Universidad
Carlos III de Madrid

Módulo III
Sistema de recuperación de Información: Crawlers

OpenCourseWare

Recuperación y Acceso a la Información

Contenidos

- Componentes de un motor de recuperación
- Crawler
- Ficheros y Base de datos

Sistema de Recuperación

Crawler/Spider
(Recopilación
Documentos)

(texto)

Indización
((palabras-localización) en BD)

Bases de datos

Algoritmos
de Recuperación
y Posicionamiento

Reformulación

Consulta
Usuario

Interfaz
de
Consulta y
SERP

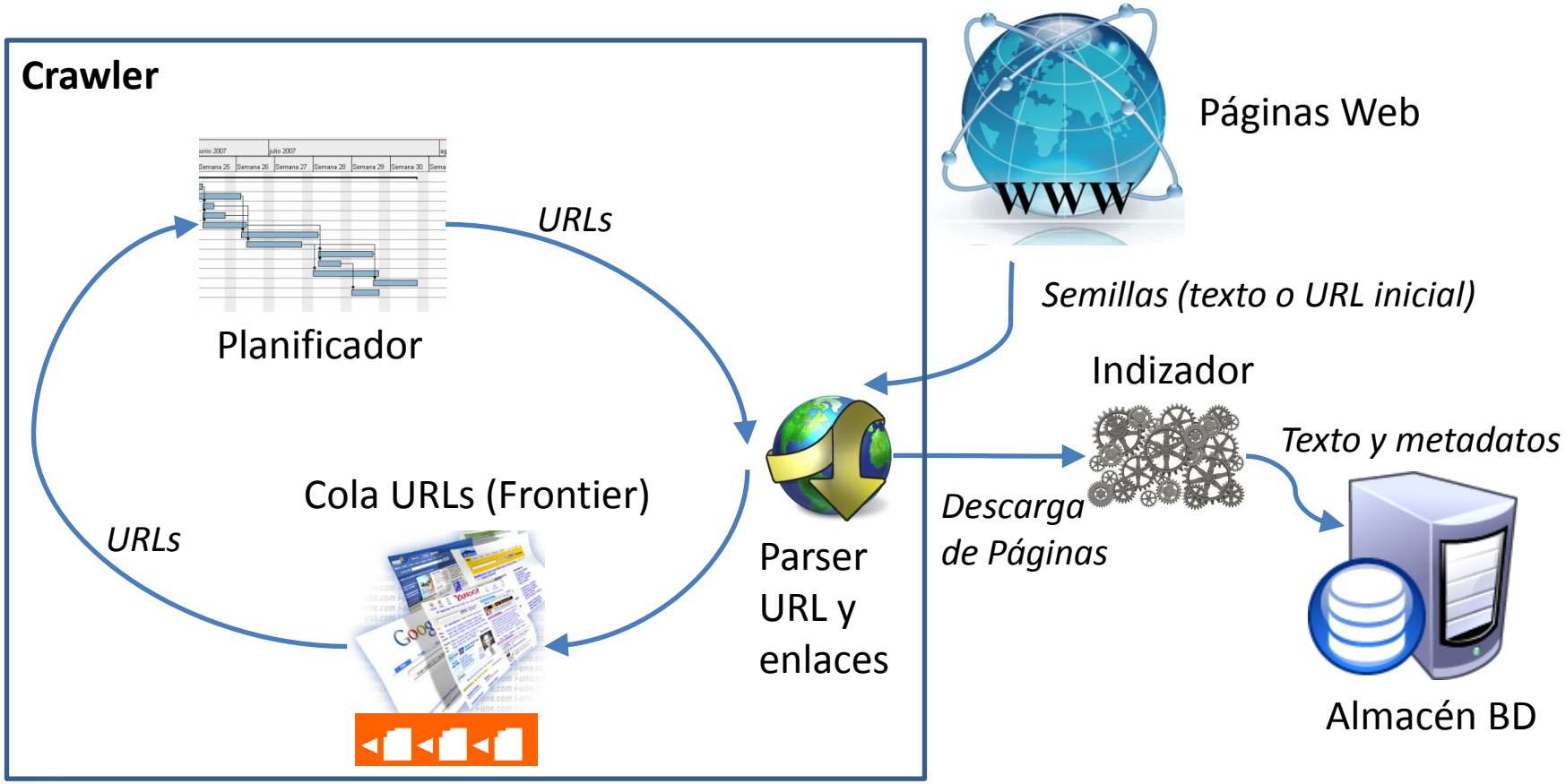
SERP (Search Engine Result Page)

Contenidos

- Componentes de un motor de recuperación
- Crawler
- Base de datos

Crawler Web

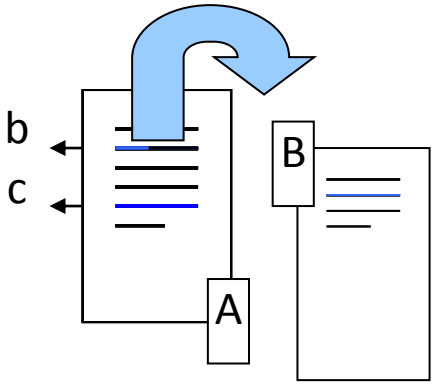
Aplicación Web para recorrer de forma sistemática las páginas web, para indizarlas (p.e. para búsquedas) u otro propósito



Crawler

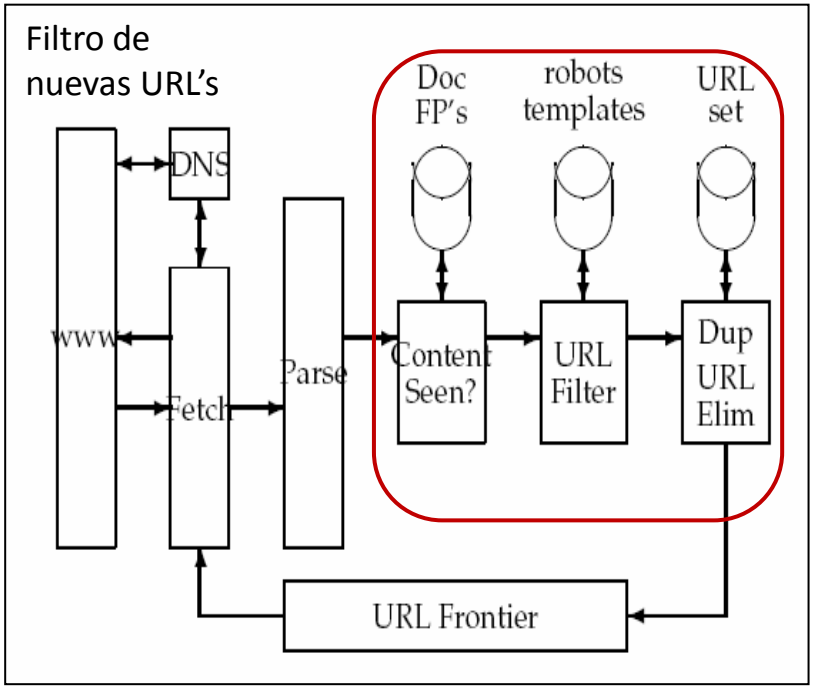
- Proceso

  1. Leer lista de URLs
  2. Seleccionar una URL de la lista, según criterio
  3. Comprobar meta follow, si está en “no”, ir a 1
  4. Parsear la pagina web
  5. Cada enlace en la página revisar si ya está en la lista y fecha última. Si no está en la lista añadir, sino leer siguiente
  6. Repetir hasta finalizar lista



- Criterios para organizar la lista a procesar:
  - Hasta que no acaba con todas las páginas de un site no pasa a las del siguiente site.
  - Profundidad: número de saltos a otras páginas desde URL semilla
- Dependiendo del objetivo del crawler: focused/topical (objetivo concreto), deep (para web oculta), incremental (va actualizándose), web scraper (recopila estructuras predefinidas), etc.
- Generalmente se introduce una función de la novedad (frecuencia de cambios) y relevancia de la página.
  - Supone introducir tiempos de espera entre visitas, o las páginas muy relevantes continuamente aparecerán en el conjunto y subirán a los primeros lugares.

Arquitectura básica de un crawler



Fuentes:

Manning C.; Raghavan, P.; Schütze, H. Introduction to Information Retrieval. Cambridge University Press, 2008

Capitulo 20. Web crawling and Indexes: http://nlp.stanford.edu/IR-book/pdf/20crawl.pdf

1º De la URL Frontier toma una URL (normalmente http)
2º Va a la página o DNS y la graba en una almacén temporal
3º Se parsea la página extrayendo enlaces
4º Se pasan filtros a la URL para saber si se almacena en la URL Frontier

- ¿Contenido en otra URL? (p.e. una fingerprint de la URL por si estuviera duplicada, en algún caso checksum, shingles o varios n-grams de bytes idénticos)
- Filtrado de URLs para quitar dominios que no interesen o páginas excluidas por el robot.txt
- Eliminación de URL visitadas o en espera en la URL frontier

Cuellos de botella:

- DNS (caché de ip para no tener que ir al DNS)
- Cache robots.txt de cada site , …

Información sobre enlaces

- Fichero robots.txt
  - Más información:http://www.robotstxt.org
  - Ej. Base de datos agentes: http://www.robotstxt.org/db.html

```
#Ejemplo de fichero robots.txt
User-agent: googlebot
Disallow: /
User-agent: bing
Disallow: /personal/
Disallow: /images/
User-agent: *
Disallow: /stuff/
```

- Etiqueta meta
  - <meta name =“robots” content=“noindex”/>
  - <meta name=“googlebot” content=“noindex”/>

Información sobre enlaces

- Páginas dinámicas: el robot debe poder acceder a todas las páginas desde un vínculo estático

Sitemaps

Documento que indica URLs del sitio y frecuencia de actualización

Más información: http://sitemaps.org

```
<?xml version="1.0" encoding="UTF-8"?>
< urlset
      xmlns="http://www.sitemaps.org/schemas/sitemap/0
      .9" >
<url >
     <loc>http://www.example.com/</loc>
     <lastmod>2005-01-01</lastmod>
      <changefreq>monthly</changefreq>
      <priority>0.8</priority>
</url>
</urlset>
```

Requisitos de un crawler

- **Debe** ser:
  - Robusto: asegurar su funcionamiento independientemente de los elementos que encuentren en los sites y de su distribución (ej. evitar bucles)
  - Respetuoso con las políticas de los servidores (ej. fichero robots.txt, frecuencia de visitas, una conexión por host)
- Es **aconsejable** que sea:
  - Distribuido y habitualmente uso de threads
  - Escalable
  - Eficiente
  - Extensible
    - Nuevos formatos
    - Nuevos protocolos
    - …

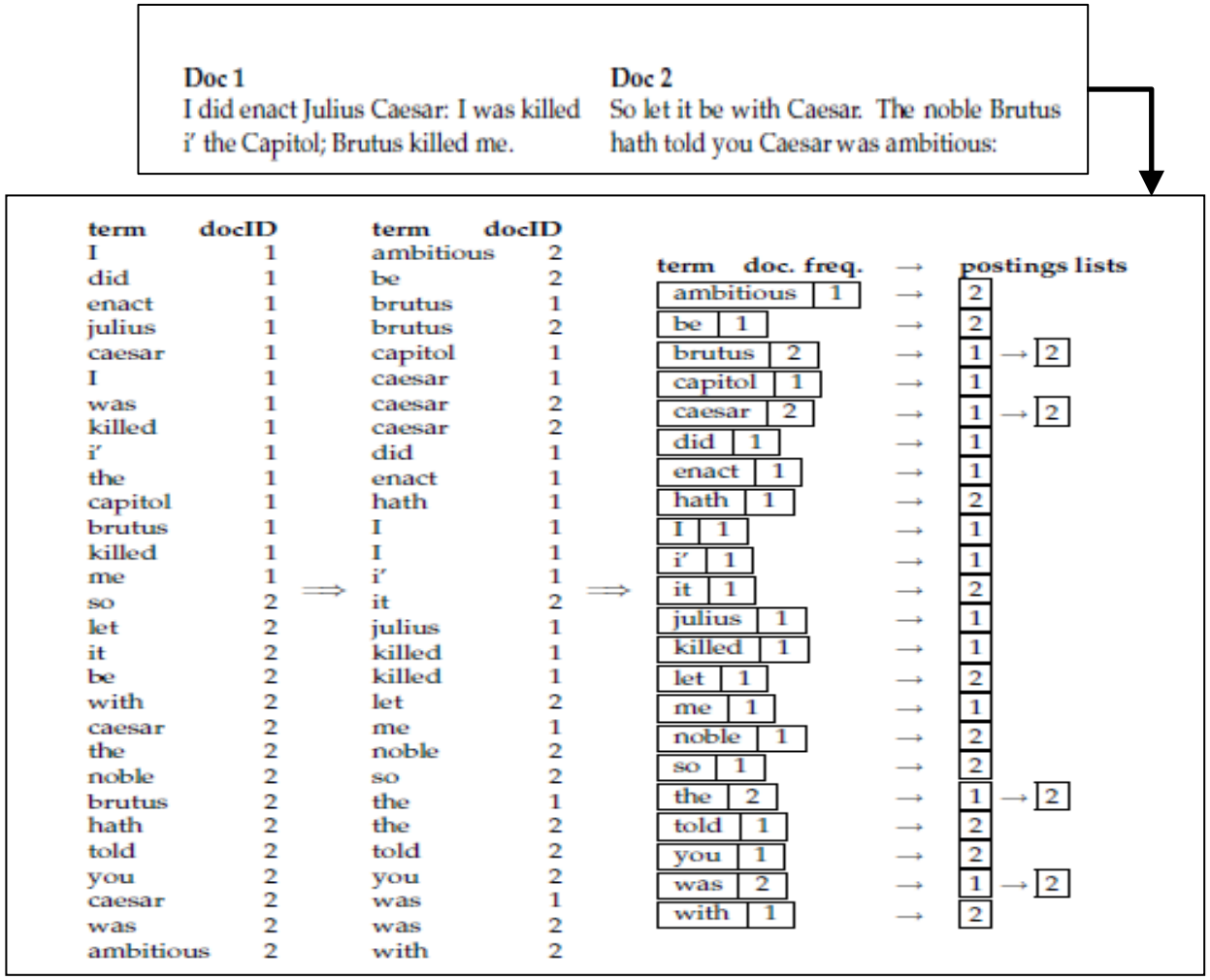
Contenidos

- Componentes de un motor de recuperación
- Crawler
- Base de datos

Indización: Ficheros

- La información se indexa mediante **ficheros inversos (o índices inversos)**
- **Lexicon y posting lists**

- Permite búsquedas rápidas de textos
- Tablas con índices
- Block addressing, bloques en vez de posiciones (ahorro espacio)
- Frecuencia? Posición? Formato?



Fuente: Manning C.; Raghavan P.; Schütze H. Information Retrieval, Cambridge University Press, 2008

Referencias. Implementación Crawler

- Wget
- En Java normalmente se utiliza la librería Jsoup
  https://www.mkyong.com/java/jsoup-basic-web-crawler-example/
  http://www.netinstructions.com/how-to-make-a-simple-web-crawler-in-java/
- Otras implementaciones más potentes en Java:
  https://github.com/yasserg/crawler4j
  https://java-source.net/open-source/crawlers o http://andreas-hess.info/programming/webcrawler/multiweb.zip
- En Python el número de líneas de código se reduce bastante con respecto a Java (150 Java en el segundo enlace, frente a 50 en Python en este)
  http://www.netinstructions.com/how-to-make-a-web-crawler-in-under-50-lines-of-python-code/
- PHP https://github.com/elboletaire/php-crawler
- C# https://www.codeproject.com/Articles/1087859/Web-crawling-with-Csharp-part-one

Universidad
Carlos III de Madrid

Módulo III
Sistema de recuperación de Información: Crawlers

Colaboradores

J.Morato, V.Palacios

M.Marrero, S.Sánchez-Cuadrado, J.Urbano