Indexación y Búsqueda

Índice

- Introducción
- Índices invertidos
- Sufijos
- Búsqueda
- Conclusiones

Introducción

Introducción

- La efectividad en un sistema IR es importante
  - Ratio satisfacción del usuario vs esfuerzo
- Un sistema IR no puede centrarse sólo en la efectividad
- Dar respuesta a miles de consultas por segundo
  - Sistema eficiente
- La eficiencia, tiene que jugar un papel central en el desarrollo de cualquier sistema IR
  - Un sistema IR tiene que ser eficaz y eficiente
    - Un sistema eficaz, pero no eficiente no será bueno
  - El usuario no está dispuesto a esperar mucho tiempo para obtener la respuesta
    - La respuesta debe ser inmediata

Introducción

- Ejemplo
  - Biblioteca con 1000 libros
    - 700 kb por libro
    - 700 Mb en total
    - Toda la colección cabe en memoria principal
    - Búsqueda de una palabra: 1-10 segundos
  - Biblioteca con 10000 libros
    - 7 GB en total
    - La colección no cabe de forma completa en memoria principal
    - Búsqueda de una palabra: 2-3 minutos
- Son necesarias estructuras de datos especiales para almacenar la información

Introducción

- En la web, es necesario
  - indexar terabytes de datos
  - responder miles de preguntas por segundo
- Búsqueda secuencia
  - Básica
  - No necesita construir/mantener ninguna estructura de datos
  - Ineficiente
- Índice: estructura de datos construida sobre el texto para agilizar las búsquedas
  - Crear y mantener un índice es complejo
  - Permite obtener tiempos de respuesta aceptables

Introducción

- La eficiencia de un sistema IR puede medirse como:
  - Tiempo de indexación: tiempo necesario para la construcción del índice
    - Lineal con respecto al tamaño de la colección
  - Espacio ocupado en indexación: espacio (máximo) utilizado durante la creación del índice
    - Lineal con respecto al tamaño de la colección
  - Almacenamiento del índice: espacio requerido para almacenar el índice
    - El espacio final debe ser mucho más pequeño que el necesario para almacenar la colección completa
  - Latencia de consulta: tiempo que transcurre desde la llegada de una consulta al sistema IR y la generación de la respuesta
  - Consultas por segundo: número medio de consultas procesadas por segundo

Introducción

- ¿Qué ocurre ante la llegada/actualización de texto en la colección?
  - El índice tiene que actualizarse
- La actualización de un índice es costosa
- La solución es utilizar colecciones semi-automáticas
  - El índice se refresca regularmente: cada día/semana
- Documentos nuevos que no están en el índice
  - Búsqueda secuencial
  - Construir un índice pequeño para ellos

Índices invertidos

Índices invertidos

- Basados en palabras
- Indexan una colección de texto para permitir agilizar las búsquedas
- Se componen de dos elementos:
  - Vocabulario, lexicón o diccionario: conjunto de todas las palabras únicas en la colección
  - Ocurrencias: documentos en los que aparece las palabras
- Para cada palabra del vocabulario, el índice almacena en qué documentos aparece
- El texto se puede reconstruir utilizando el índice invertido

Índices invertidos

REPRESENTACIÓN BÁSICA

Índices invertidos Representación básica

- Un índice invertido puede representarse de forma básica mediante una matriz $término - documento$
  - Rápida: sólo se necesita un acceso a la matriz para determinar si un documento contiene una palabra
  - En el modelo booleano no necesitaremos almacenar la frecuencia de aparición, solo si aparece en el documento o no aparece
  - Necesita demasiado espacio para almacenar el índice (proporcional al número de documentos multiplicado por el número de palabras únicas en la colección)
    - La matriz es dispersa: la mayor parte de las celdas estarán vacías o contendrán un valor de falso (modelo booleano)

Índices invertidos
Representación básica

To do is to be.
To be is to do.

d_1

To be or not to be.
I am what I am.

d_2

I think therefore I am.
Do be do be do.

d_3

Do do do, da da da.
Let it be, let it be.

d_4

Índices invertidos
Representación básica

To do is to be.
To be is to do.

d_1

To be or not to be.
I am what I am.

d_2

I think therefore I am.
Do be do be do.

d_3

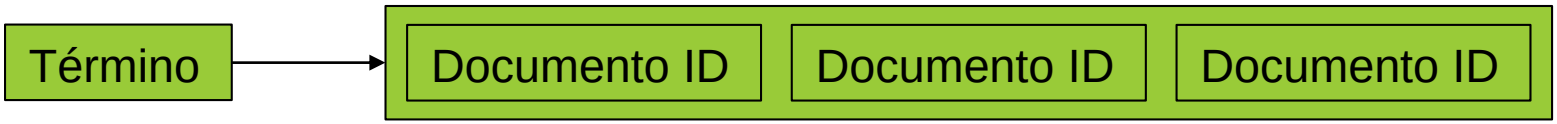
Do do do, da da da.
Let it be, let it be.

d_4

#	Término	n_i	$f_{i,1}$	$f_{i,2}$	$f_{i,3}$	$f_{i,4}$
1	to	2	4	2		
2	do	3	2		3	3
3	is	1	2			
4	be	4	2	2	2	2
5	or	1		1		
6	not	1		1		
7	I	2		2	2	
8	am	2		2	1	
9	what	1		1		
10	think	1			1	
11	therefore	1			1	
12	da	1				3
13	let	1				2
14	it	1				2

Índices invertidos Representación básica

- Solución, asociar a cada palabra una lista de documentos
  - El conjunto de listas de documentos se le llama ocurrencias



- El espacio utilizado es proporcional a las ocurrencias de las palabras en el documento
- El espacio utilizado es menor que el tamaño del documento
  - La mayoría de los documentos contiene una pequeña porción de las palabras de la colección
- El índice invertido se utiliza para albergar todos los documentos
  - Se puede adaptar a los diferentes modelos de RI

Índices invertidos
Representación básica

To do is to be.
To be is to do.

d_1

To be or not to be.
I am what I am.

d_2

I think therefore I am.
Do be do be do.

d_3

Do do do, da da da.
Let it be, let it be.

d_4

to	$[1], [2]$
do	$[1], [3], [4]$
is	$[1]$
be	$[1], [2], [3], [4]$
or	$[2]$
not	$[2]$
I	$[2], [3]$
am	$[2], [3]$
$what$	$[2]$
$think$	$[3]$
$therefore$	$[3]$
da	$[4]$
let	$[4]$
it	$[4]$

Índices invertidos Representación básica

To do is to be.
To be is to do.

d_1

To be or not to be.
I am what I am.

d_2

I think therefore I am.
Do be do be do.

d_3

Do do do, da da da.
Let it be, let it be.

d_4

to	[1,4], [2,2]
do	[1,2], [3,3], [4,3]
is	[1,2]
be	[1,2], [2,2], [3,2], [4,2]
or	[2,1]
not	[2,1]
I	[2,2], [3,2]
am	[2,2], [3,1]
$what$	[2,1]
$think$	[3,1]
$therefore$	[3,1]
da	[4,3]
let	[4,2]
it	[4,2]

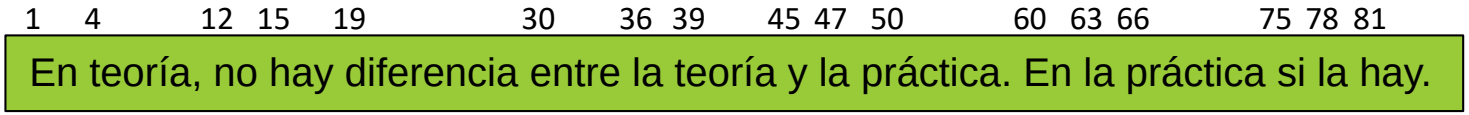
Índices invertidos

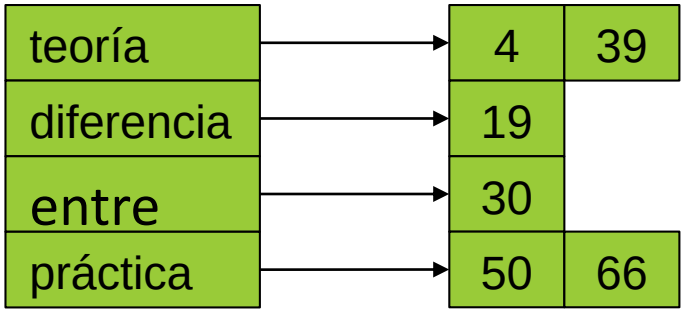
REPRESENTACIÓN COMPLETA

Índices invertidos Completos

- Los índices invertidos básicos no permiten resolver preguntas de proximidad o frases
  - No contienen información del lugar exacto del documento en el que aparece la palabra
- Solución: añadir la posición de cada palabra al índice
- La posición de la palabra puede hacer referencia a:
  - Palabra: palabra i-ésima
    - Útil para frases y búsquedas por proximidad
  - Carácter: carácter i-ésimo
    - Útil para el acceso a porciones de texto
- El espacio necesario para un índice invertido completo es proporcional al número total de ocurrencias, el cual es proporcional al tamaño del texto

Índices invertidos Completos





Índices invertidos Completos

To do is to be.
To be is to do.

d_1

To be or not to be.
I am what I am.

d_2

I think therefore I am.
Do be do be do.

d_3

Do do do, da da da.
Let it be, let it be.

d_4

to	[1,4, [1,4,6,9]], [2,2, [1,5]]
do	[1,2, [2,10]], [3,3, [6,8,10]], [4,3, [1,2,3]]
is	[1,2, [3,8]]
be	[1,2, [5,7]], [2,2, [2,6]], [3,2, [7,9]], [4,2, [9,12]]
or	[2,1, [3]]
not	[2,1, [4]]
I	[2,2, [7,10]], [3,2, [1,4]]
am	[2,2, [8,11]], [3,1, [5]]
$what$	[2,1, [9]]
$think$	[3,1, [2]]
$therefore$	[3,1, [3]]
da	[4,3, [4,5,5]]
let	[4,2, [7,10]]
it	[4,2, [8,11]]

Índices invertidos Completos

- El espacio necesario para el vocabulario sigue siendo pequeño
  - El espacio puede reducirse aún más:
    - Eliminación de las palabras vacías
    - Stemming
  - Ejemplo: el vocabulario de una colección de 1GB puede ocupar 5MB
- Las ocurrencias ocupan un mayor espacio
  - Cada palabra en un documento tiene asociadas las posiciones en las que aparece
  - 40% del texto si se omiten las palabras vacías
  - 80% del texto si se indexan las palabras vacías

Índices invertidos Completos

- El espacio utilizado se puede reducir mediante el direccionamiento **basado en bloques**:
  - El texto de los documentos se divide en bloques
  - Las ocurrencias enlazan a los bloques en donde las palabras aparecen
  - Las ocurrencias de una palabra dentro de un mismo bloque se agrupan
    - Si un bloque es un documento, el funcionamiento es similar al índice invertido básico
  - Desventaja: si necesitamos la posición exacta (búsqueda de proximidad), es necesaria una búsqueda secuencial dentro del bloque

Índices invertidos Completos

- Dependiendo de la granularidad:
  - Direccionamiento por palabras
    - Se tiene en cuenta cada palabra y las posiciones dentro de un mismo documento
  - Direccionamiento por documentos
    - El documento es la unidad lógica
    - Se direcciona una única aparición por documento
    - En un mismo documento sólo puede haber una aparición
  - Direccionamiento por bloques
    - Las ocurrencias se tienen en cuenta por bloques.
    - En un mismo bloque sólo puede haber una aparición
    - Los bloques pueden ser de 64k o 265k

Índices invertidos Completos

Bloque 1	Bloque 2	Bloque 3	Bloque 4
Esto es un texto.	Un texto tiene muchas	palabras. Las palabras	se componen de letras.

- Texto ➔ 1, 2
- Muchas ➔ 2
- Palabras ➔ 3
- Componen ➔ 4
- Letras ➔ 4

Índice invertido

Índices invertidos

BÚSQUEDA

Índices invertidos Búsqueda

- Consulta de una sola palabra
  - Buscar en el índice la palabra y recuperar la lista de documentos asociados
- Consulta de varias palabras
  - Consulta conjuntiva (AND)
    - Recuperar los documentos asociados a cada palabra
    - Realizar la intersección entre las listas de ocurrencia
      - Necesitamos los documentos en los que estén todas las palabras
    - Diferentes algoritmos para realizar la intersección
  - Consulta disyuntiva (OR)
    - Recuperar los documentos asociados a cada palabra
    - Realizar la unión entre las listas de ocurrencia
      - Documentos que contengan al menos una de las palabras

Índices invertidos Búsqueda

- Consultas complejas
  - Buscar las palabras que casen con el patrón
  - Realizar la unión de las listas dadas
- Frases y consultas de proximidad
  - Buscar cada elemento independientemente
  - Para cada documento encontrado, realizar un recorrido para buscar secuencias de aparición conjuntas de la frase o secuencias con la proximidad establecida
- Consultas booleanas
  - Resolver el árbol sintáctico → Obtener el conjunto de documentos de cada operación

Sufijos

Sufijos

- Los índices invertidos son la estructura más utilizada
- Limitaciones:
  - El texto tiene que poder dividirse fácilmente en secuencias de palabras
  - Las consultas deben recuperar sólo palabras completas o secuencias cortas de palabras
  - No debe haber un gran número de palabras diferentes
    - El vocabulario crecería sin control cayendo la eficiencia
- La mayor parte de estas condiciones ocurren en la mayoría de lenguajes
  - Español, Inglés, Italiano, Etc.

Sufijos

- Idiomas aglutinantes: alemán, finlandés
  - Concatenación de partículas menores para formar una palabra
  - Una aglutinación puede corresponder con una frase en inglés
  - Las consultas no se hace para las aglutinaciones, sino para las partículas
- Idiomas asiáticos: china, japonés, coreano, etc.
  - Secuencia de símbolos
  - Cada símbolo es un ideograma (concepto / mensaje simple)
  - El alfabeto es enorme
  - Separar las palabras de los símbolos es un problema abierto

Sufijos

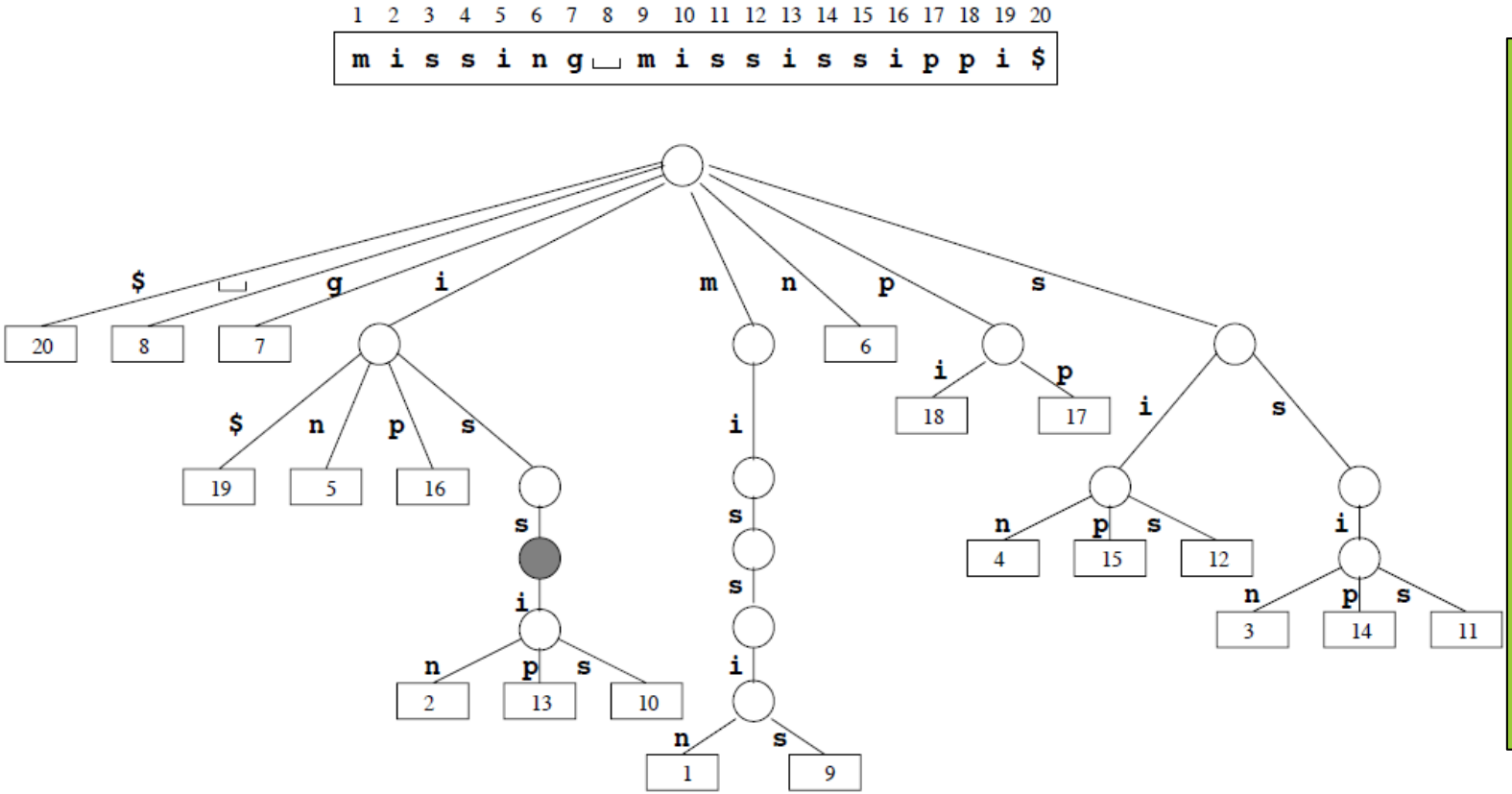
- Los índices de sufijos tratan el texto como una simple secuencia de caracteres
- Cada posición en el texto se considera un sufijo
  - Una cadena que va desde la posición actual hasta el final del texto
- Ejemplo: sufijos del texto “missing mississippi”
  - missing mississippi
  - issing mississippi
  - ssing mississippi
  - sing mississippi
  - ing mississippi
  - …
  - ppi
  - pi
  - i

Sufijos

- Los sufijos pueden almacenarse en:
  - Árboles de sufijos
    - Estructura de datos basada en Trie en la que se comprimen todas las rutas unarias
    - Ocupan mucho espacio (10 o 20 veces el tamaño de la colección)
  - Arrays de sufijos
    - Cada posición del array apunta al comienzo de un sufijo
    - Ocupan menos espacio
    - Más costosos computacionalmente

Sufijos

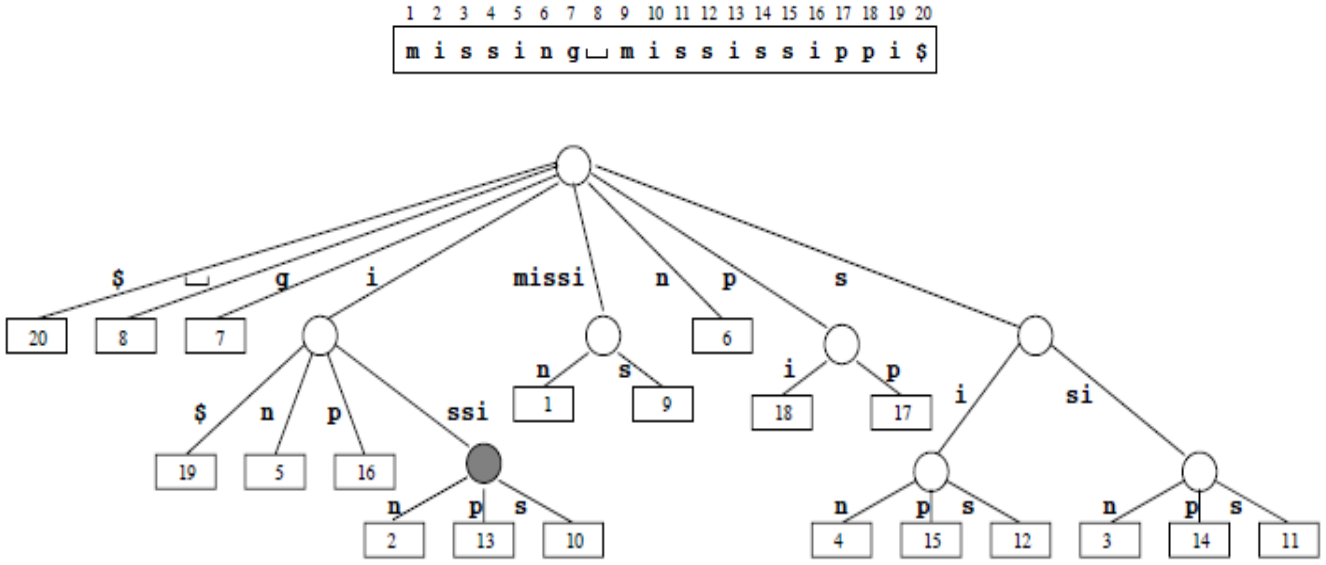
- Estructura Trie



Fuente: Modern Information Retrieval 2nd Edition. Ricardo Baeza-Yates

Sufijos

- Árbol de sufijos

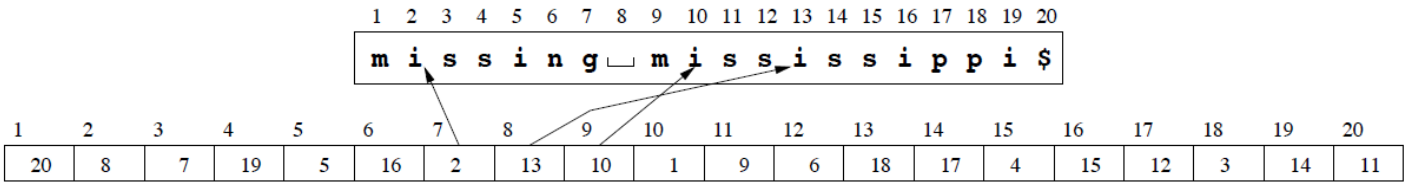


missing mississippi$
issing mississippi$
ssing mississippi$
sing mississippi$
ing mississippi$
ng mississippi$
g mississippi$
mississippi$
mississippi$
ississippi$
ssissippi$
sissippi$
issippi$
ssippi$
sippi$
ippi$
ppi$
pi$
i$
$

Fuente: Modern Information Retrieval 2nd Edition. Ricardo Baeza-Yates

Sufijos

- Array de sufijos



Fuente: Modern Information Retrieval 2nd Edition. Ricardo Baeza-Yates

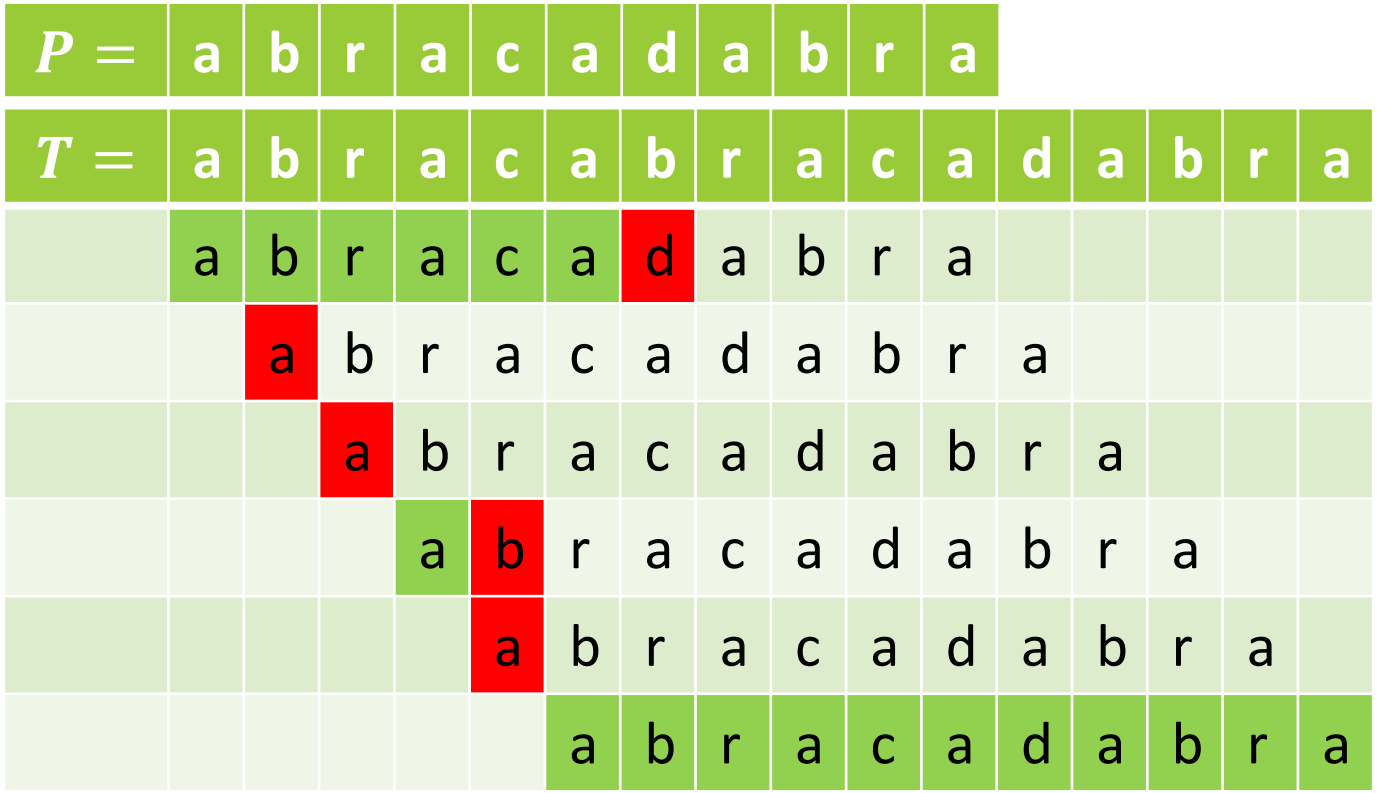
missing mississippi$
issing mississippi$
ssing mississippi$
sing mississippi$
ing mississippi$
ng mississippi$
g mississippi$
mississippi$
mississippi$
ississippi$
ssissippi$
sissippi$
issippi$
ssippi$
sippi$
ippi$
ppi$
pi$
i$
$

Búsqueda

Búsqueda

- Los índices invertidos tienen que hacer uso de una búsqueda secuencial en los modelos IR de texto completo
- El problema de **búsqueda secuencial** puede definirse como:
  - **Dado un texto** $T = t_1 t_2 \ldots t_n$ y un patrón formado por un conjunto de caracteres P, **encontrar todas las ocurrencias de los caracteres del patrón** P en T
  - Tanto P y T no tienen ninguna estructura definida
- **Búsqueda por fuerza bruta**
  - Prueba todas las posiciones posibles saber si el patrón está o no en el texto

Búsqueda


P = a b r a c a d a b r a
T = a b r a c a b r a c a d a b r a
a b r a c a d a b r a
a b r a c a d a b r a
a b r a c a d a b r a
a b r a c a d a b r a
a b r a c a d a b r a
a b r a c a d a b r a

Búsqueda

- **Algoritmo Horspool**:
  - Se basa en la idea de **desplazar la ventana del texto** de una forma más eficiente
  - Pasar por alto ciertas posiciones sin perder ninguna ocurrencia
  - Es necesario precalcular una tabla con el alfabeto de caracteres del patrón
    - Cada letra del patrón tendrá asociada cuantos saltos pueden realizarse si esta es la última letra de la ventana del texto (texto del mismo tamaño del patrón que se está evaluando en un momento dado)

Búsqueda

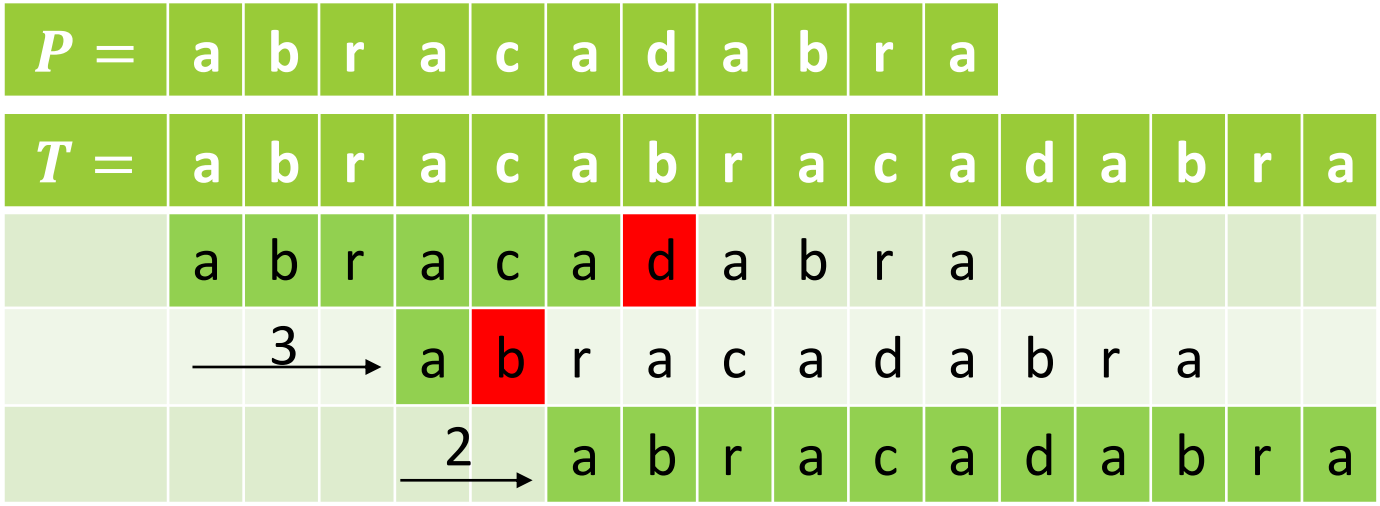


Tabla de desplazamientos

a	r	b	d	c
3	1	2	4	6