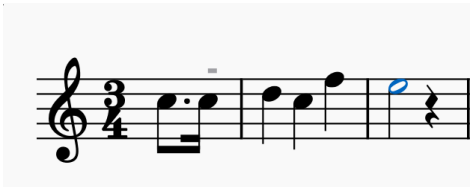
LSH 2021 1C 1

Enunciado

Voila es una aplicación que detecta la canción que suena en el momento. El sistema ya cuenta con un componente que dado un archivo de audio extrae la melodía y la codifica en un vector de longitud variable. Cada componente de este vector representa una semicorchea y el valor indica si hay un silencio (0) o una de las 12 notas posibles. Así, el *do* está codificado como 1, el *do sostenido* como 2, el *la sostenido* como 11 y finalmente el *si* como 12.

Así, por ejemplo, la melodía



se guarda como `[1,1,1,3,1,1,1,1,6,6,6,6,5,5,5,5,5,5,5,5,0,0,0,0]`.

El problema entonces es, dado un vector de petición (query), encontrar los vectores semejantes. Se cuenta con 70 millones de canciones, y el sistema recibe cientos de miles de peticiones cada minuto. Por esto, se decide resolver el proceso de búsqueda a través de LSH. Hay que tener en cuenta que las melodías (es decir, los vectores) tienen longitudes distintas, y generalmente los vectores de petición son más cortos (no son toda la canción, son un fragmento).

Le pedimos:

1) Definir qué métrica utilizar y por qué. Explicar cómo calcula su minhash a partir de un vector dado, los parámetros que tiene, y si estos parámetros tienen alguna restricción de valores dadas las condiciones del problema. (20pts)
2) Hallar cuántos minhashes se deben utilizar y en qué esquema si queremos que en 1000 peticiones LSH no devuelva más de 160 melodías no relevantes, y que la probabilidad de encontrar una melodía parecida sea igual o mayor a 88%. Una melodía parecida es a partir de 0.8 de semejanza, y una melodía no relevante es la que tiene una semejanza menor a 0.2. (15pts)
3) Describir cómo es el preprocesamiento de los vectores utilizando los minhashes descritos en 1) y el esquema de minhashes propuesto en 2), hasta guardar las referencias en la única tabla final. (35pts)
4) Describir cómo se resuelve una consulta y cómo se le devuelve al usuario una única canción relevante. (30pts)

1) En clase vimos tres métricas distintas: la métrica Coseno, la métrica Euclidea y le métrica de Jaccard. Propongo utilizar la métrica de Jaccard, ya que podemos interpretar a las melodías como un conjunto de notas, en donde la semejanza entre dos melodías, viene dada por la cantidad de notas que tienen en común sobre el total de notas que hay en estas dos melodías. Otro indicio que nos indica que debemos utilizar la métrica Jaccard es que contamos con vectores de longitud variable.

   En la teoría vimos que para obtener los minhash para la métrica de Jaccard debíamos generar una tabla de shingles, en donde aparece un 1 o 0 si el shingle se encuentra en el elemento, luego se permutaba aleatoriamente la tabla y nos quedábamos con la posición donde aparece el primer 1. Esto es muy poco eficiente en la práctica, ya que ocupa mucho espacio crear una tabla para cada conjunto. Por lo que vimos, que en la práctica es equivalente y mucho más eficiente hashear todos los shingles de cada melodía, con una familia de funciones de hashing de longitud fija, y quedarnos únicamente con el mínimo de cada función.

   Los minhashes para la métrica en cuestión son de la forma :

   $$mh(melodia) = \{ h(s) \;.\; s \in Shingles(melodia) \}$$

   Siendo h una función de hash de una familia de funciones universales de longitud fija, de la forma:

   $$h(x) = (a*x1 + \ldots + n*xn) \bmod p \bmod m$$

   Las restricciones para esta función son:
   - Los valores a, … n deben ser enteros positivos mayores a 0 y menores a p
   - m es un número real positivo
   - p es un número primo mayor o igual a m

   Podemos notar que en este caso contamos con un abecedario de 13 caracteres, entendiendo a cada una de las doce notas y al silencio como un carácter del abecedario. Como vimos en clase, la cantidad posible de shingles para este caso seria 13^n siendo n la longitud de los shingles. Como cota mínima sabemos que la cantidad posible de shingles debe ser mayor a la cantidad de entradas. En este caso contamos con 70 millones de melodías, para obtener una cantidad de shingles mayor, nos alcanza con utilizar un n = 8, ya que $13^8 > 70$ millones.

2) Sea:

- p1 la probabilidad de colisión de dos objetos similares,
- p2 la probabilidad de colisión de dos objetos distintos,
- d1 la distancia máxima para considerar a dos objetos como similares
- d2 como la distancia mínima para considerar dos elementos como disimiles.

Del enunciado se deduce:

p2 = 160 / 1000 = 0.16

p1 = 0.88

d1 = 1 – 0.8 = 0.2

d2 = 1- 0.2 = 0.8

Haciendo un Gridsearch y utilizando las ecuaciones:

p1 = (1 - (1 - (1 –d1)**r)**b)

p2 = (1 - (1 - (1 –d2)**r)**b)

Para este punto programe un gridsearch pero en el rango buscado no encontré un b y un r que satisfagan esas condiciones, por cuestiones de tiempo no seguiré buscando, pero debería seguirse buscando en la grilla y probar con distintos valores de b y r hasta obtener las probabilidades requeridas. Una vez obtenidos estos valores por tabla, la cantidad total de minhashes a utilizar es b * r.

3)

Para realizar este punto vamos a suponer valores de b y r aleatorios, propongo b = 2 y r = 2, esto quiere decir que haremos dos grupos de OR de minhashes y luego haremos un AND de estos dos grupos para hashear a la tabla final. Esto se hace con el objetivo de reducir la tasa de falsos positivos y falsos negativos en nuestra tabla de hash final.

El preprocesamiento de los datos consistiría en recorrer las 70 millones de vectores de melodías, a cada vector aplicar las 4 funciones de hash de longitud fija (como definimos b = 2 y r = 2 en total usaremos 4 minhashes), y quedarnos con el mínimo de cada una de estas funciones, con lo que obtendríamos los 4 minhash.

Como vimos en el punto a, abria que definir 4 funciones de hash de longitud fija para los minhashes. Por ejemplo

Mh1 = (1 * melodía[0] + 2 * melodía[1] + … + 7 * melodía[7]) %mod p %mod m

Entendiendo a melodía[n] como el n-esimo shingle del documento

Tendriamos que definir 3 mas variando los coeficientes que mutliplican los shingles y conservando p y m.

Una vez obtenidos los b grupos de r minhashes, debemos definir b funciones (para aplicársela a los b grupos) de hash pertenecientes a una familia de funciones de longitud fija del estilo:

$$(a*mha + b*mhb) \bmod p \bmod m$$

Teniendo los valores las mismas restricciones descriptas en el punto a, y siendo mha y mhb los r minhashes pertenecientes a un grupo.

Lo que hacen estas funciones es hashear nuestra entrada b veces en la tabla de hash final, la cual utilizaremos para hacer consultas.

Con esto en nuestra tabla de hash final, obtendríamos b veces cada una de las 70 millones de melodías. Cabe aclarar que un buen numero para m, que es el tamaño final de nuestra tabla de hash seria 2*70millones*b, para reducir el factor de carga de nuestra tabla,

4) Por ultimo, para realizar una consulta de un query, debemos aplicarle las b*r funciones de hashing de longitud fija a los shingles de la nueva melodía y quedarnos con el mínimo de cada función, luego a los b grupos de r funciones de minhash aplicarle las funciones de hash universal de longitud fija, lo cual coloca en nuestra tabla de hash final a la melodía query b veces. Por ultimo en la tabla de hash final, nos fijamos en las b posiciones en las que cayo nuestra nueva melodía, y las unión de todas las melodías con las que comparte buckets nuestra nueva melodía serán sus semejantes. Una vez obtenidos sus semejantes podemos utilizar la semenjaza de jaccard para saber cuál de todas las melodias es la más similar a nuestra melodía query y devolver la de mayor semejanza.