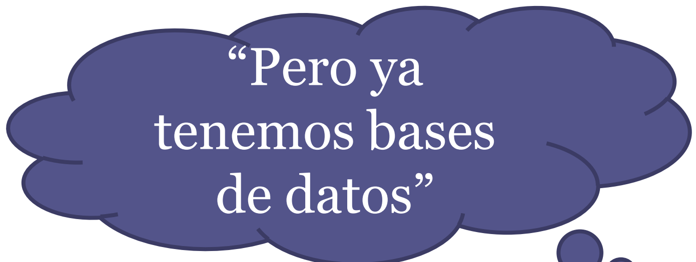
# Indexación

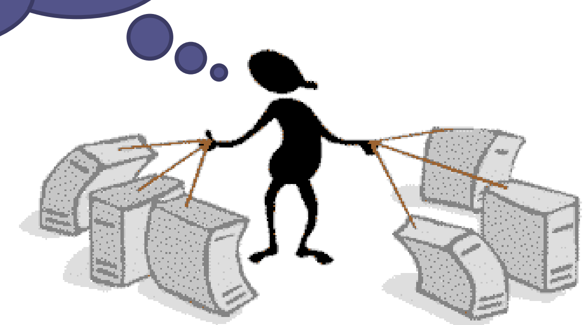
Análisis y Recuperación de Información
2016

Prof. Dr. Marcelo G. Armentano

# ¿Qué es un motor de búsqueda?

- Un sistema que:
  - Construye un índice a partir de texto
  - Responde a consultas utilizando este índice



- Un motor de búsquedas ofrece:
  - Escalabilidad
  - Ranking de relevancia
  - Integra diferentes fuentes de datos (email, páginas web, archivos, ...)

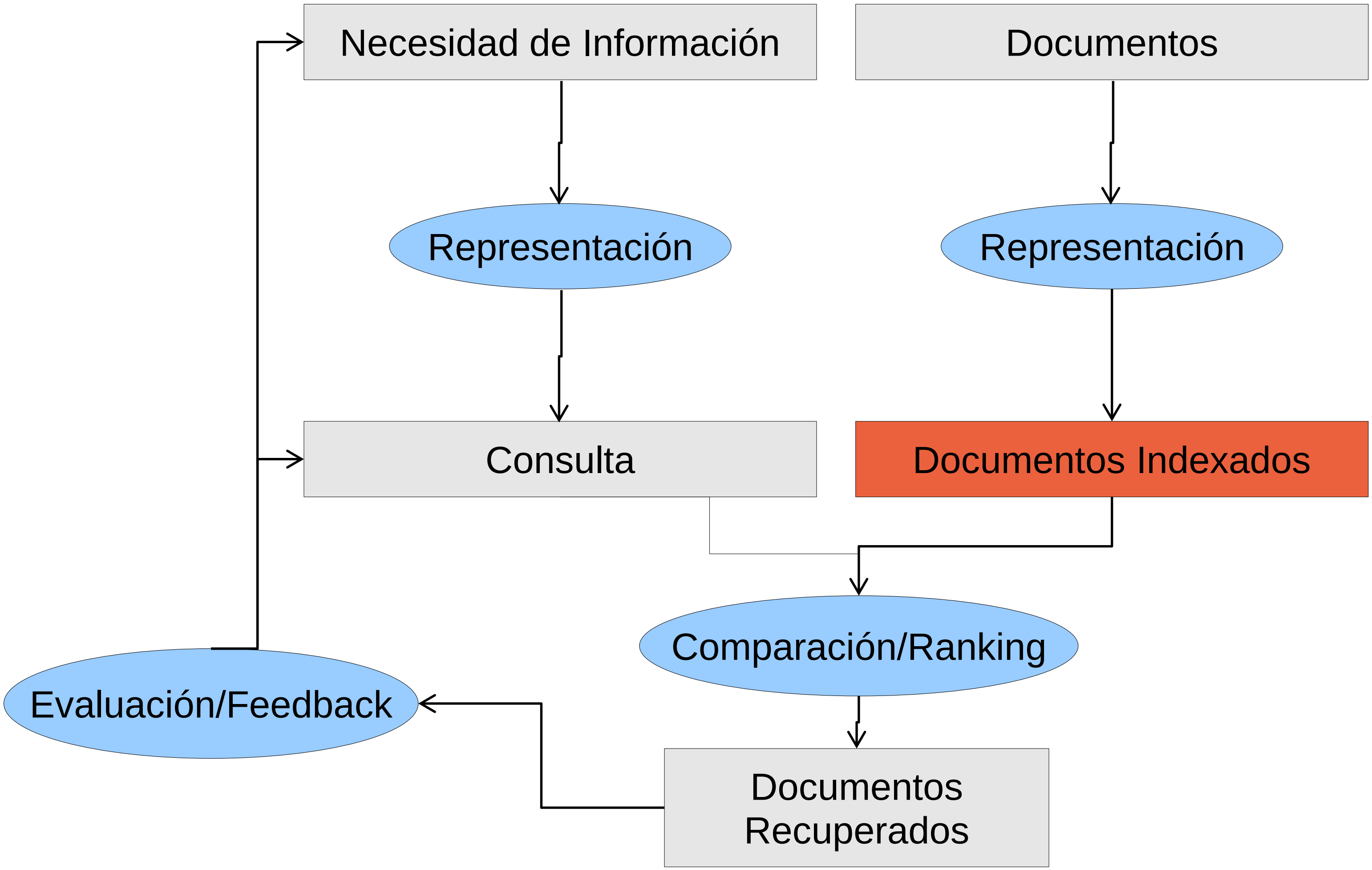
# ¿Qué es un motor de búsqueda? (cont)

- Trabaja con palabras, no con substrings
  - auto != automático, automóvil
- Proceso de indexación:
  - Convertir el documento
  - Extraer texto y metadatos
  - Normalizar el texto
  - Crear el índice invertido

# ¿Qué es un motor de búsqueda? (cont)

- Ejemplo:
  - Documento 1: “Recuperación de Información con Lucene“
  - Documento 2: “Análisis y Recuperación de Información“
- Índice invertido:
  - recuperación → 1, 2
  - análisis → 2
  - información → 1, 2
  - lucene → 1

# Recuperación de Información



# Archivos invertidos

- Los archivos invertidos son la estructura de datos primaria de los indexadores de texto

- Un archivo invertido es un mecanismo orientado a palabras para indexar una colección de texto con el objetivo de acelerar la tarea de búsqueda

- Estructura de un archivo invertido:
  - **Diccionario**: el conjunto de todas las palabras distintas que aparecen en el texto
  - **Ocurrencias** (postings): listas conteniendo toda la información necesaria para cada palabra del vocabulario (documentos donde aparece, posición, frecuencia, etc.)

# Archivos invertidos

- Un archivo invertido es esencialmente un vector invertido tal que las columnas se vuelven filas y las filas columnas

| *Docs* | *t1* | *t2* | *t3* |
|---|---|---|---|
| D1 | 1 | 0 | 1 |
| D2 | 1 | 0 | 0 |
| D3 | 0 | 1 | 1 |
| D4 | 1 | 0 | 0 |
| D5 | 1 | 1 | 1 |
| D6 | 1 | 1 | 0 |
| D7 | 0 | 1 | 0 |
| D8 | 0 | 1 | 0 |
| D9 | 0 | 0 | 1 |
| D10 | 0 | 1 | 1 |
| D11 | 1 | 0 | 1 |

| *Términos* | D1 | D2 | D3 | D4 | D5 | D6 | D7 | ... |
|---|---|---|---|---|---|---|---|---|
| *t1* | 1 | 1 | 0 | 1 | 1 | 1 | 0 | |
| *t2* | 0 | 0 | 1 | 0 | 1 | 1 | 1 | |
| *t3* | 1 | 0 | 1 | 0 | 1 | 0 | 0 | |

# Archivos invertidos

- Los archivos invertidos permiten una búsqueda rápida de los términos individuales
- Para cada término se obtiene una lista conteniendo:
  - ID de los documentos
  - frecuencia del término en el documento (opcional)
  - posición del término en el documento (opcional)
- Permiten resolver rápido consultas booleanas
  - tenis → $D_1$, $D_2$
  - fútbol → $D_2$
  - tenis AND fútbol → $D_2$

# Archivos invertidos - Construcción

- Los documentos se parsean para extraer palabras y se registran con un ID

**Doc 1**

Now is the time for all good men to come to the aid of their country

**Doc 2**

It was a dark and stormy night in the country manor. The time was past midnight

| Término | Doc # |
|---|---|
| now | 1 |
| is | 1 |
| the | 1 |
| time | 1 |
| for | 1 |
| all | 1 |
| good | 1 |
| men | 1 |
| to | 1 |
| come | 1 |
| to | 1 |
| the | 1 |
| aid | 1 |
| of | 1 |
| their | 1 |
| country | 1 |
| it | 2 |
| was | 2 |
| a | 2 |
| dark | 2 |
| and | 2 |
| stormy | 2 |
| night | 2 |
| in | 2 |
| the | 2 |
| country | 2 |
| manor | 2 |
| the | 2 |
| time | 2 |
| was | 2 |
| past | 2 |
| midnight | 2 |

# Archivos invertidos - Construcción

- Luego que todos los documentos fueron parseados el arreglo invertido se ordena por términos
- Los duplicados se retienen para mantener las estadísticas por término, se pueden eliminar si no se necesita información de las frecuencias

| Término | Doc # |
|---|---|
| now | 1 |
| is | 1 |
| the | 1 |
| time | 1 |
| for | 1 |
| all | 1 |
| good | 1 |
| men | 1 |
| to | 1 |
| come | 1 |
| to | 1 |
| the | 1 |
| aid | 1 |
| of | 1 |
| their | 1 |
| country | 1 |
| it | 2 |
| was | 2 |
| a | 2 |
| dark | 2 |
| and | 2 |
| stormy | 2 |
| night | 2 |
| in | 2 |
| the | 2 |
| country | 2 |
| manor | 2 |
| the | 2 |
| time | 2 |
| was | 2 |
| past | 2 |
| midnight | 2 |

| Término | Doc # |
|---|---|
| a | 2 |
| aid | 1 |
| all | 1 |
| and | 2 |
| come | 1 |
| country | 1 |
| country | 2 |
| dark | 2 |
| for | 1 |
| good | 1 |
| in | 2 |
| is | 1 |
| it | 2 |
| manor | 2 |
| men | 1 |
| midnight | 2 |
| night | 2 |
| now | 1 |
| of | 1 |
| past | 2 |
| stormy | 2 |
| the | 1 |
| the | 1 |
| the | 2 |
| the | 2 |
| their | 1 |
| time | 1 |
| time | 2 |
| to | 1 |
| to | 1 |
| was | 2 |
| was | 2 |

# Archivos invertidos - Construcción

- Se combinan las entradas correspondientes a cada término para un mismo documento
- Se compila la información de frecuencia
- Si se necesitan operadores de proximidad, la ubicación de cada ocurrencia también se almacena
- Para minimizar el espacio de almacenamiento, las palabras se representan unívocamente como enteros

| Término | Doc # |
|---|---|
| a | 2 |
| aid | 1 |
| all | 1 |
| and | 2 |
| come | 1 |
| country | 1 |
| country | 2 |
| dark | 2 |
| for | 1 |
| good | 1 |
| in | 2 |
| is | 1 |
| it | 2 |
| manor | 2 |
| men | 1 |
| midnight | 2 |
| night | 2 |
| now | 1 |
| of | 1 |
| past | 2 |
| stormy | 2 |
| the | 1 |
| the | 1 |
| the | 2 |
| the | 2 |
| their | 1 |
| time | 1 |
| time | 2 |
| to | 1 |
| to | 1 |
| was | 2 |
| was | 2 |

| Término | Doc # | Freq |
|---|---|---|
| a | 2 | 1 |
| aid | 1 | 1 |
| all | 1 | 1 |
| and | 2 | 1 |
| come | 1 | 1 |
| country | 1 | 1 |
| country | 2 | 1 |
| dark | 2 | 1 |
| for | 1 | 1 |
| good | 1 | 1 |
| in | 2 | 1 |
| is | 1 | 1 |
| it | 2 | 1 |
| manor | 2 | 1 |
| men | 1 | 1 |
| midnight | 2 | 1 |
| night | 2 | 1 |
| now | 1 | 1 |
| of | 1 | 1 |
| past | 2 | 1 |
| stormy | 2 | 1 |
| the | 1 | 2 |
| the | 2 | 2 |
| their | 1 | 1 |
| time | 1 | 1 |
| time | 2 | 1 |
| to | 1 | 2 |
| was | 2 | 2 |

# Archivos invertidos - Construcción

| Término | Doc # | Freq |
|---|---|---|
| a | 2 | 1 |
| aid | 1 | 1 |
| all | 1 | 1 |
| and | 2 | 1 |
| come | 1 | 1 |
| country | 1 | 1 |
| country | 2 | 1 |
| dark | 2 | 1 |
| for | 1 | 1 |
| good | 1 | 1 |
| in | 2 | 1 |
| is | 1 | 1 |
| it | 2 | 1 |
| manor | 2 | 1 |
| men | 1 | 1 |
| midnight | 2 | 1 |
| night | 2 | 1 |
| now | 1 | 1 |
| of | 1 | 1 |
| past | 2 | 1 |
| stormy | 2 | 1 |
| the | 1 | 2 |
| the | 2 | 2 |
| their | 1 | 1 |
| time | 1 | 1 |
| time | 2 | 1 |
| to | 1 | 2 |
| was | 2 | 2 |

| Término | N docs | Tot Freq |
|---|---|---|
| a | 1 | 1 |
| aid | 1 | 1 |
| all | 1 | 1 |
| and | 1 | 1 |
| come | 1 | 1 |
| country | 2 | 2 |
| dark | 1 | 1 |
| for | 1 | 1 |
| good | 1 | 1 |
| in | 1 | 1 |
| is | 1 | 1 |
| it | 1 | 1 |
| manor | 1 | 1 |
| men | 1 | 1 |
| midnight | 1 | 1 |
| night | 1 | 1 |
| now | 1 | 1 |
| of | 1 | 1 |
| past | 1 | 1 |
| stormy | 1 | 1 |
| the | 2 | 4 |
| their | 1 | 1 |
| time | 2 | 2 |
| to | 1 | 2 |
| was | 1 | 2 |

| Doc # | Freq |
|---|---|
| 2 | 1 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 2 |
| 2 | 2 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 1 | 2 |
| 2 | 2 |

El archivo se divide en diccionario y postings

# Búsquedas sobre el archivo invertido

- La búsqueda más simple es utilizando una única palabra
  - la búsqueda en el vocabulario se realiza utilizando una estructura de datos adecuada (por ej. hashtables, trees o b-trees)
- Las consultas conjuntivas implican buscar todas las palabras de la consulta, es la forma más popular en la Web
  - requiere la intersección de todas las listas invertidas
- Las consultas disyuntivas implican la unión de las listas

# Búsquedas sobre el archivo invertido

| Término | N docs | Tot Freq |
|---|---|---|
| a | 1 | 1 |
| aid | 1 | 1 |
| all | 1 | 1 |
| and | 1 | 1 |
| come | 1 | 1 |
| country | 2 | 2 |
| dark | 1 | 1 |
| for | 1 | 1 |
| good | 1 | 1 |
| in | 1 | 1 |
| is | 1 | 1 |
| it | 1 | 1 |
| manor | 1 | 1 |
| men | 1 | 1 |
| midnight | 1 | 1 |
| night | 1 | 1 |
| now | 1 | 1 |
| of | 1 | 1 |
| past | 1 | 1 |
| stormy | 1 | 1 |
| the | 2 | 4 |
| their | 1 | 1 |
| time | 2 | 2 |
| to | 1 | 2 |
| was | 1 | 2 |

| Doc # | Freq |
|---|---|
| 2 | 1 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 2 |
| 2 | 2 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 1 | 2 |
| 2 | 2 |

- Consulta: time and dark
- 2 documentos con "time" en el diccionario
- IDs 1 y 2 del archivo de postings
- 1 documento con "dark" en el diccionario
- ID 2 en el archivo de postings
- Entonces, solo el documento 2 satisface la consulta

# Búsquedas sobre el archivo invertido

- Las **consultas contextuales** son más difíciles de resolver con índices invertidos
  - lugares donde todas las palabras aparecen en secuencia (una **frase**)
  - aparecen lo suficientemente cerca (por proximidad)
  - las soluciones incluyen indexación de pares de palabras o índices posicionales, pero se incrementa el tamaño del índice
- Búsqueda por prefijo o por rango son básicamente consultas disyuntivas mayores
- Para las búsquedas de expresiones regulares no sirven las estructuras de datos construidas sobre el vocabulario, requieren una búsqueda secuencial

# Diccionario (o léxico)

- El diccionario almacena el vocabulario de términos, las frecuencias y punteros a cada lista de postings
  - requiere acceso rápido
  - preferiblemente en memoria
  - hashtables, trees, b-trees,…

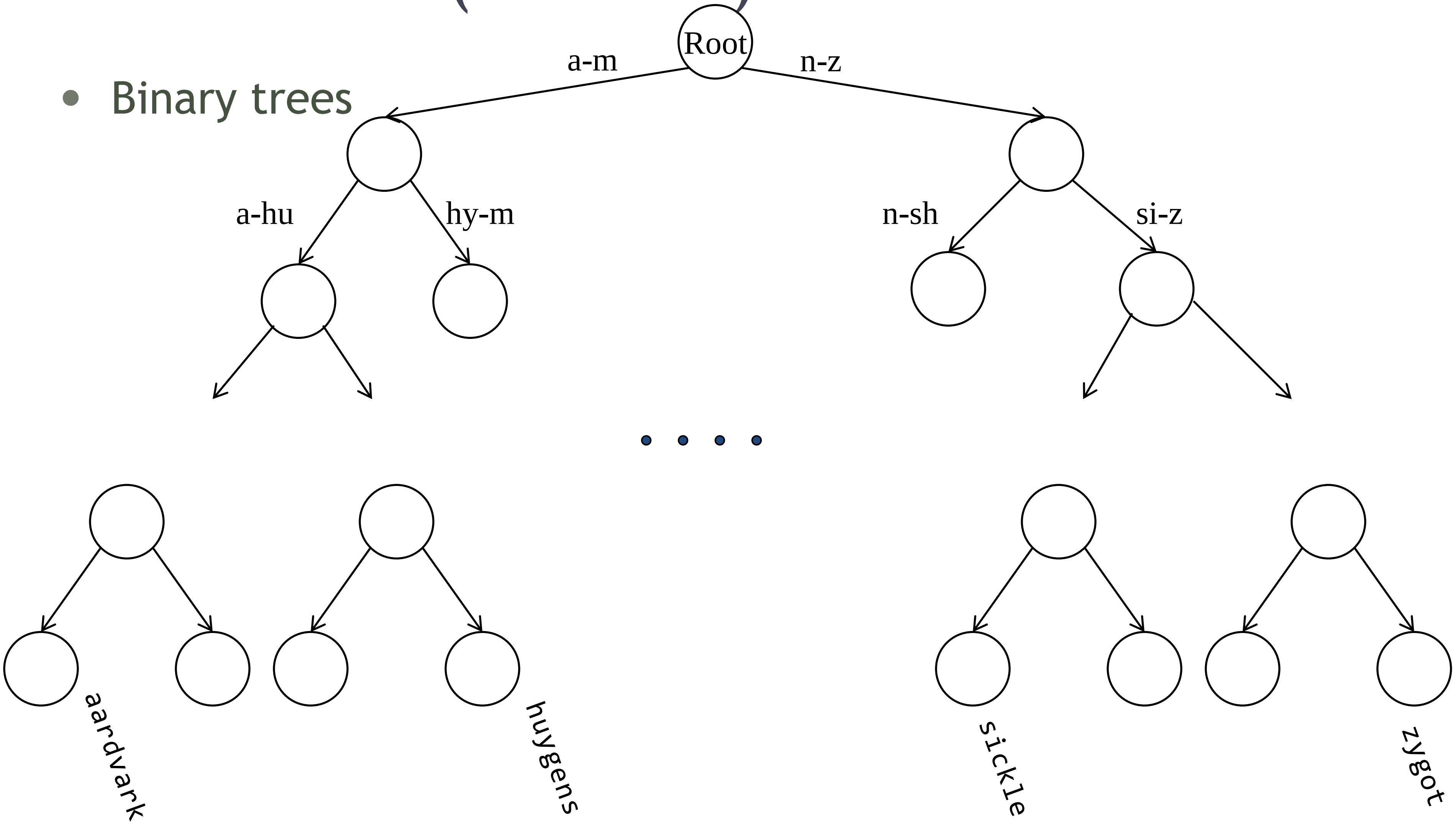
| Término | Frecuencia de documentos | Puntero al archivo de postings |
|---|---|---|
| a | 656.265 | → |
| abad | 65 | → |
| … | | |
| zutano | 20 | → |

# Diccionario (o léxico)

- Hashtables
  - cada término del vocabulario se “hashea” a un entero
- La búsqueda es más rápida que en un árbol, O(1)
- No es fácil buscar variantes menores (*septiembre/setiembre*)
- No permite búsqueda por prefijos (recuperación tolerante)

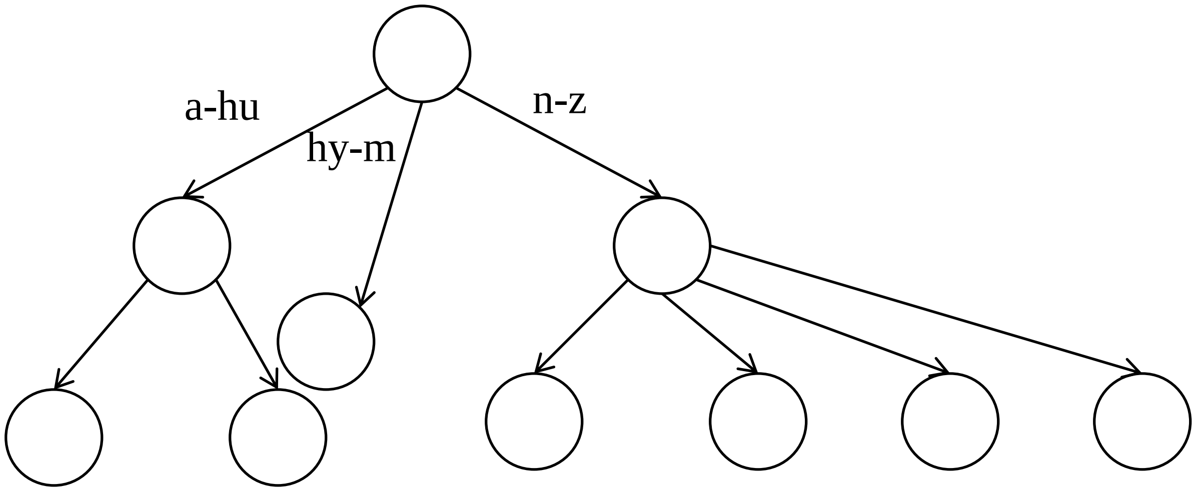
# Diccionario (o léxico)

- Binary trees



# Diccionario (o léxico)

- B-trees (árbol balanceado de búsqueda)
  - cada nodo interno tiene un número de hijos en el intervalo [a,b], donde a y b son números naturales apropiados, por ejemplo [2,4]



# Diccionario (o léxico)

- Los buscadores usan hashtables o trees, b-trees es más usual que árboles binarios
  - la búsqueda en hashtables es O(1)
  - las hashtables no tienen una forma fácil de buscar variantes de las palabras
  - en los árboles se pueden buscar prefijos y variantes mas fácilmente
  - los árboles binarios requieren estar balanceados, los b-trees mitigan este problema

# Indexación

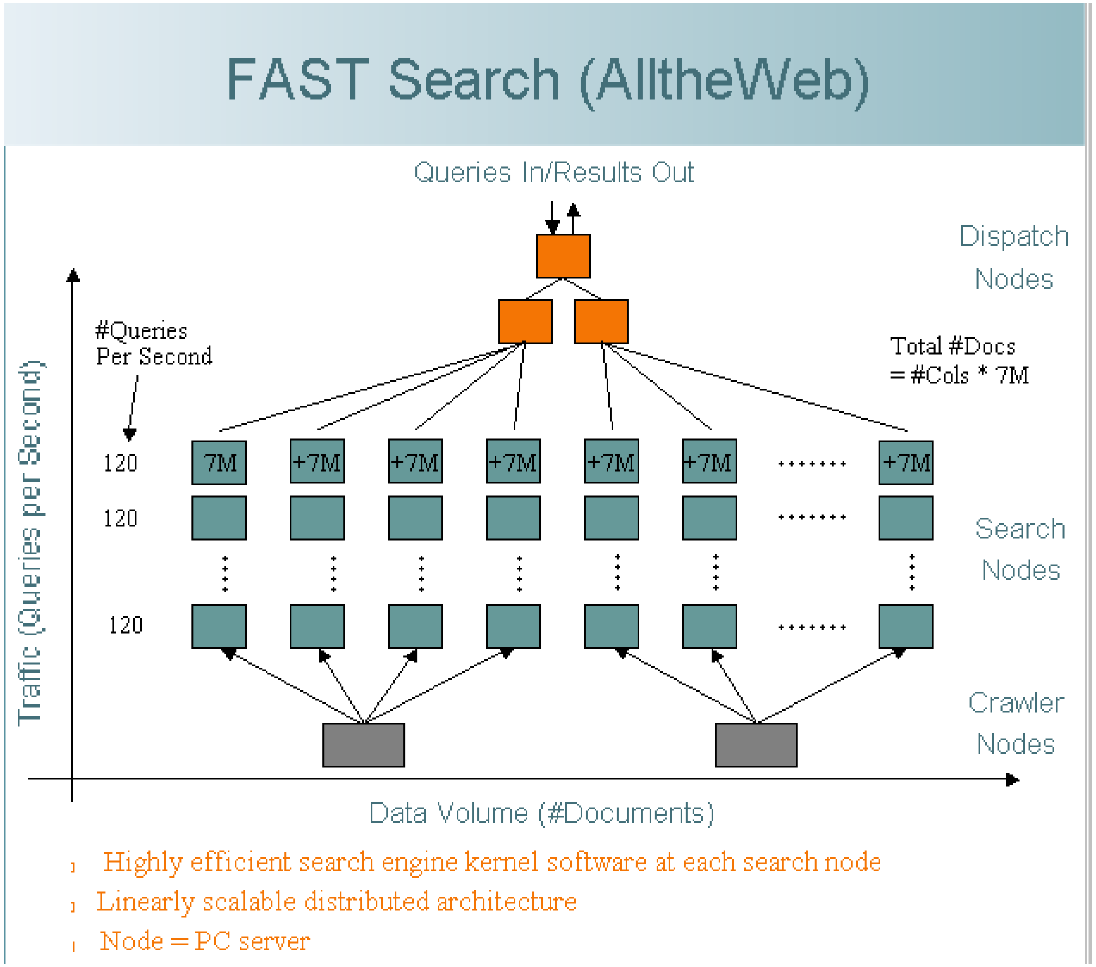
- Heap's Law:
  - el espacio requerido por el vocabulario es pequeño
    - de acuerdo a Heap's law el vocabulario crece $O(n^b)$
      - donde n es el tamaño de la colección
      - *b* es una constante dependiente de la colección entre 0.4 y 0.6 en la práctica
    - por ejemplo, TREC-3 tiene un vocabulario de 1Gb de texto que ocupa solo 5 Mb.
  - el archivo de postings demanda mucho más espacio ya que cada palabra que aparece en el texto se referencia una vez en la estructura, el espacio extra es $O(n)$

# Indexación

- La mayoría de los índices son variaciones del archivo invertido
  - algunos buscadores usan eliminación de stop-words para reducir el tamaño del índice
  - se incluye en el índice información para dar alguna idea acerca del documento recuperado, descripción corta de la página Web
- Los sistemas de IR dividen los índices en diferentes máquinas, manejando cada una diferentes porciones del archivo invertido
- Otros sistemas duplican los datos en muchas máquinas, las consultas se distribuyen entre las máquinas
- La mayoría hace una combinación de las anteriores

# Indexación distribuida

- Los datos de las páginas son divididos entre varias máquinas y, a su vez, cada partición es asignada a múltiples máquinas
- Cada fila puede manejar determinada cantidad de consultas por segundo
- Cada columna maneja cierta cantidad de páginas
- Para manejar más consultas se pueden agregar más filas

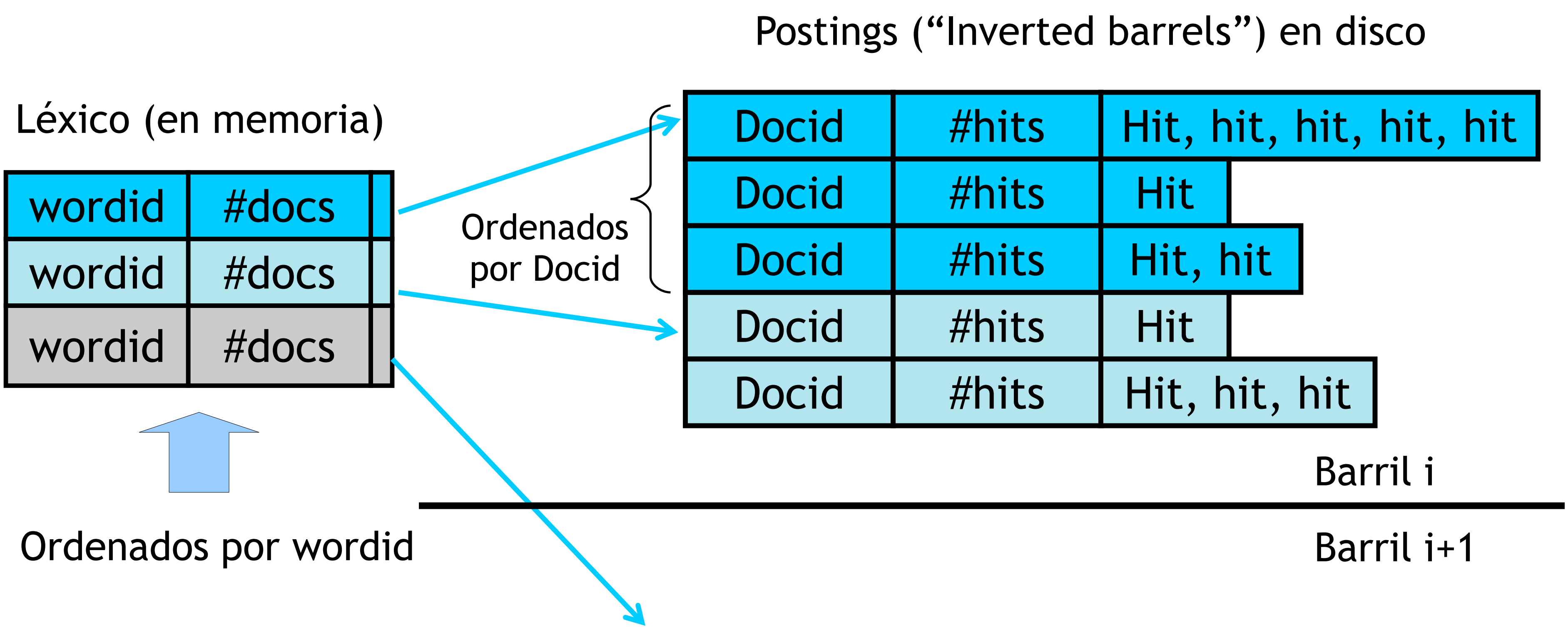


# Indexación distribuida

- Google:
  - El indexador convierte cada documento en una colección de hit lists y los pone en barriles, ordenados por ID
  - También crea una base de datos de links:
    - Hit: <wordID, posición en el doc, font info, hit type>
    - Hit type: Plain o fancy
    - Fancy hit: aparece en la URL, title, anchor text, metatag
  - Ordena cada barril por wordID para crear el archivo invertido
  - También crea un léxico:
    - Léxico: <wordID, offset en el archivo invertido>
    - El léxico es usualmente cacheado en memoria

# Indexación distribuida

- Cada barril contiene postings para un rango de wordIDs



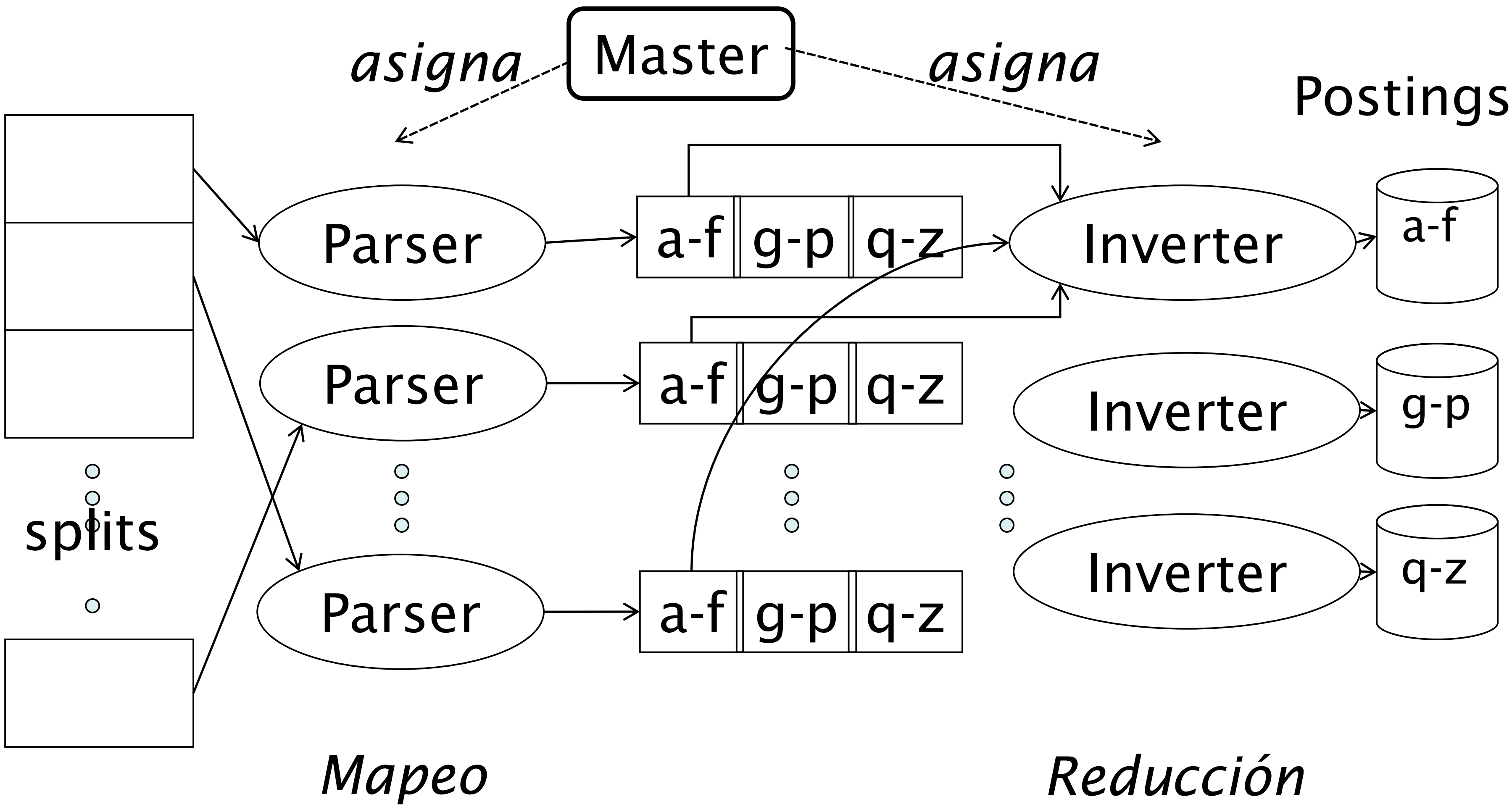
# Indexación distribuida: MapReduce

- Se divide la colección de documentos en partes (splits)
- Tiene una máquina master dirigiendo el proceso de indexación
- Particiona el indexado en un conjunto de tareas paralelas
- La máquina master asigna cada tarea a una máquina del pool
- Dos tipos de tareas paralelas:
  - Parseadores
  - Invertidores

# Indexación distribuida

- La máquina maestra asigna un split a una máquina parseadora
- El parser toma un documento a la vez y devuelve pares (término, documento)
- Los parsers escriben los pares en j particiones
- Cada partición para un rango de letras inciales
    - Por ejemplo, a-f, g-p, q-z, para j=3
- Para completar la indexación falta invertir el índice
- Un invertidor junta todos los pares (termino, documento) para una partición
- Ordena y escribe la lista de postings

# Indexación distribuida



# Lucene

## Análisis y Recuperación de Información 2016

Prof. Dr. Marcelo G. Armentano

# ¿Qué es Lucene?

- API de Recuperación de Información
  - Indexación
  - Búsqueda
- Open-Source
- Implementada en Java
- No realiza ninguna suposición acerca de qué se está indexando (y buscando)

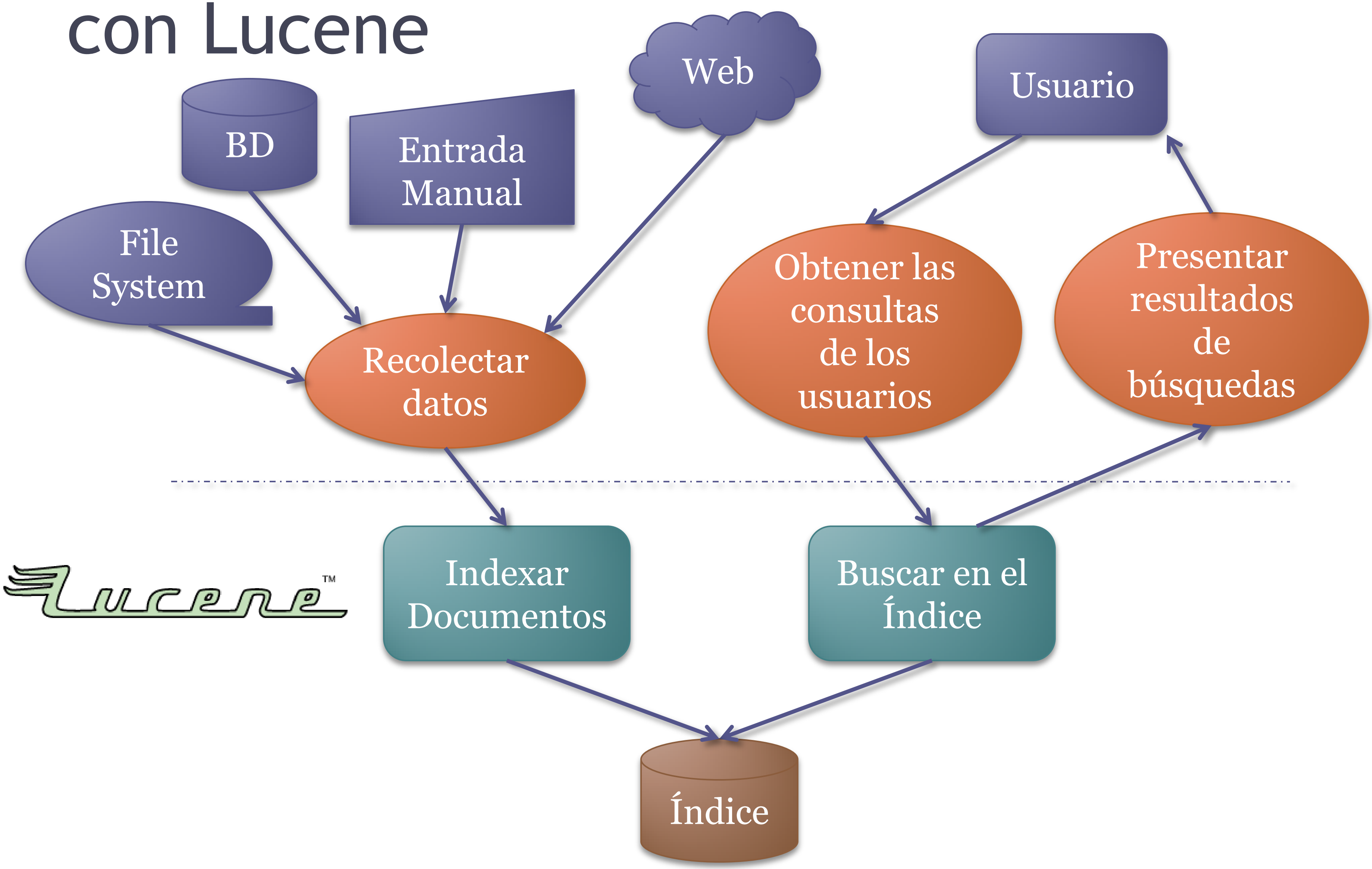
# Características de Lucene

- Indexación
  - 150GB/hora
  - Solo 1MB heap
  - Indexación incremental tan rápida como la indexación por lotes
  - El tamaño del índice es ~20-30% el tamaño del texto indexado
- Búsqueda
  - Orden de relevancia en resultados
  - Varios tipos de consulta
  - Búsqueda por campos
  - Ordenamiento por cualquier campo
  - Búsqueda en múltiples índices con combinación de resultados
  - Actualización y búsquedas simultáneas

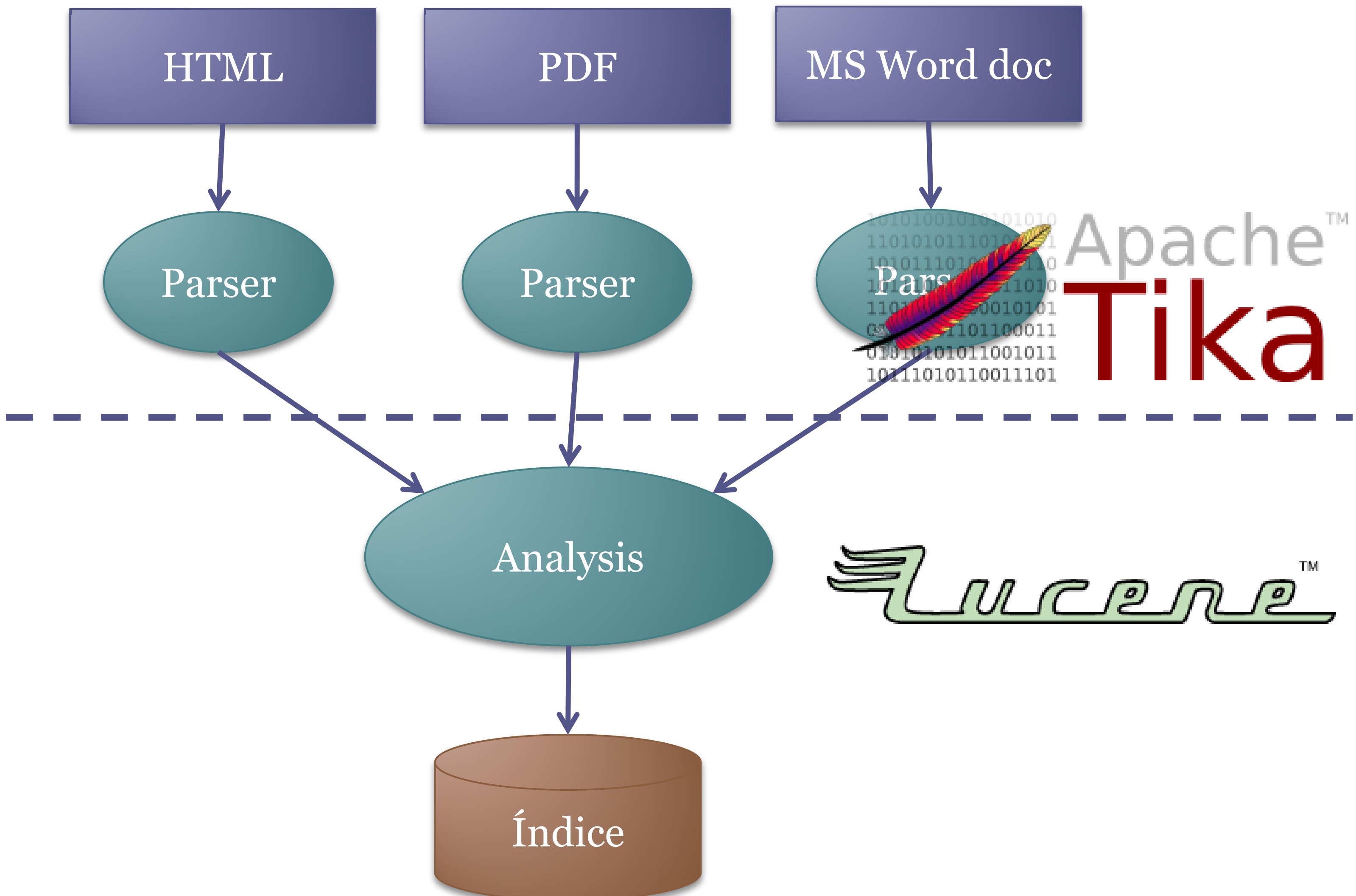
# Lucene

- Elementos de entrada para el proceso de indexación
  - objetos de tipo **Document**
    - conjunto de objetos de tipo **Field**,
      - Nombre del campo : contenido (texto plano)
- Elementos de entrada para el proceso de búsqueda:
  - query strings
  - objetos de tipo **Query**
- Almacena los índices como archivos en disco
- No proporciona un web crawler

# Integración típica de una aplicación con Lucene

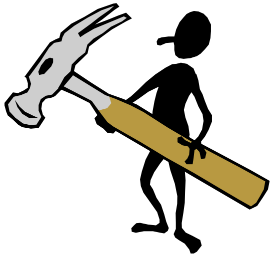


# Indexación


HTML
PDF
MS Word doc
Parser
Parser
Pars
Apache
Tika
Analysis
Lucene
Índice

# Indexación – Clases principales

- IndexWriter
- Directory
    - FSDirectory
    - RAMDirectory
- Analyzer
- Document
- Field

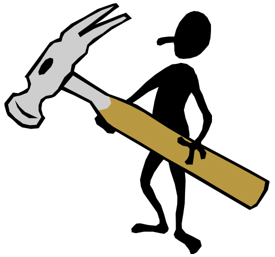


# Creación/apertura de un índice

```
Directory dir = FSDirectory.open(new File(indexPath));
Analyzer analyzer = new SimpleAnalyzer(Version.LUCENE_40);
IndexWriterConfig iwc =
        new  IndexWriterConfig(Version.LUCENE_40,  analyzer);

if (create) {
        iwc.setOpenMode(OpenMode.CREATE);
} else {
        iwc.setOpenMode(OpenMode.CREATE_OR_APPEND);
}

IndexWriter writer = new IndexWriter(dir, iwc);
```

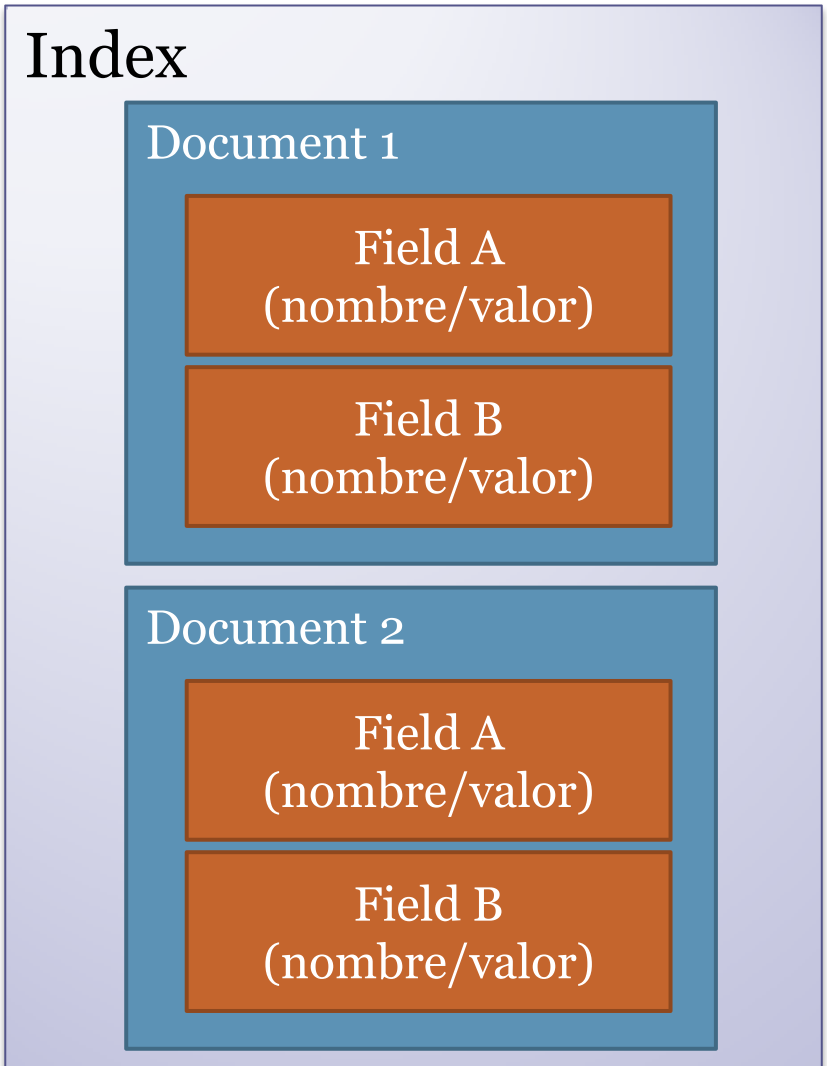


# Proceso de Indexación (1)

- Conversión a texto
  - Los datos deben convertirse a texto plano
  - Objetos `Document` y `Field` de Lucene

  - Utilizar parsers para convertir documentos pdf, word, html, xml, etc a texto plano

# Documentos y Campos




**to**: *Juan*
**from**: *María*
**subject**: *cenamos?*
**body**: *Salgamos a cenar esta noche!*

# Campos mínimos (sugeridos)

| Campo | Utilizado para |
| --- | --- |
| Path al documento | Acceder al documento original luego de una búsqueda |
| Fecha de modificación | Determinar si es necesario re-indexar el documento |
| Contenido del archivo | Campo principal por el que se realizarán las búsquedas |

Campo "**all**" → forma sencilla de permitir búsquedas por cualquier campo

# Indexación – Tipos de campos

Field(String name, String value, Store store)

- Field.Store.YES
- Field.Store.NO

# Indexación – Tipos de campos

- IntField
- LongField
- FloatField
- DoubleField
- StringField: se indexa pero no se tokeniza
- TextField: se tokeniza y se indexa
- StoredField: se almacena "tal cual es", y por lo tanto es retornado junto con el documento.
  - A cualquier tipo de campo se lo puede crear como Field.Store.YES

# Proceso de Indexación (2)

- Análisis
  - iw.addDocument(Document) *iw es un IndexWriter*
    - Se divide el texto en *tokens*
      - Pre-tokenización
        - Eliminación de tags HTML
        - Transformar o eliminar texto que coincida con ciertos patrones
      - Post-tokenización
        - Stemming
        - Filtrado de Stop words
        - Normalización del texto
        - Expansión de sinónimos

# Analizadores

- Aplican un conjunto de operaciones opcionales
  - Pasar a minúsculas → **SimpleAnalyzer**
  - Eliminación de StopWords → **StopAnalyzer**
  - Stemming → **PorterStemFilter**
  - **StandardAnalyzer**
    - Elimina signos de puntuación
    - Separa las palabras unidas por guiones
    - Reconoce direcciones de internet y dominios web
    - Convierte a minúsculas
    - Remueve stopwords

# Analizadores

- ¿Qué analizador utilizar?
  - Depende de la aplicación
  - Tener cuidado con "sobre-analizar" el texto
  - Utilizar el mismo analizador para el indexado que para las búsquedas
    - Salvo que se quiera hacer expansión de sinónimos, corrección de errores, etc. en las búsquedas

# Proceso de Indexación (3)

- Escritura en el índice
  - Índice Invertido → ¿Qué documentos tienen la palabra X?
- Operaciones básicas
  - Agregar documentos
  - Eliminar documentos
    - A través de IndexReader

```
IndexReader reader = IndexReader.open(dir);
reader.deleteDocument(new Term(“city”, “Amsterdam”));
reader.close();
```

  - Deshacer eliminación de documentos

# *Boosting* de campos

- Favorece algunos campos por sobre otros
- Por defecto los campos tienen un factor de boosting de 1.0

```
Field senderNameField = TextField(“senderName”, senderName);

Field subjectField = TextField(“subject”, subject);

subjectField.setBoost(1.2);
```

# Segmentos

- Un índice puede estar formado por múltiples subíndices o *segmentos*
  - Completamente independientes
- Un índice evoluciona:
  - Creando nuevos segmentos para los documentos agregados recientemente
  - Uniendo segmentos existentes

# Segmentos: composición

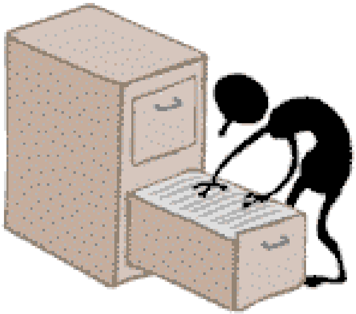
- Nombre de los campos indexados (Fields)
- Valores de los campos Stored
  - Lista de pares <atributo;valor>
- Diccionario de términos
  - Todos los términos junto a la cantidad de documentos en los que aparece el término y un puntero a los datos de frecuencia y proximidad
- Frecuencia de términos
  - Para cada término, los Ids de los documentos que lo contienen y la frecuencia del término en el documento
- Proximidad de términos
  - Para cada término, las posiciones en las que el término aparece en el documento

# Segmentos: composición

- Factores de normalización de campos
  - Para cada campo almacena el valor por el que es multiplicado el score si hay un hit en ese campo
- Vectores de términos
  - Para cada campo del documento, el vector de términos y su frecuencia
- Documentos eliminados

# Búsqueda – Clases principales

- IndexSearcher
- Term
- Query
- QueryParser
- TopDocs



# Búsqueda - Clases principales

- IndexSearcher

```
FSDirectory dir = new SimpleFSDirectory("/temp/index")
IndexSearcher is = new IndexSearcher (dir );
Query q = new TermQuery(new Term("subject", "java"));
TopDocs hits = is.search(q);
```

- Term
- Query
- QueryParser
- TopDocs

# Búsqueda - Clases principales

- IndexSearcher
- Term

```
Query q = new TermQuery(new Term("contents", "lucene"));
TopDocs hits = is.search(q);
```

- Query
- QueryParser
- TopDocs

# Búsqueda - Clases principales

- IndexSearcher
- Term
- Query
  - TermQuery
  - BooleanQuery
  - PhraseQuery
  - PrefixQuery
  - RangeQuery
  - SpanQuery
- QueryParser
- TopDocs

```
title: ”trabajo práctico” AND
text:recuperación
```

# Búsqueda - Clases principales

- IndexSearcher
- Term
- Query
- TopDocs
- QueryParser
  - “Java OR Lucene” → instancia de BooleanQuery

```
QueryParser qp = new QueryParser(“contents”,
                                  new SimpleAnalyser());

Query q = qp.parse(“+RECUPERACION +INFORMACION -JAVA”);
```

# Búsqueda - Clases principales

- MultiFieldQueryParser

```
MultiFieldQueryParser queryParser = new MultiFieldQueryParser(
            Version.LUCENE_41,
            new String[]{"title", "content", "description"},
            new StandardAnalyzer(Version.LUCENE_41));

Query query = queryParser.parse("este es el query");
```

# Búsqueda – Expresiones (1)

| Expresión | Documentos que… |
|---|---|
| java | Contienen el término *java* en el campo por defecto |
| java lucene<br>java or lucene | Contienen el término *java* o *lucene*, o ambos, en el campo por defecto |
| +java +lucene<br>java AND lucene | Contienen ambos términos, *java* y *lucene* en el campo por defecto |
| title:lucene | Contienen el término *lucene* en el campo *title* |
| title:extreme -subject:sports<br>title:extreme AND NOT subject:sports | Tienen *extreme* en el título y no tienen *sports* en el subject |
| (agile OR extreme) AND methodology | Contienen *methodology* y deben contener *agile* y/o *extreme* en el campo por defecto |

# Búsqueda – Expresiones (2)

| **Expresión** | **Documentos que…** |
|---|---|
| title:"lucene in action" | Contienen la frase exacta "*lucene in action*" en el campo title |
| title:"lucene action"~5 | Contienen los términos *lucene* y *action* a distancia 5 una de otra |
| java* | Contiene términos que comienzan con *java*, tales como *javaspaces, javaserver* y *java.net* |
| java~ | Contiene términos que son cercanos a la palabra *java*, tales como l*ava* |
| lastmodified:[1/1/09 TO 12/31/09]<br><br>title:{Aida TO Carmen} | Tienen el campo lastmodified con valores entre las fechas el 1º de enero de 2009 y el 31 de diciembre de 2009<br><br>El título está entre Aida y Carmen, no incluyendo estos extremos |

# Búsqueda – Clases principales

- IndexSearcher
- Term
- Query
- QueryParser
- TopDocs

# TopDocs

- Luego de invocar al método *search(Query)* de *IndexSearcher*, obtenemos un objeto **TopDocs**, *que contiene un array de objetos* **ScoreDoc**
  - ScoreDoc[] scoreDocs
- ScoreDoc
  - Id del documento
  - score

# Scoring

$$\mathrm{score}(q,d) = \mathrm{coord}(q,d) \cdot \mathrm{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left( \mathrm{tf}(t,d) \cdot \mathrm{idf}(t)^2 \cdot t.\mathrm{getBoost}() \cdot \mathrm{norm}(t,d) \right)$$

# Scoring

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left( \text{tf}(t \text{ en } d) \cdot \text{idf}(t)^2 \cdot t.\text{getBoost}() \cdot \text{norm}(t, d) \right)$$

Factor basado en cuántos términos de la consulta aparecen en el documento

# Scoring

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{\text{t in q}} \left( \text{tf}(\text{t en d}) \cdot \text{idf}(t)^2 \cdot \text{t.getBoost}() \cdot \text{norm}(t,d) \right)$$

$$queryNorm(q) = \frac{1}{\sqrt{q.getBoost()^2 * \sum_{t \in q} (idf(t) * t.getBoost())^2}}$$

# Scoring

$$\mathrm{score}(q,d) = \mathrm{coord}(q,d) \cdot \mathrm{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left( \mathrm{tf}(t,d) \cdot \mathrm{idf}(t)^2 \cdot t.\mathrm{getBoost}() \cdot \mathrm{norm}(t,d) \right)$$

$$tf(t,d) = \sqrt{frecuencia(t,d)}$$

# Scoring

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left( \text{tf}(t,d) \cdot \text{idf}(t)^2 \cdot t.\text{getBoost}() \cdot \text{norm}(t,d) \right)$$

$$idf(t) = 1 + \log\left(\frac{numDocs}{docFreq + 1}\right)$$

# Scoring

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left( \text{tf}(t,d) \cdot \text{idf}(t)^2 \cdot \text{t.getBoost}() \cdot \text{norm}(t,d) \right)$$

Peso dado al término en el query

# Scoring

$$\mathrm{score}(q,d) \;=\; \mathrm{coord}(q,d) \cdot \mathrm{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left( \mathrm{tf}(t,d) \cdot \mathrm{idf}(t)^2 \cdot t.\mathrm{getBoost}() \cdot \mathrm{norm}(t,d) \right)$$

$$norm(t,d) = lengthNorm * \prod_{\substack{campo\ f\ en\ d \\ llamado\ como\ t}} f.boost()$$

# TopDocs

- Paginado
  - Mantener disponibles las instancias originales de TopDocs e IndexSearcher mientras el usuario navega por los resultados
  - Realizar un nuevo query cada vez que el usuario navega a una nueva página
    - Los resultados se muestran a partir de una página determinada

# Paginado

```
Query query = qp.parse(searchTerm);
TopDocs hits = searcher.search(query, maxNumberOfResults);
ArrayLocation arrayLocation =
    paginator.calculateArrayLocation(hits.scoreDocs.length,
    pageNumber, pageSize);

for (int i = arrayLocation.getStart() - 1; i < arrayLocation.getEnd(); i++) {

    int docId = hits.scoreDocs[i].doc;

    //cargo el documento
    Document doc = searcher.doc(docId);
    String filename = doc.get("filename");
    String contents =  doc.get(searchField);

}
```

# Paginado

```
public class Paginator {

  public ArrayLocation calculateArrayLocation(int totalHits, int pageNumber, int pageSize) {
    ArrayLocation al = new ArrayLocation();
    if (totalHits < 1 || pageNumber < 1 || pageSize < 1) {
      al.setStart(0);        al.setEnd(0);        return al;
    }
    int start= 1 + (pageNumber -1) * pageSize;
    int end = Math.min(pageNumber * pageSize, totalHits);
    if (start > end) {
      start = Math.max(1, end - pageSize);
    }
    al.setStart(start);      al.setEnd(end);      return al;
  }
}
```

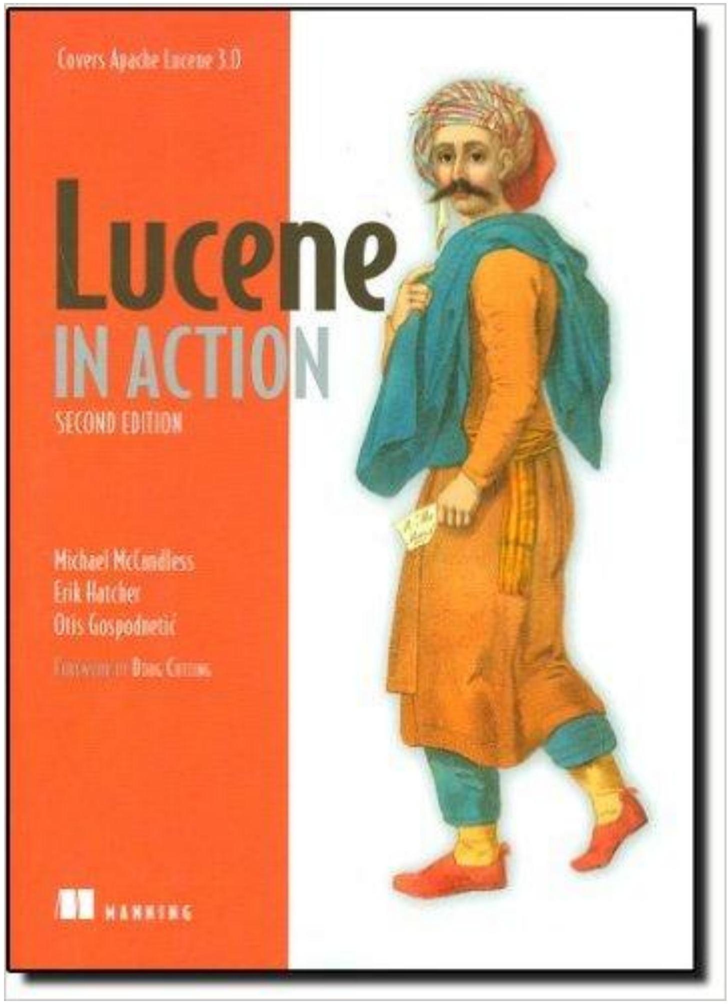
# Paginado

- Ventajas
  - Menor tiempo de respuesta
  - Menor uso de memoria
  - Menor uso de la red

→ Mejor experiencia para el usuario

| Paginado | MaxResults | Cantidad de hits | Tiempo de respuesta |
|---|---|---|---|
| Si | 500 | 500 | 5 ms |
| No | 500 | 500 | 186 ms |
| Si | 1000 | 1000 | 5 ms |
| No | 1000 | 1000 | 415 ms |
| Si | 5000 | 2110 | 5 ms |
| No | 5000 | 2110 | 1007 ms |

# Bibliografía

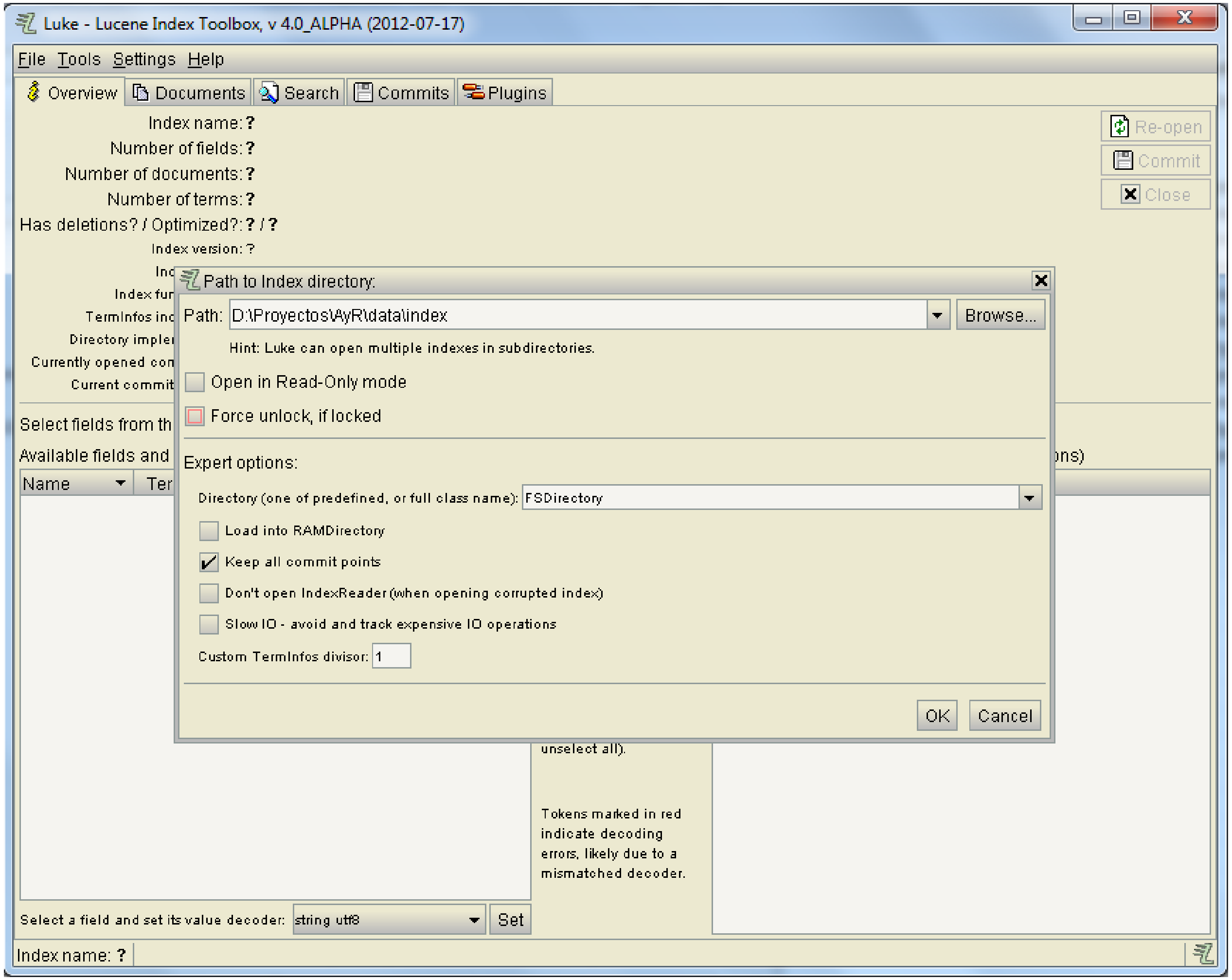


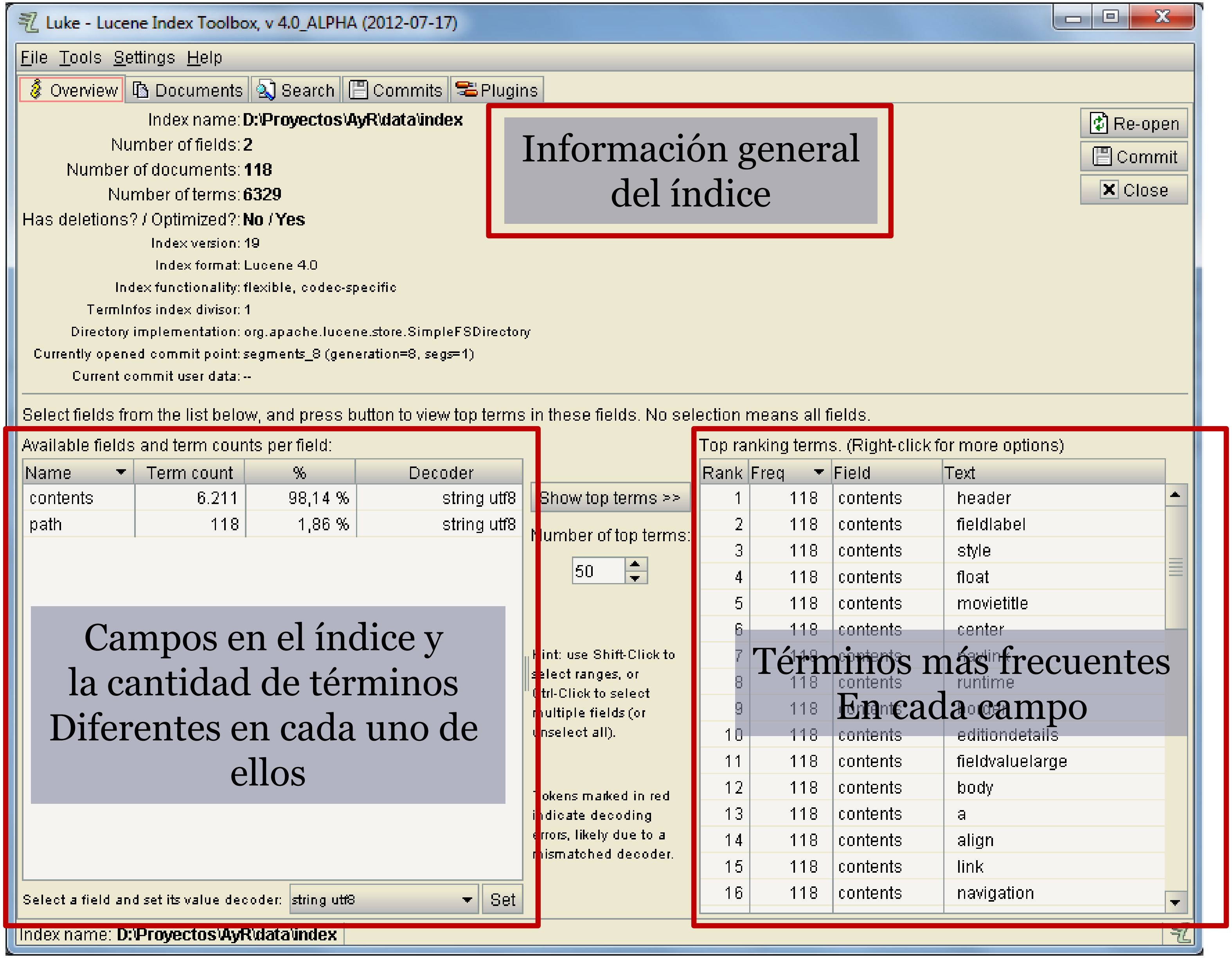

**Lucene in Action, Second Edition: Covers Apache Lucene 3.0** 2nd Edition by Michael McCandless, Erik Hatcher , Otis Gospodnetic

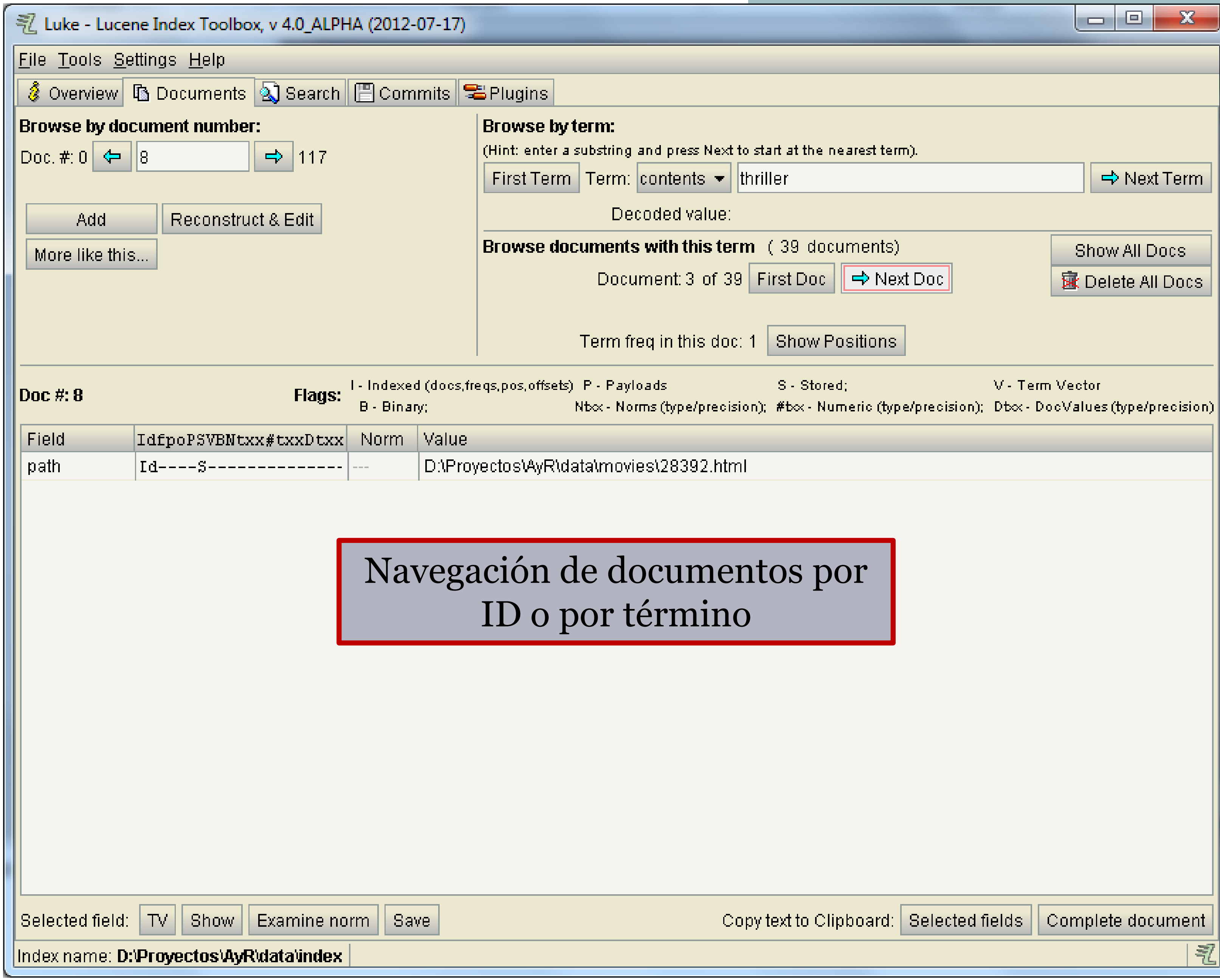
# Luke

# Luke

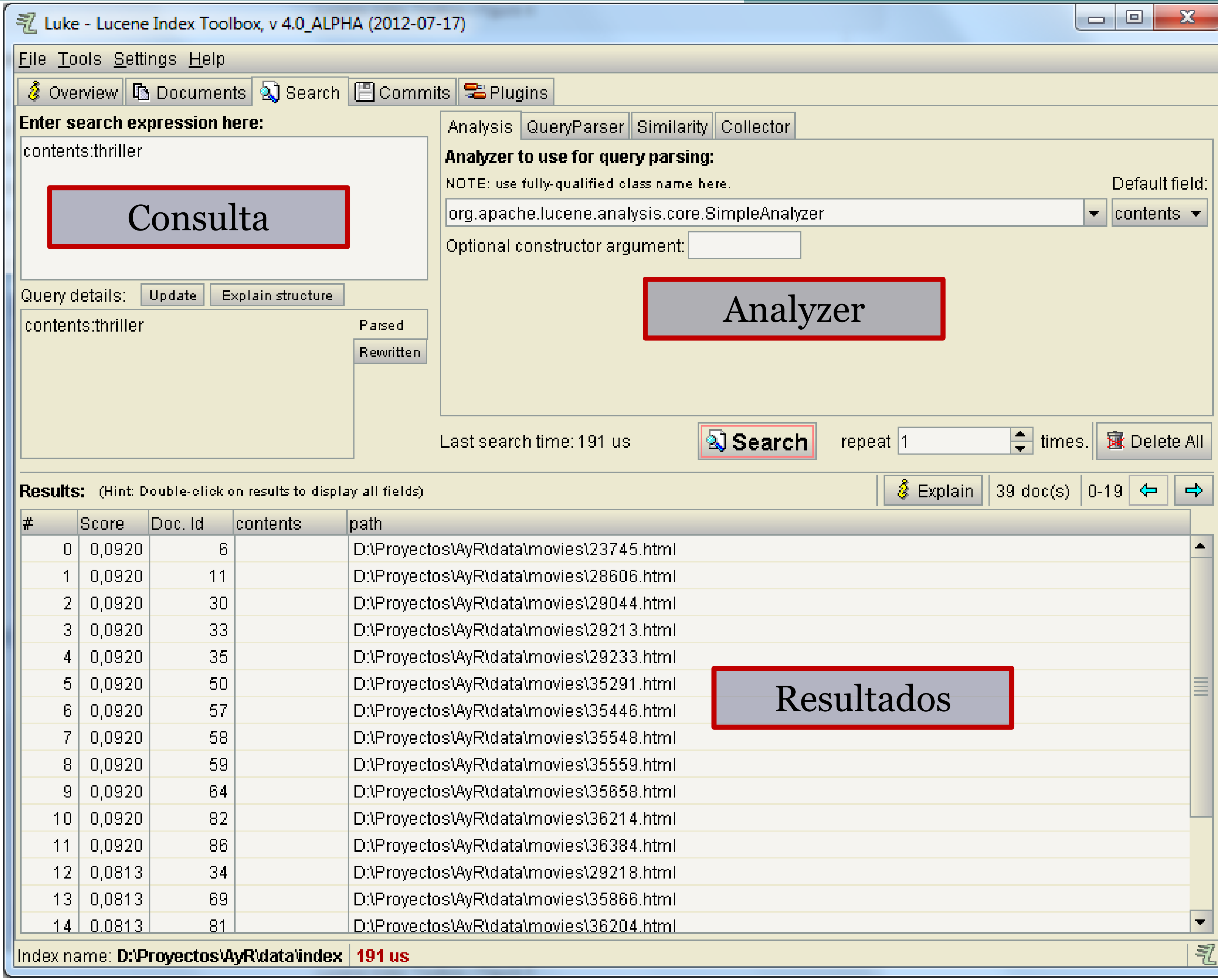
- Herramienta de diagnóstico para acceder a índices Lucene
  - Navegar documentos indexados
  - Ver documentos
  - Obtener los términos más frecuentes
  - Ejecutar búsquedas y navegar por los resultados
  - Eliminar documentos del índice
  - Optimizar índices
  - …

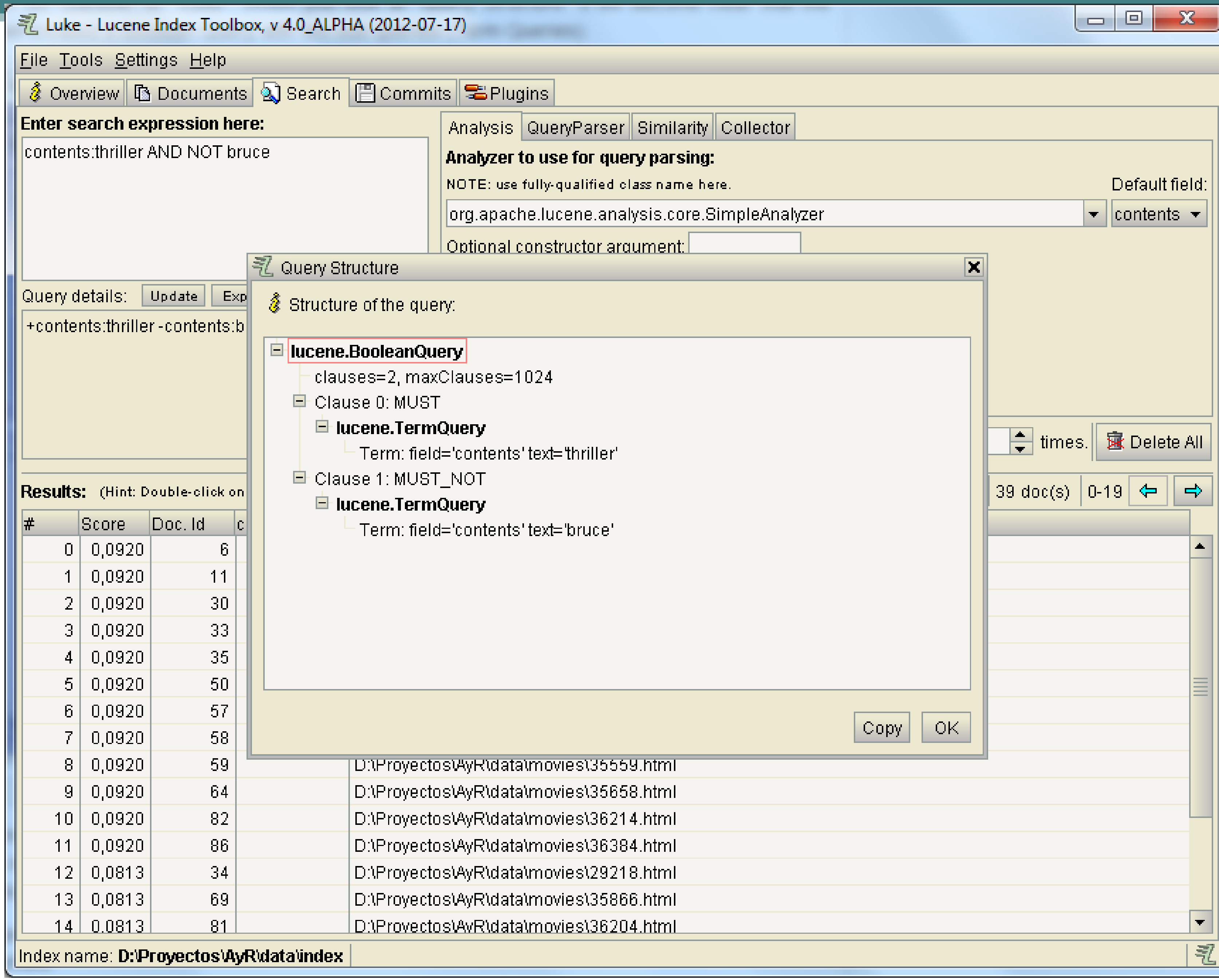

Luke - Lucene Index Toolbox, v 4.0_ALPHA (2012-07-17)
File Tools Settings Help
Overview Documents Search Commits Plugins
Index name: ?
Number of fields: ?
Number of documents: ?
Number of terms: ?
Has deletions? / Optimized?: ? / ?
Index version: ?
Re-open
Commit
Close
Path to Index directory:
Path: D:\Proyectos\AyR\data\index
Browse...
Hint: Luke can open multiple indexes in subdirectories.
Open in Read-Only mode
Force unlock, if locked
Expert options:
Directory (one of predefined, or full class name): FSDirectory
Load into RAMDirectory
Keep all commit points
Don't open IndexReader (when opening corrupted index)
Slow IO - avoid and track expensive IO operations
Custom TermInfos divisor: 1
OK
Cancel
Tokens marked in red indicate decoding errors, likely due to a mismatched decoder.
Select a field and set its value decoder: string utf8
Set
Index name: ?


Luke - Lucene Index Toolbox, v 4.0_ALPHA (2012-07-17)
File Tools Settings Help
Overview Documents Search Commits Plugins
Index name: D:\Proyectos\AyR\data\index
Number of fields: 2
Number of documents: 118
Number of terms: 6329
Has deletions? / Optimized?: No / Yes
Index version: 19
Index format: Lucene 4.0
Index functionality: flexible, codec-specific
TermInfos index divisor: 1
Directory implementation: org.apache.lucene.store.SimpleFSDirectory
Currently opened commit point: segments_8 (generation=8, segs=1)
Current commit user data: --
Información general del índice
Re-open
Commit
Close
Select fields from the list below, and press button to view top terms in these fields. No selection means all fields.
Available fields and term counts per field:
Name Term count % Decoder
contents 6.211 98,14 % string utf8
path 118 1,86 % string utf8
Campos en el índice y la cantidad de términos Diferentes en cada uno de ellos
Select a field and set its value decoder: string utf8 Set
Show top terms >>
Number of top terms:
50
Top ranking terms. (Right-click for more options)
Rank Freq Field Text
1 118 contents header
2 118 contents fieldlabel
3 118 contents style
4 118 contents float
5 118 contents movietitle
6 118 contents center
Términos más frecuentes En cada campo
10 118 contents editiondetails
11 118 contents fieldvaluelarge
12 118 contents body
13 118 contents a
14 118 contents align
15 118 contents link
16 118 contents navigation
Index name: D:\Proyectos\AyR\data\index




Navegación de documentos por ID o por término


Luke - Lucene Index Toolbox, v 4.0_ALPHA (2012-07-17)
File Tools Settings Help
Overview Documents Search Commits Plugins
Enter search expression here:
contents:thriller
Consulta
Query details: Update Explain structure
contents:thriller
Parsed
Rewritten
Analysis QueryParser Similarity Collector
Analyzer to use for query parsing:
NOTE: use fully-qualified class name here.
Default field:
org.apache.lucene.analysis.core.SimpleAnalyzer
contents
Optional constructor argument:
Analyzer
Last search time: 191 us
Search
repeat 1 times.
Delete All
Results: (Hint: Double-click on results to display all fields)
Explain
39 doc(s)
0-19
# Score Doc. Id contents path
0 0,0920 6 D:\Proyectos\AyR\data\movies\23745.html
1 0,0920 11 D:\Proyectos\AyR\data\movies\28606.html
2 0,0920 30 D:\Proyectos\AyR\data\movies\29044.html
3 0,0920 33 D:\Proyectos\AyR\data\movies\29213.html
4 0,0920 35 D:\Proyectos\AyR\data\movies\29233.html
5 0,0920 50 D:\Proyectos\AyR\data\movies\35291.html
6 0,0920 57 D:\Proyectos\AyR\data\movies\35446.html
7 0,0920 58 D:\Proyectos\AyR\data\movies\35548.html
8 0,0920 59 D:\Proyectos\AyR\data\movies\35559.html
9 0,0920 64 D:\Proyectos\AyR\data\movies\35658.html
10 0,0920 82 D:\Proyectos\AyR\data\movies\36214.html
11 0,0920 86 D:\Proyectos\AyR\data\movies\36384.html
12 0,0813 34 D:\Proyectos\AyR\data\movies\29218.html
13 0,0813 69 D:\Proyectos\AyR\data\movies\35866.html
14 0,0813 81 D:\Proyectos\AyR\data\movies\36204.html
Resultados
Index name: D:\Proyectos\AyR\data\index 191 us


Luke - Lucene Index Toolbox, v 4.0_ALPHA (2012-07-17)
File Tools Settings Help
Overview Documents Search Commits Plugins
Enter search expression here:
contents:thriller AND NOT bruce
Analysis QueryParser Similarity Collector
Analyzer to use for query parsing:
NOTE: use fully-qualified class name here.
Default field:
org.apache.lucene.analysis.core.SimpleAnalyzer
contents
Optional constructor argument:
Query details: Update Exp
+contents:thriller -contents:b
Query Structure
Structure of the query:
lucene.BooleanQuery
clauses=2, maxClauses=1024
Clause 0: MUST
lucene.TermQuery
Term: field='contents' text='thriller'
Clause 1: MUST_NOT
lucene.TermQuery
Term: field='contents' text='bruce'
Copy OK
times. Delete All
Results: (Hint: Double-click on
39 doc(s) 0-19
# Score Doc. Id
0 0,0920 6
1 0,0920 11
2 0,0920 30
3 0,0920 33
4 0,0920 35
5 0,0920 50
6 0,0920 57
7 0,0920 58
8 0,0920 59 D:\Proyectos\AyR\data\movies\35559.html
9 0,0920 64 D:\Proyectos\AyR\data\movies\35658.html
10 0,0920 82 D:\Proyectos\AyR\data\movies\36214.html
11 0,0920 86 D:\Proyectos\AyR\data\movies\36384.html
12 0,0813 34 D:\Proyectos\AyR\data\movies\29218.html
13 0,0813 69 D:\Proyectos\AyR\data\movies\35866.html
14 0,0813 81 D:\Proyectos\AyR\data\movies\36204.html
Index name: D:\Proyectos\AyR\data\index

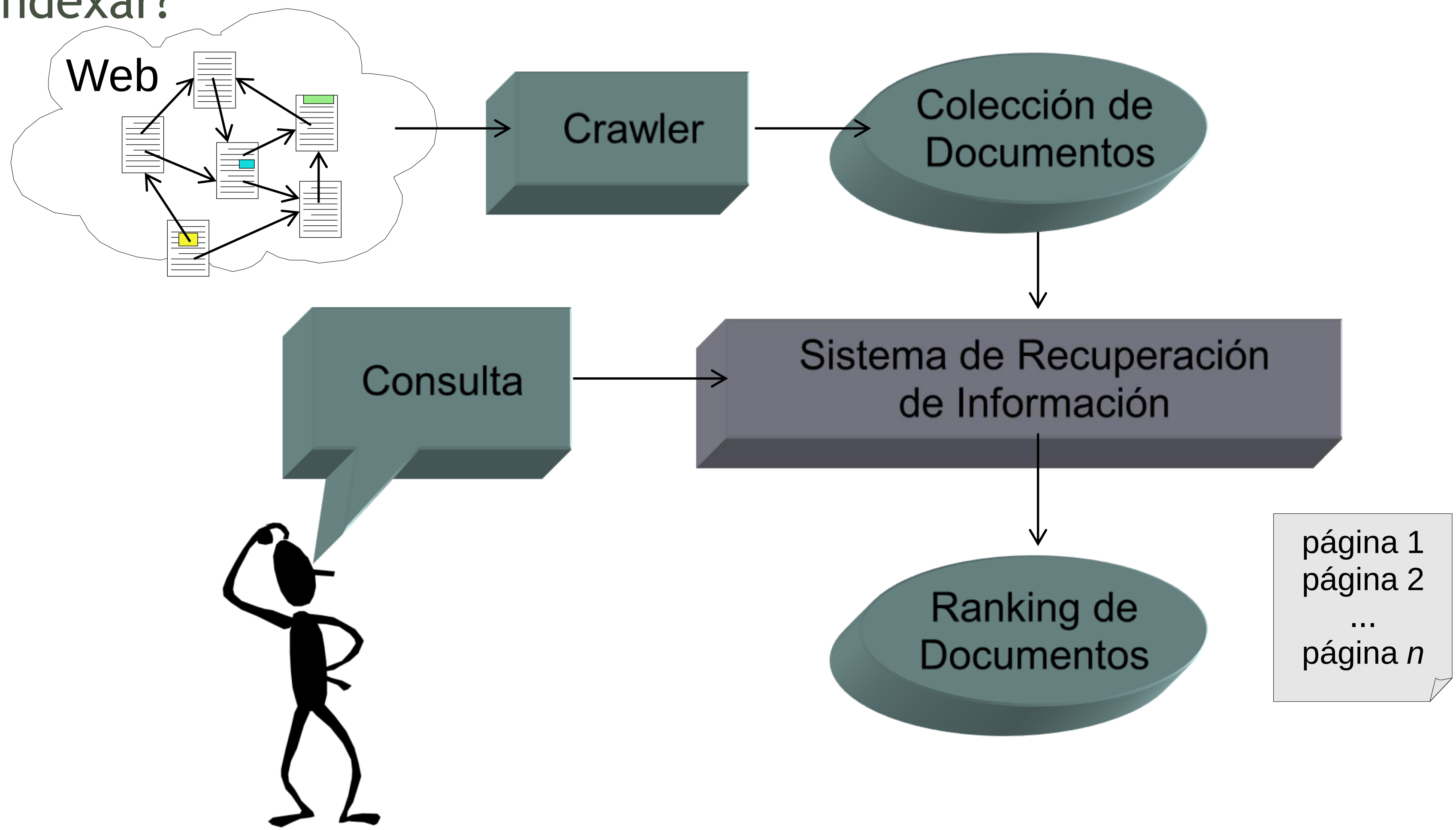
# Crawling

Análisis y Recuperación de Información
2015

Dr. Marcelo G. Armentano

# Crawling

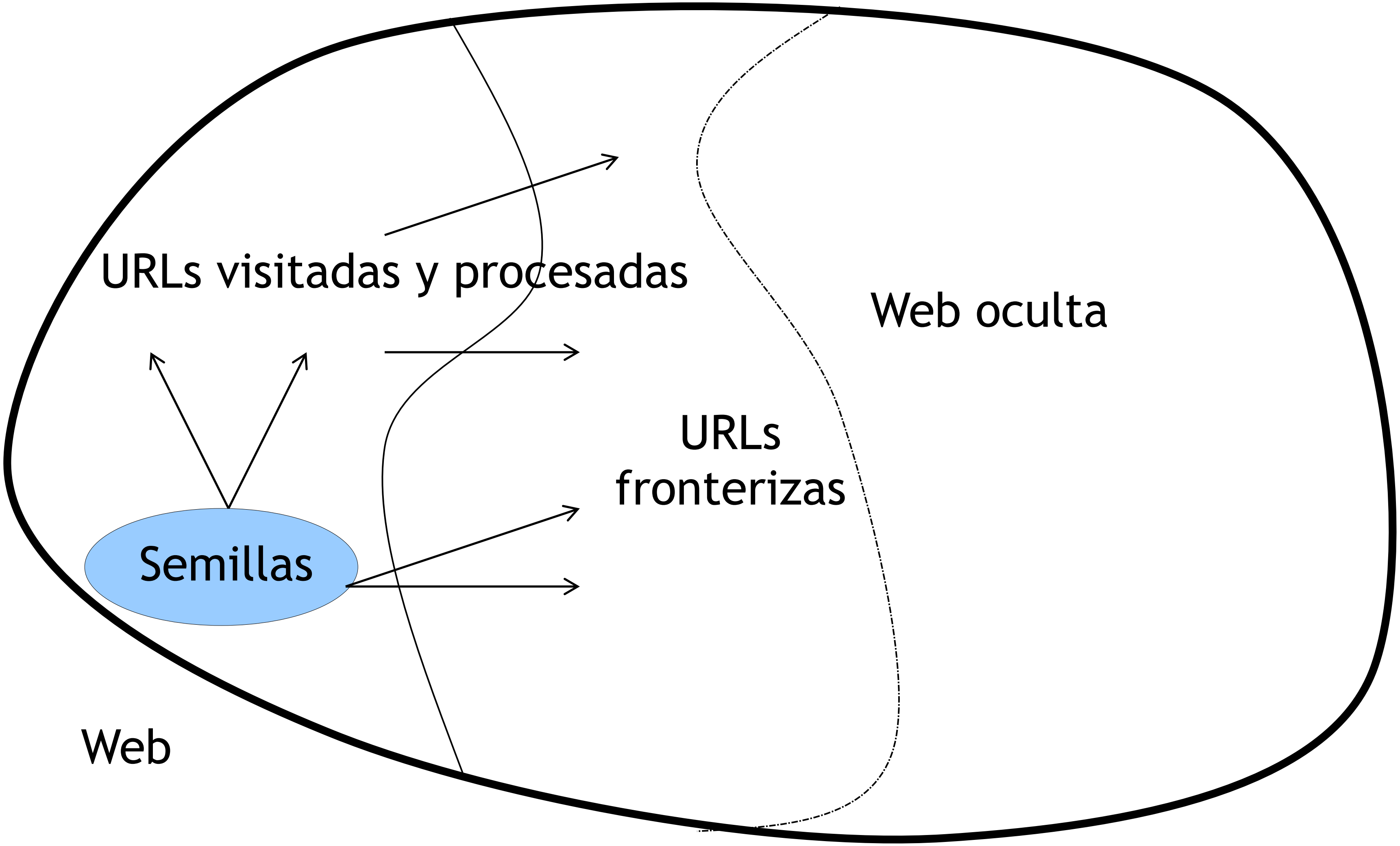
- Cómo consiguen los buscadores las páginas que deben indexar?



# Crawling

- Crawlers:
  - comienzan con sitios conocidos o “semillas”
  - descargan páginas de esos sitios
  - siguen los links de estos sitios a otros nuevos
  - descarga las páginas de los sitios nuevos
  - continúa hasta que toda la Web “alcanzable” haya sido visitada
- Los buscadores permiten a los usuarios submitir sitios Web principales para ser agregados a la lista de semillas

# Crawling


URLs visitadas y procesadas
Web oculta
URLs
fronterizas
Semillas
Web

# Crawling

- Crawler simple:
  - poner un conjunto de URLs en una cola
  - REPETIR hasta que la cola esté vacía:
    - tomar la primera página de la cola
    - SI la página no ha sido visitada todavía
      - procesar la información de la página
      - agregar a la cola todas las páginas linkeadas en la página actual
      - registrar que la página fue visitada

# Crawling

- Crawler real:
  - robustez (fallas de servidores, trampas para spiders, etc.)
  - cortesía (balance de carga, exclusión de robots, etc.)
    - los servers tienen listados de directorios fuera de límites (robots.txt)

```
User-agent:*
Disallow: /
```

Indica para todos los robots, no visitar el sitio

```
User-agent: googlebot
Disallow: /cgi-bin/
Disallow: /tmp/
Disallow: /junk/
```

El robot “googlebot” no pueden ingresar a los directorios especificados (cgi-bin, tmp, junk)

El string del robot es definido por el crawler

# Crawling

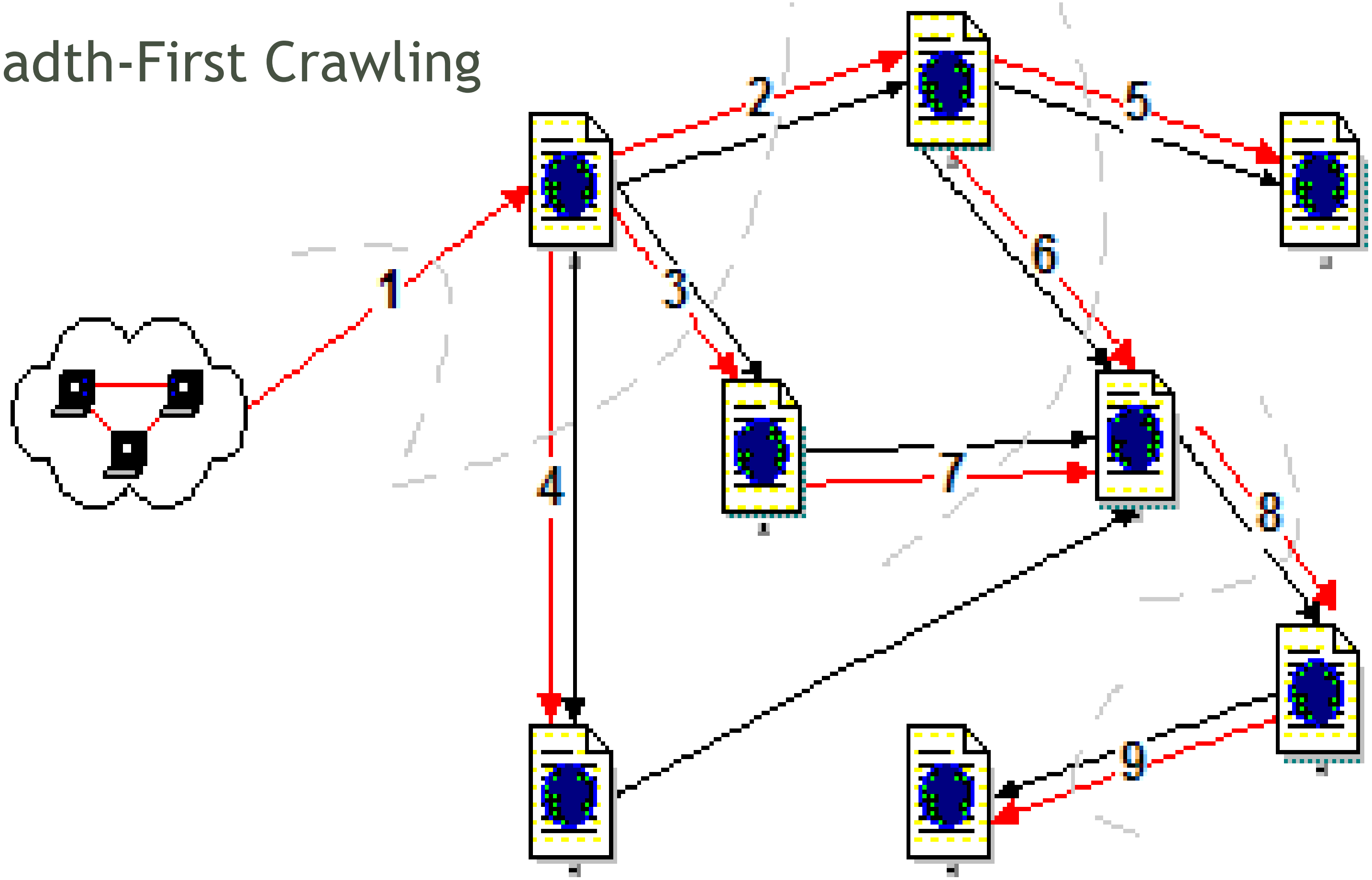
- Crawler real (cont):
  - manejo de tipos de archivos (imagenes, PDFs, etc.)
  - extensiones de URLs (scripts cgi, referencias internas, etc.)
  - reconocer páginas redundantes (idénticas y duplicadas)
  - descubrir URLs ocultas (truncadas, etc.)
  - refresco
    - determinar páginas que cambian seguido para revisitarlas más rápido

# Crawling

- Estrategia de Crawling: qué página visitar siguiente?
  - Breadth-first crawling
  - Depth-first crawling
  - Crawling enfocado
    - enfocado en un subconjunto de páginas (por ejemplo, páginas de automóviles)

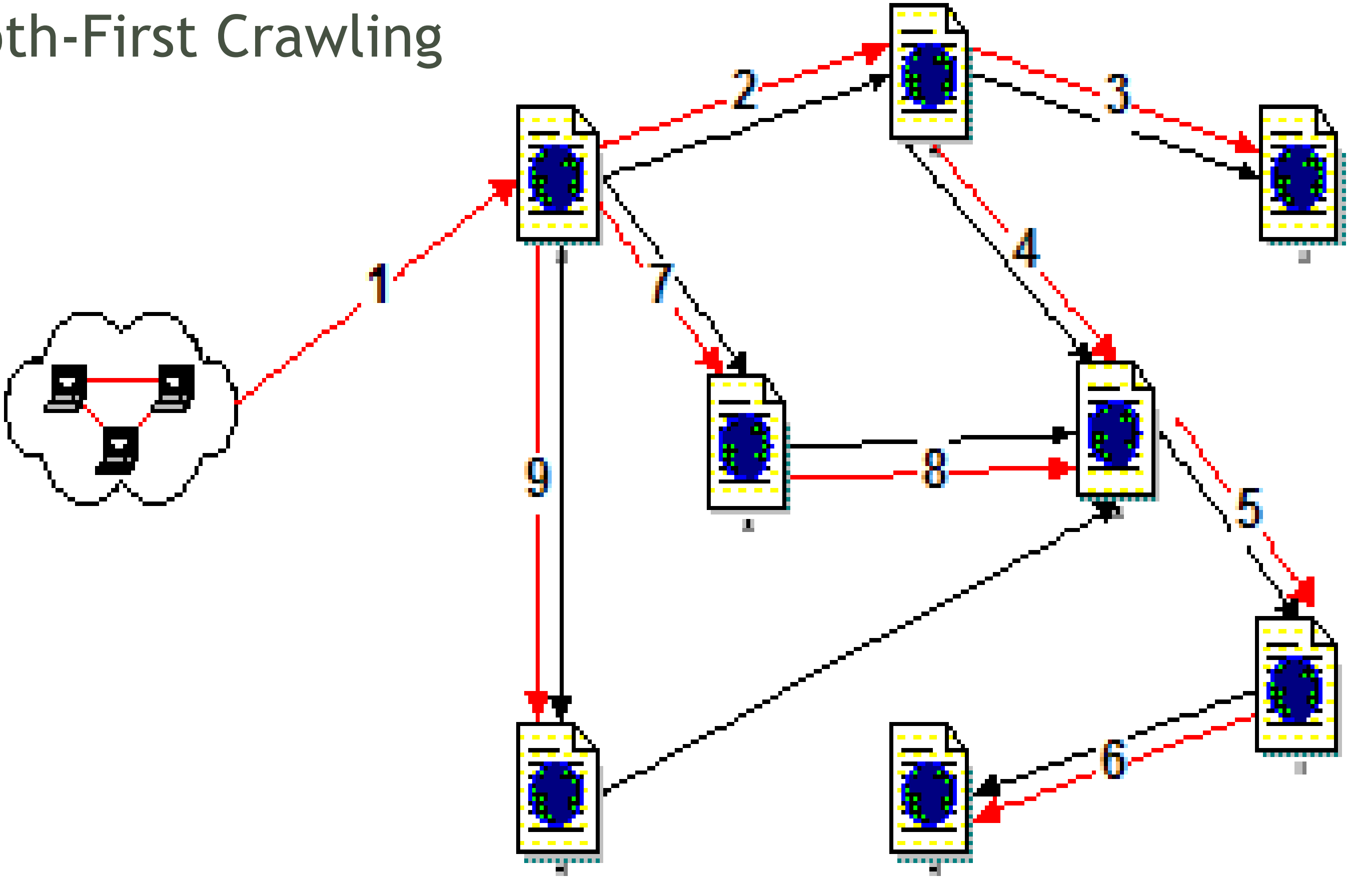
# Crawling

- Breadth-First Crawling



# Crawling

- Depth-First Crawling



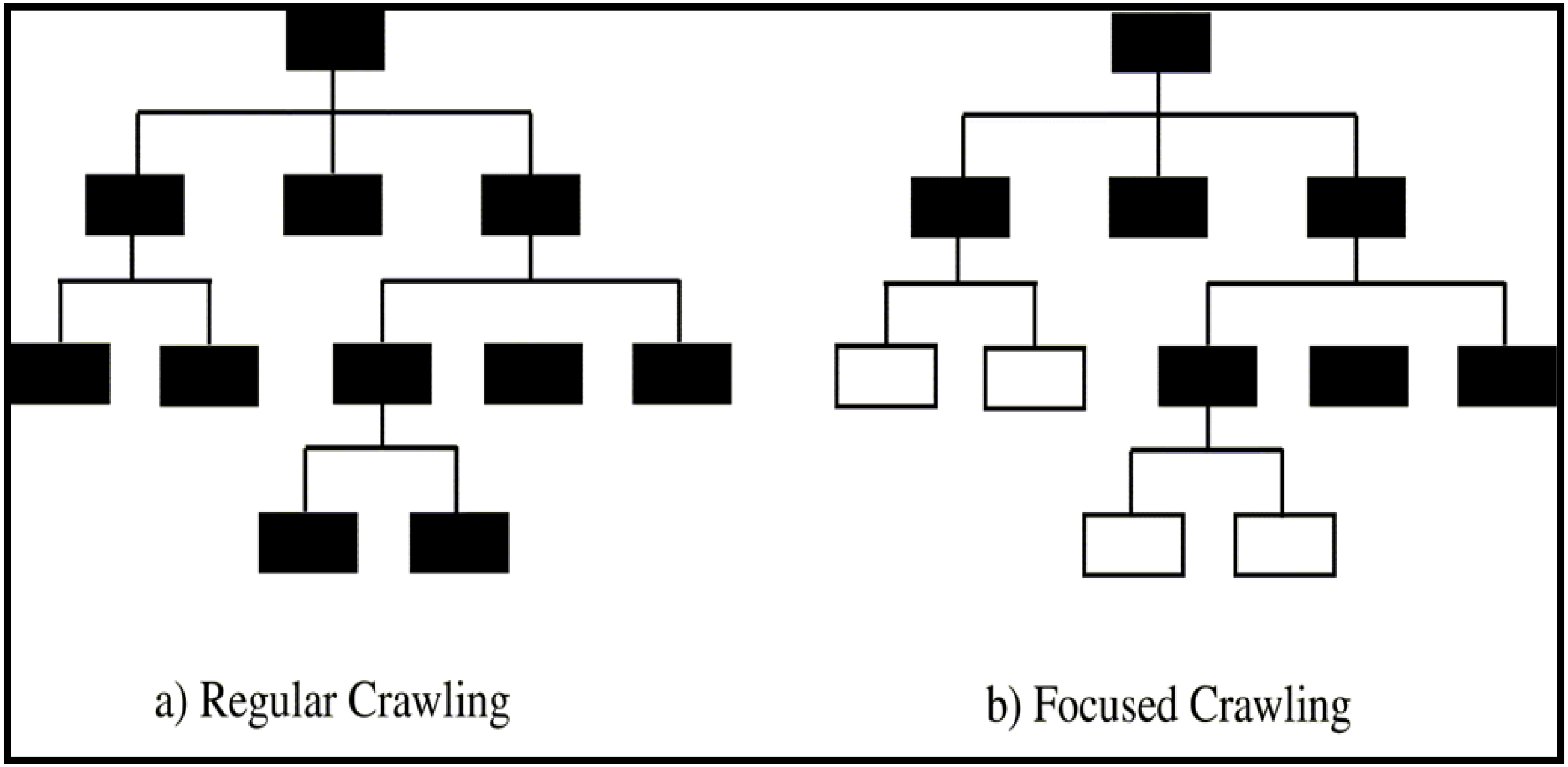
# Crawling

- Breadth-first explora uniformemente las páginas desde la raíz pero requieren memoria para todos los nodos del nivel previo (estrategia estándar)
- Depth-first sólo requiere tiempo lineal en profundidad pero puede perderse siguiendo un camino
- Ambas estrategias pueden implementarse fácilmente usando una cola de URLs

# Crawling

- Crawling enfocado:
  - sólo visita links de páginas si se determina que son relevantes
  - componentes:
    - un clasificador que asigna un score de relevancia a cada página en base a su tema
    - identificador de hubs
  - el crawler visita las páginas en base a los scores previos
- Hubs son páginas que contienen links a páginas relevantes, deben visitarse aún si no tienen buen score de relevancia

# Crawling



**Marcelo Armentano**
marcelo.armentano@isistan.unicen.edu.ar
marcelo.armentano@gmail.com

# Trabajo Práctico