Almacenamiento de los términos (léxico)

- Términos de Longitud Fija

Término	Frec.	Punteros
casa	20	xxx
casado	5	xxx
cascada	3	xxx
cascote	1	xxx
casita	2	xxx

- Concatenación de Términos
  - STRINGS : casacasadocascadacascotecasita

Offset al Término	Frec.	Punteros
0	20	xxx
4	5	xxx
10	3	xxx
17	1	xxx
24	2	xxx

Almacenamiento de los términos (léxico)

- Concatenación de Términos Mejorada
  - Guardamos un offset de cada N
  - STRINGS : 4casa6casado7cascada7cascote6casita

Frec.	Punteros
20	xxx
5	xxx
3	xxx
1	xxx
2	xxx

Offset al Término
0
12
29

Almacenamiento de los términos (léxico)

- Front Coding
  - Léxico : casa, casado, cascada, cascote, casita

Repetidos	Distintos	Chars	Frec.	Punteros
0	4	casa	20	xxx
4	2	do	5	xxx
3	4	cada	3	xxx
4	3	ote	1	xxx
3	3	ita	2	xxx

- Front Coding Parcial
  - Cada N términos, ignoramos las coincidencias

Almacenamiento de los términos (léxico)

- Sin guardar los términos - Hashing Perfecto

Frec.	Punteros
20	xxx
5	xxx
3	xxx
1	xxx
2	xxx

  - Necesitamos una función de hashing que sea :
    - Perfecta : No produce colisiones
    - Mínima : Genera todos los valores posible del espacio de direcciones
    - Preservadora del Orden : sii

      $S1 < S2 \iff h(S1) < h(S2)$

Almacenamiento de los términos (léxico)

- Hashing Perfecto (Construcción)
  - Construimos 2 funciones de hashing comunes con un espacio de direcciones M mayor a la cantidad de términos (N)
  - Definimos una función fija G que se aplicará al resultado de h1 y h2.
  - Definimos la función de hashing :

    $h(s) = (g(h1(s))+g(h2(s))) \bmod N$

Almacenamiento de los términos (léxico)

- Hashing Perfecto (Construcción - 1 de 2)
  - Para construir la función G se debe generar un gráfo donde cada uno de los valores posibles devueltos por las funciones h1 y h2 es un vértice y por cada string del léxico generamos una arista uniendo los vértices correspondientes a los valores que devolvieron h1 y h2 para dicho string.
  - El gráfico generado debe ser acíclico. Si no lo es debemos cambiar las funciones de hashing h1 y/o h2

Almacenamiento de los términos (léxico)

- Hashing Perfecto (Construcción - 2 de 2)
  - Mientras no queden vértices sin rotular :
    - Tomamos cualquier vértice no rotulado y lo rotulamos con 0 (cero)
    - A todos los vértices conectados a él le asignamos el valor absoluto de la diferencia entre el vértice desde el que venimos y la arista recorrida.
    - A todos los vértices conectados a los que acabamos de rotular los rotulamos de la misma forma, etc.
  - Los rótulos de los vértices nos dan los valores de la función G para cada valor posible devueltos por h1 y h2.

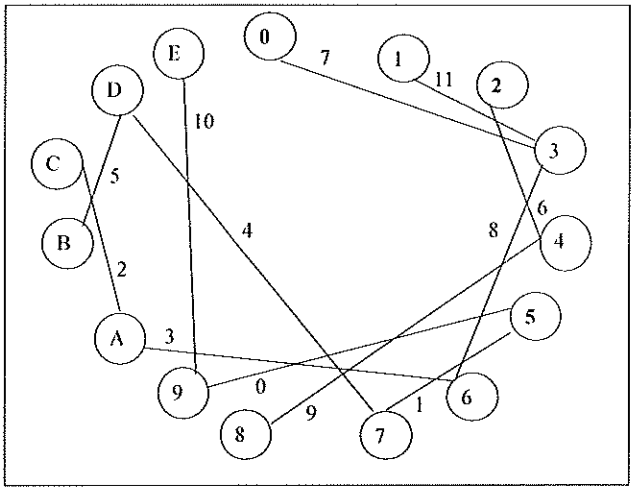
Almacenamiento de los términos (léxico)

- Hashing Perfecto (Ejemplo - 1 de 5)

Términos	h1(s)	h2(s)
string1	5	9
string2	5	7
string3	10	12
string4	6	10
string5	13	7
string6	13	11
string7	4	2
string8	0	3
string9	6	3
string10	8	4
string11	9	14
string12	3	1

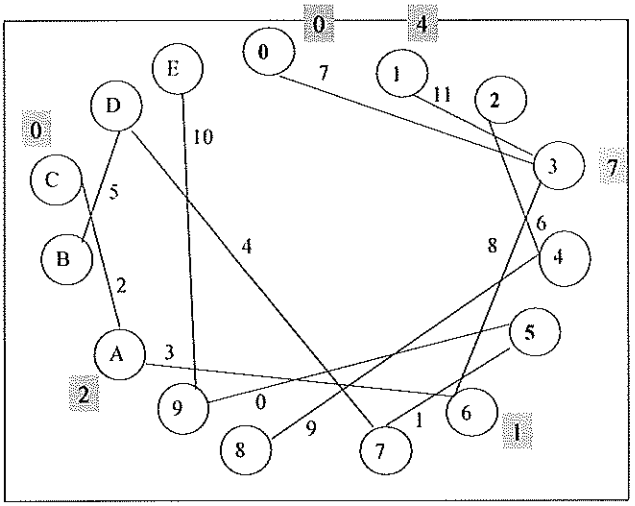
Almacenamiento de los términos (léxico)

- Hashing Perfecto (Ejemplo - 2 de 5)



Almacenamiento de los términos (léxico)

- Hashing Perfecto (Ejemplo - 3 de 5)



Almacenamiento de los términos (léxico)

- Hashing Perfecto (Ejemplo - 4 de 5)

x	g(x)
0	0
1	4
2	0
3	7
4	6
5	0
6	1
7	1
8	3
9	0
10	2
11	2
12	0
13	3
14	10

Almacenamiento de los términos (léxico)

- Hashing Perfecto (Ejemplo - 5 de 5)

Términos	h1(s)	h2(s)	g(h1(s))	g(h2(s))	h(s)
string1	5	9	0	0	0
string2	5	7	0	1	1
string3	10	12	2	0	2
string4	6	10	1	2	3
string5	13	7	3	1	4
string6	13	11	3	2	5
string7	4	2	6	0	6
string8	0	3	0	7	7
string9	6	3	1	7	8
string10	8	4	3	6	9
string11	9	14	0	10	10
string12	3	1	7	4	11

Construcción del Índice

- Inversión de Matrices :

Pedro y Pablo.
Pedro corre.
Pablo respira.
Pedro corre y respira.
Pedro corre Pedro.

	Pedro	y	Pablo	corre	respira
1	1	1	1	0	0
2	1	0	0	1	0
3	0	0	1	0	1
4	1	1	0	1	1
5	2	0	0	1	0

Transponiendo

	1	2	3	4	5
Pedro	1	1	0	1	2
y	1	0	0	1	0
Pablo	1	0	1	0	0
corre	0	1	0	1	1
Respira	0	0	1	1	0

Construcción del Índice

- Matriz en memoria :

Para la Biblia	Para TREC
• 1 Gb de memoria	• 1.4 Tb de memoria
o	o
• 1 mes	• 127 años

- Lista enlazada (solución del estudiante) :

Para la Biblia	Para TREC
• 4 horas	• 6 semanas

Construcción del Índice

- Inversión por Sort :
  - Se lee el documento. Los términos se agregan al léxico y se genera un archivo auxiliar con el siguiente formato :
    - N° de término
    - N° de Documento
    - Frecuencia del término en el documento
  - Se ordena el archivo auxiliar por n° de término y n° de documento. Se recomienda Replacement Selection y Optimal Merge

Construcción del Índice

- Inversión por Sort :
  - Se construye el índice recorriendo el archivo auxiliar de la siguiente manera :
    - Leo el primer registro
    - Mientras haya registros
      - Guardo el término y pongo el cero la frecuencia total
      - Mientras sea el mismo término
        - » agrego el documento a la lista de punteros
        - » sumo la frecuencia a la frecuencia total
        - » leo el próximo registro
      - Grabo el registro en el índice

Construcción del Índice

- Inversión por Sort - Ejemplo (1 de 3) :
  - Dados los siguientes documentos :

Pedro y Pablo
Pedro corre
Pablo respira
Pedro corre y respira
Pedro corre Pedro

  - El léxico es el siguiente :

Pedro
y
Pablo
corre
respira

Construcción del Índice

- Inversión por Sort - Ejemplo (2 de 3) :
  - El archivo auxiliar queda:

No-Termino	Documento	Freq
1	1	1
2	1	1
3	1	1
1	2	1
4	2	1
3	3	1
5	3	1
1	4	1
4	4	1
2	4	1
5	4	1
1	5	2
4	5	1

Construcción del Índice

- Inversión por Sort - Ejemplo (3 de 3) :

Luego de ordenar :

No-Termino	Documento	Freq
1	1	1
1	2	1
1	4	1
1	5	2
2	1	1
2	4	1
3	1	1
3	3	1
4	2	1
4	4	1
4	5	1
5	3	1
5	4	1

Indice generado :

Termino	Freq	Docs
Pedro	5	1,2,4,5
y	2	1,4
Pablo	2	1,3
corre	3	2,4,5
respira	2	3,4

Indices por Signature Files

- El índice tiene una entrada por cada documento, conteniendo el signature file del mismo. La longitud del signature file es una potencia de 2 (32 bits, 64, 128, etc).
- Cada signature file es un extracto de los términos que componen al documento.
- Para armar el signature file del documento hacemos un OR del resultado de hashear los términos que contiene.

Indices por Signature Files

- El hasheo de cada término se hace de la siguiente manera :
  - Se aplican, por ejemplo, 3 funciones de hashing.
  - A cada resultado de las funciones se le aplica MOD de la longitud del signature file.
  - Se encienden los bits correspondientes al los números obtenidos.

Indices por Signature Files

- Ejemplo :
  - Dado el documento: "El tractorcito rojo que silbó y bufó"
  - Queremos generar un signature file de 16 bits:
  - Aplicando las funciones de hashing a cada término MOD 16 y haciendo el OR obtenemos :

Término	*Signature*
El	0100100000100000
tractorcito	0001000010000100
rojo	1000001000100000
que	0010100000001000
silbó	0000000000100011
y	1100100000000000
bufó	0110000000100000

Signature file del documento :

1111101010101111

Indices por Signature Files

- Resolución de Consultas :
  - Para resolver consultas se calcula el hash del término a buscar y todos los documentos que tengan los bits correspondientes encendidos son candidatos a contener el término.
  - Consultas del tipo "documentos que NO contengan el siguiente término" se resulven directamente.
  - Es muy complicado resolver consultas combinadas con AND y OR.

Indices por Signature Files

- Construcción de Signature Files-Bit Slices :

Doc	Signature
1	01100001010000010
2	01000010000101010
3	11010000110010111
4	11010000010000100
5	01001110000010011

$\Longrightarrow$

Slice	Valor
0	00110
1	11111
2	10000
3	00110
4	00001
5	00001
6	01001
7	10000
8	00100
9	10110
10	01000
11	00001
12	01100
13	00010
14	11101
15	00101

Indices por Bitmaps

- Definición :

  Otra forma de indexar los documentos es utilizando un bitmap por documento, donde cada bit representa a un término y se setea en 1 si el término aparece en el documento o en 0 si no aparece.

  Esto genera bitmaps con gran preponderacia de 0 y dado que los bitmaps ocupan mucho espacio, sería bueno buscar una forma de comprimirlos aprovechando esta característica

Indices por Bitmaps

- Ejemplo :
  - Dado los siguientes documentos y el léxico :

Pedro y Pablo
Pedro corre
Pablo respira
Pedro corre y respira
Pedro corre Pedro

Pedro
y
Pablo
corre
respira

  - Los bitmaps generados serán :
    - Doc.1 : 11100
    - Doc.2 : 10010
    - Doc.3 : 00101
    - Doc.4 : 11011
    - Doc.5 : 10110

Indices por Signature Files

- Compresión del Bitmap :
  - Para comprimir un bitmap formamos grupos de N bits.
  - Armamos un árbol formando grupos de N bits en el nivel superior, un bit por cada grupo del nivel inferior. Si hay algún bit en el grupo correspondiente, se setea el bit en 1, sino se setea en 0. El árbol sigue creciendo hasta que tenga una raíz de un solo grupo.
  - Al almacenar, solo almacenamos los grupos que contienen al menos un bit en 1.

Indices por Signature Files

- Compresión del Bitmap - Ejemplo:
  - Sea el bitmap :

0000 0010 0000 0011 1000 0000 0100 0000 0000 0000 0000 0000 0000 0000 0000

  - Si N = 4 la primera etapa será :

0101 1010 0000 0000
0000 0010 0000 0011 1000 0000 0100 0000 0000 0000 0000 0000 0000 0000 0000

  - Luego agregamos el siguiente nivel :

1100
0101 1010 0000 0000
0000 0010 0000 0011 1000 0000 0100 0000 0000 0000 0000 0000 0000 0000 0000

  - En el archivo almacenamos :

1100 0101 1010 0010 0011 1000 0100 (7 x 4 = 28 bits)

Indices por Signature Files

- Descompresión del Bitmap
  - Los primeros 4 bits (por ser N=4) corresponden a la raiz. Por cada 1 leemos 4 bits del archivo de entrada, que corresponden a un hijo de la raiz. Por cada 0 agregamos un hijo con todos sus bits en cero. Luego seguimos con el siguiente nivel hasta terminar el archivo.

Indices por Signature Files

- Descompresión del Bitmap - Ejemplo :
  - Si tenemos el archivo :

    1100 0101 1010 0010 0011 1000 0100
  - Sabemos que la raiz es 1100 y que sus primeros 2 hijos son los siguientes 8 bits del archivo y los otros 2 tienen todos sus bits en 0 :

    1100

    0101 1010 0000 0000
  - Al procesar el siguiente nivel sabemos que los bloques 2,4,5 y 7 contienen los siguientes bloques del archivo y el resto está en cero. Al no haber más datos en el archivo, tenemos el bitmap original:

0000 0010 0000 0011 1000 0000 0100 0000 0000 0000 0000 0000 0000 0000 0000 0000

Resolución de Consultas

- Existen distintos tipos de consultas que podemos pretender responder :
  - Consultas Booleanas : La respuesta consiste en una lista de documentos o un mensaje de "no se encuentra" Podemos clasificarlas en :
    - Puntuales : Buscar X
    - Conjuntivas : Buscar X, Y y Z
    - Disyuntivas : Buscar X, Y o Z
    - Compuestas : Buscar ((X e Y) o (Z y (R o M)))
    - Wildcards : Buscar "lab*r"
  - Consultas Ranqueadas : Devuelven una lista ordenada por la relevancia del documento respecto de la consulta

Resolución de Consultas

- Resolución de Wildcards :
  - Fuerza bruta : Consiste en buscar en todo el léxico buscando los términos que cumplan con el patrón ingresado. Es el único método sin modificar los índices construidos.
  - N-gramas : Antes de indexar el texto, cada termino se divide en n-gramas que son porciones de n-letras. Por ejemplo si n=2, labor se indexa como l, la, bo, or, r, donde $ indica el fin y el comienzo del término. En general el índice de n-gramas se agrega como un nivel superior que apunta al léxico del indice original. Implica un incremento aproximado del 50% del espacio utilizado
  - Léxico rotado : Este método consiste en indexar todas las rotaciones de cada término. Asi labor se indexa como: $labor, abor$l, borla, orlab, r$labo.
    Este método implica un incremento aproximado del 250% del espacio utilizado

Resolución de Consultas

- Resolución de Consultas Ranqueadas :
  - Para mostrar los métodos a utilizar haremos uso de los siguientes 6 documentos :

arbol, blanco,blanco, elefante, arbol chocolate, blanco blanco chocolate arbol chocolate dedo blanco blanco blanco blanco elefante blanco chocolate blanco elefante blanco arbol chocolate blanco elefante elefante elefante chocolate elefante

  - Llamaremos a, b, c, d y e a los términos, en base a su inicial.
  - En todos los métodos el objetivo es asignar un puntaje a cada documento basados en su relevancia.

Resolución de Consultas

- Coordinate Matching
  - El puntaje se asigna simplemente contando la cantidad de palabras, de las buscadas, que contiene el documento.
- Por ejemplo para la consulta

  (elefante, arbol, blanco, dedo)
- Los pesos son:

  D1=3, D2=3, D3=2, D4=2, D5=2, D6=1

  Por lo tanto los documentos 1 y 2 son los de mayor ranking

Resolución de Consultas

- Coordinate Matching

  Si consideramos :

  – La consulta como un vector de todos los términos, donde pongamos en 1 los que estén incluidos en la consulta - en el ejemplo el vector sería (1,1,0,1,1) -

  – Que a cada documento le asociamos un vector similar, con un 1 en el lugar correspondiente a los términos que lo conforman - para D1 sería (1,1,0,0,1) -

  Podemos calcular el peso de cada documento como el producto interno del vector consulta y el vector del documento.

Resolución de Consultas

- Coordinate Matching

  El vector consulta es (1,1,0,1,1)

Los vectores de los documentos son :	Multiplicando por el vector consulta obtenemos el puntaje
D1 = (1,1,0,0,1)	. (1,1,0,1,1) = 3
D2 = (1,1,1,1,0)	. (1,1,0,1,1) = 3
D3 = (0,1,0,0,1)	. (1,1,0,1,1) = 2
D4 = (0,1,1,0,1)	. (1,1,0,1,1) = 2
D5 = (1,1,1,0,0)	. (1,1,0,1,1) = 2
D6 = (0,0,1,0,1)	. (1,1,0,1,1) = 1

Resolución de Consultas

- Coordinate Matching

  Este método tiene los algunos problemas:

  – No tiene en cuenta las frecuencias del término en el documento.

  – No tiene en cuenta el "peso" de cada término. En el ejemplo 'dedo' "pesa" más que 'blanco' ya que es menos frecuente.

  – Los documentos más largos se ven beneficiados ya que tienen más probabilidad de contener más términos que uno corto.

Resolución de Consultas

- Producto interno :

  Para solucionar el primer defecto, modificamos el vector de cada documento para que contenga la frecuencia de cada término.

  De esta forma los vectores de los documentos cambian y, por lo tanto, los puntajes obtenidos :

  D1 = (2,2,0,0,1) . (1,1,0,1,1) = 5
  D2 = (1,2,3,1,0) . (1,1,0,1,1) = 4
  D3 = (0,5,0,0,1) . (1,1,0,1,1) = 6
  D4 = (0,1,1,0,1) . (1,1,0,1,1) = 2
  D5 = (1,2,1,0,0) . (1,1,0,1,1) = 3
  D6 = (0,0,1,0,4) . (1,1,0,1,1) = 4

Resolución de Consultas

- Producto interno :

  Con esto resolvimos el primer problema, pero seguimos sin considerar la importancia de los términos.

  El valor de cada término podemos asignarselo en forma inversamente proporcional a su frecuencia. Para ello utilizamos la Ley de Zipf que llama la wt y la define:

$$wt = \log_{10} \frac{N}{ft}$$

  siendo N la cantidad de objetos del universo y ft la frecuencia del objeto t

Resolución de Consultas

- Producto interno :

  Para nuestro caso utilizaremos :

  N = cantidad total de documentos

  ft = cantidad de documentos donde aparece el término t

  Entonces, cada elemento del vector del documento será:

$$wtd = ftd * wt$$

$$wtd = ftd * \log_{10} \frac{N}{ft}$$

Resolución de Consultas

- Producto interno :

  Ahora el wt de cada término es:
  - a=log(6/3)=0,3010
  - b=log(6/5)=0,0791
  - c=log(6/4)=0,1760
  - d=log(6/1)=0,1761
  - e=log(6/4)=0,1761

  Los vectores quedan entonces asi :

Doc	a	b	c	d	e
1	3	0,778151	0	0	0,875061
2	1	1,556303	2,625184	1,477121	0
3	0	3,890756	0	0	0,875061
4	0	0,778151	0,875061	0	0,875061
5	1	1,556303	0,875061	0	0
6	0	0	0,875061	0	3,500245

Resolución de Consultas

- Producto interno :
  - Los pesos de los documentos son ahora:

Doc	Peso
1	4,653213
2	4,033424
3	4,765818
4	1,653213
5	2,556303
6	3,500245

  - Esto soluciona el segundo problema. Para solucionar el tercero. Para solucionarlo dividimos el resultado del producto interno por la longitud del documento medida en cantidad de términos.

Doc	Peso/Dd
1	0,930643
2	0,576203
3	0,794303
4	0,551071
5	0,639076
6	0,700049

Resolución de Consultas

- Modelo de espacio vectorial:

  Ya que la consulta y un documento X se representan a través de vectores, podríamos calcular el puntaje como la distancia euclidiana entre los 2 vectores :

$$P(Q,D) = \sqrt{\sum |wqi * wdi|^2}$$

  Pero lo que nos interesa no es la distancia, sino el ángulo entre ambos vectores. Esto se obtiene calculando el coseno. Cuando más alto es el coseno, más "parecidos" son los vectores en cuanto a su dirección.

Resolución de Consultas

- El método del coseno :
  - Siendo Q el vector consulta y W el vector del documento, el puntaje del documento es :

$$\cos \phi = \frac{Q * D}{|Q| * |D|}$$

$$= \frac{1}{Wq * Wd} * \sum Wqi * Wdi$$

$$Wd = \sqrt{\sum Wdi^2}$$

$$Wq = \sqrt{\sum Wqi^2}$$

$$Wdi = fdti * \log_{10} \frac{N}{fti}$$

$$Wqi = \log_{10} \frac{N}{ti} * Qi$$