Instituto Tecnológico de Costa Rica, Escuela de Computación, Análisis de Algoritmos
Prof. Rodrigo Núñez, Proyecto I – 30%

Mutant Music

El Mutant Music es una app que realiza operaciones y transformaciones a música previamente grabada en formato Wav, a 44.1Khz y 16 bit Depth en 2 canales, es decir, estéreo.

MutantMusicApp recibe por parámetro un código de operación y el nombre de dos archivos wav que llamaremos s1 y s2 a lo largo de este documento. S1 es un audio de una canción comercial de entre 3 a 4 minutos de duración, s2 es otro audio de 3 a 7 segundos. Entonces por ejemplo se puede ejecutar así:
node mutantmusicapp mt "calma.wav" "mitest.wav"

Las operaciones que puede realizar mutantmusicapp son:

Match (mt), la operación match, cuyo código parámetro es mt, busca si ***aparentemente*** s2 aparece en s1, mostrando una lista de cada segundo donde inicia cada aparición. Adicionalmente, genera un nuevo `$s1_mt.wav` el cual corresponde a una canción que contiene únicamente las apariciones en s1 de s2 concatenadas. Para probar mt, sométalo con un s2 que cumple:

- Un sample extraído de s1
- El mismo sample pasado por un LowPassFilter, LPF a 12dB
- El mismo sample con un cambio de frecuencia +-35%
- Un sample extraído de otra canción diferente

Unmatch (umt), realiza prácticamente la misma operación que match, pero la canción resultado `$s1_umt.wav`, contiene la misma canción, pero eliminando las apariciones encontradas de s2 en s1.

Dj (dj), esta operación realiza un mix nuevo de la canción s1, el parámetro s2 es ignorado. Para hacer el mix, la app extrae un ranking de las 10 tonadas/compases/secciones no repetidas de 1 a 2 segundos más dominantes de la canción, donde se entiende como dominante aquello que se usa o repite más veces en la canción. Una vez que se tengan las 10 tonadas, se procede a disponerlas en un mix de mínimo 1 minuto que contenga loops (de 4 a 7 segundos), movimiento left to right to both (4 segundos), ritmo alternando sonido y silencio (6 a 10 segundos). Los efectos y orden del mix pueden hacerse usando una matriz o mapa estático de los efectos y simplemente asignar las tonadas aleatoriamente al mapa.

Compose (cmp), componer hace uso de s1 y realiza una transformación de la canción creando una nueva canción `$s1_cmp.wav` que se comporta como s2 (envolvente).

Detalles de implementación

- Match y unmatch deben hacerse por medio de un algoritmo probabilista monte carlo de respuesta correcta siempre, donde la solución determinista no aceptable, es la de comprobar todas las posibles apariciones de s2 en s1 en todo lo largo de la canción. El problema en cuestión es encontrar los matches en la canción no necesariamente del tamaño exacto de s2 pero si de al menos una buena porción.
- Dj, debe hacerse por medio de un algoritmo probabilista con posibilidad de respuesta incorrecta, cuya solución determinista no aceptable es la búsqueda de aquellas secuencias de música que más se repiten en la canción. El mix resultado si determinista.
- Compose, genera la nueva composición basado en un algoritmo genético. Para ello debe diseñar K genotipos basado en toda la canción, y esos tipos son determinados por las transiciones ondulatorias de la canción de -1 a 1, por ejemplo, picos, valles, curvaturas, inclinaciones, subidas, planos elevados, silencios.

  Se extraen todos los cromosomas de la canción, asignándole su tipo y valor; cromosomas de 16 bits que permitan a la vez mapearse a samples de audio. Los cromosomas y el comportamiento de la canción dictado por s2 son cargados a ElasticSearch.

  Para ejecutar el GA, en el proceso de evaluación de la función de fitness, hace un query a ES de tal forma que retorne los "factores multiplicativos" que se deben usar para guiar la hacia la canción destino, la cual debe acomodarse tanto en la

línea de tiempo como en la forma dada por el envolvente de s2.

Se ejecuta las operaciones genéticas, generando una nueva población, se envía a borrar el índice anterior y se crea uno nuevo para la nueva población, y se repiten los pasos. El algoritmo genético continúa haciendo nuevas generaciones hasta que la canción quede completamente armada.

- El estudiante estructura, analiza, procesa y/o, agrupa la información de la canción a su conveniencia.

Otros aspectos

- El programa deberá hacerse en TypeScript
- Se recomienda el uso de Vscode
- Como estándar de programación deberá seguir los extends de tslint: "tslint-config-standard", "tslint:recommended", "tslint-config-airbnb"
- El proyecto puede hacerse en parejas
- Como puede ver, las soluciones algorítmicas tienen un fuerte contenido creativo, de diseño experimental, analítico, estratégicos, entre otros, es decir, hay múltiples soluciones. Se evaluará la claridad y simplicidad de las estregias algorítmicas usadas, sabiendo que queremos estar de ser posible siempre lejos de cualquier operación $O(n^2)$ o peores.
- El estudiante debe buscar en las soluciones programar poco y hacer mucho. Reutilizar código es fundamental, como también un carismático diseño de objetos, estructuras de datos y algoritmos, aspectos que tendrán un alto peso en la revisión.
- El código fuente deberá mantenerse en github sin excepción y se revisarán los commits y merges de cada estudiante. Se espera mínimo los siguientes branches: master, develop, ***ana, pedro***. Master será la versión oficial para revisión. Develop será el Branch donde hacer merge y pueden hechar a perder, debajo de develop estarían los branches personales de cada uno.
- La revisión será con cita el día 10 y 11 de mayo
- Se atenderán consultas hasta el 3 de mayo.
- Cualquier sospecha de copia anulará el trabajo.