CC5213-1 Recuperación de Información Multimedia
Profesor: Juan Manuel Barrios
Estudiante: Andrés Calderón Guardia

Mini-Control 3

Diciembre 11, 2023

P1. TF-IDF

Usando logaritmo en base 10 y dado que el vocabulario relevante de la consulta son solo las palabras "una", "nueva" y "constitución", se tiene:

a. IDF: Calculamos el IDF para cada término.
- $\text{idf}_1 = \log(\frac{10000}{1000}) = \log(10) = 1$ (casa)
- $\text{idf}_2 = \log(\frac{10000}{1000}) = \log(10) = 1$ (compré)
- $\text{idf}_3 = \log(\frac{10000}{100}) = \log(100) = 2$ (constitución)
- $\text{idf}_4 = \log(\frac{10000}{1000}) = \log(10) = 1$ (en)
- $\text{idf}_5 = \log(\frac{10000}{100}) = \log(100) = 2$ (mi)
- $\text{idf}_6 = \log(\frac{10000}{100}) = \log(100) = 2$ (no)
- $\text{idf}_7 = \log(\frac{10000}{10}) = \log(1000) = 3$ (nueva)
- $\text{idf}_6 = \log(\frac{10000}{100}) = \log(100) = 2$ (quieren)
- $\text{idf}_9 = \log(\frac{10000}{10}) = \log(1000) = 3$ (una)

b. TF-IDF normalizado: Para calcular este vector necesitamos obtener el vector TF para cada caso.
- $\text{tf}_1 = (1, 1, 1, 1, 0, 0, 1, 0, 1)$
- $\text{tf}_2 = (1, 0, 1, 1, 1, 1, 1, 1, 1)$
- $\text{tf}_Q = (0, 0, 1, 0, 0, 0, 1, 0, 1)$

Con esto obtenemos el vector TF-IDF para cada caso.
- $\text{tf-idf'}_1 = (1 \cdot 1, 1 \cdot 1, 1 \cdot 2, 1 \cdot 1, 0, 0, 1 \cdot 3, 0, 1 \cdot 3) = (1, 1, 2, 1, 0, 0, 3, 0, 3)$
- $\text{tf-idf'}_2 = (1 \cdot 1, 0, 1 \cdot 2, 1 \cdot 1, 1 \cdot 2, 1 \cdot 2, 1 \cdot 3, 1 \cdot 2, 1 \cdot 3) = (1, 0, 2, 1, 2, 2, 3, 2, 3)$
- $\text{tf-idf'}_Q = (0, 0, 1 \cdot 2, 0, 0, 0, 1 \cdot 3, 0, 1 \cdot 3) = (0, 0, 2, 0, 0, 0, 3, 0, 3)$

Obtenemos la magnitud de cada vector:
- $\|\text{tf-idf'}_1\| = \sqrt{1 + 1 + 4 + 1 + 9 + 9} = \sqrt{25} = 5$
- $\|\text{tf-idf'}_2\| = \sqrt{1 + 4 + 1 + 4 + 4 + 9 + 4 + 9} = \sqrt{36} = 6$
- $\|\text{tf-idf'}_Q\| = \sqrt{4 + 9 + 9} = \sqrt{22}$

Y finalmente normalizamos:
- $\text{tf-idf}_1 = (\frac{1}{5}, \frac{1}{5}, \frac{2}{5}, \frac{1}{5}, 0, 0, \frac{3}{5}, 0, \frac{3}{5})$
- $\text{tf-idf}_2 = (\frac{1}{6}, 0, \frac{1}{3}, \frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{2}, \frac{1}{3}, \frac{1}{2})$
- $\text{tf-idf}_Q = \left(0, 0, \frac{2}{\sqrt{22}}, 0, 0, 0, \frac{3}{\sqrt{22}}, 0, \frac{3}{\sqrt{22}}\right)$

c. Similitud coseno: Calculamos esta similitud para cada documento con respecto a la consulta.
- $\text{sim}(Q, D_1) = \frac{2}{5} \cdot \frac{2}{\sqrt{22}} + \frac{3}{5} \cdot \frac{3}{\sqrt{22}} + \frac{3}{5} \cdot \frac{3}{\sqrt{22}} = \frac{\sqrt{22}}{5} \approx 0.938$
- $\text{sim}(Q, D_2) = \frac{1}{3} \cdot \frac{2}{\sqrt{22}} + \frac{1}{2} \cdot \frac{3}{\sqrt{22}} + \frac{1}{2} \cdot \frac{3}{\sqrt{22}} = \frac{\sqrt{22}}{6} \approx 0.782$

De esto concluimos que el documento más similar a la consulta Q es D1.

P2. R-tree

a. Procedimiento del algoritmo:

- Variables iniciales:
  - pruningdist = $+\infty$
  - APL = min-heap vacío (ordenando por MINDIST)
  - candidatos = max-heap vacío (ordenando por distancia a **q**)

1. APL: [R1, R3, R2] (3 cálculos de MINDIST)
   - visitar R1: insertar R11, R12, R13
2. APL: [R13, R3, R2, R12, R11] (3 cálculos de distancia a **q**)
   - visitar R13: comparar **B**, **J**, **F**
     - candidatos: [**B**]
     - candidatos: [**B**, **J**] (nuevo pruningdist con el tope de la lista de candidatos **B**)
     - candidatos: [**F**, **J**] (nuevo pruningdist **F**)
3. APL: [R3, R2, R12, R11] (4 cálculos de MINDIST)
   - visitar R3: insertar R31, R32, no insertar R33, R34 (MINDIST > pruningdist)
4. APL: [R31, R2, R32, R12, R11] (2 cálculos de distancia a **q**)
   - visitar R31: comparar **N**, **T**
     - candidatos: [**J**, **N**] (nuevo pruningdist **J**)
     - T no es candidato (d(q, T) > pruningdist)
5. APL: [R2, R32, R12, R11] (3 cálculos de MINDIST)
   - visitar R2: insertar R21, no insertar R22, R23
6. APL: [R21, R32, R12, R11] (2 cálculos de distancia a **q**)
   - visitar R21: comparar **C**, **K**
     - C no es candidato
     - candidatos: [**N**, **K**] (nuevo pruningdist **N**)
7. APL: [R32, R12, R11] (1 cálculo de MINDIST)
   - no visitar R32 (MINDIST > pruningdist)
8. No se visitó un elemento de APL así que el resto de sus elementos ya no pueden tener elementos más cercanos, se termina el algoritmo.

b. N° de cálculos de distancia:

Sumando los cálculos de distancia mencionados durante el procedimiento del algoritmo obtenemos que se realizaron 18 cálculos.

c. N° de cálculos de distancia para linear scan:

Para este caso se realizarían 21 cálculos de distancia pues este algoritmo siempre tiene que recorrer todos los elementos del espacio para obtener los k vecinos más cercanos, de modo que calcularía la distancia de todos los puntos a **q**.

P3. k-d tree

a. $c = 1$:

Por MINDIST se visita primero el rectángulo que contiene a **q**, luego se agregaría a **G** como vecino más cercano, su distancia sería el nuevo pruningdist y como ya se visitó una región el algoritmo termina.

b. $c = 2$:

Siguiendo donde termina el caso anterior ahora se visitaría el rectángulo que contiene a **B** pues es el segundo más cercano a **q** y dentro del pruningdist, en este caso se realiza la comparación y este vector está más cerca a **q** que **G**, por lo que es el nuevo vecino más cercano y se actualiza el pruningdist acorde a ello, ya se visitaron 2 regiones y termina.

c. $c = 3$:

Ahora la siguiente región más cercana por MINDIST y dentro del pruningdist es la que contiene al vector **L**, se compara la distancia y en este caso es el nuevo vecino más cercano, se actualiza pruningdist y se termina el algoritmo.

d. $c = 4$:

Ahora se visita la siguiente región más cercana dentro del pruningdist, la cual sería la que contiene los vectores **J**, **I** y **H**, se comparan las distancias y en este caso ninguno de estos 3 vectores posee una distancia a **q** menor que con **L**, pruningdist se mantiene y el algoritmo termina.

e. $c = 5$:

Finalmente en este caso accedemos a la siguiente región más cercana dentro del pruningdist que sería aquella que contiene los vectores **C** y **D**, se comparan las distancias, D no es más cercano a **q** pero en cambio **C** si lo es, entonces se actualiza el nuevo vecino más cercano a este vector, el pruningdist a la distancia entre ambos y el algoritmo termina (notemos que en este caso se logró encontrar el vecino más cercano a **q** real, de modo que aunque se aumentase al valor de c ya no afectaría al resultado a partir de este punto).

P4. PCA

a. Matriz de transformación **W**:

Primero reordenamos los valores propios de mayor a menor:

$$\lambda_2 \geq \lambda_3 \geq \lambda_4 \geq \lambda_1$$

Con esto creamos **W'**:

$$W' = [\ v_2 \mid v_3 \mid v_4 \mid v_1\ ]$$

Para obtener **W** nos quedamos con tan solo la primera columna pues así esta matriz transformará **S** en **T** con una sola dimensión perdiendo la menor cantidad de información posible:

$$W = [v_2] = \begin{pmatrix} 0.2 \\ 0 \\ -0.6 \\ 0.8 \end{pmatrix}$$

b. Cantidad de información que se mantiene:

Usando la fórmula con los valores propios ordenados de mayor a menor se tiene:

$$p = \frac{\sum_{i=1}^{d'} \lambda_i}{\sum_{i=1}^{4} \lambda_i} = \frac{31.4}{31.4 + 23.9 + 16.8 + 8.5} = \frac{31.4}{80.6} \approx 0.39 = 39\%$$

Para PCA se considera la varianza de una dimensión como su cantidad de información para esa misma dimensión, de modo que lo que se logra retener es la varianza que tenemos dividido por la varianza total, de modo que obtuvimos un retenimiento del 39% de la información original.

c. Vector más cercano a **q**:

Primero obtenemos el vector q' correspondiente a haberlo reducido a una dimensión, por lo que para ello tenemos que realizar la operación:

$$q' = W^T \cdot q = (0.2\ \ 0\ \ -0.6\ \ 0.8) \cdot \begin{pmatrix} 9 \\ 12 \\ 5 \\ 17 \end{pmatrix} = 0.2 \cdot 9 + 0 \cdot 12 - 0.6 \cdot 5 + 0.8 \cdot 17 = 12.4$$

Con esto calculamos a cual punto es más cercano entre los 9 proyectados por PCA usando distancia Euclideana, lo cual nos entrega el vector **h**.

P5. Índices Métricos

a. Tabla de pivotes:

p	0	10	3	6	11	6	3	11	4	10
	a	b	c	d	e	f	g	h	q_1	q_2

b. Búsqueda del vecino más cercano:

1. q_1 (recorriendo en el mismo orden que en la tabla de pivotes):

cota inferior LB(q_1, u)	distancia real d(q_1, u)	candidato NN
4	4	a
6	-	-
1	3	c
2	10	c
7	-	-
2	4	c
1	7	c
7	-	-

Por ende el vecino más cercano a q_1 es c.

2. q_2:

cota inferior LB(q_2, u)	distancia real d(q_2, u)	candidato NN
10	10	a
0	2	b
7	-	-
4	-	-
1	1	e
4	-	-
7	-	-
1	7	e

Con esto se tiene que el vecino más cercano a q_2 es e.

c. Complejidad interna y externa:

Agrupando ambas búsquedas tenemos que la complejidad externa fue de 9, pues se calcularon distancias 9 veces en total, y la complejidad interna fue de 16 por parte del cálculo de las cotas inferiores, y 2 más por el cálculo de las distancias de q_1 y q_2 al pivote a. Para saber si es más eficiente que un linear scan o no calcularemos la cantidad de operaciones que se realizan en ambos casos.

1. Linear scan: Con este algoritmo se habrían realizado 16 calculos de distancias, cada uno de estos cálculos necesita realizar 2 operaciones, pues se está usando distancia Manhattan, lo que nos da un total de 32 operaciones.
2. Índice: Para cada cota inferior se realiza solo una operación, y cada distancia son 2 operaciones, lo que nos da un total de 38 operaciones (11 distancias y 16 cotas).

Concluimos que el linear scan habría sido más eficiente que el índice en este caso pues se realizan menos cálculos.