**Order-lst-vectors**

Función auxiliar que hemos creado para que pasándole la lista de vectores ordenados inserte un nuevo vector correctamente

**Pseudocódigo**

**Entrada:** vector-ref: vector que representa a una categoria, representado como una lista

vector-insert: vector a insertar

ord-lst-of-vectors: nuevo vector de vectores ordenados

**Salida:** la nueva lista ordenada con el vector insertado

**Procesamiento:**

Si la lista de vectores ordenados es vacía

Insertamos el vector a insertar en la lista

Si la distancia coseno del vector es menor que la del primer elemento de la lista ordenada

Devolvemos la lista ordenada con el vector como primer elemento

En otro caso

Devolvemos la lista con el primer elemento de la lista ordenada y la llamada recursiva de order-lst-vectors aplicada al vector y al *rest*  de la lista ordenada

**Código**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; order-lst-vectors

(defun order-lst-vectors(vector-ref vector-insert ord-lst-of-vectors)

(cond ((null ord-lst-of-vectors) (cons vector-insert ord-lst-of-vectors))

((< (cosine-distance-mapcar vector-ref vector-insert)

(cosine-distance-mapcar vector-ref (first ord-lst-of-vectors)))

(cons vector-insert ord-lst-of-vectors))

(t (cons (first ord-lst-of-vectors)

(order-lst-vectors vector-ref vector-insert (rest ord-lst-of-vectors))))))

**Comentarios**

En este caso hemos utilizado recursividad para ir insertando elementos de manera ordenada a una lista.

**Order-vectors-cosine-distance**

**Pseudocódigo**

**Entrada:** vector: vector que representa a una categoria, representado como una lista

lst-of-vectors vector de vectores

confidence-level: Nivel de confianza (parametro opcional)

**Salida:** Vectores cuya semejanza con respecto a la categoría es superior al nivel de confianza

**Procesamiento:**

Si 1 – nivel de confianza >= distancia del primero de la lista

Si *rest* == null

Insertamos el primer elemento en una lista vacia (con order-lst-vectors)

En otro caso

Recursión de order-lst-vectors para el resto de elementos

En otro caso

Si rest == null

Nil

En otro caso

Recursión de la función para aplicarla al resto de los vectores que queremos comparar con el vector referencia

**Código**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; order-vectors-cosine-distance

;;; Devuelve aquellos vectores similares a una categoria

(defun order-vectors-cosine-distance (vector lst-of-vectors &optional (confidence-level 0))

(if (>= (- 1 confidence-level)

(cosine-distance-mapcar vector (first lst-of-vectors)))

(if (null (rest lst-of-vectors))

(order-lst-vectors vector (first lst-of-vectors) '())

(order-lst-vectors vector (first lst-of-vectors) (order-vectors-cosine-distance

vector (rest lst-of-vectors) confidence-level)))

(if (null (rest lst-of-vectors))

nil

(order-vectors-cosine-distance vector (rest lst-of-vectors) confidence-level))))

**Comentarios**

Hemos utilizado la función auxiliar para ir insertando (si cumple la condición) uno a uno en la lista y de manera ordenada, aplicando recursividad sobre la propia función.

**Ejemplos**

**Get-text-category**

Función auxiliar que a partir de un text y una lista de categorías devuelve el id de la categoría que mejor lo aproxima y la distancia a ella.

**Pseudocódigo**

**Entrada:**

**Salida:**

**Procesamiento:**

**Código**

**Comentarios**

**Get-vectors category**

**Pseudocódigo**

**Entrada:**

**Salida:**

**Procesamiento:**

**Código**

**Comentarios**

**Ejemplos**