Índices de RI

UCR – ECCI

CI-2414 Recuperación de Información

Prof. Kryscia Daviana Ramírez Benavides

¿Qué es un Índice?

- Es la segunda etapa para abordar el tema de la RI.
- Es un archivo que contiene la importancia de cada término del vocabulario y los documentos de la colección que los contienen.
- Un buen índice no debe tardase demasiado en recuperar la información.
- Los índices son utilizados para la recuperación de los documentos relevantes, obtenidos del cálculo hecho por los modelos.

¿Qué es un Índice? (cont.)

- Ejemplos de índices:
  - Índices invertidos.
  - Arreglos de sufijos.
  - Índices Distribuidos.

Índices Invertidos

- Es la estructura más elemental para la recuperación de palabras.
- En su versión más básica, consta de dos partes:
  - *Vocabulario:* Conjunto de términos distintos del texto.
  - *Posteo:* Para cada término, la lista de documentos donde aparece.

Índices Invertidos (cont.)

- Variante para los modelos booleano y probalístico:
  - Es conveniente almacenar la lista de posteo de cada término en orden creciente de documento.
- Variante para el modelo vectorial:
  - Debe almacenar el *tf* correspondiente en cada entrada de posteo.
  - Debe almacenar el *idf* correspondiente en cada entrada del vocabulario.
  - Es conveniente almacenar el máximo *tf* de cada término en el vocabulario.
  - Es conveniente almacenar la lista de posteo de cada término en orden decreciente de *tf*.

Espacio Extra de los Índices Invertidos

- Caso booleano y probabilístico, al comprimirlo, requiere un 10%-25% extra sobre el texto.
- Caso vectorial requiere 15%-30% extra.
- Con direccionamiento a bloques esto puede bajar hasta un 4% para colecciones no muy grandes, al costo de mayor búsqueda secuencial.
- Se puede comprimir el texto a “5%-30% del espacio original (Huffman sobre palabras).

Búsqueda en los Índices Invertidos

- Las palabras de la consulta se buscan en el vocabulario (que está en memoria), por ejemplo usando hash.
- Se recuperan de disco las listas de posteo de cada palabra involucrada.
- Se calcula la similaridad del peso de cada documento de la lista de posteo con el peso de la consulta.
- Se hace el ranking respectivo para mostrar los documentos, esto se hace si el modelo realiza ranking.

Construcción de Índices Invertidos

- Se recorre la colección de texto secuencialmente.
- Para cada término leído, se busca en el vocabulario (que se mantiene en memoria).
- Si el término no existe aún, se agrega al vocabulario con una lista de posteo vacía.
- Se agrega el documento que se está leyendo al final de la lista de posteo del término.
- Una vez leída toda la colección, el índice se graba a disco.
- Con texto comprimido se da una pasada para generar el vocabulario y frecuencias, otra para indexar y comprimir a la vez.

Construcción de Índices Invertidos (cont.)

- El mayor problema que se presenta en la práctica es que, obviamente, la memoria RAM se terminará antes de poder procesar todo el texto.
- Cada vez que la memoria RAM se agota, se graba en disco un *índice parcial*, se libera la memoria y se comienza de cero.
- Al final, se realiza un *merge* de los índices parciales. Este *merge* no requiere demasiada memoria porque es un proceso secuencial, y resulta relativamente rápido en I/O porque el tamaño a procesar es bastante menos que el texto original.

Arreglos de Sufijos

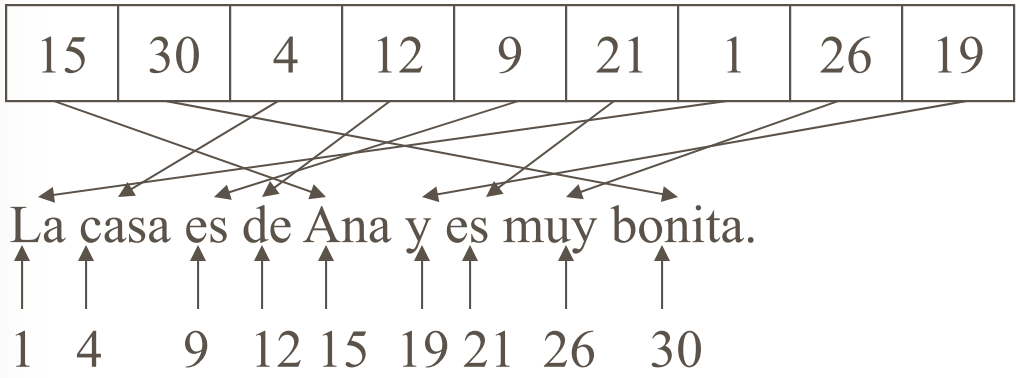
- Es un índice que no necesita que el texto esté formado por palabras.
- Es capaz de recuperar cualquier *substring* del texto, busca expresiones regulares y realizar búsqueda aproximada en tiempo sublineal.
- En RI tradicional, puede ser útil para:
  - Análisis lingüístico especializado.
  - Búsqueda de frases de varias palabras.
  - Búsqueda de patrones que trascienda palabras.

Arreglos de Sufijos (cont.)

- El espacio que requiere es cercano al 40% del texto.
- Es más costoso de construir y mantener que un índice invertido.
- Es superior al índice invertido para la búsqueda de patrones, pero no para las otras operaciones típicas en RI.

Arreglos de Sufijos (cont.)

- Es un simple arreglo con los punteros a todas las posiciones de interés en el texto.
- Cada posición define un *sufijo* del texto.
- En el arreglo, las posiciones están ordenadas *lexicográficamente* por los sufijos.



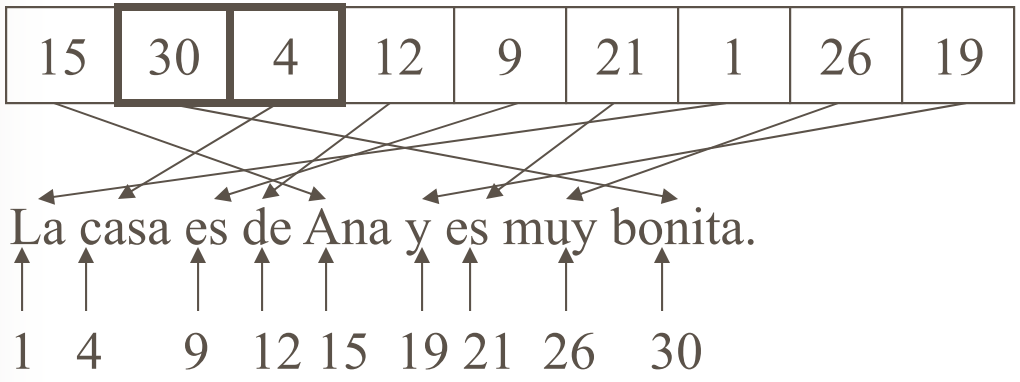
Arreglo de Sufijos

Texto

Puntos del Índice

Búsqueda en los Arreglos de Sufijos

- Todo *substring* del texto es el prefijo de un sufijo.
- Se hacen un par de búsquedas binarias en el arreglo de sufijos para obtener el rango del arreglo que contiene todas las ocurrencias de *x* (en tiempo logarítmico).



Arreglo de Sufijos

Texto

Puntos del Índice

Construcción de Arreglos de Sufijos

- En principio, no es más que ordenar un arreglo.
- Sin embargo, se puede aprovechar la estructura del problema: los sufijos son sufijos de otros sufijos.
- Sin embargo, el verdadero problema aparece en memoria secundaria, por los accesos aleatorios al texto.
- Pasos:
  - Cortar el texto en bloques que se pueden indexar en memoria.
  - Traerse el primer bloque, construir su arreglo y mandarlo a disco.
  - Al procesar el bloque *i*, el invariante es que el arreglo de sufijos para los *i-1* bloques anteriores está construido.

Construcción de Arreglos de Sufijos

- Pasos:
  - Traer el bloque *i-ésimo* y construir su arreglo de sufijos.
  - Leer el texto de los *i-1* bloques anteriores y buscar cada sufijo en el arreglo del bloque *i-ésimo*.
  - Determinar así cuántos sufijos del arreglo grande van entre cada par de posiciones del arreglo chico.
  - Realizar el *merge* secuencial de los dos arreglos con esa información.
- El método se adapta fácilmente para actualizar el índice.
- En texto comprimido se reduce el espacio, tiempo de construcción y de búsqueda.

Índices Distribuidos

- En muchos casos, por bueno que sea el algoritmo de indexación o de búsqueda, no es suficiente para cubrir la demanda:
  - El texto es demasiado grande.
  - La frecuencia de actualización es demasiado alta.
  - Llegan demasiadas consultas por segundo.
  - La velocidad de los discos no está creciendo al ritmo necesario.
- Una alternativa es utilizar paralelismo. Bien diseñado, puede expandir la capacidad de procesamiento tanto como se quiera.
- Las redes muy rápidas formadas por unas pocas máquinas muy potentes se han convertido en una alternativa de bajo costo.

Índices Distribuidos (cont.)

- En estas redes el acceso remoto cuesta aproximadamente lo mismo que el acceso al disco local.
- Normalmente, todos los procesadores pueden comunicarse de a pares sin causar congestión.
- Se puede considerar el total de RAMs como una gran memoria distribuida.
- Dos medidas de interés para las consultas:
  - *Throughput:* Cantidad de consultas respondidas por segundo.
  - *Tiempo de respuesta:* Tiempo que demora una consulta particular.

Generación Distribuida de Índices Invertidos

- Se distribuye el texto entre las máquinas equitativamente.
- Cada máquina construye su índice invertido local.
- Se aparean de a dos, jerárquicamente, hasta que una sola contiene todo el vocabulario.
- Esa máquina calcula qué parte del vocabulario será responsabilidad de cada procesador, y distribuye esa información.
- Los procesadores se aparean todos con todos intercambiando las listas de posteo.
- Secuencialmente transmiten su parte del índice a un disco central, donde se concatenan para formar un índice centralizado.

Referencias Bibliográficas

- La información fue tomada de:
  - Libro de texto del curso.