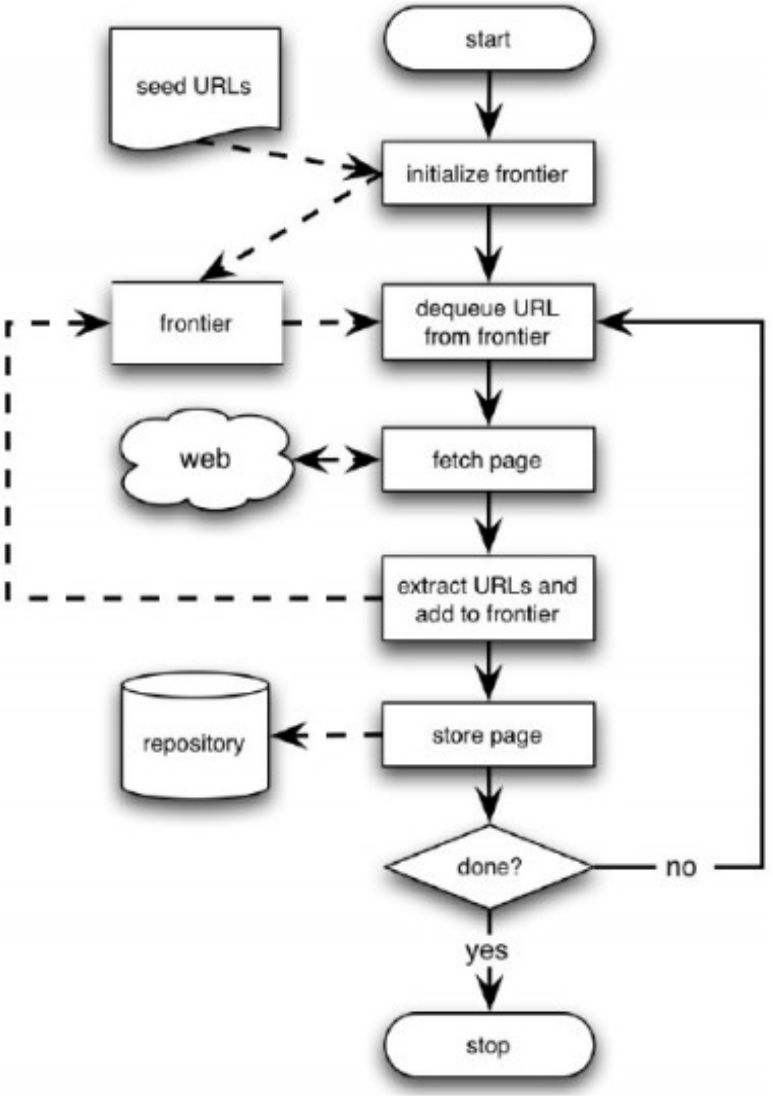
Web crawling

¿Qué hace un crawler?



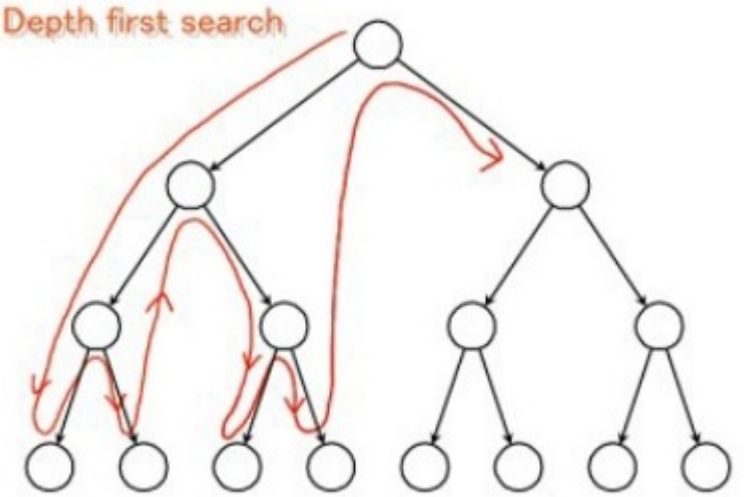
- A partir de un conjunto de URLs conocidas:
  - Toma una URL de una cola
  - Descarga la página asociada a la URL y la parsea
  - Extrae URLs de la página y las agrega a la cola
- El orden de descarga está determinado algún criterio de prioridad
- El criterio de finalización de descarga depende de la tarea de crawling.

Ejemplos de crawling

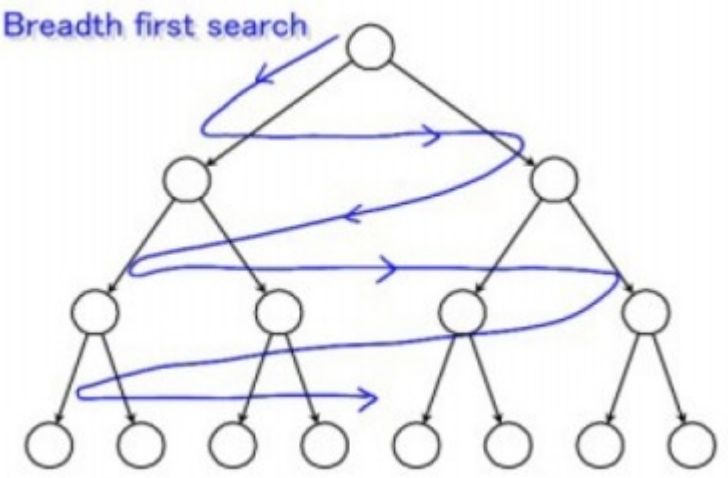
- **Common Crawl**: Se ejecuta 1 vez por mes; en Agosto 2015:
  - 181.000 millones de paginas web con su metadata.
  - 36.000 millones de imágenes únicas
  - 78,5 milles de dominios únicos.
- El resultado está disponible para cualquiera que quiera utilizarlo.
- El crawler es Apache Nutch (http://nutch.apache.org/)

¿Cómo recorrer la web?

- **Depth-First Search (DFS):**
  - Guardar URLs en una pila (LIFO)
  - Descarga páginas por caminos en profundidad, puede irse lejos del contenido interesante



- **Bread-First Search (BFS):**
  - Guardar URLs en una cola (FIFO)
  - Descarga páginas por niveles; si comenzamos con páginas buenas, esto nos mantiene cerca de los sitios interesantes.



La cosa se complica...

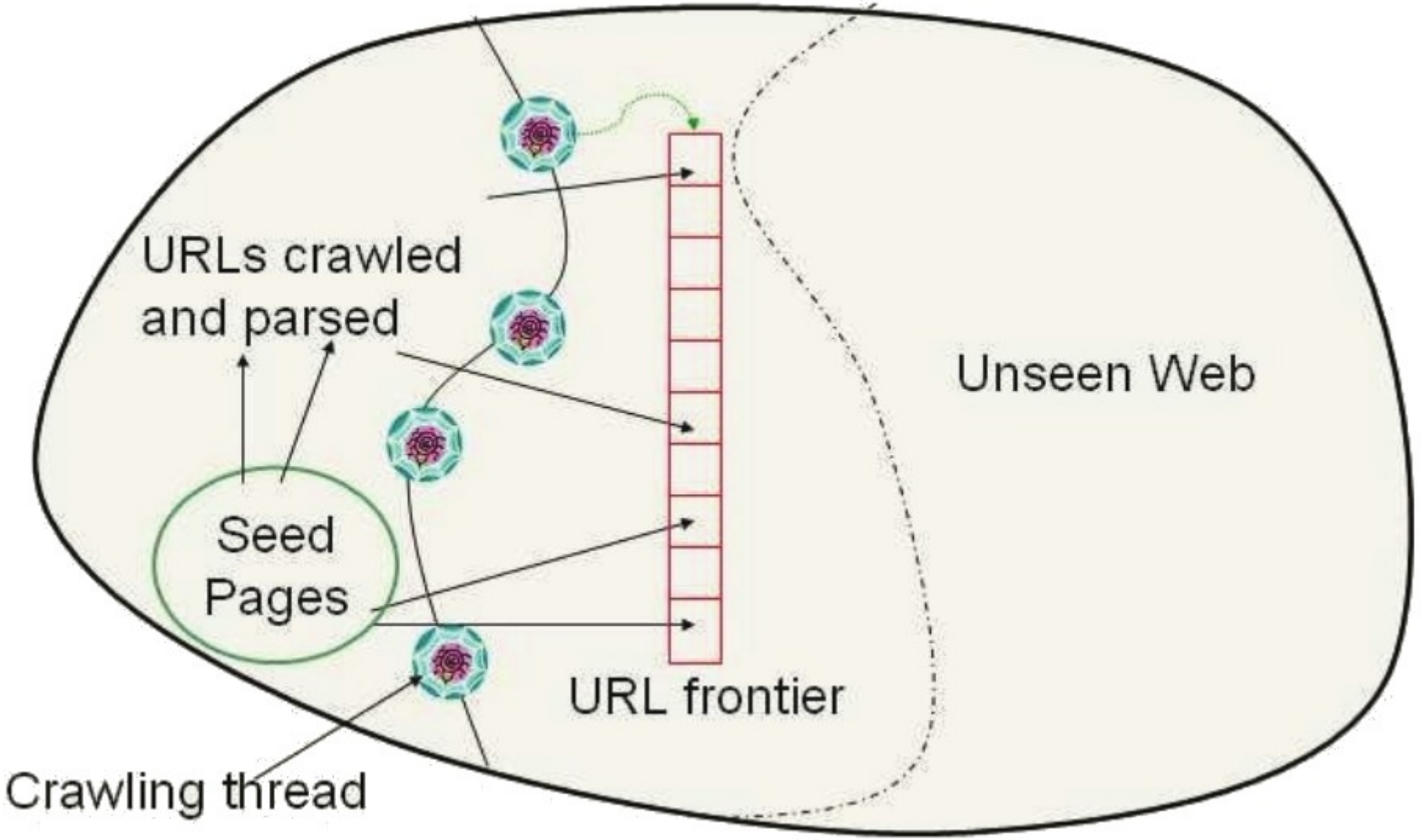
- No es posible hacer web crawling en cantidad en 1 sola máquina; todos los pasos deben ser distribuídos
- Páginas de baja calidad (spam, content farms)
- Spider traps
- Sitios que muestran diferente contenido a un crawler vs. a un usuario
- El crawler no puede pegarle a un mismo sitio continuamente
- Latencia y ancho de banda varían de sitio a sitio
- ¿Qué tan profundo vale la pena recorrer a un sitio?

Requerimientos de un crawler

Que sea…

- **Bien educado:** Respetar las limitaciones de un crawler en ancho de banda y recursos permitidos (respetar robots.txt)
- **Robusto:** Ser inmune a spider traps y otras trampas de sitios
- **Distribuído y escalabre**: correr en varias máquinas, y ser capaz de aumentar la velocidad de descarga agregando más máquinas.
- **Eficiente:**
  - Usar el ancho de banda y procesamientos locales tanto como sea posible.
  - Descargar contenido de mayor calidad lo antes posible.

Esquema de un crawler



La Frontera

- El objetivo de la frontera es mantener al todos los crawlers ocupados.
- Para evitar que el crawler sea bloqueado debe respetar robots.txt, que dice que parte del sitio puede ser descargada
- Puede incluir múltiples páginas de un mismo host, pero debe evitar descargarlas todas al mismo tiempo para no saturar al host.

robots.txt

- Es un archivo de texto ubicado en la raiz del sitio web, que especifica qué parte del sitio puede ser descargada por quien.

- **despegar.com:**

```
User-agent: *
Allow: /vuelos-baratos
Disallow: /paginas
Disallow: /georigen/
Disallow: /search/

# Bloqueos especificos para
# el bot de keywords de Adwords
User-agent: AdsBot-Google
Disallow: /book/hotels
Disallow: /book/flights
```

- **www.clarin.com:**

```
User-agent: *
Disallow: /bbtcomment/create/
Disallow: /bbtcomment/vote/
Disallow: /bbtcomment/report/
Disallow: /bbtcontent/poll/vote/
Disallow: /bbtcontent/poll/getresults/
Disallow: /bbtmail/sendEntity/
Disallow: /bbtstats/
Disallow: /modal-videos.html
Disallow: /*modal.html

Sitemap: http://www.clarin.com/sitemap_index.xml
```

¿Es obligatorio respetar robots.txt?

- No, pero la mayoría de los sitios tiene como detectar crawlers intensos.
- La detección de crawlers es una historia interminable:
  - Algunos crawlers se disfrazan de crawlers “buenos” en User-Agent
  - Acceder al sitio con espera al azar, no muy rápido y desde diferentes hosts se parece al acceso por personas.
- Contramedidas:
  - Links invisibles para una persona
  - Análisis de acceso: descarga la página pero no las imágenes.
  - Captcha y similares.
  - El orden del contenido de la página no es el orden en que se muestra: hacer rendering de la página usando javascript.
  - Si un crawler es detectado, es mejor hacer que ande muy lento y mandarle basura que bloquearlo.

Pasos de un crawler: Resolución de la URL

- DNS resuelve nombres de hosts a direcciones de IP.
- En muchos S.O. la implementación de DNS es bloqueante: 1 solo pedido a la vez, lo que frena a todos los crawlers
- Posibles soluciones: caching de DNS, resolver hosts en tandas

Pasos de un Crawler: Normalización de URLs

- Las URLs pueden ser relativas, hay que convertirlas a absolutas para descargarlas.
- Una misma página puede tener varias URLs, algunos de estos casos se pueden detectar porque son redirecciones.

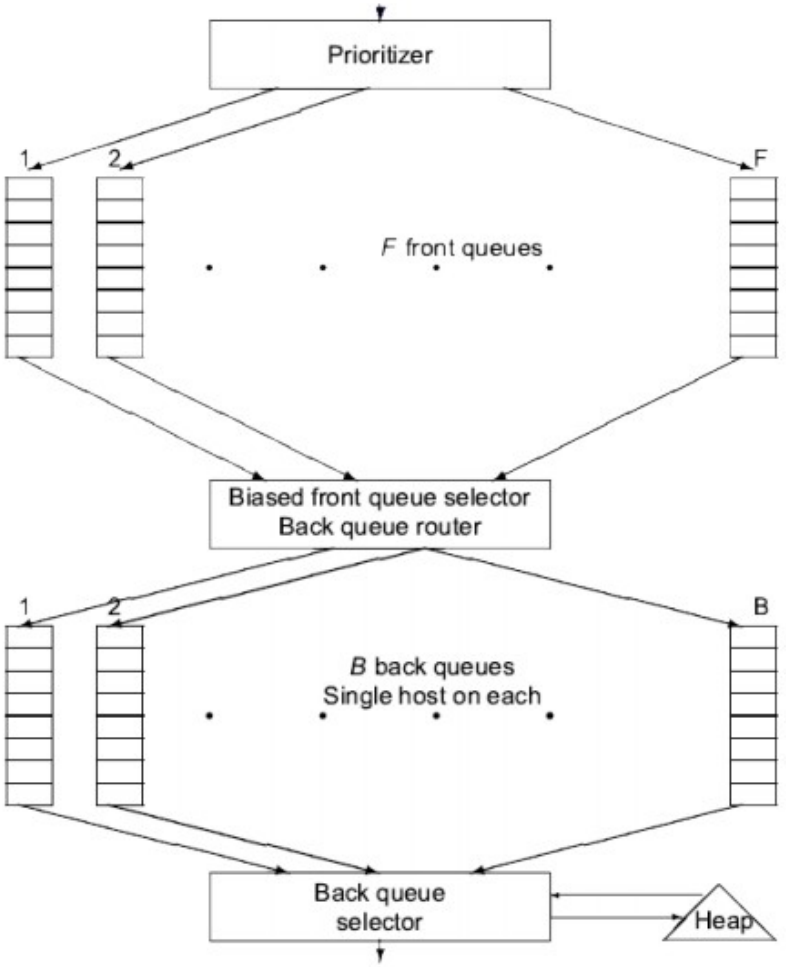
Filtrar URLs en la frontera

- Filtros: expresiones regulares sobre las URLs a descargar o no.
- Seguir las reglas de robots.txt – hace falta descargarlo 1 sola vez por host.

Frontera

- Aún si restringimos que como máximo 1 crawler descargue de un mismo host, puede hacerlo muy rápidamente.
- Normalmente se inserta un retardo entre 2 descargas desde 1 mismo host.
- Si el crawler necesita descargar 250.000 páginas, c/u en promedio con 20 referencias (css,js, imágenes, etc), a 1.935Kb en promedio por página en 2014 son 483750000 Kb en total, a 100 páginas/seg. son 2000 descargas/seg y 188Mb/seg durante 41 minutos.

Estructura de la Frontera



- URLs entrar por la parte de arriba de la frontera.
- Las colas frontales (front queues) manejan las prioridades
- Las colas traseras (back queues) manejan las reglas de descarga a un mismo server.

Partes de la frontera del crawler

- **Prioritizador:**
  - El prioritizador le asigna a cada URL una prioridad, y la URL se guarda en la cola correspondiente.
- **Colas Frontales:**
  - Hay K colas frontales, 1 cola por prioridad. La prioridad sale de heurísticas (nombre del sitio, nombre de la página, que tan seguido el contenido del sitio cambia, etc)
- **Colas Traseras:**
  - Hay N colas traseras. Todas las URLs de un mismo host van a la misma cola. El crawler mantiene una tabla de nombres de host a colas.
- **Heap:**
  - Contiene pares (numero de cola, timestamp) , ordenados por tiempo. Cada timestamp es el tiempo en el que se puede volver a sacar de esa cola.

Como conseguir una URL desde la frontera

Cada vez que el crawler necesita una nueva URL para descargar:

1. Toma la entrada mas cercana en tiempo en el heap; eso le da el numero de cola de donde sacar la url.
2. Si la cola trasera correspondiente está vacía: Se toma una url desde las colas frontales, con probabilidad proporcional a las prioridades de cada cola frontal, y se pasa a una cola trasera. Si ya hay una cola para ese host, entonces se saca otra url de la colas frontales.
3. Saca la URL en el tope la cola trasera y la descarga.
4. Agrega una entrada (cola, timestamp) en el heap para la cola desde donde se sacó la URL.

Problemas de Implementación: URLs

- Pasar URLs a forma canónica. Todas estas URLs:
  - http://www.cnn.com/TECH
  - http://WWW.CNN.COM/TECH/?
  - http://www.cnn.com:80/TECH/#
  - https://www.cnn.com/saraza/../TECH/

Son URLs equivalentes a la forma canónica: http://www.cnn.com/TECH

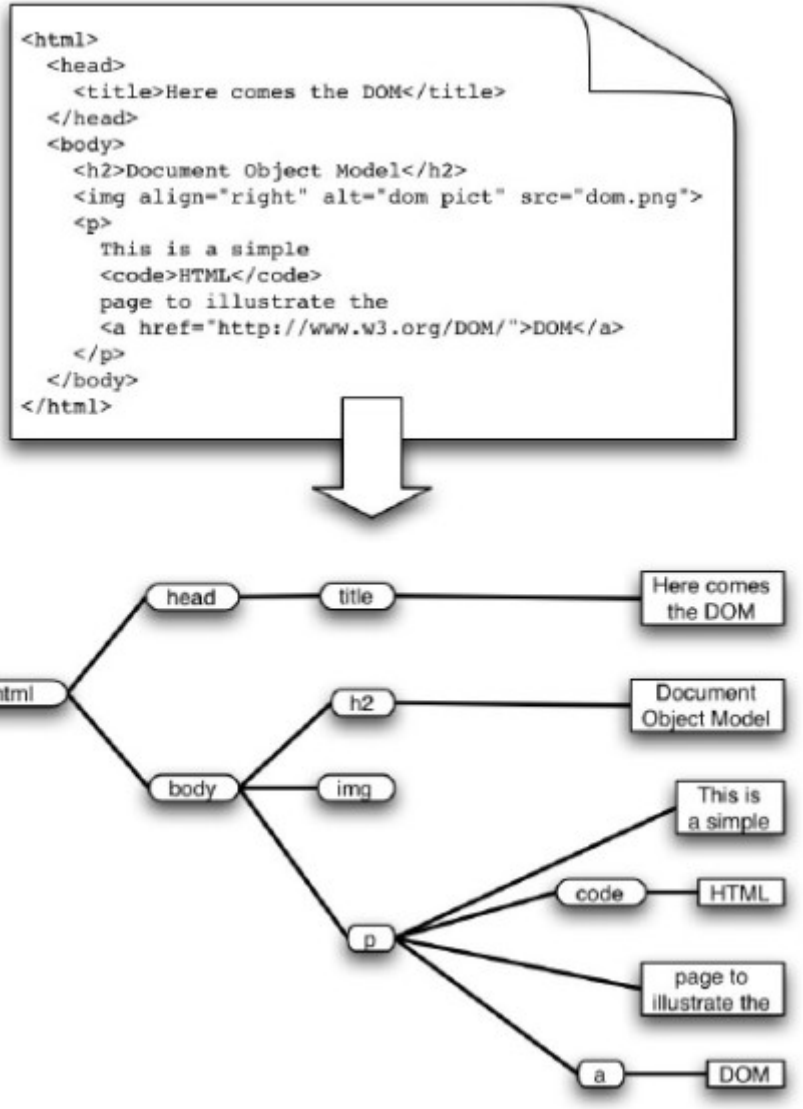
Cual es la forma canónica es arbitrario (¿con o sin número de port?) Pero lo importante es que sea consistente.

Problemas de Implementación: Spider Traps

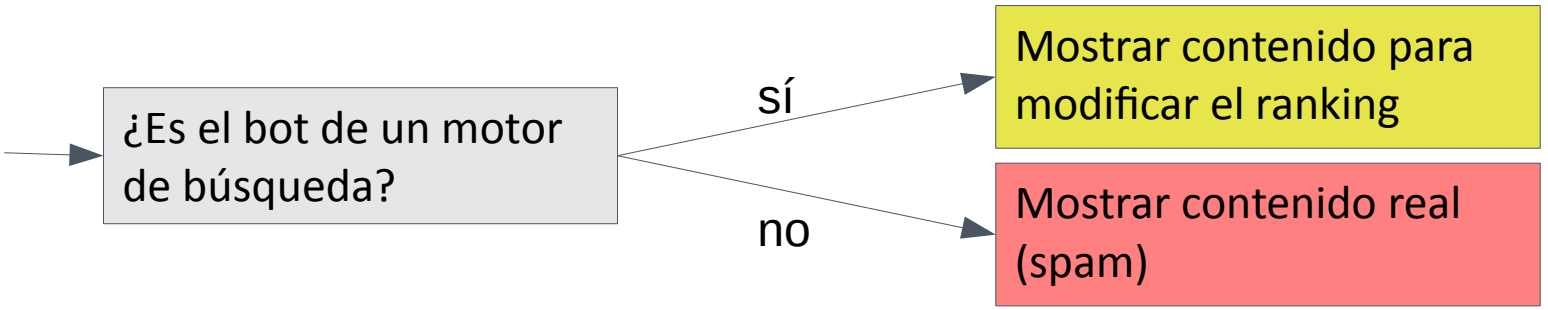
- Spider traps son links a páginas generadas dinámicamente, con profunidad infinita. Se usan para hacer perder tiempo a crawlers.
- Las páginas son generadas por código, y los links son siempre diferentes y llevan a la misma u otra manera de generar páginas.
- La única manera de detectarlas es mediante heurísticas:
  - Profundidad máxima a la que se descarga.
  - Explosión de URLs nuevas por página

Problemas de implementación: Parsing

- Una página HTML tiene una estructura en forma de árbol, llamada DOM (Document Object Model).
- Frecuentemente el HTML de una página es incorrecto. El crawler debe "enderezarlo" para poder parsearlo.
- Hay diferentes maneras de decir lo mismo: entidades de HTML, unicode.
- Más y más páginas son generadas en el cliente, por lo que hace falta interpretar javascript para poder parsearlas.



Problemas de Implementación: cloaking



- Cloaking es mostrarle al crawler un contenido diferente que al usuario.
- No siempre esto es spam: puede ser que sea porque muestra diferente contenido dependiendo del país, o porque el sitio hace personalización.

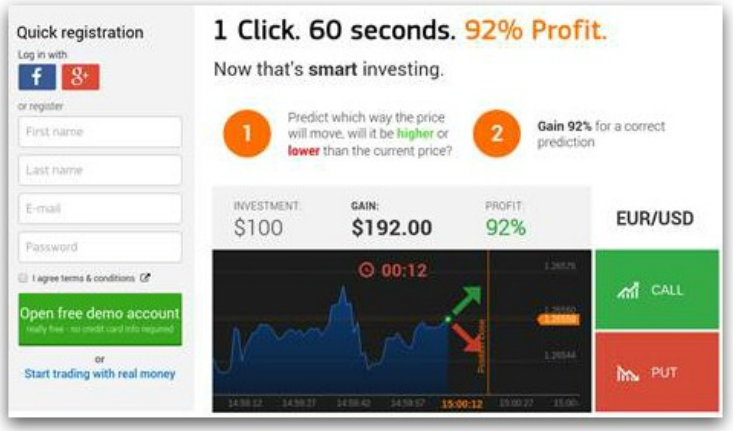
Problemas: Link Spamming

- Crear un montón de links apuntando a una página a promover. Esos links deben ir en página con alto Pagerank.
- Crear links en blogs y comentarios de sitios reputados.
- El nuevo spam es social: Los “Likes” son los nuevos links. Hay sitios que venden “likes”.
- Subir a sitios PDFs con muchas keywords y conteniendo links que llevan a sitios y a otros PDFs. Funciona porque Google (por ahora) le cree a los PDF más que HTML.

Google ve el contenido del PDF...

Click para ver el PDF...

El PDF te lleva a este sitio...



http://searchengineland.com/it-security-company-uncovers-google-link-spamming-technique-through-cloaking-pdf-documents-224941