Tarea 2

CC5213 – Recuperación de Información Multimedia
Profesor: Juan Manuel Barrios
Fecha de Entrega: 18 de octubre de 2023

El objetivo de esta tarea es implementar un **detector de canciones emitidas por radioemisoras**. Dado un conjunto de canciones originales **R** y un conjunto de grabaciones de emisoras de radio **Q**, se desea determinar para cada archivo de audio de **Q** todas las apariciones de las canciones de **R**.

Cada canción es emitida en su totalidad y puede ser emitida una o más veces. Cada emisión puede tener alguna modificación en su calidad, como mayor o menor volumen, distinta calidad de audio o tener ruido de fondo.

Para cada audio $q \in \mathbf{Q}$ la tarea debe producir una lista con detecciones, señalando el segmento de tiempo de q (tiempo de inicio y largo en segundos) y el nombre de la canción de **R** que se escucha en ese segmento.

Para implementar su tarea deberá seguir los siguientes pasos (ver Figura 1): Primero, calcular descriptores por ventanas de todos los audios. Segundo, cada descriptor de audio de **Q** se compara con todos los descriptores de audio de **R** para determinar los más similares (búsqueda k-NN). Finalmente, procesar las listas de similares para localizar secuencias de una misma canción y determinar los segmentos de tiempo con las emisiones de **R**.

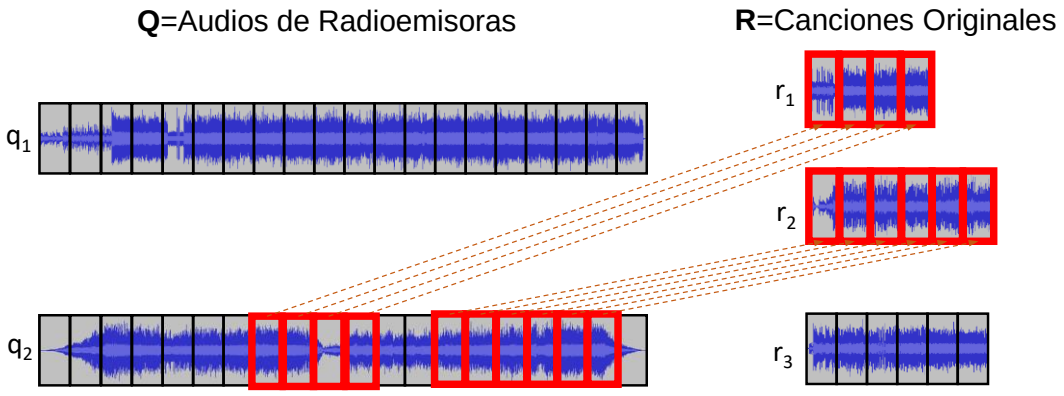

Figura 1: Cada ventana de audio en **Q** se asocia con la ventana más parecida en **R**. Cuando una canción es emitida, las ventanas más parecidas pertenecen a una misma canción y forman una secuencia.

El resultado de la tarea es un archivo de detecciones, listando todas las canciones encontradas. Cada detección se define por un audio $q_i \in \mathbf{Q}$, un segmento de tiempo de q_i (segundo de inicio y largo), nombre de la canción $r_j \in \mathbf{R}$ audible en ese segmento, y un **valor de confianza** numérico donde un valor más alto significa que está más seguro que la detección sea correcta.

Cada archivo de detecciones debe tener formato de **cinco** columnas separadas por un tabulador (`\t`), cada emisión en una línea siguiendo el siguiente formato:

```
archivo_de_q \t tiempo_inicio \t largo \t archivo_de_r \t confianza
```

Los tiempos de inicio y largo se miden en segundos. La Figura 2 muestra un archivo de detecciones de ejemplo.

```
Radio_1.m4a      87.3    30.1    cancion_8.m4a   0.2943
Radio_1.m4a     420.8    15.6    cancion_3.m4a   7.0121
Radio_1.m4a    1892.8    25.9    cancion_2.m4a   5.0315
Radio_1.m4a    2087.5    30.1    cancion_3.m4a   2.3875
```

Figura 2: Ejemplo de archivo de detecciones.

Para su tarea deberá implementar tres comandos que se describen a continuación.

Comando 1: Extracción de características de audio

```
python tarea2-extractor.py [carpeta_audios_entrada]
                           [carpeta_descriptores_salida]
```

Lee todos los archivos de audio (.m4a) que están en la carpeta de entrada, calcula un conjunto de descriptores de audio para cada uno y escribe estos descriptores en uno o más archivos en la carpeta de salida.

Utilice una herramienta como ffmpeg o sox para convertir la pista de audio en formato wav o raw. Utilice el mismo samplerate para todos los audios. Divida cada audio en ventanas de largo fijo y calcule un único vector por ventana. Los descriptores de audio los puede guardar en el formato que estime conveniente. Además, se recomienda incluir un archivo de texto describiendo la ventana de origen de cada descriptor (nombre de archivo y tiempo que representa).

Este comando va a ser llamado dos veces: una vez para **R** y otra vez para **Q**.

Comando 2: Búsqueda por similitud entre ventanas

```
python tarea2-busqueda.py [carpeta_descriptores_radio_Q]
                          [carpeta_descritores_canciones_R]
                          [carpeta_resultados_knn]
```

Lee los descriptores de audios de radio **Q** y los descriptores de canciones **R**, los compara y guarda en la carpeta de salida un archivo con los descriptores similares en **R** para cada ventana de audio de **Q**.

Debe determinar para cada ventana de audio q_i de Q la ventana de audio más parecida r_j de R, según alguna función de distancia (a su elección). Escriba en la carpeta de resultados un archivo con el nombre de cada ventana de **Q**, la ventana de **R** más parecida y opcionalmente la distancia entre ellas. La Figura 3 muestra un ejemplo de lo que podría guardar en la carpeta de resultados.

Ventana de audio de radio (Q)		Ventana de canción más parecida	
Radio_1.m4a	420.3 [s]	cancion_5.m4a	23.7 [s]
Radio_1.m4a	420.8 [s]	cancion_3.m4a	1.2 [s]
Radio_1.m4a	421.3 [s]	cancion_3.m4a	1.7 [s]
Radio_1.m4a	421.8 [s]	cancion_3.m4a	2.2 [s]
Radio_1.m4a	422.3 [s]	cancion_7.m4a	12.7 [s]
Radio_1.m4a	422.8 [s]	cancion_8.m4a	1.2 [s]
Radio_1.m4a	423.3 [s]	cancion_3.m4a	3.7 [s]
Radio_1.m4a	423.8 [s]	cancion_7.m4a	23.2 [s]
Radio_1.m4a	424.3 [s]	cancion_3.m4a	4.7 [s]
Radio_1.m4a	424.8 [s]	cancion_1.m4a	25.2 [s]
Radio_1.m4a	425.3 [s]	cancion_2.m4a	31.3 [s]
Radio_1.m4a	425.8 [s]	cancion_5.m4a	54.1 [s]

Figura 3: Ejemplo del resultado de buscar descriptores similares entre ventanas.

Comando 3: Detección de canciones

```
python tarea2-deteccion.py [carpeta_resultados_knn]
                           [archivo_detecciones]
```

Lee el o los archivos en la carpeta de descriptores similares (generado por el comando anterior), detecta las secuencias provenientes de una misma canción y escribe en el archivo de detecciones de salida todas las canciones encontradas junto con un valor de confianza.

Busque secuencias de vecinos más cercanos que provienen de una misma canción y que mantienen un mismo desfase de tiempo. En el ejemplo de la Figura 3, se puede concluir que existe un segmento de más de tres segundos de `Radio_1.m4a` (entre los segundos 420.8 y 424.3) donde aparece `cancion_3.m4a` (entre los 1.2 y 4.7 segundos) porque en ese segmento los tiempos de las ventanas similares entre ambos audios mantienen una diferencia constante de 419.6 segundos.

Calcule un valor de confianza de la detección. Por ejemplo, podría ser el número de ventanas encontradas, el porcentaje de la canción encontrada, un valor relacionado al promedio de las distancias del más cercano u otro indicador que le parezca relevante.

Datasets de prueba y evaluación

Junto con este enunciado encontrará una implementación base para `tarea2-extractor.py`, `tarea2-busqueda.py`, y `tarea2-deteccion.py`, varios conjuntos de prueba (llamados `dataset_a`, `dataset_b`, `dataset_c`, etc.) que contienen audios de radioemisoras y trozos de canciones a buscar, y un programa de evaluación con la respuesta esperada para cada conjunto de consulta.

Evaluación

El programa de evaluación se inicia con:

```
python evaluarTarea2.py
```

Este programa invoca `tarea2-extractor.py`, `tarea2-busqueda.py` y `tarea1-deteccion.py` sobre cada dataset de prueba, lee los archivos de resultados generados, los compara con la respuesta esperada, determina la distancia umbral de decisión que logra un mejor balance entre respuestas correctas y erróneas, y finalmente, dependiendo de la cantidad de detecciones correctas, señala la nota obtenida.

Su tarea será evaluada en los datasets publicados y en otros datasets similares. Su tarea no puede demorar más de 30 minutos en procesar cada dataset de prueba.

Existe la posibilidad de obtener **hasta 1 punto extra de bonus** para otras tareas si logra detectar correctamente todas las canciones en los datasets de evaluación.

Entrega

El plazo máximo de entrega es el **martes 18 de octubre** hasta las 23:59 por U-Cursos. La tarea la puede implementar en **Python 3** usando cualquier función de NumPy, SciPy, LibRosa y otras librerías gratuitas. Debe subir sólo el código fuente de su tarea (archivos .py). No envíe datasets ni descriptores ya calculados.

Se recomienda incluir un archivo de texto señalando el sistema operativo en que realizó su tarea e incluir la salida que entregó `evaluarTarea2.py` en su computador, para poder evaluarla en un ambiente similar.

La tarea es *individual*. En caso de detectar copia entre estudiantes o plagio de Internet se asignará nota 1.0 a los involucrados.