DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Reconocimiento de notas y acordes

Procesamiento avanzado de señales

Andrés Siciliani - 814/17
Federico Alamino - 316/17
Joel Cámera - 257/14

Motivación y Objetivos

Motivación

- Queremos aplicar los conocimientos de la materia en sobre un tema que tenemos como hobbie en común.
- Por ello nos decidimos en hacer algo relativo a la música.

Objetivos

- Desarrollar un sistema capaz de identificar cada nota de una melodía.
- Extender esto mismo al reconocimiento de acordes (mayores y menores en este caso).

Código

https://github.com/federicoalamino/PAS2025_Reconocimiento_Notas_Musicales

Utilizamos Python con scipy para algunos algoritmos (filtrado, find_peeks) y matplotlib para plotear.

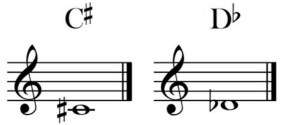
¿Que es una nota musical?

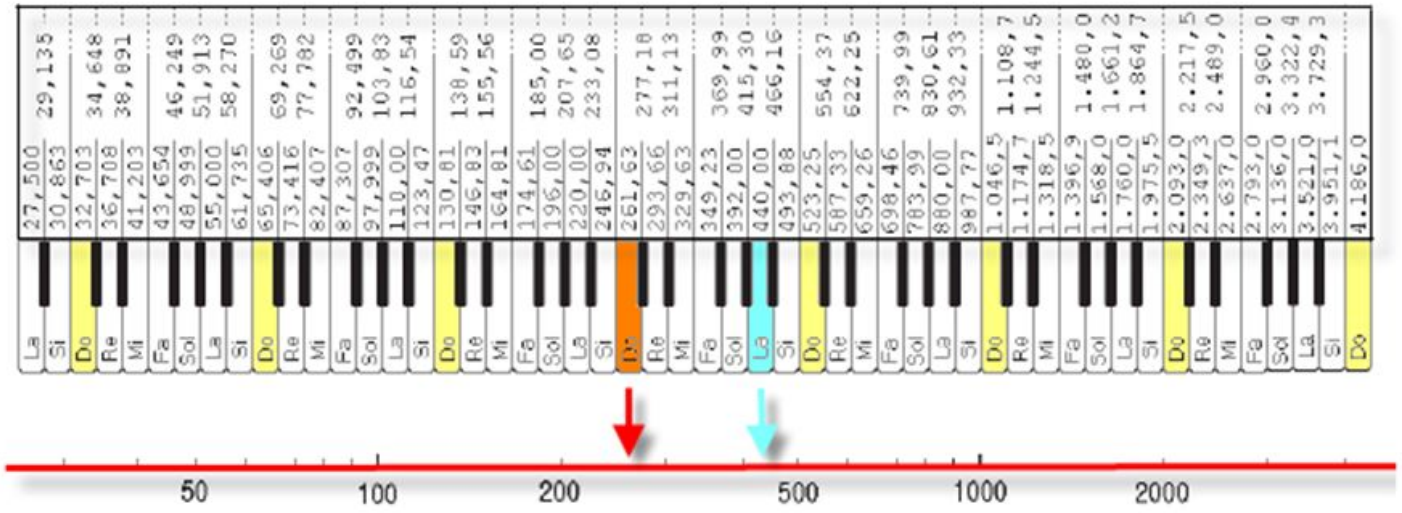
Representación de un sonido con una frecuencia fundamental bien definida.

Desde el punto de vista físico, cada nota corresponde a una vibración periódica del aire, caracterizada principalmente por su frecuencia, medida en hercios (Hz).
Esta frecuencia determina la altura del sonido percibido: frecuencias más altas generan sonidos más agudos, y frecuencias más bajas, sonidos mas graves.

Do Re Mi Fa Sol La Si Do
C D E F G A B C

Frecuencia (Hz)





Melodía de Campo: Datos Reales, No Sintéticos

La melodía seleccionada a procesar fue Arroz con leche

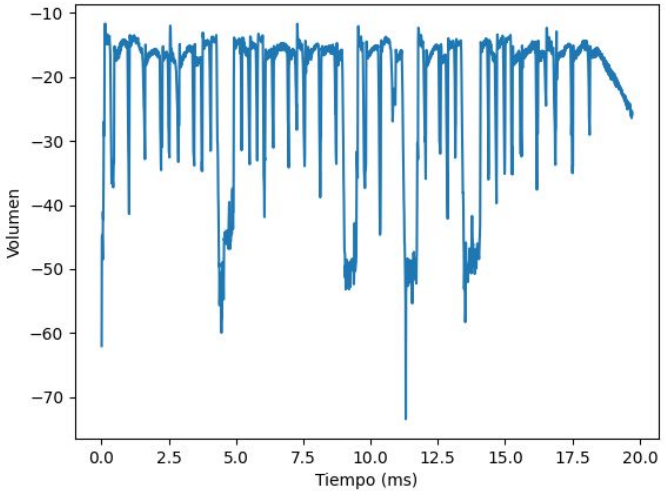
Se tomó la decisión de no procesar melodías que tengan notas con sus frecuencias exactas, sino melodías captadas por aire, para no tener casos de laboratorio, sino más bien simular la realidad.

Reconocimiento de notas

Como punto de partida, comenzamos por graficar la señal en función del tiempo.

De allí surgieron dos cuestiones a resolver:

- Reconocer en dónde están las notas.
- Identificar cuales es cada nota.

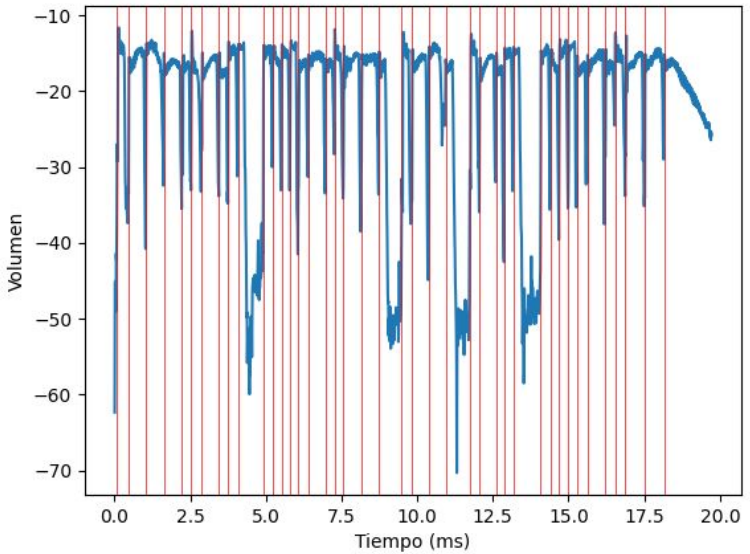


Donde se encuentran las notas

Queremos detectar los comienzos, o ataques, de cada una.

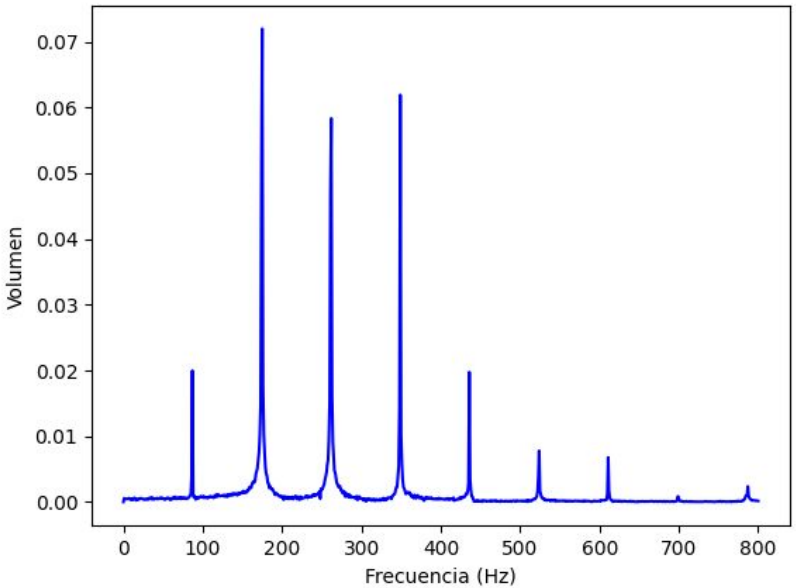
Usamos un high pass filter para quitar un poco del ruido a la señal.

Luego de esto obtenemos los tiempos de inicio aproximados de cada nota utilizando la función find_peaks que viene en SciPy.



¿Qué nota es?

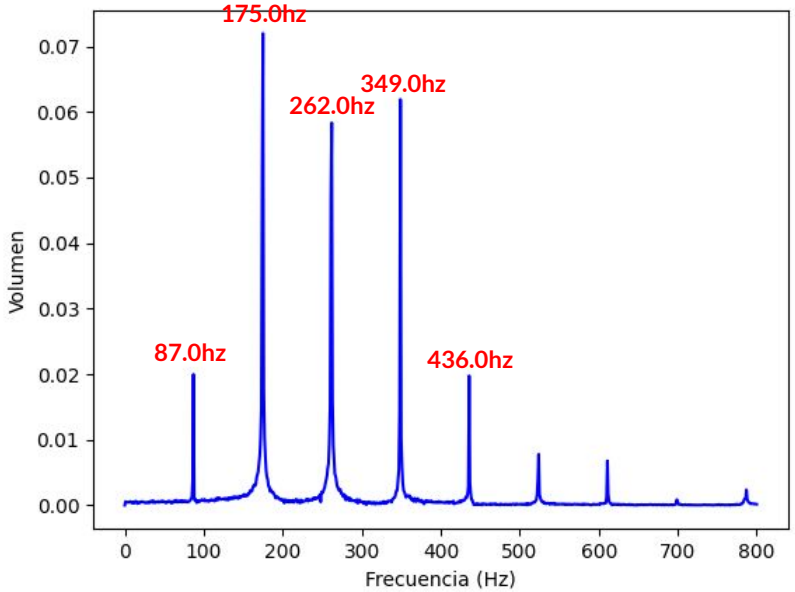
- Se utilizó FFT para convertir cada pico/ataque de la melodía en un gráfico de frecuencias.
- Con este gráfico comenzamos el análisis para ver que nota es la que estamos observando.



¿Qué nota es?

87.0hz ≈ F2 (87.31 Hz)
175.0hz ≈ F3 (174.61 Hz)
262.0hz ≈ C4 (261.63 Hz)
349.0hz ≈ F4 (349.23 Hz)
436.0hz ≈ A4 (440.0 Hz)

Tomamos como heurística la que tiene mayor volumen (ya que en pruebas funciona bastante bien) y podemos concluir que la nota es un **Fa**.

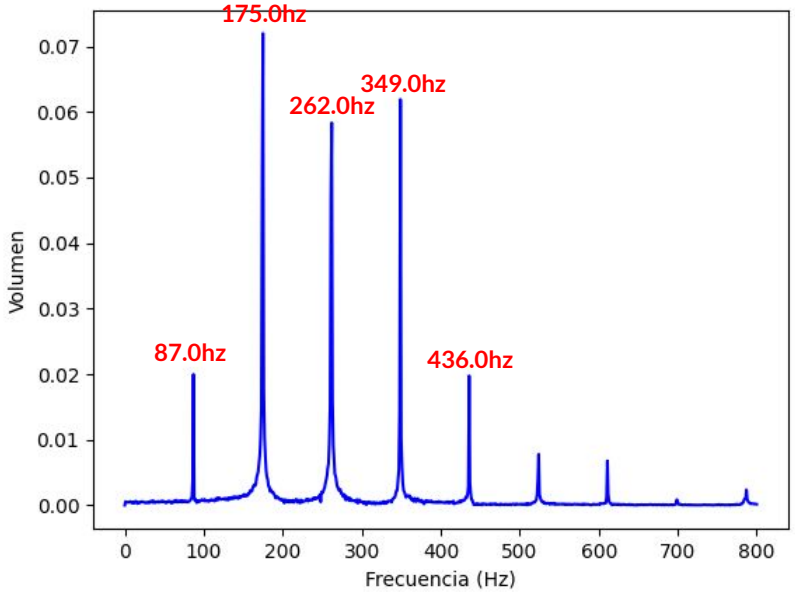


¿Qué nota es?

Claramente los valores no son perfectos, por ello tomamos como tolerancia 33 Cents.

Cents es la menor unidad que se emplea para medir intervalos musicales. Equivale a una centesima de semitono. Por ejemplo, entre Do y Do# hay 100 Cents.

No es la idea de este trabajo meterse con microtonalismo pero lo que queremos dejar es que, a pesar de que no sea perfecta cada frecuencia encontrada, se sigue percibiendo al oído la nota esperada.



Distancia de Levenshtein

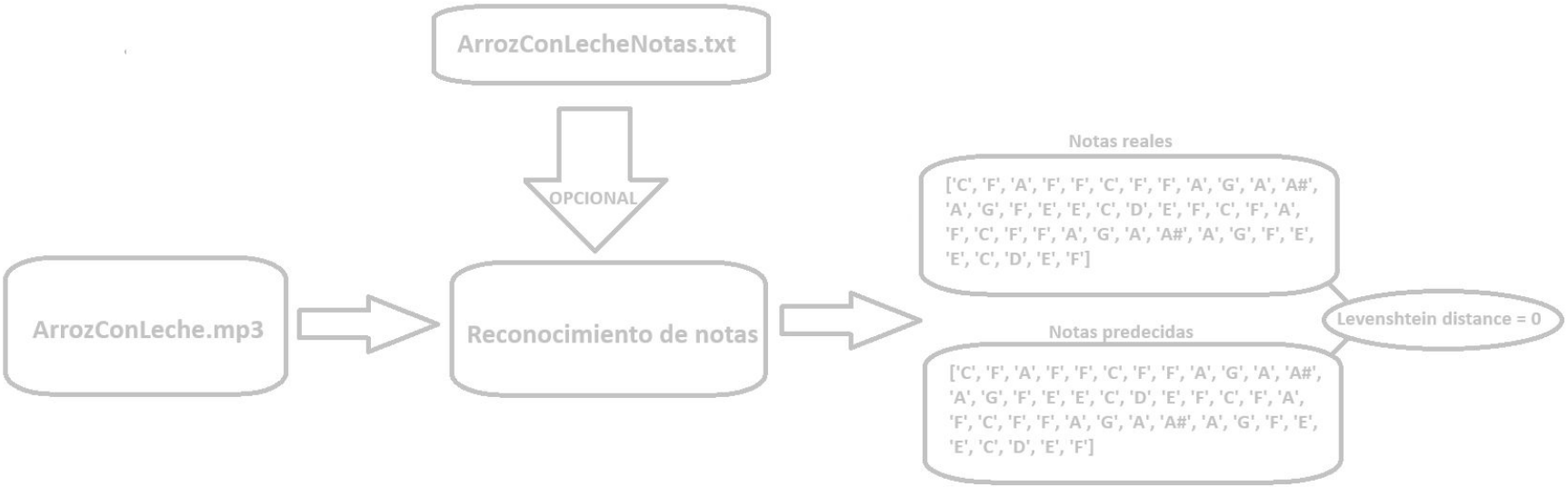
Es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra

Por ejemplo, la distancia de Levenshtein entre "casa" y "calle" es de 3 porque se necesitan al menos tres ediciones elementales para cambiar uno en el otro.

- casa → cala (sustitución de 's' por 'l')
- cala → calla (inserción de 'l' entre 'l' y 'a')
- calla → calle (sustitución de 'a' por 'e')

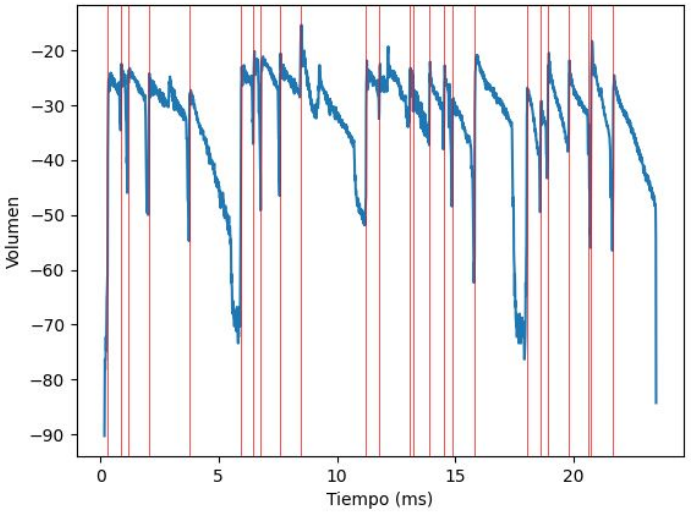
Se considera una generalización de la distancia de Hamming.

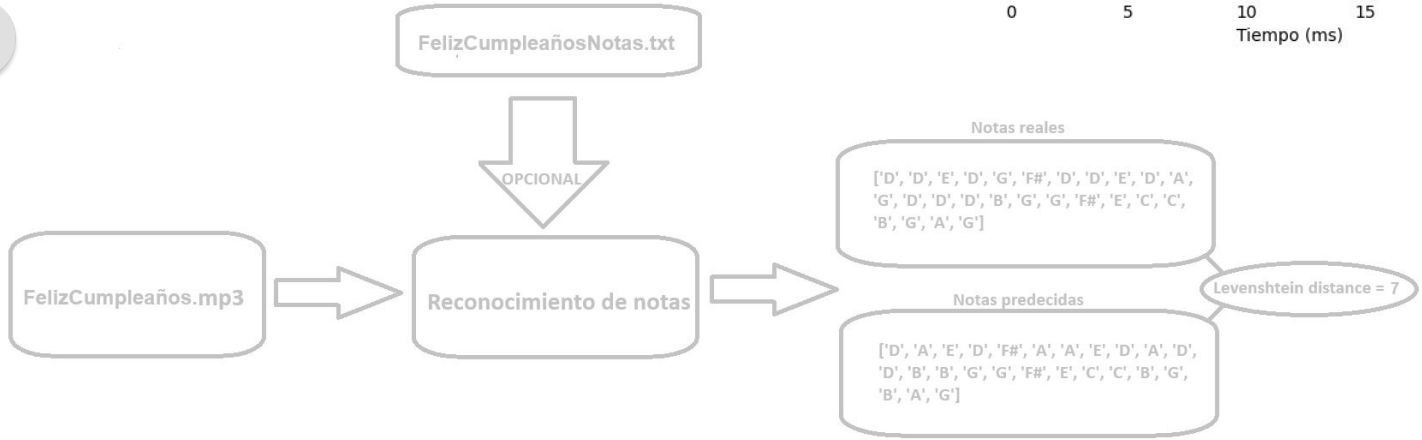
Reconocimiento de notas



Un caso no tan feliz

Probamos la siguiente melodía de feliz cumpleaños (o payaso plin plin) y no nos fue tan bien para predecir las notas.





¿Podemos utilizar lo hecho para reconocer acordes?

Anteriormente evaluamos cómo identificar notas en una melodía.

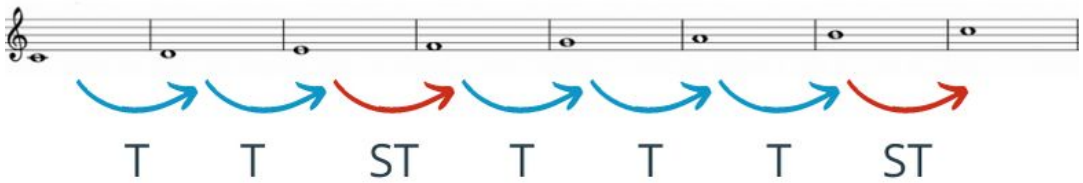
Ahora vamos a redoblar la apuesta e intentar reconocer acordes.

Un acorde es un conjunto de tres o más notas que suenan al mismo tiempo y están organizados según reglas armónicas. Un ejemplo puede ser “mi menor” que suena al tocar las notas mi, sol y si.

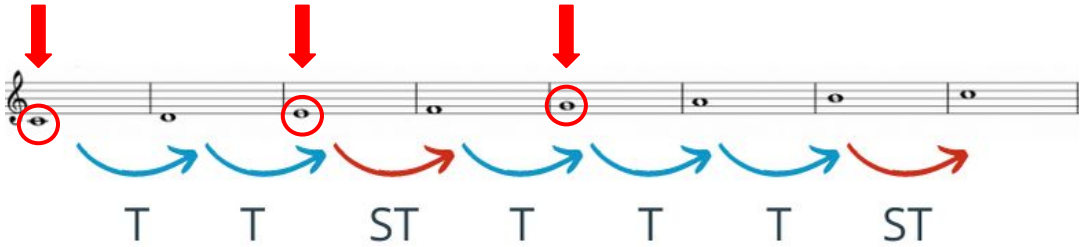
Antes de ver los acordes debemos hablar de escalas.

Una escala es una serie ordenada de notas, que siguen un patrón específico de distancias.

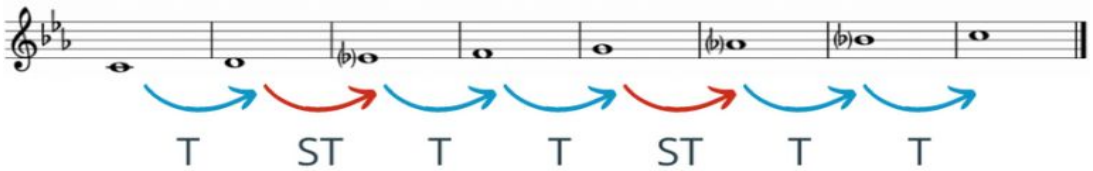
Escala de do mayor



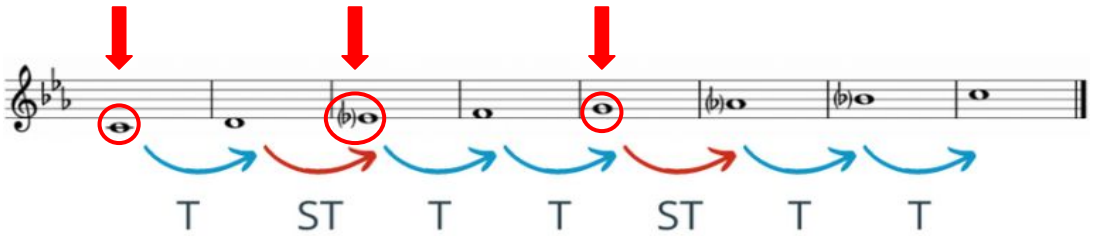
Notas del acorde de do mayor



Escala de do menor

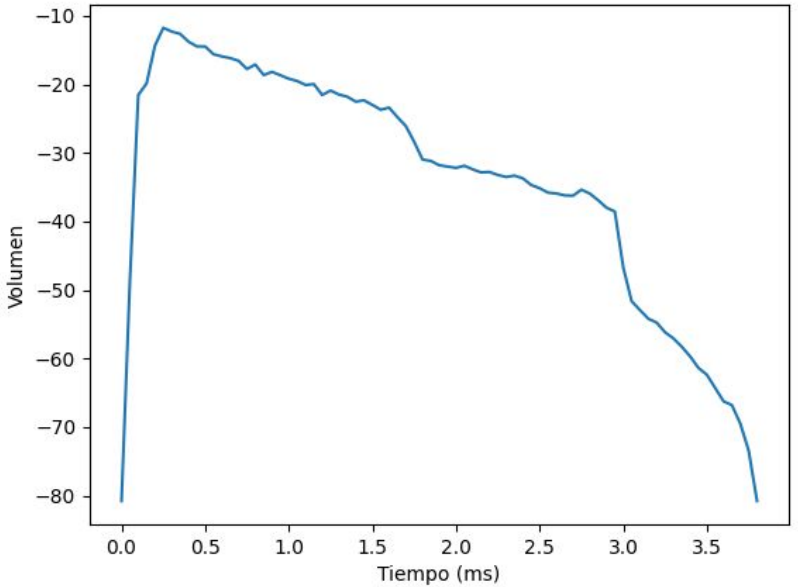


Notas del acorde de do menor



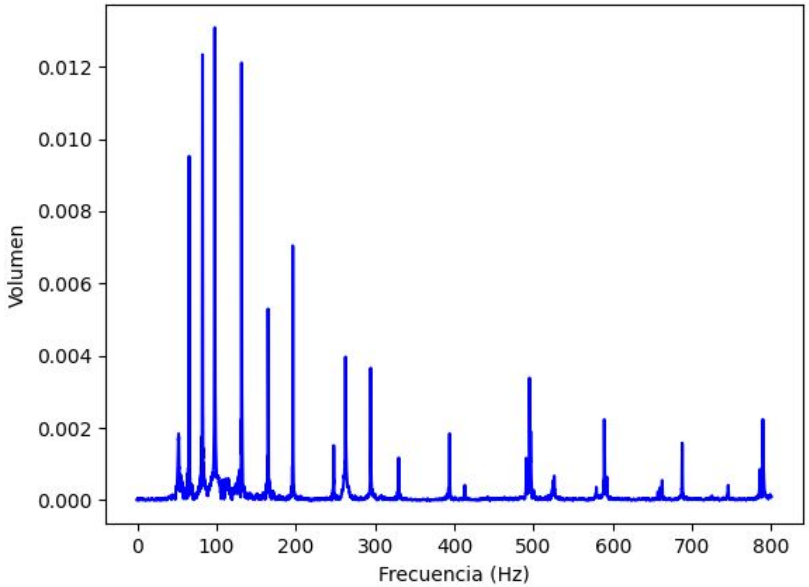
Reconocimiento de acordes

- Como punto de partida graficamos la señal.



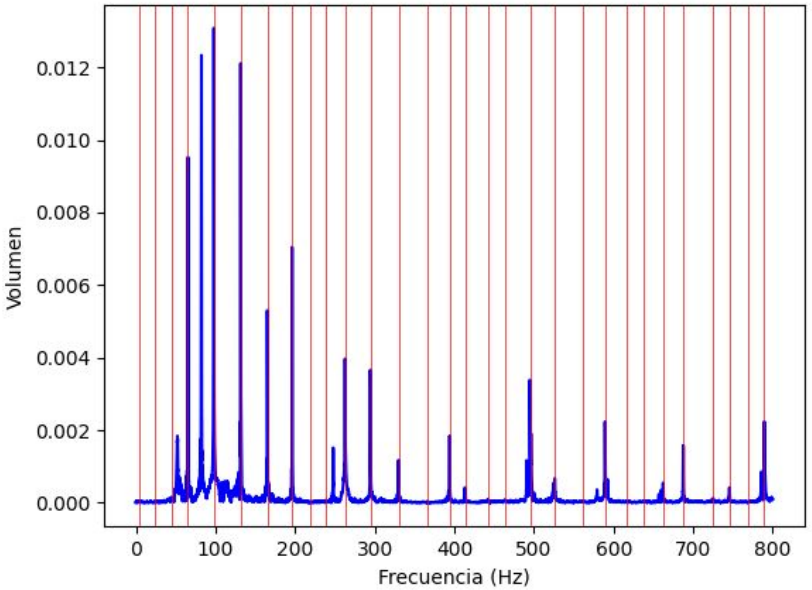
Reconocimiento de acordes

- Luego aplicamos FFT, obteniendo los valores de volumen para cada frecuencia (nota).



Reconocimiento de acordes

- Aplicamos find peaks
- Para cada pico, de la misma manera que antes, reconocemos que notas en función de su frecuencia



Reconocimiento de acordes

- Creamos una lista con (nota, volumen) para cada pico encontrado.

  [(C, 100), (E, 65), (G, 72), (C, 12), (D, 5)]

- Ordenamos esta lista por volumen de manera ascendente.

  [(D, 5) , (C, 12), (E, 65), (G, 72), (C, 100)]

- Asignamos a cada elemento un peso correspondiente al índice. (índice más pequeños, menos volumen).

  [(D, 0) , (C, 1), (E, 2), (G, 3), (C, 4)]

Reconocimiento de acordes

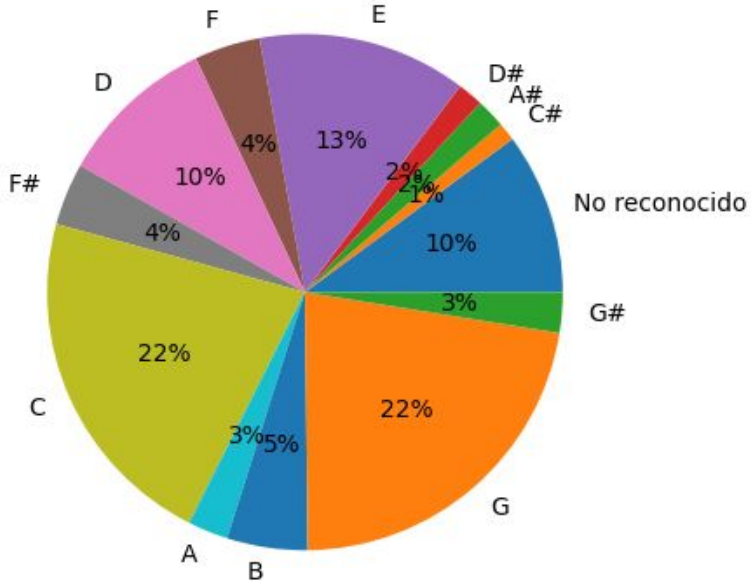
- Colapsamos las “mismas notas” sumando su peso. (pues hay muchas frecuencias que representaban la misma nota, las cuales ya fueron normalizadas).

  [(D, 0), (E, 2), (G, 3), (C, 5)]

- El resultado es una lista, donde para cada nota tenemos la suma de los pesos para todas sus apariciones.

Reconocimiento de acordes

- Aplicando lo anterior para el acorde que estamos observando obtenemos el siguiente gráfico.
- Como mencionamos antes, hay frecuencias que no pasan el nivel de tolerancia y por lo tanto no son reconocidas (lo cual tiene sentido).



Reconocimiento de acordes

- Ahora estamos en condiciones de reconocer cuáles acordes podemos formar y cual peso tienen.
- Preconfiguramos un mapa de acordes y qué notas tiene cada uno.
  Ej: "B Minor": ["B", "D", "F#"]

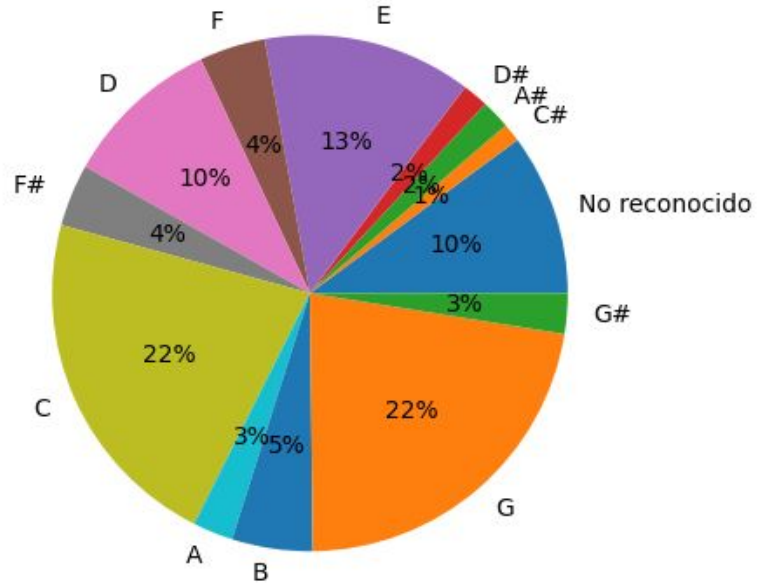
C Major: C E G	**C Minor:** C Eb G
G Major: G B D	**G Minor:** G Bb D
D Major: D F# A	**D Minor:** D F A
A Major: A C# E	**A Minor:** A C E
E Major: E G# B	**E Minor:** E G B
B Major: B D# F#	**B Minor:** B D F#
F# Major: F# A# C#	**F# Minor:** F# A C#
Db Major: Db F Ab	**C# Minor:** C# E G#
Ab Major: Ab C Eb	**G# Minor:** G# B D#
Eb Major: Eb G Bb	**D# Minor:** D# F# A#
Bb Major: Bb D F	**Bb Minor:** Bb Db F
F Major: F A C	**F Minor:** F Ab C

Reconocimiento de acordes

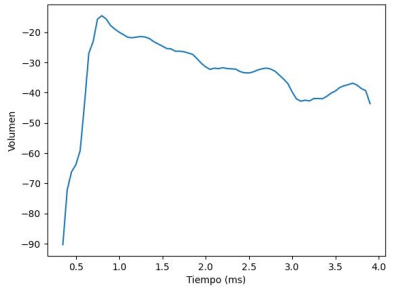
Nos quedamos con el que más peso tiene, pero también observamos los otros que reconoce.

Acorde Predicho: CMajor (C E G); peso: 249

Otros posibles en orden de probabilidad
Cminor (C D# G); peso: 199
Eminor (B E G); peso: 176
Aminor (A C E); peso: 163
GMajor (B D G); peso: 162
Gminor (A# D G); peso: 148

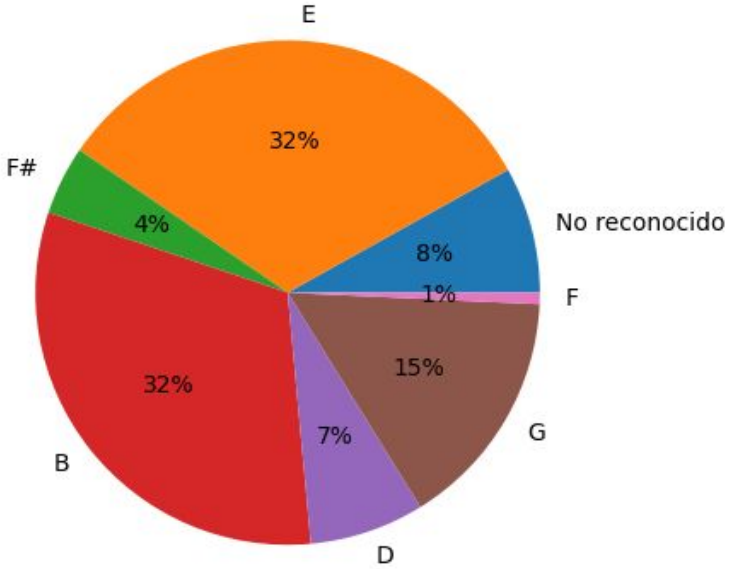


Observamos Mi Menor



Acorde Predicho: Eminor (B E G); peso: 108

Otros posibles en orden de probabilidad
GMajor (B D G); peso: 74
Bminor (B D F#); peso: 59



Conclusiones y trabajo futuro

- Problemas para detectar picos en las melodías. La función find_peaks tiene muchos parámetros para tunear que arreglan algunas cosas pero desarreglan otras.
    - Probamos otro algoritmo parecido al de los acordes con pesos en vez de tomar la heuristica del pico más alto pero no se mejoró mucho.
- Reconocer acordes es muchisimo mas facil por la cantidad de información que se tiene.
- Aprendimos muchas cosas de música.

- Nos queda pendiente poder mejorar el reconocer picos.
- Poder reconocer, en qué escala está una melodía.
- Reconocer los bmp.
- Poder reconocer una melodía cuando hay más de un instrumento pero solo uno está tocando la melodia.

MUCHAS GRACIAS