, en vez de recurrir al acorde anterior, recurrimos al siguiente. De nuevo, esta heurística suele dar buenos resultados, pero depende de la secuencia de acordes de la canción.

2.5.3. Acordes con quinta ausente

Otro caso similar a los anteriores es la ausencia de la quinta de un acorde. Esta nota se suele omitir, con la justificación de que de todas formas aparece en el espectro. En la mayoría de instrumentos acústicos, las armónicas de baja frecuencia suelen tener más energía que las de alta. Como la quinta de un acorde se corresponde con la tercera armónica del bajo (asumiendo que se toca en disposición fundamental), suele aparecer dentro de las primeras armónicas del bajo, por lo que puede escucharse aunque no se toque. Sin embargo, el programa que usamos para estimar los volúmenes identifica notas, no armónicas, por lo que las armónicas suelen ser absorbidas por su nota respectiva. Por tanto, lo usual es que el programa sea fiel a la realidad y que omita la quinta cuando está ausente. En concordancia con esto, las definiciones en nuestro diccionario también reportan un `no5` cuando la quinta está ausente. Algunos músicos se han quejado de esto y han expresado que dificulta la lectura. Como la mayoría del tiempo las quintas son justas (seis de los siete acordes de la escala mayor tienen quinta justa, y algo similar ocurre en la escala menor, dependiendo de qué tipo se use), incorporamos la opción de ignorar los mensajes de supresión de la quinta (`no5`), en cuyo caso el lector asumiría que la quinta es justa. Asumiento una distribución uniforme de los acordes de la escala, esto induciría a error solo 1/7 del tiempo. Sin embargo, en la realidad los acordes con quinta disminuida o aumentada son mucho menos frecuentes que los acordes con quinta justa. Según las estadísticas recopiladas

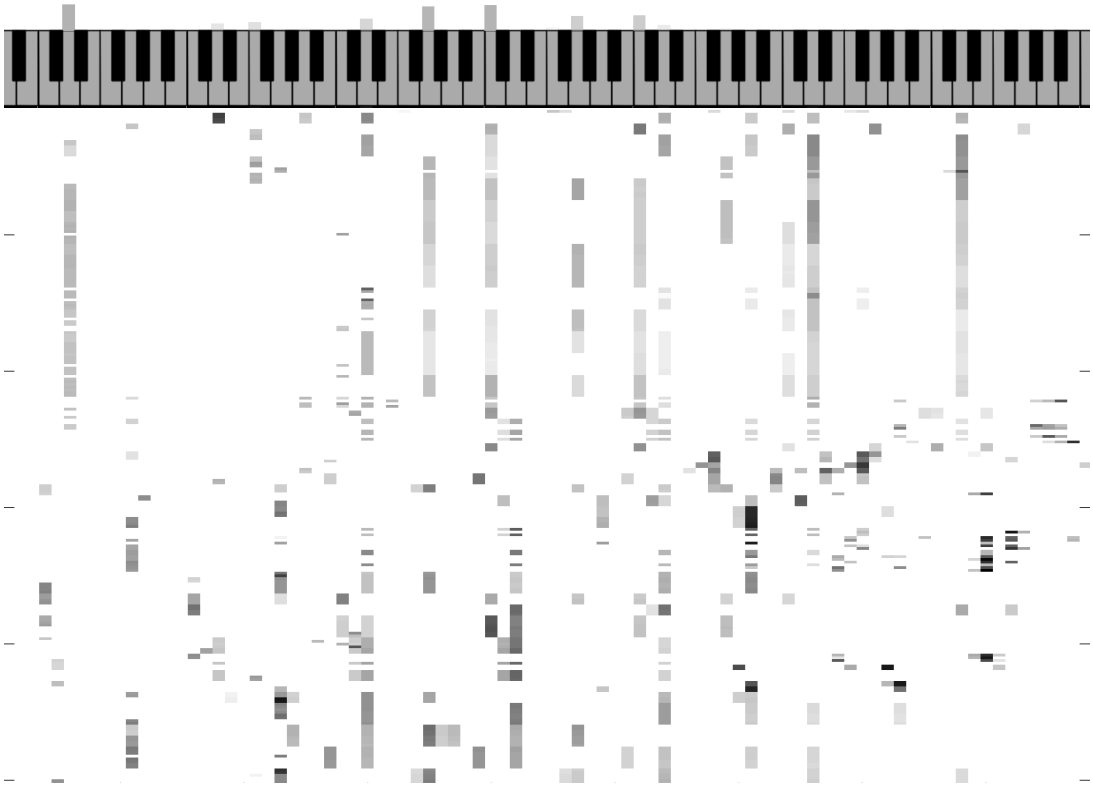

Figura 1. Volumen de las notas iniciales de la canción *A Hard Day's Night* de The Beatles. La parte debajo del teclado del piano indica el volumen de las notas durante los primeros 6 segundos de la canción: la mitad superior corresponde al famoso acorde inicial y la parte inferior a la primera frase («*It's been a hard day's night*»). Las barras encima del teclado indican el volumen promedio de las notas a lo largo del acorde inicial. Destacan el re del bajo, el do central y el sol una cuarta abajo, aunque aparecen más notas con menor volumen.

por Harte (2010) en la música de la influyente banda The Beatles, los acordes con quinta disminuida o aumentada aparecen menos del 2 % del tiempo. Por tanto, si ante la ausencia de la quinta esta se debe adivinar, lo más seguro es apostar por la quinta justa, que es lo que hace la heurística aquí propuesta.

2.6. Ejemplo

A continuación presentamos un ejemplo de reconocimiento de un acorde por parte del algoritmo. El acorde escogido es el famoso acorde inicial de la canción *A Hard Day's Night* de The Beatles, discutido en la introducción. La figura 1 muestra en forma de pianola el volumen de las notas reconocidas por el programa usado para este fin. La figura muestra los primeros seis segundos de la canción, de los cuales los

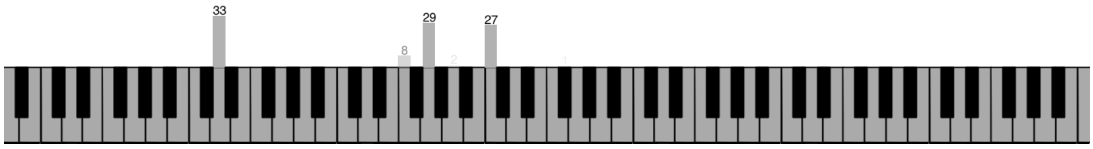

Figura 2. Notas características y volumen relativo de las clases cuyo volumen excede el 5 % del total en el acorde inicial de *A Hard Day's Night*. De la clase más grave a la más aguda: re (33 %), fa, (8 %), sol (29 %) y do (27 %).

primeros tres (arriba) corresponden al acorde en cuestión y los restantes tres (abajo) a la primera frase («*It's been a hard day's night*»). Las barras encima del teclado resumen el volumen de las notas en el intervalo correspondiente al acorde inicial. La figura 2 combina las barras de las notas que pertenecen a la misma clase y muestra la nota característica, el volumen correspondiente indicado por la altura de la barra y el valor numérico, según el procedimiento de la sección 2.2. Las clases que contribuyeron con al menos el 5 % al total, ordenadas de mayor a menor volumen, son re (33 %), sol (29 %), do (27 %) y fa (8 %). La palabra correspondiente a este ordenamiento es `5A3`, que aparece en nuestro diccionario como

```
5A3. 0,0,m7 add11 no5; 5,5, sus4; 5,5,7 sus4; A,A, sus2;
     3,3,6 sus2; 3,0,m add11 no5|;
```

Por otra parte, las clases ordenadas por nota característica, de la más grave a la más aguda, son re, fa, sol y do, que guardan intervalos de `0`, `3`, `5` y `A` (10) semitonos, respectivamente, con respecto a la clase de mayor volumen: re. Por tanto, los acordes sugeridos por el algoritmo son, ordenados por ideoneidad, `Dm7 add11 no5`, `F6 sus2`, `Dm add11 no5/F`, `G sus4`, `G7 sus4` y `C sus2`. Nuestro diccionario diferencia entre los acordes `m11` (con novena) y `m7 add11` (sin novena). Algunos músicos los consideran equivalentes, por lo que podrían reemplazar los `m7 add11` en el diccionario por `m11`. Si además se usara la heurística de asumir quintas justas cuando están ausentes (sección 2.5.3), el primer acorde sugerido sería `Dm11`. Por otra parte, si se usara la heurística de detección de vibrato, esta anularía la clase fa, con lo que se formaría la palabra `5A`:

```
5A. 0,0,7 sus4 no5; 5,5,5; 5,5, sus4; A,A, sus2;
```

En tal caso el acorde se reconocería en primer lugar como `D7 sus4` o `D7 sus4 no5`, dependiendo de si se usa o no la heurística de asumir quinta justa, respectivamente.

3. Evaluación

Para evaluar nuestro algoritmo usamos música de The Beatles y Queen y comparamos nuestros resultados con las anotaciones de Isophonics, recopiladas mayoritariamente por Harte (2010).[2] Se excluyeron de esta evaluación tres canciones: *I Want It All* de Queen, y *Love you to* y *Within you without you* de Los Beatles. La primera se excluyo porque no contábamos con la grabación (se corrompió el archivo) y las otras dos porque su estilo no es occidental, por lo que nos pareció inadecuado analizarlas con un esquema de armonía occidental. (Al final del siguiente párrafo damos una segunda razón).

El volumen de las notas fue calculado usando el programa previamente mencionado y los tiempos de inicio y finalización fueron ingresados manualmente. (Intentamos usar los tiempos incluidos en las anotaciones de Isophonics, pero los resultados obtenidos con nuestros propios tiempos fueron mejores). Para ello creamos una aplicación para dispositivo con pantalla tactil que permite registrarlos fácilmente mientras suena la canción. Todo lo que hay que hacer es pulsar la pantalla cuanto empieza un acorde y soltarla cuando termina. La pantalla muestra el volumen instantáneo de las notas en forma de una pianola, y sobre ella se colocan líneas que marcan las fronteras de los acordes. Naturalmente, este procedimiento de introducción de acordes genera lapsos vacíos. Aunque la interfaz permite mover las líneas manualmente para ajustar las fronteras, es tedioso hacerlo para cada acorde, por lo que normalmente dejamos los

[2]Las anotaciones de Isophonics están disponibles en el sitio web `http://www.isophonics.net/datasets`.

Cuadro 1. Porcentaje del tiempo que nuestro algoritmo (AC) y otros algoritmos identificaron correctamente los acordes en canciones de The Beatles y Queen, en distintas categorías (su complejidad incrementa de izquierda a derecha).

Algorithm	Root	Maj/min	Maj/Min+Bs	Sevenths	Sevenths+Bs
AC	95.20	92.98	90.38	82.43	79.45
JLCX1	86.75	86.25	84.44	75.87	74.39
JLCX2	86.51	86.05	84.23	75.64	74.17
FK2	87.38	86.80	83.43	75.55	72.60
SG1	82.03	78.67	76.84	69.20	67.55
CLSYJ1	79.89	76.50	74.33	67.79	65.82
HL2	71.21	66.97	65.63	57.75	56.58
CM1	78.66	75.51	72.58	54.78	52.36

lapsos. Como las anotaciones de Isophonics no contienen lapsos vacíos entre los acordes, al hacer la evaluación reemplazamos cada lapso vacío con el acorde anterior a este. Esta fue la otra razón para excluir las canciones *Love you to* y *Within you without you*. Además de tener un estilo con armonía no occidental, estas canciones pasan mucho tiempo en el mismo "acorde," por lo que podíamos fácilmente inflar las tasa de acierto detectando un fragmento correcto y accareándolo por el resto de la duración del acorde. En todo caso, estas dos canciones suman 8 minutos, apenas un 1.4 % de la colección total (560 segundos), por lo que no afectan los resultados significativamente.

Para hacer la evaluación usamos los procedimientos estándar de comparación de clases propuestos por Harte (2010) y ampliamente usados por la sociedad ISMIR en la evaluación MIREX, como el mapeo de categorías cuando el conjunto de estas en las anotaciones difieren de las que el algoritmo puede reconocer.[3] En particular, los acordes suspendidos fueron mapeados a acordes mayores y las quintas ausentes a quintas justas. Optamos por no usar las heurísticas de las secciones 2.5.1 y 2.5.2 porque queríamos usar la menor cantidad posible de posprocesamiento, especialmente las que involucraran interacciones entre acordes sucesivos, para evaluar lo más fielmente posible la esencia del algoritmo. Como nuestro algoritmo suele devolver múltiples sugerencias para cada acorde, en cada caso se usó la primera de ellas.

4. Resultados

Debido a que nuestro algoritmo requiere que se le den como entrada el volumen instantáneo de las notas y los tiempos de inicio y finalización del acorde, nuestros resultados no son comparables con los de los algoritmos que participan en MIREX, que se basan solo en el audio. Aún así, los mostramos referencia en el cuadro 1 (Cannam et al., 2013; Gasser and Strasser, 2018; Junyan Jiang and Xia, 2018; Lee et al., 2019), ordenados por los valores de la última columna, que corresponde a la tarea más compleja. Los algoritmos se identifican con las iniciales de los apellidos de los autores. En nuestro caso sería C, pero optamos por usar también la inicial del nombre para evitar confusiones en lo que sigue con el nombre del acorde `C`, por lo que nos referimos a él como AC. Algunos autores aportan a MIREX más de una versión de su algoritmo, por

[3]Se puede obtener información sobre ISMIR en `http://www.ismir.net`.

lo que se les agrega un número para diferenciarlas.

La segunda columna es la tarea más sencilla. Corresponde a determinar la fundamental del acorde (si la hay, sino se indica `N`). Los números indican el porcentaje del tiempo que hubo coincidencia entre el acorde resportado por el algoritmo y la anotación hecha por expertos. Nótese que no es lo mismo que porcentaje de acordes correctos. Al medir la tasa de acierto con base en el tiempo, se le da más relevancia a los acordes de larga duración que a los breves. Igualmente, se le da más énfasis a las canciones largas que a las cortas. La estimación de la fundamental es en la que más resalta AC, con una tasa de reconocimiento del 95.20 %. En parte, esto se debe a la posibilidad de manipular manualmente el inicio y final de los acordes, ya que lo hicimos tratando de obtener esta tarea bien, ya que las demás dependen de ella. (Las tareas son parcialmente acumulativas; un acierto en las demás implican un acierto en la primera).

La tercer columna corresponde a determinar si el acorde es mayor (`maj`) o menor (`min`), o si no hay acorde (`N`). En esta tarea, otros tipos de acordes básicos como disminuido, aumentado o suspendido deben mapearse a una de estas categorías. Debido a que se sabe que la mayoría de acordes en esta y otras colecciones usadas en MIREX son mayores, lo usual es mapear los aumentados y suspendidos a mayores, ya que todos poseen una tercera mayor con respecto a la fundamental. Por su parte, los acordes disminuidos se suelen mapear a menor porque ambos tipos tienen una tercera menor respecto a la fundamental. Siguiendo esta estrategia AC acertó el acorde el 92.98 % del tiempo, una caída del 2.22 % con respecto al resultado previo (reconocimiento de solo la fundamental).

La cuarta columna agrega a las demás el reconocimiento del bajo. Aunque la mayor parte del tiempo el bajo toca la fundamental, a veces no, y es importante especificar esta situación, conocida como inversión del acorde. AC logró reconocer la inversión correctamente el 90.38 %, una caída del 2.60 % con respecto a la columna anterior.

La tarea correspondiente a la quinta columna es la primera en no ser completamente acumulativa. Esta tarea no contempla inversiones, pero agrega el reconocimiento de los acordes con sétima más comunes en la música: `7`, `min7` y `maj7`. Aunque la tasa de reconocimiento de AC sigue siendo más alta que la de los otros algoritmos, en este caso 82.43 %, sufrió una caída importante con respecto a la tarea de la columna 3 (maj/min): 10.55 %. Esto se debe principalmente a que las anotaciones de Isophonics se basan en el acompañamiento, no la melodía, mientras que AC es incapaz de diferenciar entre ambos. Una sétima enfática en la melodía (con mucho volumen o larga duración) puede tener un efecto importante en el acorde reconocido por AC. Además, no está claro si los anotadores consideraron las voces secundarias (coros) como melodía o como acompañamiento. Al inspeccionar las grabaciones hallamos situaciones en que una voz secundaria cantaba la sétima del acorde y la anotación no contenía sétima.

La última tarea es la más compleja. Agrega a la de la columna 5 el reconocimiento del bajo. La tasa de reconocimiento bajó cerca de tres puntos porcentuales con respecto a la columna anterior: 79.45 %. En todas las evaluaciones, AC estuvo al menos un 5 % por encima de los demás algoritmos. Esto confirma que las canciones excluidas de la evaluación, que conforman un 1.4 % del tiempo total, no afectan significativamente los resultados.

5. Conclusiones

Presentamos un algoritmo para el reconocimiento automático de acordes. El algoritmo toma como entrada el volumen instantáneo de las notas a lo largo de la canción y los tiempos de inicio y finalización de cada uno de los acordes. El algoritmo es notablemente simple, pero requiere de un diccionario creado por un experto, aunque, con asistencia de un experto, quizá se pueda crear un algoritmo que cree o sustituya el diccionario. En nuestro caso, creamos el diccionario manualmente.

Probamos el algoritmo con acordes simples (mayores, menores, con sétimas y con inversiones) y los resultados fueron bastante satisfactorios, alcanzando en algunos casos tasas de acierto del 90 %. Aunque las pruebas reportadas no lo contemplan, el algoritmo es capaz de reconocer también acordes complejos, como acordes con novenas, onceavas y treceavas. Hemos realizado pruebas informales con ellos y los resultados han sido satisfactorios.

Una de las ventajas de nuestro enfoque con respecto a las redes neuronales, la técnica más usada en la actualidad, es que es fácilmente adaptable. Para reconocer un nuevo tipo de acorde o hacer una modificación a los reconocidos, basta con modificar el diccionario: un simple archivo de texto. Hacer esto en un sistema basado en redes neuronales requiere reentrenamiento, lo que implica generar y etiquetar una gran cantidad de datos para que la red aprenda de ellos.

Referencias

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2013). Audio chord recognition with recurrent neural networks. In de Souza Britto Jr., A., Gouyon, F., and Dixon, S., editors, *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013, Curitiba, Brazil, November 4-8, 2013*, pages 335–340.

Cannam, C., Benetos, E., Mauch, M., Davies, M., Dixon, S., Landone, C., and Noland, K. (2013). MIREX 2013 entry: Vamp plugins from the centre for digital music.

Cheng, H.-T., Yang, Y.-H., Lin, Y.-C., Liao, I.-B., and Chen, H. H. (2008). Automatic chord recognition for music classification and retrieval. pages 1505 – 1508.

Cho, T. and Bello, J. P. (2014). On the relative importance of individual components of chord recognition systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(2):477–492.

Deng, J. and Kwok, Y.-K. (2016). A hybrid gaussian-hmm-deep learning approach for automatic chord estimation with very large vocabulary. *ISMIR*, pages 812–818.

Dixon, S. (2001). Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58.

Ellis, D. P. W. (2007). Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60.

Fujishima, T. (1999). Realtime chord recognition of musical sound: a system using common lisp music. In *Proceedings of the 1999 International Computer Music Conference, ICMC 1999, Beijing, China, October 22-27, 1999*.

Gasser, S. and Strasser, F. (2018). MIREX 2018: Multi objective chord estimation. (ISMIR 2018) the 19th International Society for Music Information Retrieval Conference, MIREX special session.

Gómez, E. (2006). Tonal description of polyphonic audio for music content processing. *INFORMS J. Comput.*, 18(3):294–304.

Goto, M. (2001). An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171.

Harte, C. (2010). *Towards automatic extraction of harmony information from music signals.*

PhD thesis, Queen Mary University of London, London.

Humphrey, E. J. and Bello, J. P. (2012). Rethinking automatic chord recognition with convolutional neural networks. In *2012 11th International Conference on Machine Learning and Applications*. IEEE.

Junyan Jiang, Ke Chen, W. L. and Xia, G. (2018). MIREX 2018 submission: A structural chord representation for automatic large-vocabulary chord transcription. (ISMIR 2018) the 19th International Society for Music Information Retrieval Conference, MIREX special session.

Khadkevich, M. and Omologo, M. (2013). Reassigned spectrum-based feature extraction for GMM-based automatic chord recognition. *EURASIP Journal on Audio, Speech, and Music Processing*.

Koops, H. V., de Haas, W. B., Burgoyne, J. A., Bransen, J., Kent-Muller, A., and Volk, A. (2019). Annotator subjectivity in harmony annotations of popular music. *Journal of New Music Research*, 48(3):232–252.

Lee, K. (2006). Automatic chord recognition from audio using enhanced pitch class profile. In *Proceedings of the 2006 International Computer Music Conference, ICMC 2006, New Orleans, Louisiana, USA, November 6-11, 2006*.

Lee, K. (2007). A system for automatic chord transcription from audio using genre-specific hidden markov models. volume 4918, pages 134–146.

Lee, S.-R., Chien, I., Yeh, T.-C., and Jang, J.-S. R. (2019). MIREX 2019 submission: chord estimation. (ISMIR 2019) the 20th International Society for Music Information Retrieval Conference, MIREX special session.

McFee, B. and Bello, J. P. (2017). Structured training for large-vocabulary chord recognition. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pages 188–194, Suzhou, China. ISMIR.

Muller, M. and Ewert, S. (2010). Towards timbre-invariant audio features for harmony-based music. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):649–662.

Pauwels, J., O'Hanlon, K., Gómez, E., and Sandler, M. (2019). 20 years of automatic chord recognition from audio. In *International Society for Music Information Retrieval (ISMIR)*.

Sheh, A. and Ellis, D. P. W. (2003). Chord segmentation and recognition using EM-trained hidden Markov models. In *ISMIR*.

Sigtia, S., Boulanger-Lewandowski, N., and Dixon, S. (2015). Audio chord recognition with a hybrid recurrent neural network. In Müller, M. and Wiering, F., editors, *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 127–133.