Temas 3 y 4 del libro traducido

Recuperación da Información (Universidade da Coruña)

3

Se arrastra y se alimenta

> "Has metido tus redes en mi negocio por última vez".
>
> Doc Ock, *Hombre araña 2*

3.1 Decidir qué buscar

Este libro trata sobre los detalles de la construcción de un motor de búsqueda, desde las matemáticas detrás de la clasificación hasta los algoritmos de procesamiento de consultas. Aunque nos centramos principalmente en la tecnología que hace que los motores de búsqueda funcionen, y la gran tecnología puede hacer que un buen motor de búsqueda sea aún mejor, es la información en la colección de documentos lo que hace que los motores de búsqueda sean útiles. En otras palabras, si los documentos correctos no se almacenan en el motor de búsqueda, ninguna técnica de búsqueda podrá encontrar información relevante.

El título de esta sección implica la pregunta: "¿Qué debemos buscar?" La respuesta simple es *todo lo que puedas.* Cada documento responde al menos a una pregunta (es decir, "¿Dónde estaba ese documento de nuevo?"), Aunque los mejores documentos responden a muchas más. Cada vez que un motor de búsqueda agrega otro documento, aumenta la cantidad de preguntas que puede responder. Por otro lado, agregar muchos documentos de mala calidad aumenta la carga en el proceso de clasificación para encontrar solo los mejores documentos para mostrar al usuario. Los motores de búsqueda web, sin embargo, muestran el éxito que pueden tener los motores de búsqueda, incluso cuando contienen miles de millones de documentos de baja calidad con poco contenido útil.

Incluso los documentos útiles pueden volverse menos útiles con el tiempo. Esto es especialmente cierto en el caso de las noticias y la información financiera, donde, por ejemplo, muchas personas desean conocer el informe bursátil de hoy, pero solo unos pocos se preocupan por lo que sucedió ayer. Desafortunadamente, la frustración de encontrar páginas web y enlaces obsoletos en una lista de resultados de búsqueda es una experiencia común. Los motores de búsqueda son más eficaces cuando contienen la información más reciente además de archivos de material más antiguo.

Este capítulo presenta técnicas para encontrar documentos para buscar, ya sea en la Web, en un servidor de archivos, en el disco duro de una computadora o en un programa de correo electrónico. Discutiremos estrategias para almacenar documentos y mantener esos documentos actualizados. A lo largo del camino, discutiremos cómo extraer datos de archivos, navegando a través de problemas de codificación de caracteres, formatos de archivo obsoletos, documentos duplicados y ruido textual. Al final de este capítulo, tendrá un conocimiento sólido sobre cómo introducir datos de documentos en un motor de búsqueda, listos para ser indexados.

3.2 Rastrear la Web

Para crear un motor de búsqueda que busque páginas web, primero necesita una copia de las páginas que desea buscar. A diferencia de algunas de las otras fuentes de texto que consideraremos más adelante, las páginas web son particularmente fáciles de copiar, ya que están diseñadas para ser recuperadas a través de Internet por navegadores. Esto resuelve instantáneamente uno de los principales problemas de obtener información para buscar, que es cómo obtener los datos del lugar donde están almacenados en el motor de búsqueda.

Buscar y descargar páginas web automáticamente se denomina *gateando* y un programa que descarga páginas se llama *rastreador web*.[1] Existen algunos desafíos únicos para rastrear páginas web. El mayor problema es la enorme escala de la Web. Hay al menos decenas de miles de millones de páginas en Internet. El "al menos" en la última oración está ahí porque nadie está seguro de cuántas páginas hay. Incluso si el número de páginas que existen hoy en día pudiera medirse con exactitud, ese número sería inmediatamente incorrecto, porque las páginas se crean constantemente. Cada vez que un usuario agrega una nueva publicación de blog o carga una foto, se crea otra página web. La mayoría de las organizaciones no tienen suficiente espacio de almacenamiento para almacenar ni siquiera una gran fracción de la Web, pero los proveedores de búsqueda en la Web con abundantes recursos deben descargar constantemente contenido nuevo para mantener sus colecciones actualizadas.

Otro problema es que las páginas web no suelen estar bajo el control de las personas que crean la base de datos del motor de búsqueda. Incluso si sabe que desea copiar todas las páginas dewww.company.com, no hay una manera fácil de saber cuántas páginas hay en el sitio. Es posible que los propietarios de ese sitio no quieran que copie algunos de los datos y probablemente se enojarán si intenta copiarlos demasiado rápido o con demasiada frecuencia. Algunos de los datos que desea copiar pueden estar disponibles solo escribiendo una solicitud en un formulario, lo cual es un proceso difícil de automatizar.

[1] El rastreo también se conoce ocasionalmente como *arañas* y un rastreador a veces se llama a *araña.*

3.2.1 Recuperar páginas web

Cada página web en Internet tiene su propio *Localizador Uniforme de Recursos,* o *URL.* AnyURL que se usa para describir una página web tiene tres partes: el esquema, el nombre de host y el nombre del recurso (Figura 3.1). Las páginas web se almacenan en *servidores web,* que utilizan un protocolo llamado *Protocolo de Transferencia de Hipertexto,* o *HTTP,* para intercambiar información con el software del cliente. Por lo tanto, la mayoría de las URL que se utilizan en la Web comienzan con el esquema `http`, lo que indica que la URL representa un recurso que se puede recuperar mediante HTTP. La *nombre de host* a continuación, que es el nombre de la computadora que está ejecutando el servidor web que contiene esta página web. En la figura, el nombre de la computadora es `www.cs.umass.edu`, que es una computadora en el departamento de Ciencias de la Computación de la Universidad de Massachusetts. Esta URL hace referencia a una página en esa computadora llamada `/csinfo / people.html`.

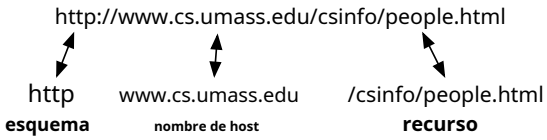


Figura 3.1. Un localizador de recursos uniforme (URL), dividido en tres partes

Los navegadores web y los rastreadores web son dos tipos diferentes de clientes web, pero ambos obtienen páginas web de la misma forma. Primero, el programa cliente se conecta a un *sistema de nombres de dominio (*Servidor DNS. El servidor DNS traduce el nombre de host en un *protocolo de Internet (*Dirección IP. Esta dirección IP es un número que suele tener una longitud de 32 bits, pero algunas redes ahora usan direcciones IP de 128 bits. Luego, el programa intenta conectarse a una computadora servidor con esa dirección IP. Dado que ese servidor puede tener muchos programas diferentes ejecutándose en él, cada uno escuchando la red en busca de nuevas conexiones, cada programa escucha en un diferente *Puerto.* Un puerto es solo un número de 16 bits que identifica un servicio en particular. Por convención, las solicitudes de páginas web se envían al puerto 80 a menos que se especifique lo contrario en la URL.

Una vez establecida la conexión, el programa cliente envía una solicitud HTTP al servidor web para solicitar una página. El tipo de solicitud HTTP más común es una solicitud GET, por ejemplo:

```
OBTENGA /csinfo/people.html HTTP / 1.0
```

Esta simple solicitud le pide al servidor que envíe la página llamada `/csinfo / people.html` de vuelta al cliente, utilizando la versión 1.0 de la especificación del protocolo HTTP. Después

Al enviar un encabezado corto, el servidor envía el contenido de ese archivo al cliente. Si el cliente desea más páginas, puede enviar solicitudes adicionales; de lo contrario, el cliente cierra la conexión.

Un cliente también puede obtener páginas web mediante solicitudes POST. Una solicitud POST es como una solicitud GET, excepto que puede enviar información de solicitud adicional al servidor. Por convención, las solicitudes GET se utilizan para recuperar datos que ya existen en el servidor, mientras que las solicitudes POST se utilizan para decirle algo al servidor. Se puede usar una solicitud POST cuando hace clic en un botón para comprar algo o editar una página web. Esta convención es útil si está ejecutando un rastreador web, ya que enviar solo solicitudes GET ayuda a asegurarse de que su rastreador no lo haga inadvertidamente pedir un producto.

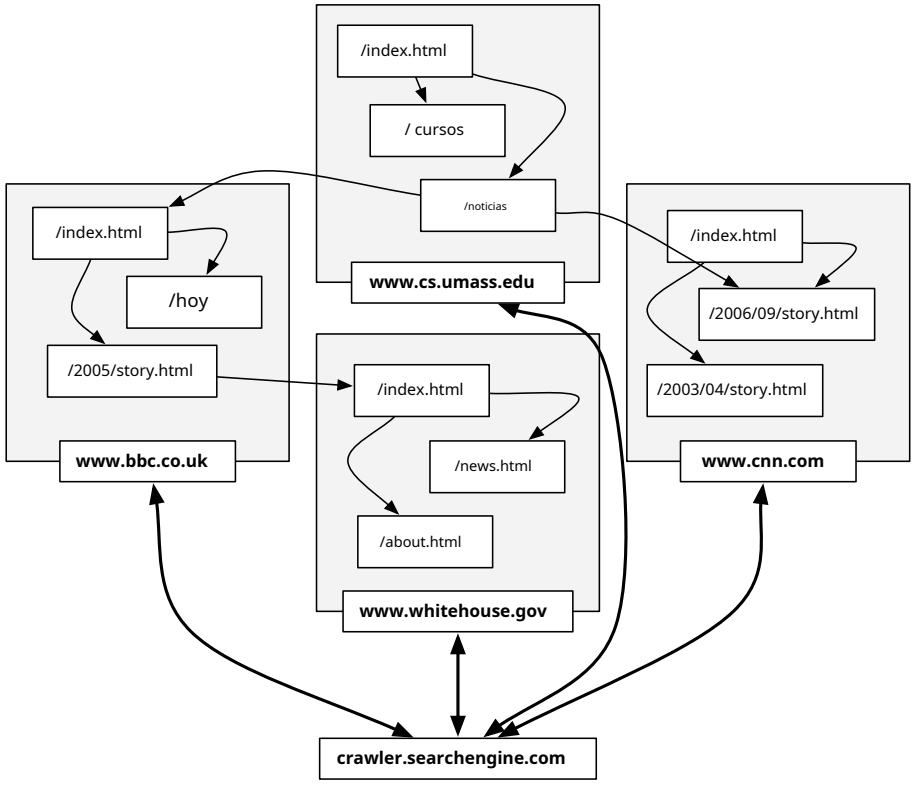


Figura 3.2. Rastrear la Web. El rastreador web se conecta a los servidores web para buscar páginas. Las páginas pueden enlazar a otras páginas en el mismo servidor o en diferentes servidores.

3.2.2 El rastreador web

La figura 3.2 muestra un diagrama de la Web desde la perspectiva de un rastreador web simple. El rastreador web tiene dos funciones: descargar páginas y buscar URL.

El rastreador comienza con un conjunto de *semillas* que son un conjunto de URL que se le asignan como parámetros. Estas semillas se agregan a una cola de solicitudes de URL. El rastreador comienza a buscar páginas de la cola de solicitudes. Una vez que se descarga una página, se analiza para buscar etiquetas de enlace que puedan contener otras URL útiles para recuperar. Si el rastreador encuentra una nueva URL que no ha visto antes, se agrega a la cola de solicitudes del rastreador, o *frontera.* La frontera puede ser una cola estándar o puede estar ordenada para que las páginas importantes pasen al principio de la lista. Este proceso continúa hasta que el rastreador se queda sin espacio en disco para almacenar páginas o se queda sin enlaces útiles para agregar a la cola de solicitudes.

Si un rastreador usara un solo hilo, no sería muy eficiente. Observe que el rastreador web pasa gran parte de su tiempo esperando respuestas: espera la respuesta del servidor DNS, luego espera que se reconozca la conexión con el servidor web y luego espera que se envíen los datos de la página web desde el servidor. Durante este tiempo de espera, la CPU de la máquina del rastreador web está inactiva y la conexión de red no se utiliza. Para reducir esta ineficiencia, los rastreadores web utilizan subprocesos y obtienen cientos de páginas a la vez.

Obtener cientos de páginas a la vez es bueno para la persona que ejecuta el rastreador web, pero no necesariamente es bueno para la persona que ejecuta el servidor web en el otro extremo. Imagínese cómo funciona la cola de solicitudes en la práctica. Cuando una página web comowww.company.com se obtiene, se analiza y todos los enlaces de esa página se agregan a la cola de solicitudes. El rastreador intentará recuperar todas esas páginas a la vez. Si el servidor web dewww.company.com no es muy potente, podría pasar todo su tiempo manejando solicitudes del rastreador en lugar de manejar solicitudes de usuarios reales. Este tipo de comportamiento de los rastreadores web tiende a enojar mucho a los administradores de servidores web.

Para evitar este problema, los rastreadores web utilizan *políticas de cortesía.* Los rastreadores web razonables no obtienen más de una página a la vez de un servidor web en particular. Además, los rastreadores web esperan al menos unos segundos, ya veces minutos, entre las solicitudes al mismo servidor web. Esto permite a los servidores web dedicar la mayor parte de su tiempo a procesar solicitudes de usuarios reales. Para admitir esto, la cola de solicitudes se divide lógicamente en una sola cola por servidor web. En cualquier momento, la mayoría de estas colas por servidor están prohibidas para el rastreo, porque el rastreador ha obtenido una página de ese servidor recientemente. El rastreador es libre de leer solicitudes de página solo de las colas a las que no se ha accedido dentro de la ventana de cortesía especificada.

Al utilizar una ventana de cortesía, la cola de solicitudes debe ser muy grande para lograr un buen rendimiento. Suponga que un rastreador web puede recuperar 100 páginas por segundo y que su política de cortesía dicta que no puede recuperar más de una página cada 30 segundos desde un servidor web en particular. El rastreador web debe tener URL de al menos 3.000 servidores web diferentes en su cola de solicitudes para lograr un alto rendimiento. Dado que muchas URL vendrán de los mismos servidores, la cola de solicitudes debe tener decenas de miles de URL antes de que un rastreador pueda alcanzar su rendimiento máximo.

```
Agente de usuario: *
No permitir: / privado /
No permitir: / confidencial / No
permitir: / otro /
Permitir: / otro / público /

Agente de usuario:
FavoredCrawler Disallow:

Mapa del sitio: http://mysite.com/sitemap.xml.gz
```

Figura 3.3. Un archivo robots.txt de ejemplo

Incluso rastrear un sitio lentamente enojará a algunos administradores de servidores web que se oponen a cualquier copia de sus datos. Los administradores del servidor web que se sientan así pueden almacenar un archivo llamado /robots.txt en sus servidores web. La Figura 3.3 contiene un ejemplo de archivo robots.txt. El archivo se divide en bloques de comandos que comienzan con un Agente de usuario: especificación. LaAgente de usuario: line identifica un rastreador, o grupo de rastreadores, afectado por las siguientes reglas. Siguiendo esta línea estánPermitir y Rechazar reglas que dictan a qué recursos puede acceder el rastreador. En la figura, el primer bloque indica que todos los rastreadores deben ignorar los recursos que comienzan con /privado y confidencial/, o /otro/, excepto aquellos que comienzan con / otro / público /. El segundo bloque indica que un rastreador llamado FavoredCrawler obtiene su propio conjunto de reglas: se le permite copiar todo.

El bloque final del ejemplo es opcional. Mapa del sitio: directiva, que se analizará más adelante en esta sección.

La Figura 3.4 muestra una implementación de un hilo de rastreo, utilizando los bloques de construcción del rastreador que hemos visto hasta ahora. Suponga que la frontera se ha inicializado

```
procedimiento C     T     (frontera)
    tiempo no frontier.done () hacer
        sitio web ← URL de frontier.nextSite
        () ← sitio web.nextURL ()
        Si website.permitsCrawl (url) luego
            texto ← retrieveURL (url)
            storeDocument (url, texto)
            por cada URL en parse (texto) hacer
                frontier.addURL (url)
            final para
        terminara si
        frontier.releaseSite (sitio web)
    terminar mientras
procedimiento final
```

Figura 3.4. Una implementación de hilo de rastreo simple

con algunas URL que actúan como semillas para el rastreo. El hilo de rastreo primero recupera un sitio web desde la frontera. A continuación, el rastreador identifica la siguiente URL en la cola del sitio web. En allowCrawl, el rastreador comprueba si la URL se puede rastrear de acuerdo con el archivo robots.txt del sitio web. Si se puede rastrear, el rastreador usaretrieveURL para recuperar el contenido del documento. Esta es la parte más cara del bucle y el hilo del rastreador puede bloquearse aquí durante muchos segundos. Una vez que se ha recuperado el texto, storeDocument almacena el texto del documento en una base de datos de documentos (que se explica más adelante en este capítulo). A continuación, se analiza el texto del documento para poder encontrar otras URL. Estas URL se agregan a la frontera, lo que las agrega a las colas de sitios web correspondientes. Cuando todo esto termina, el objeto del sitio web se devuelve a la frontera, que se encarga de hacer cumplir su política de cortesía al no ceder el sitio web a otro hilo del rastreador hasta que haya transcurrido una cantidad de tiempo adecuada. En un rastreador real, el temporizador se iniciaría inmediatamente después de que se recuperara el documento, ya que analizar y almacenar el documento podría llevar mucho tiempo.

3.2.3 Frescura

Las páginas web se agregan, eliminan y modifican constantemente. Para mantener una vista precisa de la Web, un rastreador web debe volver a visitar continuamente las páginas que ya ha rastreado para ver si han cambiado a fin de mantener el *frescura* de la colección de documentos. Lo contrario de una copia nueva es una *duro* copia, lo que significa una copia que ya no refleja el contenido real de la página web.

Solicitud de cliente:	HEAD /csinfo/people.html HTTP / 1.1 Host: www.cs.umass.edu
Respuesta del servidor:	HTTP / 1.1 200 OK Fecha: jueves, 3 de abril de 2008, 05:17:54 GMT Servidor: Apache / 2.0.52 (CentOS) Última modificación: viernes, 4 de enero de 2008, 15:28:39 GMT ETag: "239c33-2576-2a2837c0" Rangos de aceptación: bytes Longitud del contenido: 9590 Conexión: cerrar Tipo de contenido: texto / html; juego de caracteres = ISO-8859-1

Figura 3.5. Solicitud AnHTTPHEAD y respuesta del servidor

El protocolo HTTP tiene un tipo de solicitud especial llamado HEAD que facilita la verificación de cambios de página. La solicitud HEAD devuelve solo información de encabezado sobre la página, pero no la página en sí. La Figura 3.5 contiene un ejemplo de solicitud y respuesta HEAD. LaÚltima modificación El valor indica la última vez que se cambió el contenido de la página. Tenga en cuenta que la fecha también se envía junto con la respuesta, así como en respuesta a una solicitud GET. Esto permite al rastreador web comparar la fecha que recibió de una solicitud GET anterior con laÚltima modificación valor de una solicitud HEAD.

La solicitud AHEAD reduce el costo de verificar una página, pero no la elimina. Simplemente no es posible revisar cada página cada minuto. Eso no solo atraería más reacciones negativas de los administradores del servidor web, sino que causaría una carga enorme en el rastreador web y la conexión de red entrante.

Afortunadamente, la mayoría de las páginas web no se actualizan cada pocos minutos. Algunos de ellos, como los sitios web de noticias, cambian con frecuencia. Otros, como la página de inicio de una persona, cambian con mucha menos frecuencia. Incluso dentro de un tipo de página, puede haber grandes variaciones en la tasa de modificación. Por ejemplo, algunos blogs se actualizan muchas veces al día, mientras que otros pasan de una actualización a otra. De poco sirve comprobar continuamente los sitios que rara vez se actualizan. Por lo tanto, uno de los trabajos del rastreador es medir la velocidad a la que cambia cada página. Con el tiempo, estos datos se pueden utilizar para estimar la frecuencia con la que cambia cada página.

Dado que un rastreador web no puede actualizar cada página inmediatamente a medida que cambia, el rastreador debe tener alguna métrica para medir la actualización del rastreo. En este capítulo, usamos frescura como un término general, pero frescura también es el nombre de una métrica.

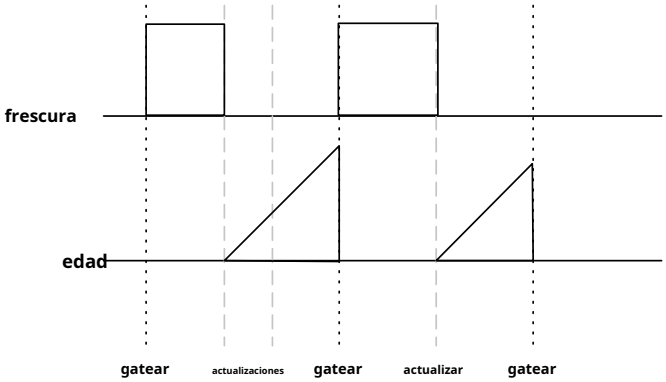


Figura 3.6. Edad y frescura de un una sola página a lo largo del tiempo

Bajo la *frescura* métrica, una página es *Fresco* si el rastreo tiene la copia más reciente de una página web, pero *duro* de lo contrario. La frescura es entonces la fracción de las páginas rastreadas que están actualizadas.

Mantener la frescura alta parece exactamente lo que le gustaría hacer, pero optimizar la frescura puede tener consecuencias no deseadas. Suponer quehttp://www.example.com es un sitio web popular que cambia ligeramente su página principal cada minuto. A menos que su rastreador realice encuestas continuamentehttp://www.example.com, casi siempre tendrá una copia obsoleta de esa página. Tenga en cuenta que si desea optimizar la actualización, la estrategia adecuada es dejar de rastrear este sitio por completo. Si nunca estará fresco, no puede ayudar a su valor de frescura. En su lugar, debe asignar los recursos de su rastreador a páginas que cambian con menos frecuencia.

Por supuesto, los usuarios se rebelarán si decide optimizar su rastreador para obtener frescura. Ellos miraránhttp://www.example.com y me pregunto por qué su copia indexada tiene meses de desactualización.

Edad es una mejor métrica para usar. Puede ver la diferencia entre edad y frescura en la Figura 3.6. En la parte superior de la figura, puede ver que las páginas se actualizan inmediatamente cuando se rastrean, pero una vez que la página cambia, la página rastreada se vuelve obsoleta. Según la métrica de edad, la página tiene una edad de 0 hasta que se cambia y, luego, su edad aumenta hasta que la página se rastrea nuevamente.

Supongamos que tenemos una página con frecuencia de cambio. λ, lo que significa que lo esperamos cambiar λ veces en uno∫-período de días Podemos calcular la edad esperada de una página t días después de la última vez que se rastreó:

$$\text{Edad}(\lambda, t) = \int_0^t \text{PAG(página cambiada en el momento } x) (t - x)\, dx$$

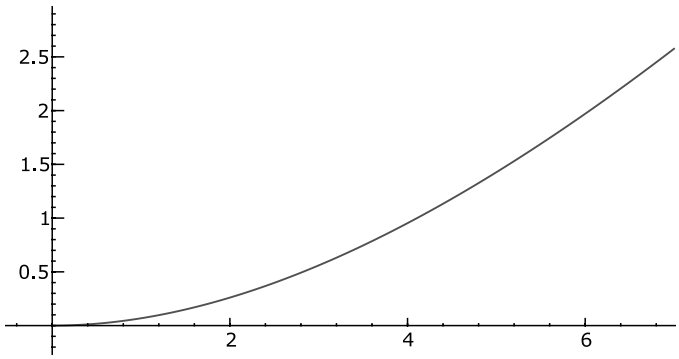


Figura 3.7. Antigüedad esperada de una página con frecuencia de cambio media $\lambda = 1/7$ (una semana)

La $(t - X)$ expresión es una edad: asumimos que la página se rastrea en el momento t, pero que cambió en el momento X. Multiplicamos eso por la probabilidad de que la página realmente haya cambiado en el momento X. Los estudios han demostrado que, en promedio, las actualizaciones de las páginas web siguen la distribución de Poisson, lo que significa que el tiempo hasta la próxima actualización se rige por una distribución exponencial (Cho y García-Molina, 2003). Esto nos da una fórmula para conectar a la $PAG($página ch∫enfadado en el momento $X)$ expresión:

$$\text{Edad}(\lambda, t) = \int_0^t \lambda e^{-\lambda x} (t - x)\, dx$$

La figura 3.7 muestra el resultado de graficar esta expresión para un $\lambda = 1/7$, lo que indica aproximadamente un cambio a la semana. Observe cómo la edad esperada comienza en cero y aumenta lentamente al principio. Esto se debe a que es poco probable que la página haya cambiado el primer día. A medida que pasan los días, aumenta la probabilidad de que la página haya cambiado. Al final de la semana, la edad esperada de la página es aproximadamente2.6 dias. Esto significa que si su rastreador rastrea cada página una vez a la semana, y cada página de su colección tiene un tiempo medio de actualización de una vez a la semana, las páginas de su índice tendrán una antigüedad media de 2,6 días justo antes de que el rastreador vuelva a ejecutarse.

Observe que la segunda derivada de la función Edad siempre es positiva. Es decir, el gráfico no solo aumenta, sino que su tasa de aumento siempre aumenta. Esta segunda derivada positiva significa que cuanto más antigua se vuelve una página, más le cuesta no rastrearla. La optimización de esta métrica nunca dará como resultado la conclusión de que la optimización para la frescura sí lo hace, donde a veces es económico no rastrear una página en absoluto.

3.2.4 Rastreo enfocado

A algunos usuarios les gustaría un motor de búsqueda que se centre en un tema específico de información. Por ejemplo, en un sitio web sobre películas, es posible que los usuarios deseen acceder a un motor de búsqueda que lleve a más información sobre películas. Si se construye correctamente, este tipo de *búsqueda vertical* puede proporcionar una mayor precisión que la búsqueda general debido a la falta de información extraña en la colección de documentos. El costo computacional de ejecutar una búsqueda vertical también será mucho menor que una búsqueda web completa, simplemente porque la colección será mucho menor.

La forma más precisa de obtener páginas web para este tipo de motor sería rastrear una copia completa de la web y luego descartar todas las páginas no relacionadas. Esta estrategia requiere una gran cantidad de espacio en disco y ancho de banda, y la mayoría de las páginas web se descartarán al final.

Un enfoque menos costoso es *enfocado* o *actual,* gateando. Un rastreador enfocado intenta descargar solo aquellas páginas que tratan sobre un tema en particular. Los rastreadores centrados se basan en el hecho de que las páginas sobre un tema tienden a tener enlaces a otras páginas sobre el mismo tema. Si esto fuera perfectamente cierto, sería posible iniciar un rastreo en una página del tema y luego rastrear todas las páginas de ese tema simplemente siguiendo los enlaces de una única página raíz. En la práctica, una serie de páginas populares para un tema específico se utilizan normalmente como semillas.

Los rastreadores enfocados requieren algunos medios automáticos para determinar si una página trata sobre un tema en particular. El capítulo 9 presentará los clasificadores de texto, que son herramientas que pueden hacer este tipo de distinción. Una vez que se descarga una página, el rastreador usa el clasificador para decidir si la página está en el tema. Si es así, la página se mantiene y los enlaces de la página se utilizan para buscar otros sitios relacionados. El texto de anclaje en los enlaces salientes es una pista importante de actualidad. Además, algunas páginas tienen más enlaces sobre el tema que otras. A medida que se visitan los enlaces de una página web en particular, el rastreador puede realizar un seguimiento de la actualidad de las páginas descargadas y utilizarlo para determinar si descargar otras páginas similares. Los datos de texto de anclaje y los datos de actualidad del enlace de la página se pueden combinar para determinar qué páginas se deben rastrear a continuación.

3.2.5 Web profunda

No todas las partes de la Web son fáciles de navegar para un rastreador. Los sitios que son difíciles de encontrar para un rastreador se denominan colectivamente *red profunda* (también llamado el *Web oculta).* Algunos estudios han estimado que la Deep Web tiene más de cien

veces más grande que la Web tradicionalmente indexada, aunque es muy difícil medir esto con precisión.

La mayoría de los sitios que forman parte de deepWeb se dividen en tres categorías generales:

- *Sitios privados* son intencionadamente privados. Es posible que no tengan enlaces entrantes, o pueden requerir que inicie sesión con una cuenta válida antes de usar el resto del sitio. Por lo general, estos sitios quieren bloquear el acceso de los rastreadores, aunque algunos editores de noticias pueden querer que su contenido sea indexado por los principales motores de búsqueda.
- *Resultados del formulario* son sitios a los que se puede acceder solo después de ingresar algunos datos en un formulario. Por ejemplo, los sitios web que venden billetes de avión suelen solicitar información sobre el viaje en la página de entrada del sitio. Se le muestra la información del vuelo solo después de enviar esta información del viaje. Aunque es posible que desee utilizar un motor de búsqueda para encontrar horarios de vuelos, la mayoría de los rastreadores no podrán acceder a este formulario para acceder a la información de horarios.
- *Páginas con guión* son páginas que utilizan JavaScript ™, Flash® u otro idioma del lado del cliente en la página web. Si un enlace no está en la fuente HTML sin procesar de la página web, sino que se genera mediante código JavaScript que se ejecuta en el navegador, el rastreador deberá ejecutar JavaScript en la página para encontrar el enlace. Aunque esto es técnicamente posible, la ejecución de JavaScript puede ralentizar significativamente el rastreador y agrega complejidad al sistema.

A veces la gente hace una distinción entre *páginas estáticas* y *páginas dinámicas.* Las páginas estáticas son archivos almacenados en un servidor web y mostrados en un navegador web sin modificar, mientras que las páginas dinámicas pueden ser el resultado de la ejecución de código en el servidor web o en el cliente. Por lo general, se asume que las páginas estáticas son fáciles de rastrear, mientras que las páginas dinámicas son difíciles. Sin embargo, esto no es del todo cierto. Muchos sitios web tienen páginas web generadas dinámicamente que son fáciles de rastrear; Los wikis son un buen ejemplo de esto. Otros sitios web tienen páginas estáticas que son imposibles de rastrear porque solo se puede acceder a ellas a través de formularios web.

Los administradores web de sitios con resultados de formularios y páginas con secuencias de comandos a menudo quieren que sus sitios estén indexados, a diferencia de los propietarios de sitios privados. De estas dos categorías, las páginas con secuencias de comandos son las más fáciles de tratar. Por lo general, el propietario del sitio puede modificar ligeramente las páginas para que los enlaces se generen por código en el servidor en lugar de por código en el navegador. El rastreador también puede ejecutar JavaScript de página, o quizás también Flash, aunque esto puede llevar mucho tiempo.

Los problemas más difíciles vienen con los resultados de forma. Por lo general, estos sitios son repositorios de datos cambiantes y el formulario envía una consulta a un sistema de base de datos. En el caso de que la base de datos contenga millones de registros, el sitio

exponer millones de enlaces al rastreador de un motor de búsqueda. Agregar un millón de enlaces a la página principal de un sitio de este tipo es claramente inviable. Otra opción es dejar que el rastreador adivine qué ingresar en los formularios, pero es difícil elegir una buena entrada de formulario. Incluso con buenas suposiciones, es poco probable que este enfoque exponga todos los datos ocultos.

3.2.6 Mapas del sitio

Como puede ver en las dos últimas secciones, los mayores problemas de rastreo surgen porque los propietarios de sitios no pueden informar adecuadamente a los rastreadores sobre sus sitios. En la sección 3.2.3, vimos cómo los rastreadores tienen que adivinar cuándo se actualizarán las páginas porque las encuestas son costosas. En la sección 3.2.5, vimos que los propietarios de sitios a veces tienen datos que les gustaría exponer a un motor de búsqueda, pero no pueden porque no hay un lugar razonable para almacenar los enlaces. *Mapas del sitio* resolver ambos problemas.

```
<? xml version = "1.0" encoding = "UTF-8"?>
<urlset xmlns = "http://www.sitemaps.org/schemas/sitemap/0.9">
   <url>
      <loc> http://www.company.com/ </loc>
      <lastmod> 2008-01-15 </lastmod>
      <changefreq> mensual </changefreq>
      <priority> 0,7 </priority>
   </url>
   <url>
      <loc> http://www.company.com/items?item=truck </loc>
      <changefreq> semanalmente </changefreq>
   </url>
   <url>
      <loc> http://www.company.com/items?item=bicycle </loc>
      <changefreq> diario </changefreq>
   </url>
</urlset>
```

Figura 3.8. Un archivo de mapa del sitio de ejemplo

Un archivo robots.txt puede contener una referencia a un mapa del sitio, como el que se muestra en la Figura 3.8. Un mapa del sitio contiene una lista de URL y datos sobre esas URL, como la hora de modificación y la frecuencia de modificación.

Hay tres entradas de URL que se muestran en el mapa del sitio de ejemplo. Cada uno contiene una URL en un`loc` etiqueta. La`changefreq` La etiqueta indica la frecuencia con la que es probable que cambie este recurso. La primera entrada incluye un`lastmod` etiqueta, que indica la última vez que se modificó. La primera entrada también incluye un`prioridad` etiqueta con un valor de 0,7, que es superior al valor predeterminado de 0,5. Esto les dice a los rastreadores que esta página es más importante que otras páginas de este sitio.

¿Por qué un administrador de servidor web se tomaría la molestia de crear un mapa del sitio? Una de las razones es que informa a los motores de búsqueda sobre páginas que de otro modo no encontrarían. Mire la segunda y tercera URL en el mapa del sitio. Suponga que se trata de dos páginas de productos. Es posible que no haya enlaces en el sitio web a estas páginas; en su lugar, es posible que el usuario tenga que utilizar un formulario de búsqueda para acceder a ellos. Un rastreador web simple no intentará ingresar nada en un formulario (aunque algunos rastreadores avanzados lo hacen), por lo que estas páginas serían invisibles para los motores de búsqueda. Un mapa del sitio permite a los rastreadores encontrar este contenido oculto.

El mapa del sitio también expone los tiempos de modificación. En la discusión sobre la actualización de la página, mencionamos que un rastreador generalmente tiene que adivinar cuándo es probable que cambien las páginas. La`changefreq` le da al rastreador una pista sobre cuándo volver a comprobar si hay cambios en una página, y `lastmod` La etiqueta le dice al rastreador cuando una página ha cambiado. Esto ayuda a reducir la cantidad de solicitudes que el rastreador envía a un sitio web sin sacrificar la actualización de la página.

3.2.7 Rastreo distribuido

Para rastrear sitios web individuales, una sola computadora es suficiente. Sin embargo, rastrear toda la Web requiere muchas computadoras dedicadas al rastreo. ¿Por qué no sería suficiente una sola computadora rastrera? Consideraremos tres razones.

Una razón para usar varios equipos es colocar el rastreador *más cerca de los sitios que rastrea.* Las conexiones de red de larga distancia tienden a tener un menor rendimiento (menos bytes copiados por segundo) y una mayor latencia (los bytes tardan más en cruzar la red). La disminución del rendimiento y el aumento de la latencia funcionan en conjunto para hacer que cada solicitud de página demore más. A medida que disminuye el rendimiento y aumenta la latencia, el rastreador tiene que abrir más conexiones para copiar páginas a la misma velocidad.

Por ejemplo, suponga que un rastreador tiene una conexión de red que puede transferir 1 MB por segundo. Con un tamaño de página web promedio de 20K, puede copiar 50 páginas por segundo. Si los sitios que se están rastreando están cerca, la velocidad de transferencia de datos desde ellos puede ser de 1 MB por segundo. Sin embargo, el sitio puede tardar 80 ms en comenzar a enviar datos, ya que hay cierto retraso en la transmisión al abrir la conexión.

y enviando la solicitud. Supongamos que cada solicitud toma 100 ms (80 ms de latencia y 20 ms de transferencia de datos). Multiplicando 50 por 100 ms, vemos que hay 5 segundos de espera involucrados en la transferencia de 50 páginas. Esto significa que se necesitarán cinco conexiones para transferir 50 páginas en un segundo. Si los sitios están más lejos, con un rendimiento promedio de 100 K por segundo y 500 ms de latencia, cada solicitud ahora tomaría 600 ms. Desde $50 \times 600\text{ms} = 30$ segundos, el rastreador necesitaría mantener abiertas 30 conexiones para transferir páginas a la misma velocidad.

Otro motivo para el rastreo de varios equipos es reducir la *número de sitios que el rastreador debe recordar.* Un rastreador debe recordar todas las URL que ya ha rastreado y todas las URL que ha puesto en cola para rastrear. Estas URL deben ser de fácil acceso, porque cada página que se rastrea contiene nuevos vínculos que deben agregarse a la cola de rastreo. Dado que la cola del rastreador no debe contener duplicados o sitios que ya se hayan rastreado, cada URL nueva debe comprobarse con todo lo que hay en la cola y todo lo que se ha rastreado. La estructura de datos para esta búsqueda debe estar en RAM; de lo contrario, la velocidad de rastreo de la computadora se verá muy limitada. Distribuir las tareas de rastreo entre muchas computadoras reduce esta carga contable.

Otra razón más es que el rastreo puede usar una gran cantidad de *recursos informáticos,* incluyendo recursos de CPU para analizar y ancho de banda de red para rastrear páginas. Rastrear una gran parte de la Web es demasiado trabajo para que lo pueda manejar una sola computadora.

Un rastreador distribuido es muy parecido a un rastreador en una sola computadora, excepto que en lugar de una única cola de URL, hay muchas colas. El rastreador distribuido utiliza una función hash para asignar URL a los equipos de rastreo. Cuando un rastreador ve una nueva URL, calcula una función hash en esa URL para decidir qué computadora de rastreo es responsable de ella. Estas URL se recopilan en lotes y luego se envían periódicamente para reducir la sobrecarga de red de enviar una sola URL a la vez.

La función hash debe calcularse solo en la parte del host de cada URL. Esto asigna todas las URL de un host en particular a un único rastreador. Aunque esto puede promover un desequilibrio, ya que algunos hosts tienen más páginas que otros, las reglas de cortesía requieren un retraso de tiempo entre las búsquedas de URL en el mismo host. Es más fácil mantener ese tipo de demora utilizando los mismos equipos de rastreo para todas las URL del mismo host. Además, esperaríamos que los sitios de dominio.com tendrá muchos enlaces a otras páginas en dominio.com. Asignando dominio.com a un único host de rastreo, minimizamos la cantidad de URL que deben intercambiarse entre los equipos de rastreo.

3.3 Rastreo de documentos y correo electrónico

Aunque la Web es un recurso de información enorme, una gran cantidad de información digital no se almacena en los sitios web. En esta sección, consideraremos la información que puede encontrar en una computadora de escritorio normal, como correo electrónico, documentos de procesamiento de texto, presentaciones u hojas de cálculo. Esta información se puede buscar mediante un *búsqueda de escritorio* herramienta. En empresas y organizaciones, *búsqueda empresarial* hará uso de documentos en servidores de archivos, o incluso en las computadoras de escritorio de los empleados, además de las páginas web locales.

Muchos de los problemas del rastreo web cambian cuando miramos los datos del escritorio. En el rastreo web, solo encontrar los datos puede resultar complicado. En una computadora de escritorio, los datos interesantes se almacenan en un sistema de archivos con semántica familiar. Encontrar todos los archivos en un disco duro no es particularmente difícil, ya que los sistemas de archivos tienen directorios que son fáciles de descubrir. De alguna manera, un sistema de archivos es como un servidor web, pero con un mapa del sitio generado automáticamente.

Sin embargo, existen desafíos únicos en el rastreo de datos de escritorio. La primera se refiere a la velocidad de actualización. En las aplicaciones de búsqueda de escritorio, los usuarios exigen resultados de búsqueda basados en el contenido actual de sus archivos. Esto significa, por ejemplo, poder buscar un correo electrónico en el instante en que se recibe y poder buscar un documento tan pronto como se haya guardado. Tenga en cuenta que esta es una expectativa muy diferente a la de la búsqueda web, donde los usuarios pueden tolerar demoras de rastreo de horas o días. Rastrear el sistema de archivos cada segundo no es práctico, pero los sistemas de archivos modernos pueden enviar notificaciones de cambios directamente al proceso del rastreador para que pueda copiar nuevos archivos inmediatamente. Los sistemas de archivos remotos de los servidores de archivos no suelen proporcionar este tipo de notificación de cambios, por lo que deben rastrearse como un servidor web.

El espacio en disco es otra preocupación. Con un rastreador web, asumimos que necesitamos guardar una copia de cada documento que se encuentre. Esto es menos cierto en un sistema de escritorio, donde los documentos ya están almacenados localmente y donde los usuarios no estarán contentos si el indexador toma una gran parte del disco duro. En cambio, un rastreador de escritorio puede necesitar leer documentos en la memoria y enviarlos directamente al indexador. Discutiremos más sobre la indexación en el Capítulo 5.

Dado que los sitios web están destinados a ser vistos con navegadores web, la mayor parte del contenido web se almacena en HTML. Por otro lado, cada programa de escritorio (el procesador de texto, la herramienta de presentación, el programa de correo electrónico, etc.) tiene su propio formato de archivo. Entonces, simplemente encontrar estos archivos no es suficiente; eventualmente deberán convertirse a un formato que el indexador pueda entender. En la sección 3.5 revisaremos este tema de conversión.

Finalmente, y quizás lo más importante, el rastreo de datos de escritorio requiere un enfoque en la privacidad de los datos. Los sistemas de escritorio pueden tener varios usuarios con diferentes cuentas y usuarios*A* no debería poder encontrar correos electrónicos del usuario *B*cuenta a través de la función de búsqueda. Esto es especialmente importante cuando consideramos el rastreo de sistemas de archivos de red compartidos, como en una red corporativa. Los permisos de acceso a archivos de cada archivo deben registrarse junto con los datos rastreados y deben mantenerse actualizados.

3.4 Alimentación de documentos

En general, en el rastreo web o de escritorio, asumimos que cualquier documento puede ser creado o modificado en cualquier momento. Sin embargo, muchos documentos son*publicado,* lo que significa que se crean en un momento fijo y rara vez se actualizan de nuevo. Artículos de noticias, publicaciones de blogs, comunicados de prensa y correos electrónicos son algunos de los documentos que se ajustan a este modelo de publicación. Se publica la mayor parte de la información urgente.

Dado que cada documento publicado tiene un tiempo asociado, los documentos publicados de una sola fuente se pueden ordenar en una secuencia llamada *alimentación de documentos.* Una alimentación de documentos es particularmente interesante para los rastreadores, ya que el rastreador puede encontrar fácilmente todos los documentos nuevos examinando solo el final de la alimentación.

Podemos distinguir dos tipos de alimentación de documentos, *empujar* y *jalar*. A *empujar* feed alerta al suscriptor sobre nuevos documentos. Es como un teléfono que le avisa de una llamada telefónica entrante; no es necesario que revise continuamente el teléfono para ver si alguien está llamando. A*jalar* el feed requiere que el suscriptor revise periódicamente si hay nuevos documentos; esto es como revisar su buzón de correo para ver si llega nuevo correo. Los feeds de noticias de las agencias de noticias comerciales suelen ser push feeds, pero los pull feeds son abrumadoramente populares para los servicios gratuitos. Nos centraremos principalmente en pull feeds en esta sección.

El formato más común para pull feeds se llama *RSS.* RSS tiene al menos tres definiciones: Really Simple Syndication, RDF Site Summary o Rich Site Summary. Como era de esperar, RSS también tiene una serie de implementaciones ligeramente incompatibles, y existe un formato competitivo similar llamado el*Formato de distribución Atom.* La proliferación de estándares es el resultado de una idea que ganó popularidad demasiado rápido para que los desarrolladores se pusieran de acuerdo sobre un estándar único.

La figura 3.9 muestra un feed RSS2.0 de un sitio de ejemplo llamadohttp://www.searchengine-news.org. Este feed contiene dos artículos: uno sobre una próxima conferencia SIGIR y el otro sobre un libro de texto. Tenga en cuenta que cada entrada contiene una hora que indica cuándo se publicó. Además, cerca de la parte superior de la fuente RSS hay una etiqueta llamadattl, lo que significa *tiempo para vivir,* medido en minutos. Esto

```
<? xml version = "1.0"?>
<rss version = "2.0">
   <canal>
      <title> Noticias sobre motores de búsqueda </title> <link> http://
      www.search-engine-news.org/ </link> <description> Noticias sobre
      motores de búsqueda. </description> <language> en-us < / idioma>

      <pubDate> Mar, 19 de junio de 2008 05:17:00 GMT </pubDate>
      <ttl> 60 </ttl>

      <item>
         <title> Próxima Conferencia SIGIR </title> <link> http://www.sigir.org/
         conference </link> <description> ¡Se acerca la conferencia anual
         SIGIR!
            Marque sus calendarios y busque vuelos baratos. </
            description>
         <pubDate> Mar, 05 de junio de 2008 09:50:11 GMT </pubDate>
        <guid> http: //search-engine-news.org#500 </guid> </item>


      <item>
         <title> Nuevo libro de texto para motores de búsqueda </title> <link>
         http://www.cs.umass.edu/search-book </link> <description> Un
         nuevo libro de texto sobre motores de búsqueda
            se publicará pronto. </description> <pubDate> Mar, 05 de
        junio de 2008 09:33:01 GMT </pubDate> <guid> http: //search-
        engine-news.org#499 </guid> </ item >

   </canal>
</rss>
```

Figura 3.9. Un ejemplo de feed RSS 2.0

feed establece que su contenido debe almacenarse en caché solo durante 60 minutos, y la información de más de una hora debe considerarse obsoleta. Esto le da al rastreador una indicación de la frecuencia con la que se debe rastrear este archivo de feed.

Se accede a las fuentes RSS como a una página web tradicional, utilizando solicitudes HTTP GET a los servidores web que las alojan. Por lo tanto, algunas de las técnicas de rastreo que discutimos antes también se aplican aquí, como el uso de solicitudes HTTP HEAD para detectar cuándo cambian los feeds RSS.

Desde una perspectiva de rastreo, la alimentación de documentos tiene una serie de ventajas sobre las páginas tradicionales. Los feeds dan una estructura natural a los datos; incluso más que con un mapa del sitio, una fuente web implica alguna relación entre los elementos de datos. Los feeds son fáciles de analizar y contienen información de tiempo detallada, como un mapa del sitio, pero también incluyen un campo de descripción sobre cada página (y este campo de descripción a veces contiene el texto completo de la página a la que se hace referencia en la URL). Más importante aún, como un mapa del sitio, los feeds proporcionan una única ubicación para buscar nuevos datos, en lugar de tener que rastrear un sitio completo para encontrar algunos documentos nuevos.

3.5 El problema de la conversión

Los motores de búsqueda están diseñados para buscar a través de texto. Desafortunadamente, el texto se almacena en las computadoras en cientos de formatos de archivo incompatibles. Los formatos de archivo de texto estándar incluyen texto sin formato, RTF, HTML, XML, Microsoft Word, ODF (OpenDocument Format) y PDF (PortableDocument Format). Hay decenas de otros procesadores de texto menos comunes con sus propios formatos de archivo. Pero los documentos de texto no son el único tipo de documento que debe buscarse; otros tipos de archivos también contienen texto importante, como diapositivas de PowerPoint y hojas de cálculo de Excel®. Además de todos estos formatos, las personas a menudo desean buscar documentos antiguos, lo que significa que los motores de búsqueda pueden necesitar admitir formatos de archivo obsoletos. No es raro que un motor de búsqueda comercial admita más de cien tipos de archivos.

La forma más común de manejar un nuevo formato de archivo es utilizar una herramienta de conversión que convierta el contenido del documento en un formato de texto etiquetado como HTML o XML. Estos formatos son fáciles de analizar y conservan parte de la información de formato importante (tamaño de fuente, por ejemplo). Puede ver esto en cualquier motor de búsqueda web importante. Busque un documento PDF, pero luego haga clic en el enlace "En caché" en la parte inferior de un resultado de búsqueda. Se le llevará a la vista de la página del motor de búsqueda, que suele ser una versión HTML del documento original. Para algunos tipos de documentos, como PowerPoint, esta versión en caché puede resultar casi ilegible. Afortunadamente, la legibilidad no es la principal preocupación del motor de búsqueda.

El punto es copiar estos datos en el motor de búsqueda para que puedan indexarse y recuperarse. Sin embargo, traducir los datos a HTML tiene una ventaja: el usuario no necesita tener una aplicación que pueda leer el formato de archivo del documento para poder verlo. Esto es fundamental para los formatos de archivo obsoletos.

Los documentos se pueden convertir a texto sin formato en lugar de HTML o XML. Sin embargo, hacer esto quitaría al archivo información importante sobre encabezados y tamaños de fuente que podrían ser útiles para el indexador. Como veremos más adelante, los títulos y el texto en negrita tienden a contener palabras que describen bien el contenido del documento, por lo que queremos darle a estas palabras un trato preferencial durante la puntuación. La conversión precisa de la información de formato permite al indexador extraer estas características importantes.

3.5.1 Codificaciones de caracteres

Incluso los archivos HTML no son necesariamente compatibles entre sí debido a *codificación de caracteres* asuntos. El texto que ves en esta página es una serie de pequeñas imágenes que llamamos*letras* o *glifos.* Por supuesto, un archivo de computadora es un flujo de bits, no una colección de imágenes. Una codificación de caracteres es un mapeo entre bits y glifos. Para el inglés, la codificación de caracteres básica que ha existido desde 1963 es ASCII. ASCII codifica 128 letras, números, caracteres especiales y caracteres de control en 7 bits, ampliado con un bit adicional para almacenamiento en bytes. Este esquema está bien para el alfabeto inglés de 26 letras, pero hay muchos otros idiomas, y algunos de ellos tienen muchos más glifos. El idioma chino, por ejemplo, tiene más de 40.000 caracteres, con más de 3.000 de uso común. Para la familia CJK (chino-japonés-coreano) de idiomas de Asia oriental, esto llevó al desarrollo de varios estándares diferentes de 2 bytes. Otros idiomas, como el hindi o el árabe, también tienen una variedad de codificaciones diferentes. Tenga en cuenta que no todas las codificaciones coinciden en inglés. La codificación EBCDIC utilizada en los marcos principales, por ejemplo, es completamente diferente a la codificación ASCII utilizada por las computadoras personales.

La industria informática se ha movido lentamente en el manejo de conjuntos de caracteres complicados, como el chino y el árabe. Hasta hace poco, el enfoque típico era utilizar diferentes codificaciones específicas del idioma, a veces llamadas*páginas de códigos.* Los primeros 128 valores de cada codificación están reservados para caracteres típicos en inglés, puntuación y números. Los números superiores a 128 se asignan a glifos en el idioma de destino, desde el hebreo hasta el árabe. Sin embargo, si usa una codificación diferente para cada idioma, no puede escribir en hebreo y japonés en el mismo documento. Además, el texto en sí ya no es autodescriptivo. No es suficiente almacenar datos en un archivo de texto; también debe registrar qué codificación se utilizó.

Para resolver este lío de problemas de codificación, *Unicode* fue desarrollado. Unicode es un mapeo único de números a glifos que intenta incluir todos los glifos de uso común en todos los idiomas conocidos. Esto resuelve el problema de usar varios idiomas en un solo archivo. Desafortunadamente, no resuelve por completo los problemas de las codificaciones binarias, porque Unicode es un mapeo entre números y glifos, no bits y glifos. ¡Resulta que hay muchas formas de traducir números Unicode a glifos! Algunos de los más populares incluyen UTF-8, UTF-16, UTF-32 y UCS-2 (que está en desuso).

La proliferación de codificaciones proviene de la necesidad de compatibilidad y de ahorrar espacio. La codificación de texto en inglés en UTF-8 es idéntica a la codificación ASCII. Cada letra ASCII requiere solo un byte. Sin embargo, algunos caracteres chinos tradicionales pueden requerir hasta 4 bytes. La compensación por la compacidad para los lenguajes occidentales es que cada carácter requiere un número variable de bytes, lo que hace que sea difícil calcular rápidamente el número de caracteres en una cadena o saltar a una ubicación aleatoria en una cadena. Por el contrario, UTF-32 (también conocido como UCS-4) utiliza exactamente 4 bytes para cada carácter. Saltar al vigésimo carácter en una cadena UTF-32 es fácil: simplemente salta al octogésimo byte y comienza a leer. Desafortunadamente, las cadenas UTF-32 son incompatibles con todo el software ASCII antiguo, y los archivos UTF-32 requieren cuatro veces más espacio que UTF-8. Debido a esto, muchas aplicaciones usan UTF-32 como su codificación de texto interna (donde el acceso aleatorio es importante), pero use UTF-8 para almacenar texto en disco.

Decimal	Hexadecimal	Codificación
0-127	0–7F	0xxxxxxx
128-2047	80–7FF	110xxxxx 10xxxxxx
2048–55295	800 – D7FF	1110xxxx 10xxxxxx 10xxxxxx
55296–57343	D800 – DFFF	Indefinido
57344–65535	E000 – FFFF	1110xxxx 10xxxxxx 10xxxxxx
65536–1114111	10000–10FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Cuadro 3.1. Codificación UTF-8

La Tabla 3.1 muestra una tabla de codificación para UTF-8. Las columnas de la izquierda representan rangos de valores decimales y la columna de la derecha muestra cómo estos valores se codifican en binario. Lax los caracteres representan dígitos binarios. Por ejemplo, la letra griega pi (π) es el número de símbolo Unicode 960. En binario, ese número es 00000011 11000000 (3C0 en hexadecimal). La segunda fila de la tabla nos dice

que esta letra requerirá 2 bytes para codificarse en UTF-8. Los 5 bits más altos del carácter van en el primer byte y los siguientes 6 bits van en el segundo byte. La codificación final es**110**01111 **10**000000 (CF80 en hexadecimal). Los dígitos binarios en negrita son los mismos que los dígitos de la tabla, mientras que x las letras de la tabla se han rellenado con dígitos binarios del número Unicode.

3.6 Almacenamiento de documentos

Una vez que los documentos se han convertido a algún formato común, deben almacenarse como preparación para la indexación. El almacenamiento de documentos más simple no es el almacenamiento de documentos, y para algunas aplicaciones esto es preferible. En la búsqueda de escritorio, por ejemplo, los documentos ya están almacenados en el sistema de archivos y no es necesario copiarlos en otro lugar. A medida que se ejecuta el proceso de rastreo, puede enviar documentos convertidos inmediatamente a un proceso de indexación. Al no almacenar los documentos convertidos intermedios, los sistemas de búsqueda de escritorio pueden ahorrar espacio en disco y mejorar la latencia de indexación.

La mayoría de los otros tipos de motores de búsqueda necesitan almacenar documentos en algún lugar. Se requiere acceso rápido al texto del documento para crear fragmentos de documento[2] para cada resultado de búsqueda. Estos fragmentos de texto le dan al usuario una idea de lo que hay dentro del documento recuperado sin necesidad de hacer clic en un enlace.

Incluso si los fragmentos no son necesarios, existen otras razones para guardar una copia de cada documento. La búsqueda de documentos puede resultar costosa tanto en términos de CPU como de carga de red. Tiene sentido mantener copias de los documentos en lugar de intentar recuperarlos la próxima vez que desee crear un índice. Mantener documentos antiguos le permite usar solicitudes HEAD en su rastreador para ahorrar ancho de banda, o rastrear solo un subconjunto de las páginas en su índice.

Finalmente, los sistemas de almacenamiento de documentos pueden ser un punto de partida para la extracción de información (descrito en el Capítulo 4). El tipo de extracción de información más generalizado ocurre en los motores de búsqueda web, que extraen el texto de anclaje de los enlaces para almacenarlos con los documentos web de destino. Son posibles otros tipos de extracción, como identificar nombres de personas o lugares en documentos. Tenga en cuenta que si se utiliza la extracción de información en la aplicación de búsqueda, el sistema de almacenamiento de documentos debe admitir la modificación de los datos del documento.

Ahora discutimos algunos de los requisitos básicos para un sistema de almacenamiento de documentos, incluido el acceso aleatorio, la compresión y la actualización, y consideramos los aspectos relativos

[2] Analizamos la generación de fragmentos en el Capítulo 6.

beneficios de utilizar un sistema de base de datos o un sistema de almacenamiento personalizado como BigTable de Google.

3.6.1 Uso de un sistema de base de datos

Si ha utilizado una base de datos relacional anteriormente, es posible que esté pensando que una base de datos sería un buen lugar para almacenar datos de documentos. Para muchas aplicaciones, de hecho, una base de datos es un lugar excelente para almacenar documentos. Una base de datos se encarga de los detalles difíciles de almacenar pequeños fragmentos de datos, como páginas web, y facilita su actualización posterior. La mayoría de las bases de datos también se ejecutan como un servidor de red, por lo que los documentos están fácilmente disponibles en la red. Esto podría admitir, por ejemplo, una sola computadora que sirva documentos para fragmentos mientras que muchas otras computadoras manejan consultas. Las bases de datos también suelen incluir herramientas útiles de importación y análisis que pueden facilitar la gestión de la colección de documentos.

Muchas empresas que ejecutan motores de búsqueda web se muestran reacias a hablar sobre sus tecnologías internas. Sin embargo, parece que pocos, si alguno, de los principales motores de búsqueda utilizan bases de datos relacionales convencionales para almacenar documentos. Un problema es el gran volumen de datos de documentos, que puede abrumar a los sistemas de bases de datos tradicionales. Los proveedores de bases de datos también tienden a esperar que los servidores de bases de datos utilicen los sistemas de disco más costosos, lo cual no es práctico dado el tamaño de la colección. Discutimos una alternativa a una base de datos relacional al final de esta sección que aborda algunas de estas preocupaciones.

3.6.2 Acceso aleatorio

Para recuperar documentos rápidamente a fin de calcular un fragmento para un resultado de búsqueda, el almacén de documentos debe admitir el acceso aleatorio. Sin embargo, en comparación con una base de datos relacional completa, solo se necesita un criterio de búsqueda relativamente simple. Queremos un almacén de datos en el que podamos solicitar el contenido de un documento en función de su URL.

La forma más sencilla de manejar este tipo de búsqueda es con hash. El uso de una función hash en la URL nos da un número que podemos usar para encontrar los datos. Para instalaciones pequeñas, la función hash puede decirnos qué archivo contiene el documento. Para instalaciones más grandes, la función hash nos dice qué servidor contiene el documento. Una vez que la ubicación del documento se ha reducido a un solo archivo, se puede utilizar un árbol B o una estructura de datos ordenados para encontrar el conjunto de datos del documento dentro del archivo.

3.6.3 Compresión y archivos grandes

Independientemente de si la aplicación requiere acceso aleatorio a los documentos, el sistema de almacenamiento de documentos debe utilizar archivos grandes y compresión.

Incluso un documento que a una persona le parece largo es pequeño para los estándares informáticos modernos. Por ejemplo, este capítulo tiene aproximadamente 10,000 palabras y esas palabras requieren alrededor de 70 K de espacio en disco para almacenar. Eso es mucho más grande que la página web promedio, pero un disco duro moderno puede transferir 70K de datos en aproximadamente un milisegundo. Sin embargo, el disco duro requiere 10 milisegundos para buscar ese archivo y comenzar a leer. Es por eso que almacenar cada documento en su propio archivo no es una muy buena idea; leer estos pequeños archivos requiere una sobrecarga considerable para abrirlos. Una mejor solución es almacenar muchos documentos en un solo archivo, y que ese archivo sea lo suficientemente grande como para que transferir el contenido del archivo lleve mucho más tiempo que buscar el principio. Una buena elección de tamaño podría estar en los cientos de megabytes. Al almacenar documentos muy juntos,

El motor de búsqueda de TheGalago incluye analizadores para tres formatos de documentos compuestos: ARC, TRECText y TRECWeb. En cada formato, muchos documentos de texto se almacenan en el mismo archivo, con regiones cortas de metadatos del documento que separan los documentos. La figura 3.10 muestra un ejemplo del formato TRECWeb. Observe que cada bloque de documentos comienza con un <DOC> etiqueta y termina con un </DOC> etiqueta. Al comienzo del documento, el <DOCHDR> marca una sección que contiene la información sobre la solicitud de la página, como su URL, la fecha en que se rastreó y los encabezados HTTP devueltos por el servidor web. Cada registro de documento también contiene un <DOCNO> campo que incluye un identificador único para el documento.

Aunque los archivos grandes tienen sentido para la transferencia de datos desde el disco, la reducción de los requisitos de almacenamiento total para las colecciones de documentos tiene ventajas obvias. Afortunadamente, el texto escrito por personas es muy redundante. Por ejemplo, la letra *q* es casi siempre seguida por la letra *u.* Shannon (1951) demostró que los hablantes nativos de inglés pueden adivinar la siguiente letra de un pasaje de un texto en inglés con un 69% de precisión. Las etiquetas HTML y XML son aún más redundantes. *Compresión* Las técnicas aprovechan esta redundancia para reducir el tamaño de los archivos sin perder nada del contenido. Cubriremos la compresión como se usa para indexar documentos en el Capítulo 5, en parte porque la compresión para indexar es bastante especializada. Mientras continúa la investigación sobre la compresión de texto, algoritmos populares como DEFLATE (Deutsch,
1996) y LZW (Welch, 1984) pueden comprimir texto HTML y XML en un 80%. Este ahorro de espacio reduce el costo de almacenar una gran cantidad de documentos y también reduce

<DOC>
<DOCNO> WTX001-B01-10 </DOCNO>
<DOCHDR>
http://www.example.com/test.html 204.244.59.33 19970101013145 texto / html 440 HTTP / 1.0 200 OK

Fecha: miércoles, 01 de enero de 1997 01:21:13
GMT Servidor: Apache / 1.0.3
Tipo de contenido: texto / html
Longitud del contenido: 270
Última modificación: lunes 25 de noviembre de 1996 a las 05:31:24 GMT </
DOCHDR>
<HTML>
<TITLE> Tienda de pescado tropical </TITLE>
¡Próximamente!
</HTML>
</DOC>
<DOC>
<DOCNO> WTX001-B01-109 </DOCNO>
<DOCHDR>
http://www.example.com/fish.html 204.244.59.33 19970101013149 texto / html 440 HTTP / 1.0 200 OK

Fecha: miércoles, 01 de enero de 1997 01:21:19
GMT Servidor: Apache / 1.0.3
Tipo de contenido: texto / html
Longitud del contenido: 270
Última modificación: lunes 25 de noviembre de 1996 a las 05:31:24 GMT </
DOCHDR>
<HTML>
<TITLE> Información sobre peces </TITLE>
Esta página pronto contendrá información
interesante sobre peces tropicales. </HTML>

</DOC>

Figura 3.10. Un ejemplo de texto en el formato de documento compuesto TRECWeb

la cantidad de tiempo que se tarda en leer un documento del disco, ya que hay menos bytes para leer.

La compresión funciona mejor con grandes bloques de datos, lo que la convierte en una buena opción para archivos grandes con muchos documentos. Sin embargo, no es necesariamente una buena idea comprimir todo el archivo como un solo bloque. La mayoría de los métodos de compresión no permiten el acceso aleatorio, por lo que cada bloque solo se puede descomprimir de forma secuencial. Si desea un acceso aleatorio a los datos, es mejor considerar la posibilidad de comprimirlos en bloques más pequeños, tal vez un bloque por documento o un bloque para algunos documentos. Los bloques pequeños reducen las relaciones de compresión (la cantidad de espacio ahorrado) pero mejoran la latencia de las solicitudes.

3.6.4 Actualizar

A medida que ingresan nuevas versiones de documentos del rastreador, tiene sentido actualizar el almacén de documentos. La alternativa es crear un almacén de documentos completamente nuevo fusionando los documentos nuevos y modificados del rastreador con los datos del documento del antiguo almacén de documentos para los documentos que no cambiaron. Si los datos del documento no cambian mucho, este proceso de fusión será mucho más costoso que actualizar los datos en su lugar.

```
<a href="http://example.com"> Sitio web de ejemplo </a>
```

Figura 3.11. Un enlace de ejemplo con texto de anclaje

Otra razón importante para admitir la actualización es manejar el texto de anclaje. La figura 3.11 muestra un ejemplo de texto de anclaje en una etiqueta de enlace HTML. El código HTML de la figura se mostrará en el navegador web como un enlace, con el textoSitio web de ejemplo que, al hacer clic, dirigirá al usuario a http://example.com. El texto de anclaje es una característica importante porque proporciona un resumen conciso de lo que trata la página de destino. Si el enlace proviene de un sitio web diferente, también podemos creer que el resumen es imparcial, lo que también nos ayuda a clasificar los documentos (consulte los Capítulos 4 y 7).

Recopilar correctamente el texto de anclaje es difícil porque el texto de anclaje debe estar asociado con la página de destino. Una forma sencilla de abordar esto es utilizar un almacén de datos que admita la actualización.Cuando se encuentra un documento que contiene texto de anclaje, buscamos el registro de la página de destino y actualizamos la parte del texto de anclaje del registro. el documento, el texto de anclaje está todo junto y listo para indexar.

3.6.5 BigTable

Aunque una base de datos puede realizar las tareas de un almacén de datos de documentos, las colecciones de documentos más grandes exigen sistemas de almacenamiento de documentos personalizados. BigTable es el más conocido de estos sistemas (Chang et al., 2006). BigTable es un sistema en funcionamiento que se utiliza internamente en Google, aunque al menos dos proyectos de código abierto están adoptando un enfoque similar. En los siguientes párrafos, veremos la arquitectura de BigTable para ver cómo el problema del almacenamiento de documentos influyó en su diseño.

BigTable es un sistema de base de datos distribuido originalmente construido para la tarea de almacenar páginas web. Una instancia de BigTable es realmente una *grande* mesa; puede tener un tamaño superior a un petabyte, pero cada base de datos contiene solo una tabla. La mesa se divide en pedazos pequeños, llamados *tabletas,* que son atendidos por miles de máquinas (Figura 3.12).

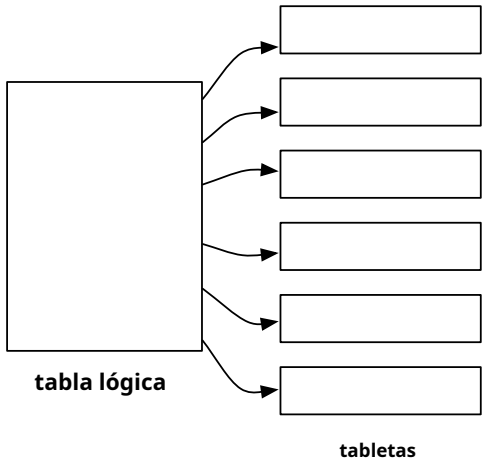


Figura 3.12. BigTable almacena datos en una sola tabla lógica, que se divide en muchas tabletas más pequeñas

Si está familiarizado con las bases de datos relacionales, habrá encontrado SQL (lenguaje de consulta estructurado). SQL permite a los usuarios escribir consultas complejas y computacionalmente costosas, y una de las tareas del sistema de base de datos es optimizar el procesamiento de estas consultas para que sean lo más rápido posible. Debido a que algunas de estas consultas pueden tardar mucho en completarse, una base de datos relacional grande requiere un sistema de bloqueo complejo para garantizar que muchos usuarios de la base de datos no la corrompan leyendo o escribiendo datos simultáneamente. Aislar a los usuarios entre sí es un trabajo difícil, y se han escrito muchos artículos y libros sobre cómo hacerlo bien.

El enfoque de BigTable es bastante diferente. No hay un lenguaje de consulta y, por lo tanto, no hay consultas complejas, e incluye solo transacciones de nivel de fila, que se considerarían bastante simples según los estándares de bases de datos relacionales. Sin embargo, la simplicidad del modelo permite a BigTable escalar hasta tamaños de base de datos muy grandes mientras se utilizan computadoras económicas, aunque pueden ser propensas a fallar.

La mayor parte de la ingeniería en BigTable implica la recuperación de fallas. Las tabletas, que son las secciones pequeñas de la tabla, se almacenan en un sistema de archivos replicado al que pueden acceder todos los servidores de tabletas BigTable. Cualquier cambio en una tableta BigTable se registra en un registro de transacciones, que también se almacena en un sistema de archivos compartido. Si algún servidor de tableta falla, otro servidor puede leer inmediatamente los datos de la tableta y el registro de transacciones del sistema de archivos y tomar el control.

La mayoría de las bases de datos relacionales almacenan sus datos en archivos que se modifican constantemente. Por el contrario, BigTable almacena sus datos en archivos inmutables (inmutables). Una vez que los datos del archivo se escriben en un archivo BigTable, nunca se cambian. Esto también ayuda en la recuperación de fallas. En los sistemas de bases de datos relacionales, la recuperación de fallas requiere una serie compleja de operaciones para asegurarse de que los archivos no se corrompan porque solo algunas de las escrituras pendientes se completaron antes de que la computadora fallara. En BigTable, un archivo está incompleto (en cuyo caso se puede desechar y volver a crear a partir de otros archivos BigTable y el registro de transacciones) o está completo y, por lo tanto, no está dañado. Para permitir las actualizaciones de la tabla, los datos más recientes se almacenan en la RAM, mientras que los datos más antiguos se almacenan en una serie de archivos. Periódicamente, los archivos se fusionan para
reducir el número total de archivos de disco.

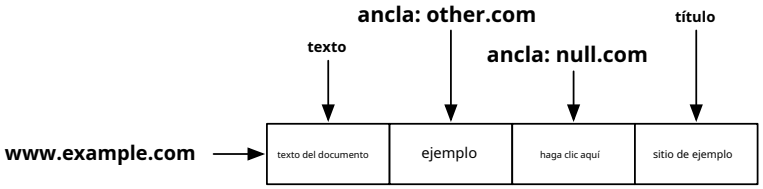


Figura 3.13. Una fila de BigTable

Las BigTables están organizadas lógicamente por filas (Figura 3.13). En la figura, la fila almacena los datos de una sola página web. La URLwww.example.com, es la clave de fila, que se puede usar para encontrar esta fila. La fila tiene muchas columnas, cada una con un nombre único. Cada columna puede tener muchas marcas de tiempo diferentes, aunque eso no se muestra en la figura. La combinación de una clave de fila, una clave de columna y un tiempo

apisonar punto a un solo *célula* en la fila. La celda contiene una serie de bytes, que pueden ser un número, una cadena o algún otro tipo de datos.

En la figura, observe que hay una columna de texto para el texto completo del documento, así como una columna de título, lo que facilita la búsqueda rápida del título del documento sin analizar el texto completo del documento. Hay dos columnas para el texto de anclaje. Uno, llamadoancla: other.com, incluye texto de anclaje de un enlace del sitio other.com a example.com; el texto del enlace es "ejemplo", como se muestra en la celda. Laancla: null.comdescribe un enlace denull.coma example.comcon texto de anclaje "haga clic aquí". Ambas columnas están en elancla *grupo de columnas.* Se podrían agregar otras columnas a este grupo de columnas para agregar información sobre más enlaces.

BigTable puede tener una gran cantidad de columnas por fila y, aunque todas las filas tienen los mismos grupos de columnas, no todas las filas tienen las mismas columnas. Esta es una diferencia importante de los sistemas de bases de datos tradicionales, pero esta flexibilidad es importante, en parte debido a la falta de tablas. En un sistema de base de datos relacional, las columnas de anclaje se almacenarían en una tabla y el texto del documento en otra. Debido a que BigTable tiene solo una tabla, toda la información de anclaje debe empaquetarse en un solo registro. Con todos los datos ancla almacenados juntos, solo se necesita una lectura de disco para leer todos los datos del documento. En una base de datos relacional de dos tablas, serían necesarias al menos dos lecturas para recuperar estos datos.

Las filas se dividen en tabletas según sus claves de fila. Por ejemplo, todas las URL que comienzan conA podrían ubicarse en una tableta, mientras que todos los que comienzan con B podría estar en otra tableta. El uso de este tipo de particionamiento basado en rangos hace que sea fácil para un cliente de BigTable determinar qué servidor está sirviendo cada fila. Para buscar una fila en particular, el cliente consulta una lista de rangos de claves de fila para determinar qué tableta contendría la fila deseada. A continuación, el cliente se pone en contacto con el servidor de tabletas apropiado para buscar la fila. Los rangos de claves de fila se almacenan en caché en el cliente, por lo que la mayor parte del tráfico de la red se encuentra entre los clientes y los servidores de tabletas.

La arquitectura de BigTable está diseñada para la velocidad y la escala a través de cantidades masivas de servidores, y para la economía mediante el uso de computadoras económicas que se espera que fallen. Para lograr estos objetivos, BigTable sacrifica algunas características clave de la base de datos relacional, como un lenguaje de consulta complejo y bases de datos de múltiples tablas. Sin embargo, esta arquitectura es adecuada para la tarea de almacenar y encontrar páginas web, donde la tarea principal son búsquedas y actualizaciones eficientes en filas individuales.

3.7 Detección de duplicados

Duplicar y *casi duplicado* Los documentos ocurren en muchas situaciones. Hacer copias y crear nuevas versiones de documentos es una actividad constante en las oficinas, y hacer un seguimiento de estos es una parte importante de la gestión de la información. En la Web, sin embargo, la situación es más extrema. Además de las fuentes normales de duplicación,*plagio* y *correo no deseado* son comunes, y el uso de varias URL para apuntar a la misma página web y *sitios espejo* puede hacer que un rastreador genere un gran número de páginas duplicadas. Los estudios han demostrado que aproximadamente el 30% de las páginas web en un rastreo extenso son exactas o casi duplicadas de páginas en el otro 70% (por ejemplo, Fetterly et al., 2003).

Los documentos con contenido muy similar generalmente brindan poca o ninguna información nueva al usuario, pero consumen importantes recursos durante el rastreo, la indexación y la búsqueda. En respuesta a este problema, se han desarrollado algoritmos para detectar documentos duplicados para que puedan ser eliminados o tratados como un grupo durante la indexación y clasificación.

La detección de duplicados exactos es una tarea relativamente sencilla que se puede realizar utilizando *suma de comprobación* técnicas. Una suma de comprobación es un valor que se calcula en función del contenido del documento. La suma de comprobación más sencilla es la suma de los bytes del archivo del documento. Por ejemplo, la suma de comprobación de un archivo que contiene el texto "peces tropicales" se calcularía de la siguiente manera (en hexadecimal):

T rop	icalfish *Suma*
54 72 6F 70	69 63 61 6C 20 66 69 73 68 508

Cualquier archivo de documento que contenga el mismo texto tendría la misma suma de comprobación. De Por supuesto, cualquier archivo de documento que contenga texto que tenga la misma suma de verificación también se tratará como un duplicado. Un archivo que contenga los mismos caracteres en un orden diferente tendría la misma suma de comprobación, por ejemplo. Funciones más sofisticadas, como un*verificación de redundancia cíclica (*CRC), se han desarrollado que consideran las posiciones de los bytes.

La detección de documentos casi duplicados es más difícil. Incluso definir un casi duplicado es un desafío. Las páginas web, por ejemplo, podrían tener el mismo contenido de texto pero diferir en los anuncios, las fechas o el formato. Otras páginas pueden tener pequeñas diferencias en su contenido con respecto a las revisiones o actualizaciones. En general, un casi duplicado se define utilizando un valor umbral para alguna medida de similitud entre entre pares de documentos. Por ejemplo, un documento D_1 podría definirse como un casi duplicado del documento D_2 si más del 90% de las palabras de los documentos fueran iguales.

Hay dos escenarios para la detección de casi duplicados. Uno es el *buscar* escenario, donde el objetivo es encontrar casi duplicados de un documento dado D. Esto, como todos los problemas de búsqueda, implica conceptualmente la comparación del documento de consulta con todos los demás documentos. Para una colección que contiene *norte* documentos, el número de comparaciones requeridas será $O(N)$. El otro escenario, *descubrimiento,* implica encontrar todos los pares de documentos casi duplicados en la colección. Este proceso requiere $O(n^2)$ comparaciones. Aunque se ha demostrado que las técnicas de recuperación de información que miden la similitud utilizando representaciones de documentos basadas en palabras son efectivas para identificar casi duplicados en el escenario de búsqueda, los requisitos computacionales del escenario de descubrimiento han significado que se han desarrollado nuevas técnicas para derivar datos compactos. representaciones de documentos. Estas representaciones compactas se conocen como *huellas dactilares.*

El proceso básico para generar huellas dactilares es el siguiente:

1. El documento se descompone en palabras. El contenido que no es una palabra, como la puntuación, las etiquetas HTML y los espacios en blanco adicionales, se elimina (consulte la sección 4.3).
2. Las palabras se agrupan en contiguas *n-gramos* para algunos *norte.* Suelen ser secuencias de palabras superpuestas (consulte la sección 4.3.5), aunque algunas técnicas utilizan secuencias que no se superponen.
3. Algunos de los n-gramas se seleccionan para representar el documento.
4. Los n-gramas seleccionados tienen un hash para mejorar la eficiencia de recuperación y reducir aún más el tamaño de la representación.
5. Los valores hash se almacenan, normalmente en un índice invertido.

Hay varios algoritmos de huellas dactilares que utilizan este enfoque general, y difieren principalmente en cómo se seleccionan los subconjuntos de n-gramas. Seleccionar un número fijo de n-gramas al azar no conduce a un buen desempeño en términos de encontrar casi duplicados. Considere dos documentos idénticos D_1 y D_2. La huellas dactilares generadas a partir de n-gramas seleccionados al azar del documento D_1 Es poco probable que tengan una superposición alta con las huellas dactilares generadas a partir de diferentes conjunto de n-gramas seleccionados al azar de D_2. Una técnica más eficaz utiliza combinaciones de caracteres preespecificadas y selecciona n-gramas que comienzan con esos caracteres. Otra técnica popular, llamada $0 \bmod p$, es seleccionar todos los n-gramos cuyo valor hash *modulo pag* es cero, donde *pag* es un parámetro.

La figura 3.14 ilustra el proceso de toma de huellas dactilares usando 3 gramos superpuestos, valores hash hipotéticos y la $0 \bmod p$ método de selección con un *pag* valor de 4. Tenga en cuenta que después del proceso de selección, el documento (o frase en este caso) está representado por huellas dactilares para los n-gramos "pescado incluye pescado", "encontrado en tropical",

Los peces tropicales incluyen peces que se encuentran en ambientes tropicales de todo el mundo, incluidas especies de agua dulce y salada.

(a) Texto original

los peces tropicales incluyen, los peces incluyen peces, incluyen peces encontrados, peces encontrados, encontrados en ambientes tropicales, en ambientes tropicales, ambientes tropicales alrededor, ambientes alrededor del, alrededor del mundo, el mundo incluyendo, mundo incluyendo ambos, incluyendo tanto de agua dulce, ambos de agua dulce y, agua dulce y salada, y agua salada, especies de agua salada

(b) 3 gramos

938 664 463 822 492 798 78 969 143 236 913 908 694 553 870 779

(c) Valores hash

664 492236 908

(d) Valores hash seleccionados usando 0 mod 4

Figura 3.14. Ejemplo de proceso de toma de huellas dactilares

"El mundo incluido" y "incluido el agua dulce". En aplicaciones a gran escala, como la búsqueda de casi duplicados en la Web, los n-gramas suelen tener entre 5 y 10 palabras de longitud y los valores hash son de 64 bits.[3]

Los documentos casi duplicados se encuentran comparando las huellas digitales que los representan. Los pares casi duplicados se definen por el número de huellas dactilares compartidas o la proporción de huellas dactilares compartidas con el número total de huellas dactilares utilizadas para representar el par de documentos. Sin embargo, las huellas dactilares no capturan toda la información del documento y, en consecuencia, esto conduce a errores en la detección de casi duplicados. Las técnicas de selección adecuadas pueden reducir estos errores, pero no eliminarlos. Como mencionamos, las evaluaciones han demostrado que comparar representaciones basadas en palabras usando una medida de similitud como la correlación coseno (ver sección 7.1.2) es generalmente significativamente más efectivo que los métodos de huellas dactilares para encontrar casi duplicados. El problema de estos métodos es su eficacia.

[3] Los valores hash generalmente se generan usando *Toma de huellas dactilares de Rabin*Broder et al., 1997), que lleva el nombre del científico informático israelí Michael Rabin.

Una técnica de toma de huellas dactilares desarrollada recientemente llamadasimhash Charikar, 2002) combina las ventajas de las medidas de similitud basadas en palabras con la eficiencia de las huellas dactilares basadas en hash. Tiene la propiedad inusual para una función hash de que documentos similares tienen valores hash similares. Más precisamente, la similitud de dos páginas medida por la medida de correlación del coseno es proporcional al número de bits que son iguales en las huellas dactilares generadas porsimhash.

El procedimiento para calcular un Simhash la huella dactilar es la siguiente:

1. Procese el documento en un conjunto de características con pesos asociados. Asumiremos el caso simple donde las características son palabras ponderadas por su frecuencia. Otros esquemas de ponderación se analizan en el Capítulo 7.
2. Genere un valor hash con B bits (el tamaño deseado de la huella dactilar) para cada palabra. El valor hash debe ser único para cada palabra.
3. En vector b-dimensional V, actualice los componentes del vector agregando el peso de una palabra a cada componente para el que el bit correspondiente en el valor hash de la palabra sea 1 y restando el peso si el valor es 0.
4. Una vez procesadas todas las palabras, genere un B-huella digital de bits configurando el ith bit a 1 si el ith componente de V es positivo o 0 en caso contrario.

La figura 3.15 muestra un ejemplo de este proceso para una huella dactilar de 8 bits. Tenga en cuenta que las palabras comunes (palabras vacías) se eliminan como parte del procesamiento de texto. En la práctica, valores mucho mayores de$B$ Henzinger (2006) describe una evaluación basada en Web a gran escala donde las huellas dactilares tenían 384 bits. Una página web se define como casi un duplicado de otra página si elSimhash las huellas dactilares coinciden en más de 372 bits. Este estudio mostró ventajas de efectividad significativas para losSimhash
enfoque en comparación con las huellas dactilares basadas en n-gramas.

3.8 Eliminación de ruido

Muchas páginas web contienen texto, enlaces e imágenes que no están directamente relacionados con el contenido principal de la página. Por ejemplo, la Figura 3.16 muestra una página web que contiene una noticia. El contenido principal de la página (la historia) está delineado en negro. Esto *bloque de contenido* ocupa menos del 20% del área de visualización de la página, y el resto se compone de banners, anuncios, imágenes, enlaces de navegación general, servicios (como búsquedas y alertas) e información diversa, como copy- derecho. Desde la perspectiva del motor de búsqueda, este material adicional en la página web es principalmente*ruido* eso podría afectar negativamente la clasificación de la página. Un componente importante de la representación de una página utilizada en un motor de búsqueda se basa

Los peces tropicales incluyen peces que se encuentran en ambientes tropicales de todo el mundo, incluidas especies de agua dulce y salada.

(a) Texto original

tropical 2 peces 2 incluyen 1 encontrado 1 ambientes 1 alrededor de 1 mundo 1 incluyendo 1 ambos 1 agua dulce 1 agua salada 1 agua 1 especie 1

(b) Palabras con pesos

tropical	01100001	pescado	10101011	incluir	11100110
encontró	00011110	ambientes	00101101	alrededor	10001011
mundo	00101010	incluso	11000000	ambas cosas	10101110
agua dulce	00111111	sal	10110101	agua	00100101
especies	11101110				

(c) valores hash de 8 bits

1! 5 9! 9 3 1 3 3

(d) Vector V formado sumando pesos

1 0 1 0 1 1 1 1

(e) Huella digital de 8 bits formada a partir deV

Figura 3.15. Ejemplo de Simhash proceso de toma de huellas dactilares

en el recuento de palabras, y la presencia de una gran cantidad de palabras no relacionadas con el tema principal puede ser un problema. Por esta razón, se han desarrollado técnicas para detectar los bloques de contenido en una página web e ignorar el otro material o reducir su importancia en el proceso de indexación.

Finn y col. (2001) describen una técnica relativamente simple basada en la observación de que hay menos etiquetas HTML en el texto del contenido principal de las páginas web típicas que en el material adicional. Figura 3.17 (también conocida como *curva de pendiente del documento)* muestra la distribución acumulativa de etiquetas en la página web de ejemplo de la Figura 3.16, como una función del número total de tokens (palabras u otras cadenas que no son etiquetas) en la página. El contenido de texto principal de la página corresponde a la "meseta" en el medio de la distribución. Esta área plana es relativamente pequeña debido a la gran cantidad de información de formato y presentación en la fuente HTML de la página.

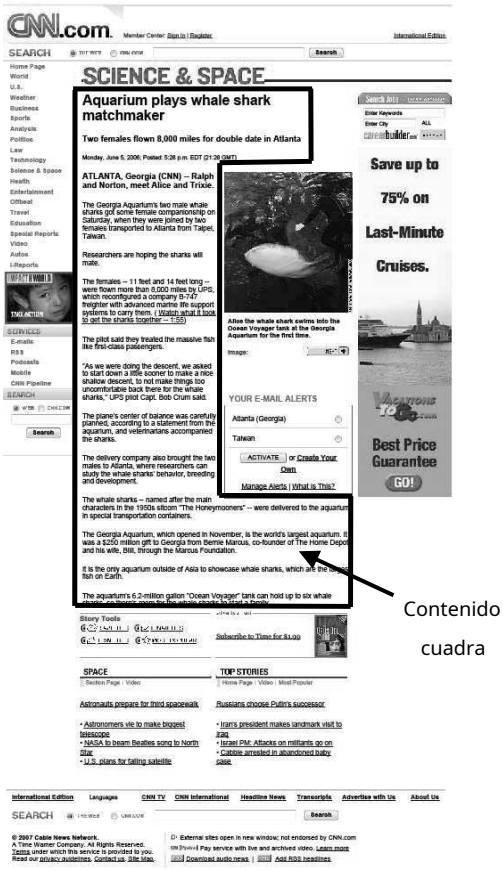


Figura 3.16. Bloque de contenido principal en una página web

Una forma de detectar el área plana más grande de la distribución es representar una red página como una secuencia de bits, donde $B_n = 1$ indica que el *norte*El token es una etiqueta, y $B_n = 0$ de lo contrario. Ciertas etiquetas que se utilizan principalmente para formatear texto, como cambios de fuente, encabezados y etiquetas de tabla, se ignoran (es decir, se representan con un bit 0). La La detección del contenido principal puede verse como un problema de optimización en el que encontramos valores de i y j para maximizar tanto el número de etiquetas a continuación i y por encima j y la cantidad de tokens sin etiqueta entre i y j. Esto corresponde a maximizando el correspondiente función objetiva:

$$\sum_{n=0}^{i-1} B_{norte} + \sum_{n=i}^{j} (1 - B_n) + \sum_{n=j+1}^{norte-1} B_{norte}$$

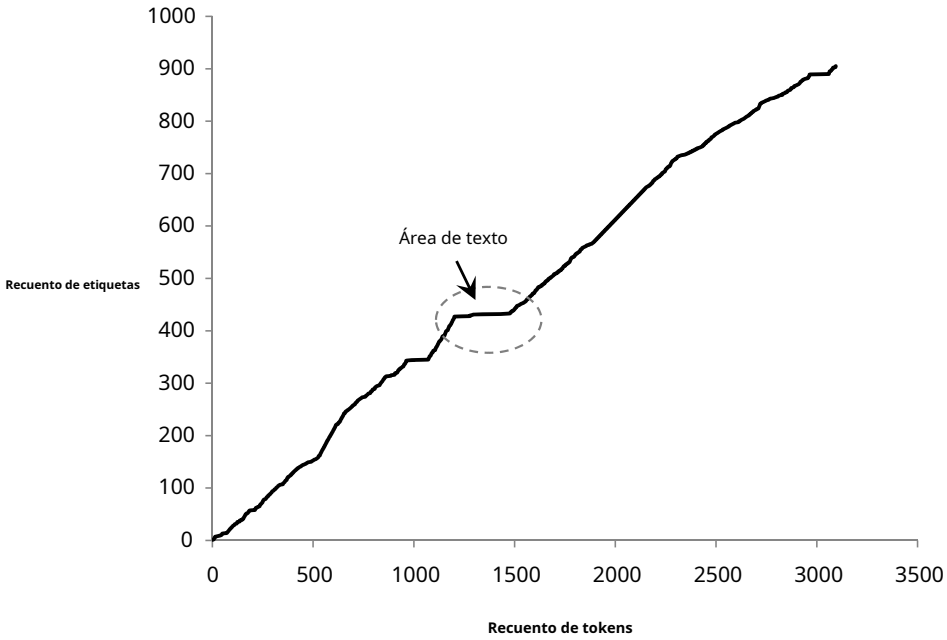


Figura 3.17. Conteo de etiquetas utilizado para identificar bloques de texto en una página web

dónde*norte* es el número de tokens en la página. Esto se puede hacer simplemente escaneando los posibles valores para*I* y *j* y calcular la función objetivo. Tenga en cuenta que este procedimiento solo funcionará cuando la proporción de tokens de texto en la sección sin contenido sea menor que la proporción de etiquetas, lo que no es el caso de la página web de la Figura 3.17. Pinto y col. (2002) modificó este enfoque para usar una ventana de texto para buscar secciones de pendiente baja de la curva de pendiente del documento.

La estructura de la página web también se puede utilizar de forma más directa para identificar los bloques de contenido en la página. Para mostrar una página web usando un navegador, un analizador HTML interpreta la estructura de la página especificada usando las etiquetas y crea una representación del Modelo de objetos de documento (DOM). La estructura en forma de árbol representada por el DOM se puede utilizar para identificar los componentes principales de la página web. La figura 3.18 muestra parte de la estructura DOM[4] para la página web de ejemplo en la Figura 3.16. La parte de la estructura que contiene el texto de la historia está indicada por el comentario.cnnArticleContent. Gupta y col. (2003) describen un enfoque que

[4] Esto se generó utilizando la herramienta DOM Inspector en el navegador Firefox.

navega por el árbol DOM de forma recursiva, utilizando una variedad de técnicas de filtrado para eliminar y modificar nodos en el árbol y dejar solo contenido. Los elementos HTML, como imágenes y secuencias de comandos, se eliminan mediante filtros simples. Los filtros más complejos eliminan anuncios, listas de enlaces y tablas que no tienen contenido "sustancial".

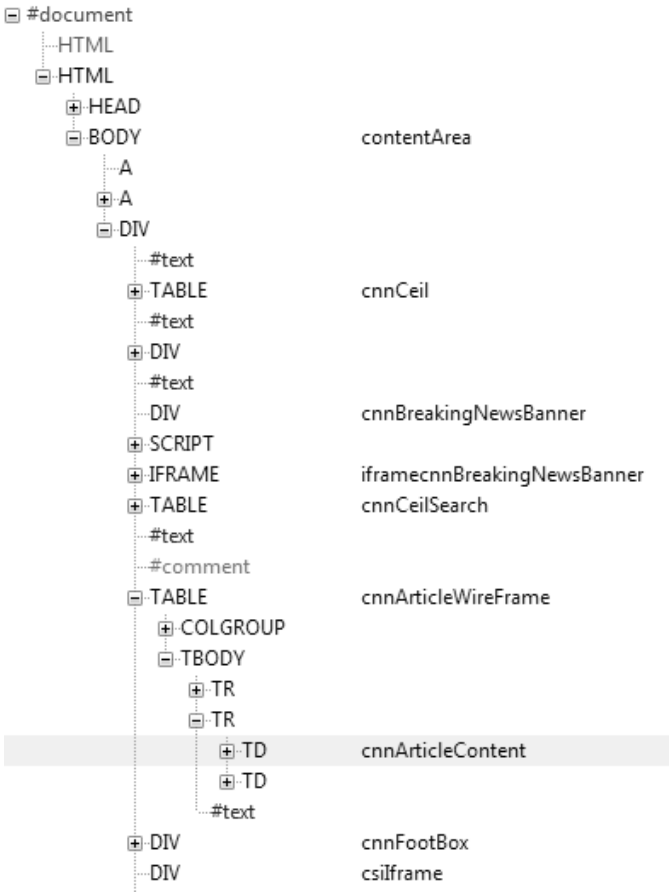


Figura 3.18. Parte de la estructura DOM para la página web de ejemplo

La estructura DOM proporciona información útil sobre los componentes de una página web, pero es compleja y es una mezcla de componentes lógicos y de diseño. En la Figura 3.18, por ejemplo, el contenido del artículo está enterrado en una celda de la tabla (TD etiqueta) en una fila (TR etiqueta) de una tabla HTML (MESA etiqueta). La tabla se utiliza en este caso para especificar el diseño en lugar de datos relacionados semánticamente. Otro enfoque para

La identificación de los bloques de contenido de una página se centra en el diseño y la presentación de la página web. En otras palabras, las características visuales, como la posición del bloque, el tamaño de la fuente utilizada, los colores de fondo y fuente, y la presencia de separadores (como líneas y espacios), se utilizan para definir bloques de información que ser evidente para el usuario en la página web mostrada. Yu y col. (2003) describen un algoritmo que construye una jerarquía de bloques visuales a partir del árbol DOM y características visuales.

El primer algoritmo que discutimos, basado en la distribución de etiquetas, es bastante efectivo para páginas web con un solo bloque de contenido. Los algoritmos que utilizan la estructura DOM y el análisis visual pueden tratar con páginas que pueden tener varios bloques de contenido. En el caso de que haya varios bloques de contenido, el proceso de indexación puede utilizar la importancia relativa de cada bloque para producir una representación más eficaz. Un enfoque para juzgar la importancia de los bloques en una página web es capacitar a un *clasificador* que asignará una categoría de importancia basada en características visuales y de contenido (R. Song et al., 2004).

Referencias y lecturas adicionales

Cho y García-Molina (2002, 2003) escribieron una serie de artículos influyentes sobre el diseño de rastreadores web. Nuestro análisis de las políticas de actualización de páginas se basa en gran medida en Cho y García-Molina (2003), y la sección 3.2.7 se basa en Cho y García-Molina. (2002).

Hay muchos rastreadores web de código abierto. El rastreador Heritrix,[5] desarrollado para el proyecto Internet Archive, es un ejemplo capaz y escalable. El sistema está desarrollado en módulos que son altamente configurables en tiempo de ejecución, lo que lo hace particularmente adecuado para la experimentación.

El rastreo enfocado atrajo mucha atención en los primeros días de la búsqueda web. Menczer y Belew (1998) y Chakrabarti et al. (1999) escribió dos de los artículos más influyentes. Menczer y Belew (1998) visualizan un rastreador enfocado hecho de agentes de software autónomos, principalmente para un solo usuario. El usuario ingresa una lista de URL y palabras clave. Luego, el agente intenta encontrar páginas web que sean útiles para el usuario, y el usuario puede calificar esas páginas para dar retroalimentación al sistema. Chakrabarti et al. (1999) se centran en rastrear índices temáticos especializados. Su rastreador utiliza un clasificador para determinar la actualidad de las páginas rastreadas, así como un destilador, que juzga la calidad de una página como fuente de enlaces a otros temas.

[5] http://crawler.archive.org

páginas. Evalúan su sistema frente a un rastreador tradicional desenfocado para mostrar que un rastreador desenfocado con enlaces de actualidad no es suficiente para lograr un rastreo de actualidad. La amplia estructura de enlaces de la Web hace que el rastreador no enfocado se desvíe rápidamente a otros temas, mientras que el rastreador enfocado se mantiene exitosamente en el tema.

La especificación Unicode es un trabajo increíblemente detallado, que cubre decenas de miles de caracteres (Unicode Consortium, 2006). Debido a la naturaleza de algunas escrituras no occidentales, muchos glifos se forman al agrupar varios caracteres Unicode, por lo que la especificación debe detallar no solo qué son los caracteres, sino cómo se pueden unir. Los caracteres todavía se agregan a Unicode periódicamente.

Bergman (2001) es un estudio extenso de deepWeb. Aunque este estudio es antiguo según los estándares web, muestra cómo el muestreo a través de los motores de búsqueda se puede utilizar para ayudar a estimar la cantidad de contenido no indexado en la Web. Este estudio estimó que existían 550 mil millones de páginas web en la deepWeb, en comparación con mil millones en la web accesible. He et al. (2007) describen una encuesta más reciente que muestra que la Deep Web ha seguido expandiéndose rápidamente en los últimos años. Ipeirotis y Gravano (2004) describen un ejemplo de una técnica para generar representaciones de búsqueda de bases de datos de la Web profunda, llamado sondeo de consultas.

Los mapas del sitio, los archivos robots.txt, las fuentes RSS y las fuentes Atom tienen sus propias especificaciones, que están disponibles en la Web.[6] Estos formatos muestran que los estándares web exitosos suelen ser bastante simples.

Como mencionamos, los sistemas de bases de datos se pueden usar para almacenar documentos de un rastreo web para algunas aplicaciones. Sin embargo, nuestra discusión sobre los sistemas de bases de datos se limitó principalmente a una comparación con BigTable. Hay varios libros de texto, como García-Molina et al. (2008), que proporcionan mucha más información sobre cómo funcionan las bases de datos, incluidos detalles sobre características importantes como lenguajes de consulta, bloqueo y recuperación. BigTable, a la que nos referimos con frecuencia, se describió en Chang et al. (2006). Otras grandes empresas de Internet han creado sus propios sistemas de bases de datos con objetivos similares: distribución a gran escala y alto rendimiento, pero sin un lenguaje de consulta expresivo ni un soporte de transacciones detallado. El sistema Dynamo de Amazon tiene garantías de latencia baja (DeCandia et al., 2007) y Yahoo!

[6] http://www.sitemaps.org http://www.robotstxt.org http://www.rssboard.org/rss-specification http://www.rfc-editor.org/rfc/rfc5023.txt

Mencionamos DEFLATE (Deutsch, 1996) y LZW (Welch, 1984) como algoritmos específicos de compresión de documentos en el texto. DEFLATE es la base de las populares herramientas de compresión Zip, gzip y zlib. LZW es la base de Unixcomprimir comando, y también se encuentra en formatos de archivo como GIF, PostScript y PDF. El texto de Witten et al. (1999) proporciona discusiones detalladas sobre algoritmos de compresión de texto e imágenes.

Hoad y Zobel (2003) proporcionan tanto una revisión de las técnicas de toma de huellas dactilares como una comparación con las medidas de similitud basadas en palabras para la detección de casi duplicados. Su evaluación se centró en encontrar versiones de documentos y documentos plagiados. Bernstein y Zobel (2006) describen una técnica para utilizar huellas dactilares completas (sin selección) para la tarea de encontrar*co-derivados,* que son documentos derivados de la misma fuente. Bernstein y Zobel (2005) examinaron el impacto de la duplicación en las evaluaciones de la efectividad de la recuperación. Demostraron que alrededor del 15% de los documentos relevantes para una de las pistas TREC eran redundantes, lo que podría afectar significativamente el impacto de los resultados desde la perspectiva del usuario.

Henzinger (2006) describe una evaluación a gran escala de la detección de casi duplicados en la Web. Las dos técnicas comparadas fueron una versión del algoritmo de "brillo" de Broder (Broder et al., 1997; Fetterly et al., 2003) ysimhashCharikar, 2002) .El estudio de Henzinger, que usó 1.600 millones de páginas, mostró que ninguno de los métodos funcionaba bien para detectar documentos redundantes. *en el mismo sitio* debido al uso frecuente de texto "repetitivo" que hace que diferentes páginas se vean similares. Para páginas en diferentes sitios, elSimhash El algoritmo alcanzó una precisión del 50% (es decir, de aquellas páginas que se declararon "casi duplicadas" según el umbral de similitud, el 50% fueron correctas), mientras que el algoritmo de Broder produjo una precisión del 38%.

Se han escrito varios artículos sobre técnicas para extraer contenido de páginas web. Yu y col. (2003) y Gupta et al. (2003) son buenas fuentes de referencias a estos artículos.

Ejercicios

3.1. Suponga que tiene dos colecciones de documentos. La colección más pequeña está llena de información útil, precisa y de alta calidad. La colección más grande contiene algunos documentos de alta calidad, pero también contiene texto de menor calidad que es antiguo, desactualizado o mal escrito. ¿Cuáles son algunas de las razones para crear un motor de búsqueda solo para una colección pequeña? ¿Cuáles son algunas de las razones para crear un motor de búsqueda que cubra ambas colecciones?

3.2. Suponga que tiene una conexión de red que puede transferir 10 MB por segundo. Si cada página web es de 10K y requiere 500 milisegundos para transferirse, ¿cuántos subprocesos necesita su rastreador web para utilizar completamente la conexión de red? Si su rastreador necesita esperar 10 segundos entre solicitudes al mismo servidor web, ¿cuál es el número mínimo de servidores web distintos con los que el sistema necesita contactar cada minuto para mantener la conexión de red completamente utilizada?

3.3. ¿Cuál es la ventaja de usar HEADrequests en lugar de GETrequests durante el rastreo? ¿Cuándo utilizaría un rastreador una solicitud GET en lugar de una solicitud HEAD?

3.4. ¿Por qué los rastreadores no utilizan solicitudes POST?

3.5. Nombra los tres tipos de sitios mencionados en el capítulo que componen el deepWeb.

3.6. ¿Cómo diseñaría un sistema para ingresar datos automáticamente en formularios web con el fin de rastrear páginas web profundas? ¿Qué medidas usaría para asegurarse de que las acciones de su rastreador no fueran destructivas (por ejemplo, para que no agregue comentarios de blog aleatorios).

3.7. Escribe un programa que pueda crear un mapa del sitio válido basado en el contenido de un directorio en el disco duro de tu computadora. Suponga que se puede acceder a los archivos desde un sitio web en la URLhttp://www.example.com. Por ejemplo, si hay un archivo en su directorio llamado tarea.pdf, esto estaría disponible en http://www.example.com/homework.pdf. Utilice la fecha de modificación real en el archivo como la última hora modificada en el mapa del sitio y para ayudar a estimar la frecuencia de cambio.

3.8. Suponga que, en un esfuerzo por rastrear páginas web más rápido, configura dos máquinas de rastreo con diferentes URL iniciales de inicio. ¿Es esta una estrategia eficaz para el rastreo distribuido? ¿Por qué o por qué no?

3.9. Escriba un rastreador web simple de un solo subproceso. A partir de una única inputURL (quizás la página web de un profesor), el rastreador debe descargar una página y luego esperar al menos cinco segundos antes de descargar la página siguiente. Su programa debería encontrar otras páginas para rastrear analizando las etiquetas de enlace que se encuentran en documentos rastreados previamente.

3.10. UTF-16 se utiliza en Java y Windows®. Compárelo con UTF-8.

3.11. ¿Cómo maneja BigTable las fallas de hardware?

3.12. Diseñe un algoritmo de compresión que comprima etiquetas HTML. Su algoritmo debería detectar etiquetas en un archivo HTML y reemplazarlas con un código de su propio diseño que sea más pequeño que la etiqueta misma. Escriba un programa de codificador y descodificador.

3.13. Genere sumas de verificación para un documento agregando los bytes del documento y usando el comando Unix cksum. Edite el documento y vea si cambian ambas sumas de verificación. ¿Puede cambiar el documento para que la suma de comprobación simple no cambie?

3.14. Escribe un programa para generar Simhash huellas dactilares para documentos. Puede utilizar cualquier función hash razonable para las palabras. Utilice el programa para detectar duplicados en la computadora de su hogar. Informe sobre la precisión de la detección. ¿Cómo varía la precisión de detección con el tamaño de la huella digital?

3.15. Trace las curvas de pendiente del documento para una muestra de páginas web. La muestra debe incluir al menos una página que contenga un artículo de noticias. Pruebe la precisión del algoritmo de optimización simple para detectar el bloque de contenido principal. Escribir su propio programa o use el código de http://www.aidanf.net/software/bte-body-extracción de texto. Describe los casos en los que falla el algoritmo. ¿Sería exitoso en estos casos un algoritmo que buscara explícitamente áreas de pendiente baja de la curva de pendiente del documento?

3.16. Brinde un esquema de alto nivel de un algoritmo que usaría la estructura DOM para identificar la información del contenido en una página web. En particular, describa la heurística que usaría para identificar el contenido y los elementos que no lo son de la estructura.

4

Procesando texto

"Estaba tratando de comprender el significado de las palabras".

Spock, *Star Trek: La última frontera*

4.1 De las palabras a los términos

Después de recopilar el texto que queremos buscar, el siguiente paso es decidir si debe modificarse o reestructurarse de alguna manera para simplificar la búsqueda. Los tipos de cambios que se realizan en esta etapa se denominan *transformación de texto* o, más a menudo, *procesamiento de texto.* El objetivo del procesamiento de texto es convertir las muchas formas en las que las palabras pueden aparecer en formas más consistentes. *Términos del Índice.* Los términos de índice son la representación del contenido de un documento que se utiliza para realizar búsquedas.

La decisión más simple sobre el procesamiento de texto sería no hacerlo en absoluto. Un buen ejemplo de esto es la función "buscar" en su procesador de texto favorito. En el momento en que utiliza el comando de búsqueda, el texto que desea buscar ya se ha recopilado: está en la pantalla. Después de escribir la palabra que desea buscar, el procesador de textos escanea el documento e intenta encontrar la secuencia exacta de letras que acaba de escribir. Esta función es extremadamente útil y casi todos los programas de edición de texto pueden hacerlo porque los usuarios lo exigen.

El problema es que la búsqueda de texto exacto es bastante restrictiva. La restricción más molesta es la distinción entre mayúsculas y minúsculas: suponga que desea encontrar "hardware de computadora" y hay una oración en el documento que comienza con "Hardware de computadora". Su consulta de búsqueda no coincide exactamente con el texto de la oración, porque la primera letra de la oración está en mayúscula. Afortunadamente, la mayoría de los procesadores de texto tienen la opción de ignorar las mayúsculas y minúsculas durante las búsquedas. Puede pensar en esto como una forma muy rudimentaria de procesamiento de texto en línea. Como la mayoría de las técnicas de procesamiento de texto, ignorar mayúsculas y minúsculas aumenta la probabilidad de que encuentre una coincidencia para su consulta en el documento.

Muchos motores de búsqueda no distinguen entre letras mayúsculas y minúsculas. Sin embargo, van mucho más lejos. Como veremos en este capítulo, los motores de búsqueda

puede quitar la puntuación de las palabras para que sean más fáciles de encontrar. Las palabras se dividen en un proceso llamado *tokenización.* Algunas palabras pueden ignorarse por completo para que el procesamiento de consultas sea más efectivo y eficiente; se llama *parada.*
El sistema puede utilizar *derivando* para permitir que palabras similares (como "correr" y "correr") coincidan entre sí. Algunos documentos, como las páginas web, pueden tener cambios de formato (como texto grande o en negrita) o explícitos. *estructura (*como títulos, capítulos y subtítulos) que también puede utilizar el sistema. Las páginas web también contienen *Enlaces* a otras páginas web, que se pueden utilizar para mejorar la clasificación de los documentos. Todas estas técnicas se analizan en este capítulo.

Estas técnicas de procesamiento de texto son bastante simples, aunque sus efectos en los resultados de búsqueda pueden ser profundos. Ninguna de estas técnicas implica que la computadora realice ningún tipo de razonamiento complejo o comprensión del texto. Los motores de búsqueda funcionan porque gran parte del significado del texto se captura mediante el recuento de apariciones y coincidencias de palabras,[1] especialmente cuando esos datos se recopilan de las enormes colecciones de texto disponibles en la Web. Comprender la naturaleza estadística del texto es fundamental para comprender los modelos de recuperación y los algoritmos de clasificación, por lo que comenzamos este capítulo con una discusión de *estadísticas de texto.* Técnicas más sofisticadas para *procesamiento natural del lenguaje* que implican análisis sintáctico y semántico de texto se han estudiado durante décadas, incluida su aplicación a la recuperación de información, pero hasta la fecha han tenido poco impacto en los algoritmos de clasificación para motores de búsqueda. Sin embargo, estas técnicas se están utilizando para la tarea de responder a preguntas, que se describe en el Capítulo 11. Además, se están utilizando técnicas que implican un procesamiento de texto más complejo para identificar términos de índice o características adicionales para la búsqueda. *Extracción de información* Aquí se analizan las técnicas para identificar los nombres de las personas, los nombres de las organizaciones, las direcciones y muchos otros tipos especiales de características, y *clasificación,* que se puede utilizar para identificar categorías semánticas, se analiza en el Capítulo 9.

Por último, aunque este libro se centra en la recuperación de documentos en inglés, las técnicas de recuperación de información se pueden utilizar con texto en muchos idiomas diferentes. En este capítulo, mostramos cómo diferentes lenguajes requieren diferentes tipos de representación y procesamiento de texto.

---

1 La co-ocurrencia de palabras mide la cantidad de veces que los grupos de palabras (generalmente pares) ocurren juntos en los documentos. A *colocación* es el nombre que se le da a un par, grupo o secuencia de palabras que aparecen juntas con más frecuencia de lo que cabría esperar por casualidad. La *medidas de asociación de términos* que se utilizan para buscar colocaciones se analizan en el Capítulo 6.

4.2 Estadísticas de texto

Aunque el lenguaje es increíblemente rico y variado, también es muy predecible. Hay muchas formas de describir un tema o evento en particular, pero si se cuentan las palabras que aparecen en muchas descripciones de un evento, algunas palabras aparecerán con mucha más frecuencia que otras. Algunas de estas palabras frecuentes, como "y" o "el", serán comunes en la descripción de cualquier evento, pero otras serán características de ese evento en particular. Esto fue observado ya en 1958 por Luhn, cuando propuso que el significado de una palabra dependía de su frecuencia en el documento. Los modelos estadísticos de apariciones de palabras son muy importantes en la recuperación de información y se utilizan en muchos de los componentes centrales de los motores de búsqueda, como los algoritmos de clasificación, la transformación de consultas y las técnicas de indexación. Estos modelos se discutirán en capítulos posteriores,

Una de las características más obvias del texto desde un punto de vista estadístico es que la distribución de frecuencias de palabras es muy *sesgado.* Hay algunas palabras que tienen frecuencias muy altas y muchas palabras que tienen frecuencias bajas. De hecho, las dos palabras más frecuentes en inglés ("el" y "de") representan aproximadamente el 10% de todas las apariciones de palabras. Las seis palabras más frecuentes representan el 20% de las ocurrencias, y las 50 palabras más frecuentes son aproximadamente el 40% de todo el texto. Por otro lado, dada una muestra grande de texto, típicamente alrededor de la mitad de todas las palabras únicas en esa muestra ocurren solo una vez. Esta distribución está descrita por *La ley de Zipf*[2] que establece que la frecuencia de la rLa palabra más común es inversamente proporcional a r o, alternativamente, el *rango* de una palabra por su frecuencia (F) es aproximadamente una constante (k):

$$r \cdot f = k$$

A menudo queremos hablar sobre la probabilidad de aparición de una palabra, que es solo la frecuencia de la palabra dividida por el número total de apariciones de palabras en el texto. En este caso, la ley de Zipf es:

$$r \cdot PAG_r = C$$

dónde PAG_r es la probabilidad de ocurrencia para el $r$la palabra clasificada, y C es una constante. Para inglés, $C \approx 0{,}1$. La figura 4.1 muestra la gráfica de la ley de Zipf con este constante. Esto muestra claramente cómo la frecuencia de aparición de palabras cae rápidamente después de las primeras palabras más comunes.

[2] El nombre del lingüista estadounidense George Kingsley Zipf.

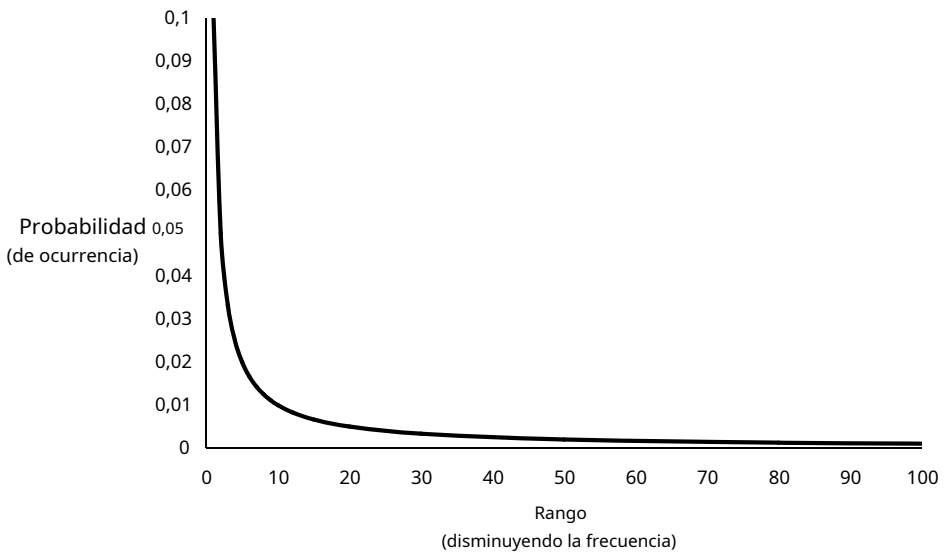


Figura 4.1. Rango versus probabilidad de ocurrencia de palabras asumiendo la ley (rango × probabilidad de Zipf = 0.1)

Para ver qué tan bien predice la ley de Zipf las ocurrencias de palabras en colección de texto Como ejemplo, usaremos la colección AssociatedPress de noticias de 1989 (llamada AP89). Esta colección se utilizó en las evaluaciones de TREC durante varios años. La Tabla 4.1 muestra algunas estadísticas para las ocurrencias de palabras en AP89. El tamaño del vocabulario es el número de palabras únicas en la colección. Incluso en esta colección relativamente pequeña, el tamaño del vocabulario es bastante grande (casi 200.000 palabras únicas). Una gran proporción de estas palabras (70.000) aparecen solo una vez. Las palabras que aparecen una vez en un corpus de texto o en un libro se han considerado durante mucho tiempo importantes en el análisis de textos y se les ha dado el nombre especial de *Hapax Legomena.*[3]

La Tabla 4.2 muestra las 50 palabras más frecuentes de la colección AP89, junto con sus frecuencias, rangos, probabilidad de ocurrencia (convertidas a porcentaje del total de ocurrencias), y el rP_r valor. Desde esta tabla, podemos ver

[3] El nombre fue creado por eruditos que estudiaban la Biblia. Desde el siglo XIII, la gente ha estudiado las apariciones de palabras en la Biblia y, de particular interés, ha creado *concordancias,* que son índices de dónde aparecen las palabras en el texto. Las concordancias son los antepasados de los archivos invertidos que se utilizan en los motores de búsqueda modernos. Se dice que la primera concordancia requirió 500 monjes para crearla.

Total de documentos	84,678
Total de ocurrencias de palabras	39,749,179
Tamaño del vocabulario	198,763
Palabras que ocurren> 1000 veces	4.169
Palabras que aparecen una vez	70,064

Cuadro 4.1. Estadísticas de la colección AP89

que la ley de Zipf es bastante precisa, ya que el valor de rP_r es aproximadamente estafa- stant, y cerca de 0.1. Las mayores variaciones corresponden a algunas de las palabras más frecuentes. De hecho, generalmente se observa que la ley de Zipf es inexacta para rangos bajos y altos (palabras de alta y baja frecuencia). La Tabla 4.3 da algunos ejemplos de palabras de menor frecuencia de AP89.

La figura 4.2 muestra una gráfica logarítmica[4] de El rP_r valores para todas las palabras en el AP89 colección. La ley de Zipf se muestra como una línea recta en este gráfico, ya queIniciar sesiónPAG_r = Iniciar sesión($C \cdot r^{-1}$) = Iniciar sesión C - Iniciar sesión r. Esta figura muestra claramente cómo la el lacionalismo se rompe en los rangos altos (aproximadamente el rango 10,000 y superior). Se han propuesto varias modificaciones a la ley de Zipf,[5] algunos de los cuales tienen conexiones interesantes con modelos cognitivos del lenguaje.

Es posible derivar una fórmula simple para predecir la proporción de palabras con una frecuencia dada a partir de la ley de Zipf. Una palabra que ocurre$norte$ veces tiene rango $r_n = k / n$. En general, más de una palabra puede tener la misma frecuencia. Nosotros asumir que el rango r_{norte} está asociado con el *último* del grupo de palabras con la misma frecuencia. En ese caso, el número de palabras con la misma frecuencia$norte$voluntad ser dado por $r_{norte} - r_{n+1}$, cuál es el rango de la última palabra en el grupo menos el rango de la última palabra del grupo anterior de palabras con mayor frecuencia (recuerde que las palabras de mayor frecuencia tienen rangos más bajos). Por ejemplo, Table 4.4 tiene un ejemplo de una clasificación de palabras en orden decreciente de frecuencia. El número de palabras con frecuencia 5,099 es el rango del último miembro de ese

[4] La X y y Los ejes de una gráfica logarítmica muestran el logaritmo de los valores de X y y, no los valores en sí mismos.

[5] La más conocida es la derivación del matemático Benoit Mandelbrot (el misma persona que desarrolló la geometría fractal), que es $(r + \beta)^{\alpha} \cdot PAG_r = y$, dónde β, α, y y son parámetros que se pueden ajustar para un texto en particular. En el caso del AP89 recopilación, sin embargo, el ajuste para los datos de frecuencia no es notablemente mejor que la distribución Zipf.

Palabra	*Frec.*	*r*	*PAGr (%)*	*r.PAGr*	*Palabra*	*Frec*	*r*	*PAGr (%)*	*r.PAGr*
la	2,420,778	1	6,49	0.065	tiene	136,007	26	0,37	0,095
de	1.045.733	2	2,80	0.056	son	130,322	27	0,35	0,094
a	968,882	3	2,60	0.078	no	127,493	28	0,34	0,096
a	892,429	4	2,39	0.096	quien	116,364	29	0,31	0.090
y	865,644	5	2.32	0.120	ellos	111,024	30	0,30	0,089
en	847,825	6	2.27	0.140	su	111,021	31	0,30	0.092
dicho	504,593	7	1,35	0.095	tenía	103,943	32	0,28	0,089
por	363,865	8	0,98	0.078	voluntad	102,949	33	0,28	0.091
que	347,072	9	0,93	0.084	haría	99.503	34	0,27	0.091
estaba	293.027	10	0,79	0.079	sobre	92,983	35	0,25	0.087
en	291,947	11	0,78	0,086	i	92,005	36	0,25	0,089
él	250,919	12	0,67	0.081	sido	88,786	37	0,24	0,088
es	245,843	13	0,65	0.086	esto	87,286	38	0,23	0,089
con	223,846	14	0,60	0.084	su	84,638	39	0,23	0,089
a	210,064	15	0,56	0,085	nuevo	83,449	40	0,22	0.090
por	209.586	16	0,56	0.090	o	81,796	41	0,22	0.090
eso	195,621	17	0,52	0.089	que	80,385	42	0,22	0.091
de	189.451	18	0,51	0.091	nosotros	80,245	43	0,22	0.093
como	181,714	19	0,49	0.093	más	76,388	44	0,21	0.090
ser	157,300	20	0,42	0.084	después	75,165	45	0,20	0.091
fueron	153,913	21	0,41	0.087	nosotros	72,045	46	0,19	0,089
un	152,576	22	0,41	0.090	por ciento	71,956	47	0,19	0.091
tengo	149,749	23	0,40	0.092	arriba	71.082	48	0,19	0.092
su	142,285	24	0,38	0.092	uno	70.266	49	0,19	0.092
pero	140,880	25	0,38	0.094	personas	68,988	50	0,19	0.093

Cuadro 4.2. 50 palabras más frecuentes de AP89

Palabra	*Frec.*	*r*	*PAGr (%)*	*r.PAGr*
asistente	5.095	1.021	. 013	0,13
alcantarillas	100	17,110	. 000256	0,04
cepillo de dientes	10	51,555	. 000025	0,01
hazmat	1	166,945	. 000002	0,04

Cuadro 4.3. Palabras de baja frecuencia de AP89

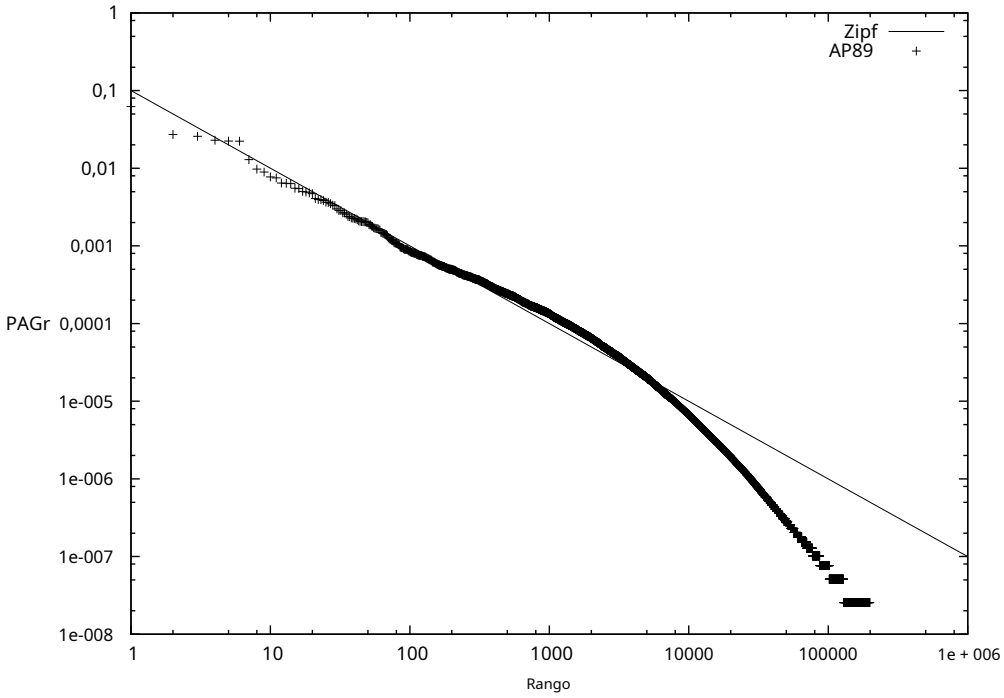


Figura 4.2. Una gráfica logarítmica de la ley de Zipf comparada con los datos reales de AP89. La relación predicha entre la probabilidad de ocurrencia y el rango desciende considerablemente en los rangos altos.

grupo ("químico") menos el rango del último miembro del grupo anterior con mayor frecuencia ("cumbre"), que es 1006 - 1002 = 4.

Rango	*Palabra*	*Frecuencia*
1000	preocupación	5.100
1001	habló	5.100
1002	cumbre	5.100
1003	traer	5.099
1004	estrella	5.099
1005	inmediato	5.099
1006	químico	5.099
1007	africano	5,098

Cuadro 4.4. Ejemplo de clasificación de frecuencia de palabras

Dado que el número de palabras con frecuencia *norte* es $r_{norte} - r_{n+1} = k / n - k / (n + 1) = k / n (n + 1)$, luego la proporción de palabras con esta frecuencia se puede encontrar dividiendo este número por el número total de palabras, que será el rango de la última palabra con frecuencia 1. El rango de la última palabra en el vocabulario es $k / 1 = k$. La proporción de palabras con frecuencia. *norte,* por lo tanto, está dado por $1 / n (n + 1)$. Esta fórmula predice, por ejemplo, que 1/2 de las palabras del vocabulario aparecerán una vez. La tabla 4.5 compara las predicciones de esta fórmula con datos reales de una colección TREC diferente.

Número de Ocurrencias *(norte)*	*Predicho Proporción (1 / n (n + 1))*	*Real Proporción*	*Real Número de Palabras*
1	0.500	0.402	204,357
2	0,167	0,132	67,082
3	0.083	0,069	35.083
4	0,050	0,046	23.271
5	0.033	0.032	16,332
6	0,024	0,024	12,421
7	0,018	0,019	9,766
8	0,014	0,016	8.200
9	0,011	0,014	6,907
10	0,009	0,012	5.893

Cuadro 4.5. Proporciones de palabras que ocurren *norte* veces en 336,310 documentos del TRECVolumen 3 corpus. El tamaño total del vocabulario (número de palabras únicas) es 508,209.

4.2.1 Crecimiento de vocabulario

Otra predicción útil relacionada con la ocurrencia es *crecimiento de vocabulario.* A medida que aumenta el tamaño del corpus, aparecen nuevas palabras. Basado en el supuesto de una distribución Zipf para palabras, esperaríamos que el número de palabras nuevas que ocurren en una cantidad dada de texto nuevo disminuya a medida que aumenta el tamaño del corpus. Sin embargo, siempre aparecerán nuevas palabras debido a fuentes como palabras inventadas (piense en todos esos nombres de medicamentos y nombres de empresas emergentes), errores ortográficos, números de productos, nombres de personas, direcciones de correo electrónico y muchos otros. Heaps (1978) encontró empíricamente que la relación entre el tamaño del corpus y el tamaño del vocabulario era:

$$v = k \cdot norte^{\beta}$$

dónde v es el tamaño del vocabulario para un corpus de tamaño *norte* palabras, y k y β son parámetros que varían para cada colección. Esto a veces se denomina *Ley de montones.*
Valores típicos para k y β a menudo se dice que son $10 \le k \le 100$ y $\beta \approx 0{,}5$.
La ley de Heaps predice que el número de palabras nuevas aumentará muy rápidamente cuando el corpus es pequeño y seguirá aumentando indefinidamente, pero a un ritmo más lento para los corpus más grandes. La figura 4.3 muestra una gráfica de crecimiento de vocabulario para la colección AP89 en comparación con una gráfica de la ley de Heaps con $k = 62{,}95$ y $\beta = 0{,}455$. Claramente, la ley de Heaps encaja bien. Los valores de los parámetros son similares para muchas de las otras colecciones de noticias de TREC. Como ejemplo de la precisión de esta predicción, si se escanean las primeras 10,879,522 palabras de la colección AP89, la ley de Heaps predice que el número de palabras únicas será 100,151, mientras que el número real
el valor es 100.024. Predicciones son mucho menos precisos para pequeñas cantidades de palabras (<1000).

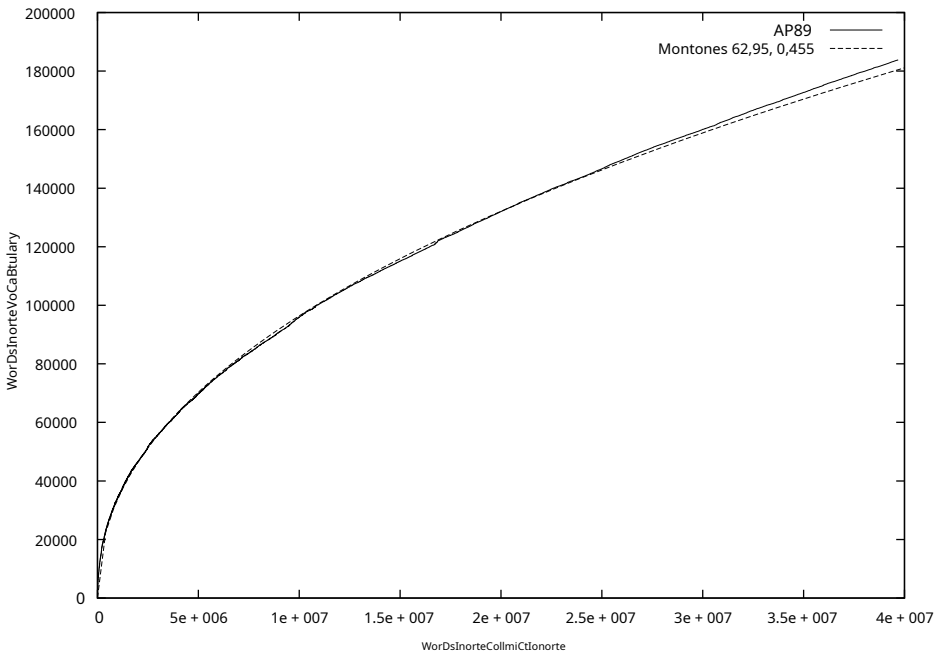


Figura 4.3. Crecimiento de vocabulario para la colección TRECAP89 en comparación con la ley de Heaps

Las colecciones a escala web son considerablemente más grandes que la colección AP89. La colección AP89 contiene alrededor de 40 millones de palabras, pero la (relativamente pequeña) colección TREC Web GOV2[6] contiene más de 20 *mil millones* palabras. Con tantas palabras, parece probable que la cantidad de palabras nuevas eventualmente caiga casi a cero y la ley de Hell no sea aplicable. Resulta que este no es el caso. La figura 4.4 muestra una gráfica de crecimiento de vocabulario para GOV2 junto con una gráfica de la ley de Heaps con $k = 7.34$ y $\beta = 0,648$. Estos datos indican que la cantidad de palabras únicas continúa creciendo de manera constante incluso después de alcanzar los 30 millones. Esto tiene implicaciones significativas para el diseño de motores de búsqueda, que se analizarán en el Capítulo 5. La ley de Heaps proporciona un buen ajuste para estos datos, aunque los valores de los parámetros son muy diferentes a los de otras colecciones de TREC y fuera de límites establecidos como típicos con estas y otras colecciones más pequeñas.

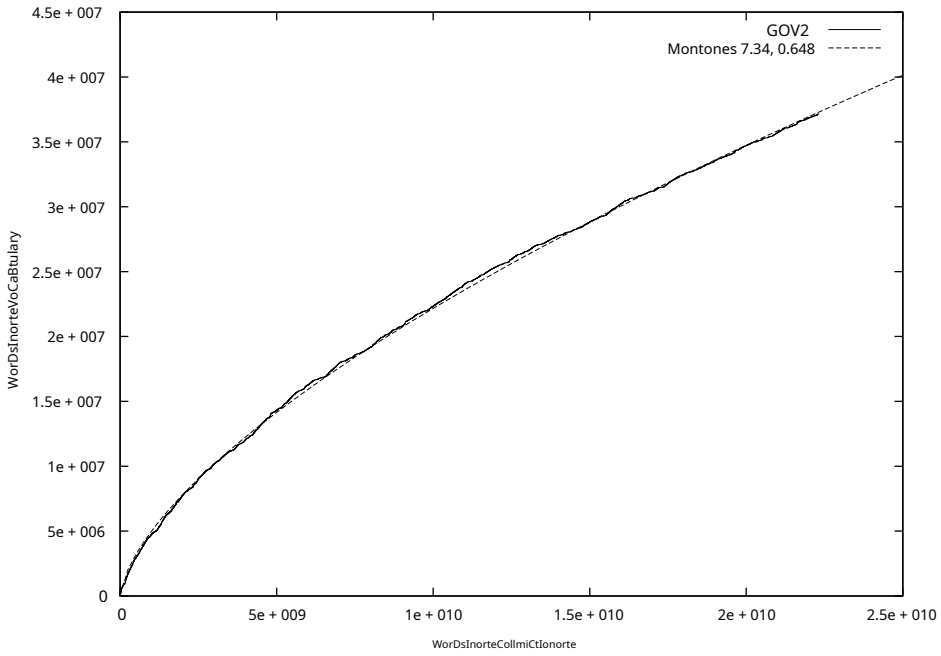


Figura 4.4. Crecimiento de vocabulario para la colección TRECGOV2 en comparación con la ley de Heaps

[6] Las páginas web se rastrearon desde sitios web en formato.gov dominio durante principios de 2004. Consulte la sección 8.2 para obtener más detalles.

4.2.2 Estimación del tamaño de la colección y el conjunto de resultados

Las estadísticas de ocurrencia mundial también pueden usarse para estimar el tamaño de los resultados de una búsqueda en la web. Todos los motores de búsqueda web tienen alguna versión de la interfaz de consulta que se muestra en la Figura 4.5, donde inmediatamente después de la consulta ("acuario de peces tropicales" en este caso) y antes de la lista clasificada de resultados, se proporciona una estimación del número total de resultados. Por lo general, se trata de un número muy grande, y las descripciones de estos sistemas siempre señalan que es solo una estimación. Sin embargo, siempre está incluido.

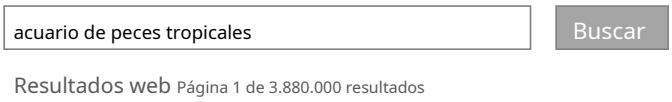


Figura 4.5. Estimación del tamaño del resultado para la búsqueda web

Para estimar el tamaño de un conjunto de resultados, primero debemos definir "resultados". A los efectos de esta estimación, un resultado es cualquier documento (o página web) que contiene *todas* de las palabras de consulta. Algunas aplicaciones de búsqueda clasificarán los documentos que no contienen todas las palabras de la consulta, pero dado el enorme tamaño de la Web, esto generalmente no es necesario. Si asumimos que las palabras ocurren *independientemente* entre sí, la probabilidad de que un documento contenga todas las palabras de la consulta es simplemente el producto de las probabilidades de que las palabras individuales aparezcan en un documento. Por ejemplo, si hay tres palabras de consulta *a, b,* y *C,* luego:

$$Pensilvania \cap B \cap c) = P(a) \cdot P(b) \cdot Ordenador\ personal)$$

dónde $Pensilvania \cap B \cap C)$ es la probabilidad conjunta, o la probabilidad de que las tres palabras aparezcan en un documento, y $P(a), P(b),$ y $Ordenador\ personal)$ son las probabilidades de que cada palabra aparezca en un documento. Un motor de búsqueda siempre tendrá acceso al número de documenta que una palabra aparece en ($F_a, F_B,$ y $F_{C),7}$ y el número de documentos de la colección ($NORTE),$ por lo que estas probabilidades se pueden estimar fácilmente como $P(a) = F_a/N$, $P(b) = f_{B/}NORTE,$ y $P(c) = f_{C/}NORTE.$ Esto nos da

$$F_{abc} = norte \cdot F_a/norte \cdot F_{B/}norte \cdot F_{C/}N = (f_a \cdot F_B \cdot F_{C)/}norte_2$$

dónde $F_{a\ B\ C}$ es el tamaño estimado del conjunto de resultados.

7 Tenga en cuenta que estos son *documentar las frecuencias de ocurrencia,* no el número total de apariciones de palabras (puede haber muchas apariciones de una palabra en un documento).

Palabras)	*Documento Frecuencia*	*Estimado Frecuencia*
tropical	120,990	
pescado	1,131,855	
acuario	26,480	
cría	81,885	
pez tropical	18,472	5.433
acuario tropical	1,921	127
cría tropical	5.510	393
acuario de peces	9,722	1,189
cría de peces	36,427	3.677
cría de acuario	1.848	86
acuario de peces tropicales	1,529	6
cría de peces tropicales	3.629	18

Cuadro 4.6. Documente las frecuencias y las frecuencias estimadas para las combinaciones de palabras (asumiendo independencia) en la colección GOV2Web. Tamaño de la colección (*NORTE)* es 25,205,179.

La Tabla 4.6 proporciona frecuencias de ocurrencia de documentos para las palabras “tropical”, “pez”, “acuario” y “cría”, y para combinaciones de esas palabras en la colección TRECGOV2Web. También proporciona el tamaño estimado de las frecuencias de las combinaciones basándose en el supuesto de independencia. Claramente, esta suposición no conduce a buenas estimaciones para el tamaño del resultado, especialmente para combinaciones de tres palabras. El problema es que las palabras en estas combinaciones no ocurren independientemente unas de otras. Si vemos la palabra "pez" en un documento, por ejemplo, es más probable que la palabra "acuario" aparezca en este documento que en uno que no contenga "peces".

Es posible obtener mejores estimaciones si el motor de búsqueda también dispone de información sobre la co-ocurrencia de palabras. Obviamente, esto daría respuestas exactas para consultas de dos palabras. Para consultas más largas, podemos mejorar la estimación al no asumir independencia. En general, por tres palabras

$$Pensilvania \cap B \cap c) = P (a \cap B) \cdot P (c \mid (a \cap B))$$

dónde $Pensilvania \cap B)$ es la probabilidad de que las palabras a y B coexisten en un documento, y $P (c \mid (a \cap B))$ es la probabilidad de que la palabra C ocurre en un documento *dado* que las palabras a y B ocurren en el documento.[8] Si tenemos información de co-ocurrencia,

[8] A esto se le llama *condicional* probabilidad.

podemos aproximar esta probabilidad usando $P(c \mid a)$ o $P(c \mid b)$, cualquiera que sea el más grande. Para la consulta de ejemplo "acuario de peces tropicales" en la Tabla 4.6, esto significa que estimamos el tamaño del conjunto de resultados multiplicando el número de documentos que contienen tanto "tropical" como "acuario" por la probabilidad de que un documento contenga "peces" dado que contiene "acuario" , o:

$$F_{tropical \cap pescado \cap acuario} = F_{tropical \cap acuario} \cdot F_{pescado \cap acuario} / F_{acuario}$$
$$= 1921 \cdot 9722/26480 = 705$$

Del mismo modo, para la consulta "cría de peces tropicales":

$$F_{tropical \cap pescado \cap cría} = F_{tropical \cap cría} \cdot F_{pescado \cap crianza} / F_{cría}$$
$$= 5510 \cdot 36427/81885 = 2451$$

Estas estimaciones son mucho mejores que las producidas asumiendo la independencia, pero siguen siendo demasiado bajas. En lugar de almacenar aún más información, como el número de apariciones de triples de palabras, resulta que se pueden hacer estimaciones razonables del tamaño del resultado utilizando solo la frecuencia de palabras y el tamaño de la palabra.*Actual* conjunto resultante. Los motores de búsqueda estiman el tamaño del resultado porque no clasifican todos los documentos que contienen las palabras de consulta. En cambio, clasifican un subconjunto mucho más pequeño de los documentos que probablemente sean los más relevantes. Si conocemos la proporción del total de documentos que se han clasificado (s) y el número de documentos encontrados que contienen todas las palabras de consulta (C), simplemente podemos estimar el tamaño del resultado como C / s, lo que supone que los documentos que contienen todas las palabras se distribuyen uniformemente.[9] La proporción de documentos procesados se mide por la proporción de documentos que contienen la palabra menos frecuente que se han procesado, ya que todos los resultados deben contener esa palabra.

Por ejemplo, si la consulta "acuario de peces tropicales" se utiliza para clasificar los documentos GOV2 en el motor de búsqueda de Galago, después de procesar 3.000 de los 26.480 documentos que contienen "acuario", la cantidad de documentos que contienen las tres palabras es 258 . Esto da una estimación del tamaño del resultado de 258 / (3000 ÷ 26.480) = 2.277. Después de procesar algo más del 20% de los documentos, la estimación es de 1.778 (en comparación con la cifra real de 1.529). Para la consulta "cría de peces tropicales", las estimaciones después de procesar el 10% y el 20% de los documentos que contienen "cría" son 4.076

[9] También estamos asumiendo *documento a la vez* procesamiento, donde las listas invertidas para todas las palabras de consulta se procesan al mismo tiempo, dando puntajes completos del documento (ver Capítulo 5).

y 3.762 (en comparación con 3.629). Estas estimaciones, además de ser bastante precisas, no requieren el conocimiento del número total de documentos de la colección.

Estimar el número total de documentos almacenados en un motor de búsqueda es, de hecho, de gran interés tanto para el mundo académico (¿cuán grande es la Web?) Como para las empresas (¿qué motor de búsqueda tiene mejor cobertura de la Web?). Se han escrito varios artículos sobre técnicas para hacer esto, y uno de ellos se basa en el concepto de independencia de palabras que usamos antes. Si a y B son dos palabras que ocurren independientemente, entonces

$$F_{ab}/N = f_a/norte \cdot F_B/norte$$

y

$$N = (f_a \cdot F_B)/F_{ab}$$

Para obtener una estimación razonable de *NORTE,* las dos palabras deben ser independientes y, como hemos visto en los ejemplos de la tabla 4.6, este no suele ser el caso. Sin embargo, podemos tener más cuidado con la elección de las palabras de consulta. Por ejemplo, si usamos la palabra “lincoln” (frecuencia de documento 771,326 en GOV2), esperaríamos que las palabras en la consulta “tropical lincoln” fueran más independientes que los pares de palabras en la Tabla 4.6 (ya que las primeras están menos relacionadas semánticamente). La frecuencia de documentos de "tropical lincoln" en GOV2 es 3,018, lo que significa que podemos estimar el tamaño de la colección como $N = (120{,}990 \cdot 771{,}326) / 3{,}018 = 30{,}922{,}045$. Esto se acerca bastante al número real de 25.205.179.

4.3 Análisis de documentos

4.3.1 Resumen

El análisis de documentos implica el reconocimiento del contenido y la estructura de los documentos de texto. El contenido principal de la mayoría de los documentos son las palabras que estábamos contando y modelando usando la distribución Zipf en la sección anterior. Reconocer la aparición de cada palabra en la secuencia de caracteres de un documento se denomina *tokenizar* o *análisis léxico.* Aparte de estas palabras, puede haber muchos otros tipos de contenido en un documento, como metadatos, imágenes, gráficos, código y tablas. Como se mencionó en el Capítulo 2, los metadatos son información sobre un documento que no forma parte del contenido del texto. El contenido de los metadatos incluye atributos del documento como la fecha y el autor y, lo más importante, la *etiquetas* que son usados por *lenguajes de marcado* para identificar los componentes del documento. El más popular

Los lenguajes de marcado son HTML (lenguaje de marcado de hipertexto) y XML (lenguaje de marcado extensible).

El analizador usa las etiquetas y otros metadatos reconocidos en el documento para interpretar la estructura del documento según la sintaxis del lenguaje de marcado (*análisis sintáctico)* y producir una representación del documento que incluya tanto la estructura como el contenido. Por ejemplo, un analizador HTML interpreta la estructura de una página web como se especifica mediante etiquetas HTML y crea una representación del Modelo de objetos de documento (DOM) de la página que utiliza un navegador web. En un motor de búsqueda, la salida de un analizador de documentos es una representación del contenido y la estructura que se utilizará para crear índices. Dado que es importante que un índice de búsqueda represente todos los documentos de una colección, un analizador de documentos para un motor de búsqueda suele ser más tolerante con los errores de sintaxis que los analizadores utilizados en otras aplicaciones.

En la primera parte de nuestra discusión sobre el análisis de documentos, nos enfocamos en el reconocimiento de los tokens, palabras y frases que componen el contenido de los documentos. En secciones posteriores, discutimos por separado los temas importantes relacionados con la estructura del documento, a saber, marcado, enlaces y extracción de estructura del contenido del texto.

4.3.2 Tokenización

Tokenizar es el proceso de formar palabras a partir de la secuencia de caracteres en un documento. En el texto en inglés, esto parece simple. En muchos de los primeros sistemas, una "palabra" se definía como cualquier secuencia de caracteres alfanuméricos de longitud 3 o más, terminados por un espacio u otro carácter especial. Todas las letras mayúsculas también se convirtieron a minúsculas.[10] Esto significa, por ejemplo, que el texto

> El informe semestral de 2007 de Bigcorp mostró que las ganancias aumentaron un 10%.

produciría la siguiente secuencia de tokens:

> El informe anual de Bigcorp 2007 mostró que las ganancias aumentaron

Aunque este sencillo proceso de tokenización fue adecuado para experimentos con pequeñas colecciones de prueba, no parece apropiado para la mayoría de las aplicaciones de búsqueda o incluso experimentos con colecciones TREC, porque se descarta demasiada información. Algunos ejemplos de problemas relacionados con la tokenización que pueden tener un impacto significativo en la efectividad de la búsqueda son:

[10] Esto a veces se denomina *plegado de estuches, normalización de estuches,* o *Downcasing.*

- Las palabras pequeñas (uno o dos caracteres) pueden ser importantes en algunas consultas, generalmente en combinación con otras palabras. Por ejemplo,xp, ma, pm, ben e king, el paso, master p, gm, j lo, segunda guerra mundial.[11]
- Son comunes las formas con guiones y sin guiones de muchas palabras. En algunos casos, el guión no es necesario. Por ejemplo,e-bay, wal-mart, active-x, cd-rom, camisetas. En otras ocasiones, los guiones deben considerarse como parte de la palabra o como separador de palabras. Por ejemplo,winston-salem, mazda rx-7, tarjetas electrónicas, prediabetes, t-mobile, de habla hispana.
- Los caracteres especiales son una parte importante de las etiquetas, URL, código y otras partes importantes de los documentos que deben estar correctamente tokenizados.
- Las palabras en mayúscula pueden tener un significado diferente al de las palabras en minúscula. Por ejemplo, “Bush” y “Apple”.
- Los apóstrofos pueden ser parte de una palabra, parte de un posesivo o simplemente un error. Por ejemplo, rosie o'donnell, can, don't, 80's, 1890's, sombreros de paja para hombres, maestría, las diez ciudades más grandes de inglaterra, shriner's.
- Los números pueden ser importantes, incluidos los decimales. Por ejemplo,Nokia 3250, parte superior 10 campos, united 93, quicktime 6.5 pro, 92.3 the beat, 288358 (si, esto fue un consulta real; es un número de patente).
- Los puntos pueden aparecer en números, abreviaturas (por ejemplo, “IBM”, “Ph.D.”), URL, finales de oraciones y otras situaciones.

A partir de estos ejemplos, la creación de tokens parece más complicado de lo que parece. El hecho de que estos ejemplos provengan de consultas también enfatiza que el procesamiento de texto para consultas *deber* ser el mismo que se utiliza para los documentos. Si se utilizan diferentes procesos de tokenización para consultas y documentos, muchos de los términos de índice utilizados para documentos simplemente no coincidirán con los términos correspondientes de las consultas. Los errores en la tokenización se vuelven obvios muy rápidamente a través de fallas de recuperación.

Para poder incorporar el rango de procesamiento de lenguaje requerido para que la comparación sea efectiva, el proceso de tokenización debe ser simple y flexible. Un enfoque para hacer esto es que el primer paso de la tokenización se centre por completo en identificar el marcado o las etiquetas en el documento. Esto podría hacerse usando un tokenizador y analizador diseñado para el lenguaje de marcado específico usado (por ejemplo, HTML), pero debería acomodar errores de sintaxis en la estructura, como se mencionó anteriormente. Luego, se puede realizar una segunda pasada de tokenización en las partes apropiadas de la estructura del documento. Algunas partes que no se utilizan para la búsqueda, como las que contienen código HTML, se ignorarán en esta pasada.

[11] Estos y otros ejemplos se tomaron de una pequeña muestra de consultas web.

Dado que casi todo en el texto de un documento puede ser importante para alguna consulta, las reglas de tokenización tienen que convertir la mayor parte del contenido en tokens de búsqueda. En lugar de intentar hacer todo en el tokenizador, algunos de los problemas más difíciles, como identificar variantes de palabras o reconocer que una cadena es un nombre o una fecha, pueden manejarse mediante procesos separados, que incluyen la derivación, la extracción de información y transformación de consultas. La extracción de información generalmente requiere la forma completa del texto como entrada, incluidas las mayúsculas y la puntuación, por lo que esta información debe conservarse hasta que se haya realizado la extracción. Aparte de esta restricción, las mayúsculas rara vez son importantes para la búsqueda y el texto se puede reducir a minúsculas para indexar. Esto no significa que las palabras en mayúscula no se utilicen en las consultas. Son, de hecho, se utiliza con bastante frecuencia, pero en consultas en las que las mayúsculas no reducen la ambigüedad y, por lo tanto, no afectan la eficacia. Palabras como “Apple” que se utilizan a menudo en ejemplos (pero no tan a menudo en consultas reales) pueden manejarse mediante técnicas de reformulación de consultas (Capítulo 6) o simplemente confiando en las páginas más populares (sección 4.5).

Si consideramos que los problemas complicados son manejados por otros procesos, la estrategia más general para guiones, apóstrofos y puntos sería tratarlos como terminadores de palabras (como espacios). Es importante que todos los tokens producidos estén indexados, incluidos los caracteres individuales como "s" y "o". Esto significará, por ejemplo, que la consulta[12] "O'connor" es equivalente a "O connor", "bob" es equivalente a "Bob s", y "Rx-7" es equivalente a ”Rx 7”. Tenga en cuenta que esto también significará que una palabra como "rx7" será un token diferente a "rx-7" y, por lo tanto, se indexará por separado. La tarea de relacionar las consultasrx 7, rx7, y rx-7 luego será manejado por el componente de transformación de consultas del motor de búsqueda.

Por otro lado, si confiamos completamente en el componente de transformación de consultas para realizar las conexiones apropiadas o inferencias entre palabras, existe el riesgo de que la efectividad se reduzca, particularmente en aplicaciones donde no hay suficientes datos para una expansión confiable de consultas. En estos casos, se pueden incorporar más reglas en el tokenizador para garantizar que los tokens producidos por el texto de la consulta coincidan con los tokens producidos a partir del texto del documento. Por ejemplo, en el caso de las colecciones TREC, una regla que simboliza todas las palabras que contienen apóstrofos por la cadena sin el apóstrofo es muy eficaz. Con esta regla, "O'Connor" sería tokenizado como "oconnor" y "Bob's" produciría el token "bobs". Otra regla eficaz para las colecciones de TREC es tokenizar todas las abreviaturas

[12] Asumimos la sintaxis común para consultas web donde "<palabras>" significa coincidir exactamente la frase contenida en las comillas.

manteniendo puntos como la cadena sin puntos. En este caso, una abreviatura es cualquier cadena de caracteres alfabéticos individuales separados por puntos. Esta regla convertiría "IBM" en "ibm", pero "Ph.D." todavía estaría tokenizado como "ph d".

En resumen, el proceso de tokenización más general implicará primero identificar la estructura del documento y luego identificar las palabras en el texto como cualquier secuencia de caracteres alfanuméricos, terminados por un espacio o carácter especial, con todo convertido a minúsculas. Esto no es mucho más complicado que el proceso simple que describimos al comienzo de la sección, pero se basa en la extracción de información y la transformación de consultas para manejar los problemas difíciles. En muchos casos, se agregan reglas adicionales al tokenizador para manejar algunos de los caracteres especiales, para garantizar que los tokens de consulta y documento coincidan.

4.3.3 Detención

El lenguaje humano está lleno de palabras funcionales: palabras que tienen poco significado aparte de otras palabras. Los más populares: "el", "un", "un", "eso" y "esos", son *determinantes.* Estas palabras son parte de cómo describimos los sustantivos en el texto y expresamos conceptos como ubicación o cantidad. Las preposiciones, como "encima", "debajo", "arriba" y "abajo", representan la posición relativa entre dos sustantivos.

Dos propiedades de estas palabras de función hacen que deseemos tratarlas de una manera especial en el procesamiento de texto. Primero, estas palabras de función son extremadamente comunes. La Tabla 4.2 muestra que casi todas las palabras más frecuentes en la colección AP89 entran en esta categoría. Hacer un seguimiento de la cantidad de estas palabras en cada documento requiere mucho espacio en el disco. En segundo lugar, tanto por su carácter común como por su función, estas palabras rara vez indican algo sobre la relevancia del documento por sí mismas. Si estamos considerando palabras individuales en el proceso de recuperación y no frases, estas palabras funcionales nos ayudarán muy poco.

En la recuperación de información, estas palabras funcionales tienen un segundo nombre: *Para las palabras.* Las llamamos palabras vacías porque el procesamiento de texto se detiene cuando se ve una y se descartan. Tirar estas palabras disminuye el tamaño del índice, aumenta la eficacia de la recuperación y, en general, mejora la eficacia de la recuperación.

La construcción de una lista de palabras vacías debe hacerse con precaución. Eliminar demasiadas palabras perjudicará la efectividad de la recuperación de formas particularmente frustrantes para el usuario. Por ejemplo, la consulta"Ser o no ser" consiste enteramente en palabras que generalmente se consideran palabras vacías. Aunque no eliminar las palabras vacías puede causar algunos problemas en la clasificación, eliminar las palabras vacías puede hacer que las consultas perfectamente válidas no arrojen resultados.

Se puede construir una lista de palabras vacías simplemente usando la parte superior *n* (por ejemplo, 50) la mayoría palabras frecuentes en una colección. Sin embargo, esto puede llevar a que se incluyan palabras que son importantes para algunas consultas. Más típicamente, se usa una lista estándar de palabras vacías,[13] o una lista de palabras frecuentes y palabras vacías estándar se edita manualmente para eliminar cualquier palabra que pueda ser significativa para una aplicación en particular. También es posible crear listas de palabras vacías personalizadas para partes específicas de la estructura del documento (también llamado *campos).* Por ejemplo, las palabras "haga clic", "aquí" y "privacidad" pueden ser palabras vacías razonables para usar al procesar texto de anclaje.

Si los requisitos de espacio de almacenamiento lo permiten, es mejor indexar al menos todas las palabras de los documentos. Si es necesario detener, las palabras vacías siempre se pueden eliminar de las consultas. Al mantener las palabras vacías en el índice, habrá varias formas posibles de ejecutar una consulta con palabras vacías. Por ejemplo, muchos sistemas eliminarán las palabras vacías de una consulta a menos que la palabra esté precedida por un signo más (+). Si no es posible mantener las palabras vacías en un índice debido a los requisitos de espacio, se deben eliminar las menos posibles para mantener la máxima flexibilidad.

4.3.4 Derivado

Parte de la expresividad del lenguaje natural proviene de la gran cantidad de formas de transmitir una sola idea. Esto puede ser un problema para los motores de búsqueda, que dependen de la coincidencia de palabras para encontrar documentos relevantes. En lugar de restringir las coincidencias a palabras que son idénticas, se han desarrollado varias técnicas para permitir que un motor de búsqueda combine palabras que están relacionadas semánticamente. *Derivado,* también llamado *fusión* es un componente del procesamiento de texto que captura las relaciones entre diferentes variaciones de una palabra. Más precisamente, la derivación reduce las diferentes formas de una palabra que ocurren debido a *inflexión* ej., plurales, tiempos verbales) o *derivación* por ejemplo, convertir un verbo en un sustantivo agregando la su x -ción) a una raíz común.

Suponga que desea buscar artículos de noticias sobre la carrera de natación olímpica de Mark Spitz. Puede escribir "mark spitz nadando" en un motor de búsqueda. Sin embargo, muchos artículos de noticias suelen ser resúmenes de eventos que ya han sucedido, por lo que es probable que contengan la palabra "nadó" en lugar de "nadar". Es el trabajo de la lectora reducir "nadar" y "nadar" al mismo tallo (probablemente "nadar") y así permitir que el motor de búsqueda determine que hay una coincidencia entre estas dos palabras.

[13] Como el que se distribuye con el kit de herramientas Lemur y se incluye con Galago.

En general, el uso de un lematizador para aplicaciones de búsqueda con texto en inglés produce una mejora pequeña pero notable en la calidad de los resultados. En aplicaciones que involucran idiomas con muchas inflexiones, como el árabe o el ruso, la derivación es una parte crucial de la búsqueda efectiva.

Hay dos tipos básicos de lematizadores: *algorítmico* y *basado en diccionario.* Un lematizador algorítmico utiliza un pequeño programa para decidir si dos palabras están relacionadas, por lo general basándose en el conocimiento de los sufijos de palabras de un idioma en particular. Por el contrario, un lematizador basado en diccionario no tiene lógica propia, sino que se basa en diccionarios creados previamente de términos relacionados para almacenar relaciones de términos.

El tipo más simple de lematizador algorítmico en inglés es el *su ffi x-s* despalillador. Este tipo de lematización asume que cualquier palabra que termine en la letra "s" es plural, por lo que tortas → pastel, perros → perro. Por supuesto, esta regla no es perfecta. No puede detectar muchas relaciones plurales, como "siglo" y "siglos". En casos muy raros, detecta una relación donde no existe, como con "yo" y "es". El primer tipo de error se llama *falso negativo,* y el segundo tipo de error se llama *falso positivo.*[14]

Los lematizadores algorítmicos más complicados reducen el número de falsos negativos al considerar más tipos de su xos, como -ing o -ed. Al manejar más tipos de su ffi x, el lematizador puede encontrar más relaciones de términos; en otras palabras, se reduce la tasa de falsos negativos. Sin embargo, la tasa de falsos positivos (encontrar una relación donde no existe) generalmente aumenta.

El lematizador algorítmico más popular es el *Porter despalillador.*[15] Esto se ha utilizado en muchos experimentos y sistemas de recuperación de información desde la década de 1970, y hay varias implementaciones disponibles. El lematizador consta de una serie de pasos, cada uno de los cuales contiene un conjunto de reglas para eliminar su jos. En cada paso, se ejecuta la regla para la su x aplicable más larga. Algunas de las reglas son obvias, mientras que otras requieren un poco de reflexión para resolver lo que están haciendo. Como ejemplo, aquí están las dos primeras partes del paso 1 (de 5 pasos):

Paso 1a:

- Reemplazar *sses* por *ss*p.ej, destaca→estrés).
- Borrar *s* si la parte de la palabra precedente contiene una vocal no inmediatamente antes de la *s (*p.ej, brechas→brecha pero gas→gas).
- Reemplazar *ied* o *ies* por *I* si va precedido por más de una letra, en caso contrario por *es decir* (p.ej, corbatas→ corbata, gritos→cri).

[14] Estos términos se utilizan en cualquier proceso de decisión binaria para describir los dos tipos de errores. Esto incluye evaluación (Capítulo 8) y clasificación (Capítulo 9).

[15] http://tartarus.org/martin/PorterStemmer/

- Si su ffi x es *nosotros* o *ss* no hacer nada (p. ej., estrés →estrés).

Paso 1b:

- Reemplazar *eed, eedly* por *ee* si está en la parte de la palabra después de la primera no vocal que sigue a una vocal (por ejemplo, acordado →de acuerdo, alimentar → alimentación).
- Borrar *ed, edly, ing, ingly* si la parte de la palabra anterior contiene una vocal, y luego si la palabra termina en *en, bl,* o *iz* agregar *e (*p.ej, pescado → pescado, pirateo → pirata), o si la palabra termina con una letra doble que no es *ll, ss,* o *zz,* eliminar la última letra (p. ej., descendente →caer, gotear →goteo), o si la palabra es corta, agregue *e (*p.ej, esperando →esperanza).
- ¡Uf!

Se ha demostrado que la lectora Porter es eficaz en varias evaluaciones de TREC y aplicaciones de búsqueda. Sin embargo, es difícil capturar todas las sutilezas de un lenguaje en un algoritmo relativamente simple. La versión original del lematizador Porter cometió una serie de errores, tanto falsos positivos como falsos negativos. Mesa 4.7 muestra algunos de estos errores. Es fácil imaginar cómo la confusión de "ejecutar" con "ejecutivo" u "organización" con "órgano" podría causar problemas importantes en la clasificación. Una forma más reciente de lematizador (llamado Porter2)dieciséis corrige algunos de estos problemas y proporciona un mecanismo para especificar excepciones.

Falsos positivos	*Falsos negativos*
organización / órgano	europeo / europa
generalización / genérico	cilindro / cilíndrico
numérico / numeroso	matrices / matriz
política / policía	urgencia / urgente
universidad / universo	crear / crear
adición / aditivo	análisis / análisis
insignificante / negligente	útil / útil
ejecutar / ejecutivo	ruido / ruidoso
pasado / pegar	descomponer /
ignorar / ignorante	descomposición escasa / escasa
especial / especializado	resolver / resolución
cabeza / encabezado	triángulo / triangular

Cuadro 4.7. Ejemplos de errores cometidos por el tallador Porter original. Los falsos positivos son pares de palabras que tienen la misma raíz. Los falsos negativos son pares que tienen diferentes raíces.

dieciséis http://snowball.tartarus.org

Un lematizador basado en diccionario proporciona un enfoque diferente al problema de los errores de lematización. En lugar de intentar detectar relaciones de palabras a partir de patrones de letras, podemos almacenar listas de palabras relacionadas en un diccionario grande. Dado que estas listas de palabras pueden ser creadas por humanos, podemos esperar que la tasa de falsos positivos sea muy baja para estas palabras. Las palabras relacionadas ni siquiera necesitan tener un aspecto similar; un lematizador de diccionario puede reconocer que “es”, “ser” y “era” son todas formas del mismo verbo. Desafortunadamente, el diccionario no puede ser infinitamente extenso, por lo que no puede reaccionar automáticamente a nuevas palabras. Este es un problema importante ya que el lenguaje evoluciona constantemente. Es posible construir diccionarios troncales automáticamente mediante el análisis estadístico de un corpus de texto. Dado que esto es particularmente útil cuando se utiliza la derivación para la expansión de consultas,

Otra estrategia es combinar un lematizador algorítmico con un lematizador basado en diccionario. Normalmente, las palabras irregulares como el verbo "to be" son las más antiguas del idioma, mientras que las palabras nuevas siguen convenciones gramaticales más regulares. Esto significa que es probable que las palabras recién inventadas funcionen bien con un lematizador algorítmico. Se puede usar un diccionario para detectar relaciones entre palabras comunes, y el lematizador algorítmico se puede usar para palabras no reconocidas.

Un ejemplo bien conocido de este enfoque híbrido es el *Despalilladora Krovetz (* Krovetz, 1993). Este lematizador hace un uso constante de un diccionario para comprobar si la palabra es válida. El diccionario del lematizador Krovetz se basa en un diccionario de inglés general, pero también utiliza excepciones que se generan manualmente. Antes de derivar, se comprueba el diccionario para ver si hay una palabra presente; si es así, se deja solo (si está en el diccionario general) o se deriva en función de la entrada de excepción. Si la palabra no está en el diccionario, se comprueba si hay una lista de su jos comunes de inflexión y derivación. Si se encuentra uno, se elimina y se vuelve a comprobar el diccionario para ver si la palabra está presente. Si no se encuentra, la terminación de la palabra se puede modificar según la terminación que se eliminó. Por ejemplo, si el final -ies se encuentra, se reemplaza por -es decir y comprobado en el diccionario. Si se encuentra en el diccionario, se acepta la raíz; de lo contrario, el final es reemplazado por*y.* Esto resultará en calorías→caloría, por ejemplo. Los sufijos se comprueban en una secuencia (por ejemplo, plurales antes de -ion terminaciones), por lo que se pueden eliminar múltiples su xos.

El lematizador de Krovetz tiene una tasa de falsos positivos más baja que la de Porter, pero también tiende a tener una tasa de falsos negativos más alta, dependiendo del tamaño de los diccionarios de excepción. En general, la efectividad de los dos lematizadores es comparable cuando se utilizan en las evaluaciones de búsqueda. El tallador Krovetz tiene la ventaja adicional de producir tallos que, en la mayoría de los casos, son palabras completas, mientras que el Porter

Texto original:

El documento describirá las estrategias de marketing llevadas a cabo por las empresas estadounidenses para sus productos químicos agrícolas, informará las predicciones de la participación de mercado de dichos productos químicos o informará las estadísticas de mercado de agroquímicos, pesticidas, herbicidas, fungicidas, insecticidas, fertilizantes, ventas previstas, participación de mercado, estimular la demanda, reducción de precios, volumen de ventas.

Lechador Porter:

documento describe estrategia de mercado carri compani agricultur informe químico predecir participación de mercado informe químico estadístico de mercado agroquímico pesticida herbicida fungicida insecticida fertilizante predecir venta participación de mercado estimul demanda reducción de precios volumen venta

Trituradora de Krovetz:

documento describir estrategia de marketing llevar empresa agricultura informe químico predicción cuota de mercado informe químico estadística de mercado pesticida agroquímico herbicida fungicida insecticida fertilizante predecir venta estimular demanda reducción de precio volumen venta

Figura 4.6. Comparación de la salida del lematizador para una consulta TREC. También se han eliminado las palabras vacías.

Stemmer a menudo produce raíces que son fragmentos de palabras. Esta es una preocupación si los tallos se utilizan en la interfaz de búsqueda.

La Figura 4.6 compara la salida de los lematizadores de Porter y Krovetz en el texto de una consulta TREC. La salida del lematizador Krovetz es similar en cuanto a que las palabras se reducen a las mismas raíces, aunque "marketing" no se reduce a "mercado" porque estaba en el diccionario. Los tallos producidos por el lematizador Krovetz son en su mayoría palabras. La excepción es la raíz "agroquímico", que se produjo porque "agroquímico" no estaba en el diccionario. Tenga en cuenta que el procesamiento de texto en este ejemplo ha eliminado las palabras vacías, incluidos los caracteres individuales. Esto dio como resultado la eliminación de “EE.UU.” del texto, lo que podría tener consecuencias importantes para algunas consultas. Esto puede manejarse mediante una mejor tokenización o extracción de información, como discutimos en la sección 4.6.

Como en el caso de las palabras vacías, el motor de búsqueda tendrá más flexibilidad para responder a una amplia gama de consultas si las palabras del documento no se derivan sino que están indexadas en su forma original. La derivación se puede realizar como un tipo de expansión de consultas, como se explica en la sección 6.2.1. En algunas aplicaciones, tanto las palabras completas como sus raíces están indexadas para proporcionar flexibilidad y tiempos de procesamiento de consultas eficientes.

Mencionamos anteriormente que la derivación puede ser particularmente importante para algunos idiomas y prácticamente no tiene impacto en otros. Incorporación de idiomas específicos

algoritmos de derivación es uno de los aspectos más importantes de la personalización, o *internacionalización,* un motor de búsqueda para varios idiomas. Discutimos otros aspectos de la internacionalización en la sección 4.7, pero aquí nos enfocamos en las cuestiones derivadas.

Como ejemplo, la Tabla 4.8 muestra algunas de las palabras árabes derivadas de la misma raíz. Un algoritmo de lematización que redujera las palabras árabes a sus raíces claramente no funcionaría (hay menos de 2000 raíces en árabe), pero se debe considerar una amplia gama de prefijos y sufijos. Los lenguajes con muchas inflexiones como el árabe tienen muchas variantes de palabras, y la derivación puede marcar una gran diferencia en la precisión de la clasificación. Un motor de búsqueda árabe con derivación de alta calidad puede ser más del 50% más eficaz, en promedio, para encontrar documentos relevantes que un sistema sin derivación. Por el contrario, las mejoras para un motor de búsqueda en inglés varían de menos del 5% en promedio para colecciones grandes a alrededor del 10% para colecciones pequeñas de dominios específicos.

kItaB	*un libro*
kItaBI	*mi libro*
AlabamakItaB	*el libro*
kItaBuki	*tu libro (*f)
kItaBuka	*tu libro (m)*
kItaBuhu	*su libro*
kataBa	*escribir*
mamáktaBa	*biblioteca, librería*
mamáktaB	*oficina*

Cuadro 4.8. Ejemplos de palabras con raíz árabe ktb

Afortunadamente, ya se han desarrollado lematizadores para varios idiomas y están disponibles como software de código abierto. Por ejemplo, la lectora Porter está disponible en francés, español, portugués, italiano, rumano, alemán, holandés, sueco, noruego, danés, ruso, finlandés, húngaro y turco.[17] Además, el enfoque estadístico para construir un lematizador que se describe en la sección 6.2.1 se puede utilizar cuando solo se dispone de un corpus de texto.

[17] http://snowball.tartarus.org/

4.3.5 Frases y N-gramos

Las frases son claramente importantes en la recuperación de información. Muchas de las consultas de dos y tres palabras enviadas a los motores de búsqueda son frases, y encontrar documentos que contengan esas frases será parte de cualquier algoritmo de clasificación eficaz. Por ejemplo, dada la consulta "mar negro", es mucho más probable que los documentos que contienen esa frase sean relevantes que los documentos que contienen texto como "el mar se volvió negro". Las frases son más precisas que las palabras simples como descripciones de temas (p. Ej., “Peces tropicales” versus “peces”) y generalmente menos ambiguas (p. Ej., “Manzana podrida” versus “manzana”). Sin embargo, el impacto de las frases en la recuperación puede ser complejo. Dada una consulta como "suministros de pesca", si los documentos recuperados contienen exactamente esa frase, o deben recibir crédito por contener las palabras "pescado", "pesca" y "suministros" en el mismo párrafo, o incluso el mismo documento? Los detalles de cómo las frases afectan la clasificación dependerán del modelo de recuperación específico que se incorpore en el motor de búsqueda, por lo que pospondremos esta discusión hasta el Capítulo 7. Desde la perspectiva del procesamiento de texto, la cuestión es si las frases deben identificarse en al mismo tiempo que la tokenización y la derivación, para que puedan indexarse para un procesamiento de consultas más rápido.

Hay varias definiciones posibles de una frase, y la mayoría de ellas se han estudiado en experimentos de recuperación a lo largo de los años. Dado que una frase tiene una definición gramatical, parece razonable identificar frases usando la estructura sintáctica de las oraciones. La definición que se ha utilizado con mayor frecuencia en la investigación de recuperación de información es que una frase equivale a una simple *frase nominal.* Esto a menudo se restringe aún más para incluir solo secuencias de sustantivos o adjetivos seguidos de sustantivos. Las frases definidas por estos criterios se pueden identificar mediante un *etiquetador de parte del discurso (POS).* Un etiquetador POS marca las palabras en un texto con etiquetas que corresponden a la parte gramatical de la palabra en ese contexto. Los etiquetadores se basan en enfoques estadísticos o basados en reglas y se entrenan usando grandes corpora que han sido etiquetados manualmente. Las etiquetas típicas que se utilizan para etiquetar las palabras incluyen NN (sustantivo singular), NNS (sustantivo plural), VB (verbo), VBD (verbo, tiempo pasado), VBN (verbo, participio pasado), IN (preposición), JJ (adjetivo), CC (conjunción, por ejemplo, "y", "o"), PRP (pronombre) y MD (auxiliar modal, por ejemplo, "puede", "voluntad").

La Figura 4.7 muestra la salida de un etiquetador POS para el texto de consulta TREC usado en la Figura 4.6. Este ejemplo muestra que el etiquetador puede identificar frases que son secuencias de sustantivos, como “marketing / estrategias de NN / NNS”, o adjetivos seguidos de sustantivos, como “agrícolas / químicos JJ / NNS”. Sin embargo, los etiquetadores cometen errores. Las palabras "predicho / VBN sales / NNS" no se identificarían como un sintagma nominal, porque "predicho" está etiquetado como un verbo.

Texto original:

El documento describirá las estrategias de marketing llevadas a cabo por las empresas estadounidenses para sus productos químicos agrícolas, informará las predicciones de la participación de mercado de dichos productos químicos o informará las estadísticas de mercado de agroquímicos, pesticidas, herbicidas, fungicidas, insecticidas, fertilizantes, ventas previstas, participación de mercado, estimular la demanda, recorte de precios, volumen de ventas.

Brill tagger:

Documento / NN will / MD describirá / VB marketing / NN estrategias / NNS llevado / VBD out / IN by / IN US / NNP Companies / NNS for / IN its / PRP agrícola / JJ chemical / NNS, /, report / NN predictions / NNS para / IN market / NN share / NN of / IN such / JJ chemical / NNS, / o / CC report / NN market / NN statistics / NNS for / IN agroquímicos / NNS, /, pesticida / NN, /, herbicida / NN, /, fungicida / NN, /, insecticida / NN, /, fertilizante / NN , /, ventas previstas / VBN / NNS, /, mercado / cuota de NN / NN, /, estimular / demanda de VB / NN, /, precio / reducción de NN / NN , /, volumen / NN de / IN ventas / NNS ./.

Figura 4.7. Salida de un etiquetador POS para una consulta TREC

La Tabla 4.9 muestra los sintagmas nominales simples de alta frecuencia de un corpus TREC que consisten principalmente en noticias y un corpus de tamaño comparable que consiste en todas las patentes de 1996 emitidas por la Oficina de Patentes y Marcas Registradas de los Estados Unidos (PTO). Las frases se identificaron mediante etiquetado POS. Las frecuencias de las frases de ejemplo indican que las frases se utilizan con más frecuencia en la colección PTO, porque las patentes están escritas en un estilo formal y legal con una considerable repetición. Había 1,100,000 frases en la colección TREC que ocurrieron más de cinco veces, y 3,700,000 frases en la colección PTO. Muchas de las frases TREC son nombres propios, como “los angeles” o “unión europea”, o son temas que serán importantes para la recuperación , como "proceso de paz" y "derechos humanos". Dos frases están asociadas al formato de los documentos (“tipo de artículo”, “fin de grabación”). Por otro lado, la mayoría de las frases de alta frecuencia en la colección de PTO son términos estándar que se utilizan para describir todas las patentes, como “invención actual” y “realización preferida”, y relativamente pocas están relacionadas con el contenido de las patentes. , como "átomos de carbono" y "acetato de etilo". Una de las frases, "grupo consistente", fue el resultado de un error de etiquetado frecuente.

Aunque el etiquetado POS produce frases razonables y se usa en varias aplicaciones, en general es demasiado lento para usarse como base para la indexación de frases de colecciones grandes. Existen alternativas más simples y rápidas que son igual de efectivas. Un enfoque consiste en almacenar información sobre la posición de las palabras en los índices y utilizar esta información para identificar frases solo cuando se procesa una consulta. Esto proporciona una flexibilidad considerable en el sentido de que el usuario puede identificar las frases o mediante el etiquetado de POS en la consulta, y no se restringen a grupos adyacentes de

Datos TREC		Datos de patentes	
Frecuencia	*Frase*	*Frecuencia*	*Frase*
65824	Estados Unidos	975362	presente invención
61327	tipo de artículo	191625	nosotros palmaditas
33864	los Angeles	147352	realización preferida
18062	Hong Kong	95097	Átomos de carbón
17788	Corea del Norte	87903	grupo que consiste
17308	Nueva York	81809	temperatura ambiente
15513	San Diego	78458	ID de secuencia
15009	condado de Orange	75850	breve descripción
12869	Primer ministro	66407	arte previo
12799	primera vez	59828	vista de perspectiva
12067	Unión Soviética	58724	primera encarnación
10811	Federación Rusa	56715	mezcla de reacción
9912	Naciones Unidas	54619	Descripción detallada
8127	Sureste de california	54117	acetato de etilo
7640	Corea del Sur	52195	Ejemplo 1
7620	finalizar la grabación	52003	diagrama de bloques
7524	Unión Europea	46299	segunda encarnación
7436	Sudáfrica	41694	dibujos adjuntos
7362	San Francisco	40554	señal de salida
7086	conferencia de prensa	37911	primer fin
6792	Ayuntamiento	35827	segundo fin
6348	Oriente Medio	34881	reclamaciones adjuntas
6157	proceso de paz	33947	fin distante
5955	derechos humanos	32338	vista en sección transversal
5837	casa Blanca	30193	Superficie exterior

Cuadro 4.9. Frases nominales de alta frecuencia de una colección TREC y patentes de EE. UU. de 1996

palabras. La identificación de frases sintácticas se reemplaza probando restricciones de proximidad de palabras, como si dos palabras ocurren dentro de una ventana de texto especificada. Describimos la indexación de posiciones en el Capítulo 5 y los modelos de recuperación que explotan la proximidad de palabras en el Capítulo 7.

En aplicaciones con colecciones grandes y restricciones estrictas en el tiempo de respuesta, como la búsqueda web, también es probable que probar la proximidad de palabras en el momento de la consulta sea demasiado lento. En ese caso, podemos volver a identificar frases en los documentos durante

procesamiento de texto, pero use una definición mucho más simple de una frase: cualquier secuencia de *norte* palabras. Esto también se conoce como *n-gramo.* Las secuencias de dos palabras se llaman *bigrams,* y las secuencias de tres palabras se llaman *trigramas.* Las palabras sueltas se llaman *unigramos.* Los N-gramas se han utilizado en muchas aplicaciones de texto y los mencionaremos de nuevo con frecuencia en este libro, particularmente en asociación con *modelos de lenguaje* sección 7.3). En esta discusión, nos enfocamos en *palabra* n-gramos, pero *personaje*
Los n-gramas también se utilizan en aplicaciones como OCR, donde el texto es "ruidoso" y la coincidencia de palabras puede resultar difícil (sección 11.6). Los n-gramas de caracteres también se utilizan para indexar idiomas como el chino que no tienen saltos de palabras (sección 4.7). Los N-gramas, tanto de caracteres como de palabras, se generan eligiendo un valor particular para *norte* y luego mover esa "ventana" hacia adelante una unidad (carácter o palabra) a la vez. En otras palabras, n-gramos *superposición.* Por ejemplo, la palabra "tropical" contiene los siguientes bigramas de caracteres: tr, ro, op, pi, ic, ca y al. Los índices basados en n-gramas son obviamente más grandes que los índices de palabras.

Cuanto más frecuentemente aparece una palabra n-gramo, más probable es que corresponda a una frase significativa en el idioma. N-gramas de todas las longitudes forman una distribución Zipf, con algunas frases comunes que ocurren con mucha frecuencia y un gran número ocurre con la frecuencia 1. De hecho, los datos de frecuencia de rango para n-gramas (que incluyen palabras simples) se ajustan a la Distribución Zipf mejor que solo palabras. Algunos de los n-gramas más comunes estarán compuestos por palabras vacías (por ejemplo, "y el", "hay") y podrían ignorarse, aunque al igual que con las palabras, debemos tener cuidado al descartar información. Nuestra consulta de ejemplo anterior "ser o no ser" ciertamente podría hacer uso de n-gramas. Potencialmente, podríamos indexar todos los n-gramas en un texto de documento hasta una longitud específica y ponerlos a disposición del algoritmo de clasificación. Esto parecería ser un uso extravagante del tiempo de indexación y del espacio en disco debido a la gran cantidad de n-gramas posibles. Un documento que contenga 1000 palabras, por ejemplo, contendría 3990 instancias de n-gramos de palabras de longitud
$2 \leq norte \leq 5$. Sin embargo, muchos motores de búsqueda web utilizan la indexación de n-gramas porque proporciona un método rápido para incorporar características de frases en la clasificación.

Recientemente, Google puso a disposición un archivo de n-gramas derivado de páginas web.[18] Las estadísticas de esta muestra se muestran en la Tabla 4.10. Un análisis de n-gramas en la Web (Yang et al., 2007) encontró que "todos los derechos reservados" era el trigrama más frecuente en inglés, mientras que "sociedad de responsabilidad limitada" era el más frecuente en chino. En ambos casos, esto se debió a la gran cantidad de sitios corporativos, pero

[18] http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

También indica que los n-gramas no están dominados por patrones comunes de habla. como "y será".

Numero de tokens:	1.024.908.267.229
Numero de sentencias:	95,119,665,584
Número de unigramos:	13,588,391
Número de bigramas:	314,843,401
Número de trigramas:	977,069,902
Número de cuatro gramos:	1.313.818.354
Número de cinco gramos:	1,176,470,663

Cuadro 4.10. Estadísticas de la muestra de n-gramas de Google

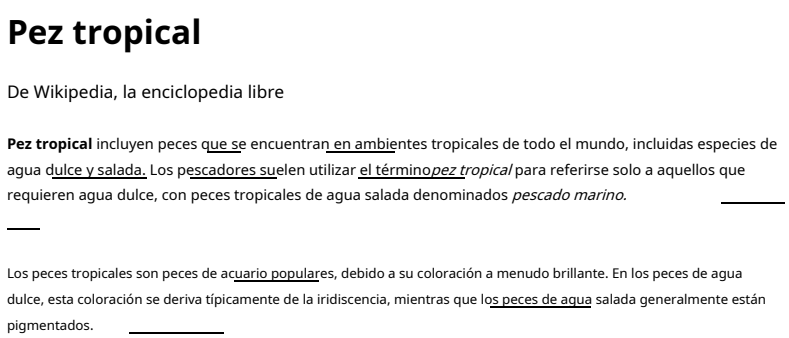
4.4 Estructura y marcado del documento

En las aplicaciones de bases de datos, los campos o atributos de los registros de la base de datos son una parte fundamental de la búsqueda. Las consultas se especifican en términos de los valores obligatorios de estos campos. En algunas aplicaciones de texto, como el correo electrónico o la búsqueda de literatura, campos como *autor* y *fecha* tendrá una importancia similar y será parte de la especificación de la consulta. En el caso de la búsqueda web, las consultas no suelen referirse a la estructura o los campos del documento, pero eso no significa que esta estructura no sea importante. Algunas partes de la estructura de las páginas web, indicadas por HTMLmarkup, son características muy importantes utilizadas por el algoritmo de clasificación. El analizador de documentos debe reconocer esta estructura y ponerla a disposición para la indexación.

Como ejemplo, la Figura 4.8 muestra parte de una página web para una Wikipedia[19] entrada. La página tiene una estructura obvia que podría usarse en un algoritmo de clasificación. El título principal de la página, “peces tropicales”, indica que esta frase es particularmente importante. La misma frase también está en negrita y cursiva en el cuerpo del texto, lo que es una prueba más de su importancia. Otras palabras y frases se utilizan como *texto de anclaje* para enlaces y es probable que sean buenos términos para representar el contenido de la página.

El código fuente HTML de esta página web (Figura 4.9) muestra que hay aún más estructura que debería representarse para la búsqueda. Cada campo o*elemento* en HTML se indica mediante una etiqueta de inicio (como <h1>) y una etiqueta de cierre opcional (p. ej.,

[19] Enciclopedia TheWeb, http://en.wikipedia.org/.



Pez tropical

De Wikipedia, la enciclopedia libre

Pez tropical incluyen peces que se encuentran en ambientes tropicales de todo el mundo, incluidas especies de agua dulce y salada. Los pescadores suelen utilizar el término *pez tropical* para referirse solo a aquellos que requieren agua dulce, con peces tropicales de agua salada denominados *pescado marino.*

Los peces tropicales son peces de acuario populares, debido a su coloración a menudo brillante. En los peces de agua dulce, esta coloración se deriva típicamente de la iridiscencia, mientras que los peces de agua salada generalmente están pigmentados.

Figura 4.8. Parte de una página web de Wikipedia

</h1>).[20] Los elementos también pueden tener atributos (con valores), dados por atributo_name = "valor" pares. El <cabeza> El elemento de un documento HTML contiene metadatos que no se muestran en un navegador. El elemento de metadatos para palabras clave (<meta nombre = "palabras clave") proporciona una lista de palabras y frases que se pueden utilizar como términos de contenido adicionales para la página. En este caso, estos son los títulos de otras páginas de Wikipedia. El <título> El elemento de metadatos proporciona el título de la página (que es diferente del encabezado principal).

El <cuerpo> elemento del documento contiene el contenido que se muestra. El encabezado principal está indicado por el <h1> etiqueta. Otros encabezados, de diferentes tamaños y potencialmente diferente importancia, serían indicados por <h2> hasta <h6> etiquetas. Los términos que deben mostrarse en negrita o cursiva se indican con <b> y <i> etiquetas. A diferencia de los campos de base de datos típicos, estas etiquetas se utilizan principalmente para formatear y pueden aparecer muchas veces en un documento. También pueden, como hemos dicho, interpretarse como una etiqueta que indica una palabra o frase de cierta importancia.

Enlaces, como pescado , son muy comunes. Ellos son la base de algoritmos de análisis de enlaces como PageRank (Brin & Page, 1998), pero también definen el texto de anclaje. Los enlaces y el texto de anclaje son de particular importancia para la búsqueda web y se describirán en la siguiente sección. Latítulo El atributo de un enlace se utiliza para proporcionar información adicional sobre ese enlace, aunque en nuestro ejemplo son las palabras en la última parte de la URL de la página de Wikipedia asociada. Los motores de búsqueda web también utilizan la URL de una página como fuente de metadatos adicionales. La URL de esta página es:

http://en.wikipedia.org/wiki/Tropical_fish

[20] En XML, la etiqueta final no es opcional.

El hecho de que las palabras “tropical” y “pez” aparezcan en la URL aumentará la importancia de esas palabras para esta página. La profundidad de una URL (es decir, el número de directorios de profundidad de la página) también puede ser importante. Por ejemplo, la URL www.ibm.com es más probable que sea la página de inicio de IBM que una página con la URL:

www.pcworld.com/businesscenter/article/698/ibm_buys_apt!

```
<html>
<cabeza>
<meta name = "keywords" content = "Peces tropicales, Airstone, Albinismo, Devorador de algas, Acuario,
Comedero para peces de acuario, Muebles de acuario, Paisajismo acuático, Tratamiento de baño
(mantenimiento de peces), Método Berlín, Biotopo" />
...
<title> Peces tropicales - Wikipedia, la enciclopedia libre </title> </
head>
<cuerpo>
 ...
<h1 class = "firstHeading"> Peces tropicales </
h1>...
<p> Los <b> peces tropicales </b> incluyen <a href="/wiki/Fish" title="Fish"> peces </a> que se
encuentran en <a href = "/ wiki / Tropics" title = "Tropics "> entornos tropicales </a> de todo el mundo,
incluidos <a href="/wiki/Fresh_water" title="Fresh water"> agua dulce </a> y <a href =" / wiki / Sea_water
"title =" Especies de agua de mar "> agua salada </a>. <a
href = "/ wiki / Fishkeeping" title = "Fishkeeping"> Los cuidadores de peces </a> a menudo usan el término <i>
peces tropicales </i> para referirse solo a aquellos que requieren agua dulce, y los peces tropicales de agua
salada se denominan <i> <a href="/wiki/List_of_marine_aquarium_fish_species" title="Lista de especies de peces
de acuario marino"> peces marinos </a> </i>. </p>
<p> Los peces tropicales son peces de <a href="/wiki/Aquarium" title="Aquarium"> acuario </a> populares debido
a su coloración a menudo brillante. En los peces de agua dulce, esta coloración generalmente se deriva de <a
href="/wiki/Iridescence" title="Iridescence"> iridiscencia </a>, mientras que los peces de agua salada son
generalmente <a href = "/ wiki / Pigment" title = " Pigmento "> pigmentado </a>. </p>...

</body> </html>
```

Figura 4.9. Fuente HTML, por ejemplo, página de Wikipedia.

En HTML, los tipos de elementos están predefinidos y son los mismos para todos los documentos. XML, por el contrario, permite que cada aplicación defina cuáles son los tipos de elementos y qué etiquetas se utilizan para representarlos. Los documentos XML pueden describirse mediante un*esquema,* similar a un esquema de base de datos. En consecuencia, los elementos XML están más estrechamente vinculados a la semántica de los datos que los elementos HTML. Solicitud de búsqueda

a menudo utilizan XML para registrar *anotaciones semánticas* en los documentos que se producen mediante técnicas de extracción de información, como se describe en el apartado 4.6. Un analizador de documentos para estas aplicaciones registraría las anotaciones, junto con la otra estructura del documento, y las pondría a disposición para su indexación.

El lenguaje de consulta XQuery[21] ha sido definido por la comunidad de bases de datos para la búsqueda de datos estructurados descritos en XML. XQuery admite consultas que especifican restricciones estructurales y de contenido, lo que plantea la cuestión de si un enfoque de recuperación de información o de base de datos es mejor para construir un motor de búsqueda de datos XML. Discutimos este tema con más detalle en la sección 11.4, pero la respuesta general es que dependerá de los datos, la aplicación y las necesidades del usuario. Para datos XML que contienen una proporción sustancial de texto, el enfoque de recuperación de información es superior. En el Capítulo 7, describiremos los modelos de recuperación que están diseñados para documentos de texto que contienen tanto estructura como metadatos.

4.5 Análisis de enlaces

Los enlaces que conectan las páginas son un componente clave de la Web. Los enlaces son una poderosa ayuda de navegación para las personas que navegan por la Web, pero también ayudan a los motores de búsqueda a comprender las relaciones entre las páginas. Estas relaciones detectadas ayudan a los motores de búsqueda a clasificar las páginas web de forma más eficaz. Sin embargo, debe recordarse que muchas colecciones de documentos que se utilizan en aplicaciones de búsqueda, como la búsqueda de escritorio o empresarial, no tienen vínculos o tienen muy poca estructura de vínculos. Para estas colecciones, el análisis de enlaces no tendrá ningún impacto en el rendimiento de la búsqueda.

Como vimos en la última sección, un enlace en una página web está codificado en HTML con una declaración como:

> Para obtener más información sobre este tema, vaya a la página de algún lugar .

Cuando esta página aparezca en su navegador web, las palabras "la página de algún lugar" se mostrarán de manera diferente al texto normal, generalmente subrayadas o en un color diferente (o ambos). Cuando haga clic en ese enlace, su navegador cargará la página web http://www.somewhere.com. En este enlace, "la página de algún lugar" se llama *texto de anclaje,* y http://www.somewhere.com es el *destino.* Ambos componentes son útiles en el proceso de clasificación.

[21] http://www.w3.org/XML/Query/

4.5.1 Texto de anclaje

El texto de anclaje tiene dos propiedades que lo hacen particularmente útil para clasificar páginas web. Primero, tiende a ser muy corto, quizás dos o tres palabras, y esas palabras a menudo describen de manera sucinta el tema de la página vinculada. Por ejemplo, enlaces awww.ebay.com es muy probable que contengan la palabra "eBay" en el texto de anclaje. Muchas consultas son muy similares al texto de anclaje en el sentido de que también son breves descripciones temáticas de páginas web. Esto sugiere un algoritmo muy simple para clasificar las páginas: busque en todos los enlaces de la colección, buscando un texto de anclaje que coincida exactamente con la consulta del usuario. Cada vez que haya una coincidencia, agregue 1 a la puntuación de la página de destino. Las páginas se clasificarían en orden decreciente de esta puntuación. Este algoritmo tiene algunas fallas evidentes, una de las cuales es cómo manejar la consulta "haga clic aquí". De manera más general, la colección de todo el texto de anclaje en los enlaces que apuntan a una página puede usarse como un campo de texto adicional para esa página e incorporarse al algoritmo de clasificación.

El texto de anclaje generalmente lo escriben personas que no son los autores de la página de destino. Esto significa que el texto de anclaje puede describir una página de destino desde una perspectiva diferente o enfatizar el aspecto más importante de la página desde el punto de vista de la comunidad. El hecho de que el enlace exista es un voto de importancia para la página de destino. Aunque el texto de anclaje no se menciona con tanta frecuencia como los algoritmos de análisis de enlaces (por ejemplo, PageRank) en las discusiones de los motores de búsqueda web, las evaluaciones de TREC han demostrado que es la parte más importante de la representación de una página para algunos tipos de búsqueda web. En particular, es esencial para las búsquedas en las que el usuario intenta encontrar una página de inicio para un tema, una persona u organización en particular.

4.5.2 PageRank

Hay decenas de miles de millones de páginas web, pero la mayoría de ellas no son muy interesantes. Muchas de esas páginas son spam y contienen contenido útil. Otras páginas son blogs personales, anuncios de bodas o álbumes de fotos familiares. Estas páginas son interesantes para una pequeña audiencia, pero probablemente no en general. Por otro lado, hay algunas páginas que son populares y útiles para muchas personas, incluidos los sitios de noticias y los sitios web de empresas populares.

El enorme tamaño de la Web hace que este sea un problema difícil para los motores de búsqueda. Suponga que un amigo le ha dicho que visite el sitio de eBay y usted no lo supiera www.ebay.com era la URL a utilizar. Puede escribir "eBay" en un motor de búsqueda, pero hay millones de páginas web que contienen la palabra "eBay". ¿Cómo puede el

¿El motor de búsqueda elige el más popular (y probablemente el correcto)? Un enfoque muy eficaz es utilizar los enlaces entre páginas web como una forma de medir la popularidad. La medida más obvia es contar el número de*inlinks*enlaces que apuntan a una página) para cada página y utilícelo como una característica o pieza de evidencia en el algoritmo de clasificación. Aunque se ha demostrado que es bastante eficaz, es muy susceptible al spam. Las medidas basadas en algoritmos de análisis de enlaces están diseñadas para proporcionar valoraciones más fiables de las páginas web. De estas medidas, el PageRank, que está asociado con el motor de búsqueda de Google, es el que se menciona con mayor frecuencia.

PageRank se basa en la idea de un *surfista aleatorio*como en web surfer). Imagina a una persona llamada Alice que está usando su navegador web. Alice está extremadamente aburrida, por lo que deambula sin rumbo fijo entre páginas web. Su navegador tiene un botón especial "Sorpréndeme" en la parte superior que saltará a una página web aleatoria cuando haga clic en ella. Cada vez que se carga una página web, elige si hace clic en el botón "Sorpréndeme" o si hace clic en uno de los enlaces de la página web. Si hace clic en un enlace de la página, no tiene preferencia por ningún enlace en particular; en cambio, solo elige uno al azar. Alice está lo suficientemente aburrida como para seguir navegando en la Web de esta manera para siempre.[22]

Para poner esto en una forma más estructurada, Alice navega por la Web usando este algoritmo:

1. Elija un número aleatorio r entre 0 y 1.
2. Si $r < \lambda$:
   - Haga clic en el botón "Sorpréndame".
3. Si $r \geq \lambda$:
   - Haga clic en un enlace al azar en la página actual.
4. Empiece de nuevo.

Normalmente asumimos que λ es bastante pequeño, por lo que es mucho más probable que Alice haga clic en un enlace que elegir el botón "Sorpréndeme". Aunque la ruta de Alice a través de las páginas web es aleatoria, Alice seguirá viendo páginas populares con más frecuencia que las impopulares. Eso es porque Alice a menudo sigue enlaces, y los enlaces tienden a apuntar a páginas populares. Por lo tanto, esperamos que Alice termine en el sitio web de una universidad, por ejemplo, con más frecuencia que en un sitio web personal, pero con menos frecuencia que en el sitio web de CNN.

[22] El cálculo de PageRank corresponde a encontrar lo que se conoce como problema estacionario. distribución de habilidades de un *Caminata aleatoria* en el gráfico de la Web. Un paseo aleatorio es un caso especial de un*Cadena de Markov* en el que el siguiente estado (la siguiente página visitada) depende únicamente del estado actual (página actual). Las transiciones permitidas entre estados son todas igualmente probables y están dadas por los enlaces.

Suponga que CNN ha publicado una historia que contiene un enlace a la página web de un profesor. Ahora es mucho más probable que Alice visite la página de ese profesor, porque Alice visita el sitio web de CNN con frecuencia. Un solo enlace en CNN puede influir en la actividad de Alice más que cientos de enlaces en sitios menos populares, porque Alice visita CNN con mucha más frecuencia que esos sitios menos populares.

Debido al botón especial "Sorpréndeme" de Alice, podemos estar seguros de que eventualmente llegará a todas las páginas de Internet.[23] Dado que planea navegar por la Web durante mucho tiempo, y dado que el número de páginas web es finito, visitará cada página una gran cantidad de veces. Sin embargo, es probable que visite un sitio popular miles de veces más que uno impopular. Tenga en cuenta que si no tuviera el botón "sorprenderme", se quedaría atascada en páginas que no tenían enlaces, páginas cuyos enlaces ya no apuntaban a ninguna página o páginas que formaban un bucle. Los enlaces que apuntan a los dos primeros tipos de páginas, o páginas que aún no se han rastreado, se denominan *enlaces colgantes.*

Ahora suponga que mientras Alice está navegando, usted entró en su habitación y miró la página web en su pantalla. ¿Cuál es la probabilidad de que esté mirando la página web de CNN cuando entre? Esa probabilidad es el PageRank de CNN. Cada página web en Internet tiene un PageRank, y está determinado de forma única por la estructura de enlaces de las páginas web. Como muestra este ejemplo, PageRank tiene la capacidad de distinguir entre páginas populares (aquellas con muchos enlaces entrantes o aquellas que tienen enlaces de páginas populares) y las impopulares. El valor de PageRank puede ayudar a los motores de búsqueda a examinar los millones de páginas que contienen la palabra "eBay" para encontrar la más popular (www.ebay.com).

Alice tendría que hacer clic en muchos miles de millones de enlaces para que podamos obtener una estimación razonable de PageRank, por lo que no podemos esperar calcularlo utilizando personas reales. Afortunadamente, podemos calcular el PageRank de una manera mucho más eficiente.

Supongamos por el momento que la Web consta de solo tres páginas, *A, B,* y *C.* Supondremos que esa pagina *A* enlaces a páginas *B* y *C,* página *B* enlaces a la página *C,* y pagina *C* enlaces a la página *A,* como se muestra en la Figura 4.10.

El PageRank de la página *C,* cuál es la probabilidad de que Alice esté mirando esta página, dependerá del PageRank de las páginas *A* y *B.* Dado que Alice elige al azar entre los enlaces de una página determinada, si comienza en la página *A,* hay un 50% de probabilidad de que vaya a la página *C* (porque hay dos enlaces salientes). Otra forma de decir esto es que el PageRank de una página se divide uniformemente entre todos los

[23] El "botón sorpresa" hace que el modelo de surfista aleatorio sea un *ergódico* Cadena de Markov, que garantiza que el cálculo iterativo de PageRank convergerá.

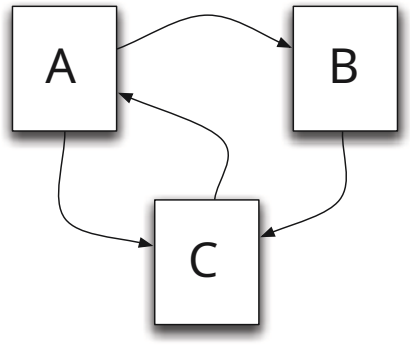


Figura 4.10. Un ejemplo de "Internet" que consta de solo tres páginas web. Las flechas indican vínculos entre las páginas.

enlaces salientes. Si ignoramos el botón "Sorpréndeme", esto significa que el Page Rank de la página *C,* representado como *PR (C),* se puede calcular como:

$$PR(C) = \frac{PR(A)}{2} + \frac{PR(B)}{1}$$

De manera más general, podríamos calcular la PageRank para cualquier página *tu* como:

$$PR(u) = \sum_{v \in B_{tu}} \frac{PR(v)}{L_v}$$

dónde B_{tu} es el conjunto de páginas que apuntan a *u,* y L_v es el número de enlaces salientes de la página *v* (sin contar los enlaces duplicados).

Hay un problema obvio aquí: no conocemos los valores de PageRank para las páginas, porque eso es lo que estamos tratando de calcular. Si comenzamos asumiendo que los valores de PageRank para todas las páginas son iguales (1/3 en este caso), entonces es fácil ver que podríamos realizar múltiples iteraciones del cálculo. Por ejemplo, en la primera iteración, *PR (C)* = 0,33 / 2 + 0,33 = 0,5, *PR (A)* = 0,33 y *PR (B)* = 0,17. En la siguiente iteración, *PR (C)* = 0,33 / 2 + 0,17 = 0,33, *PR (A)* = 0.5, y *PR (B)* = 0,17. En la tercera iteración, *PR (C)* = 0,42, *PR (A)* = 0,33 y *PR (B)* = 0,25. Después de algunas iteraciones más, los valores de PageRank convergen a los valores finales de *PR (C)* = 0,4, *PR (A)* = 0.4, y *PR (B)* = 0,2.

Si tenemos en cuenta el botón "sorpréndame", parte del PageRank de la página *C* será debido a la probabilidad de acceder a esa página presionando el botón. Dado que existe una probabilidad de 1/3 de ir a cualquier página cuando se presiona el botón,

la contribución al PageRank para C porque el botón será $\lambda/3$. Esto significa que el PageRank total para C es ahora:

$$PR(C) = \frac{\lambda}{3} + (1 - \lambda) \cdot \left(\frac{PR(A)}{2} + \frac{PR(B)}{1} \right)$$

Del mismo modo, la fórmula general para PageRank es: $\sum PR()$

$$PR(u) = \frac{\lambda}{norte} + (1 - \lambda) \cdot \sum_{v \in B_u} \frac{v}{L_v}$$

dónde $norte$ es el número de páginas que se están considerando. El valor típico para λ es 0,15.

Esto también se puede expresar como una ecuación matricial:

$$R = TR$$

dónde R es el vector de valores de PageRank y T es la matriz que representa el probabilidades de transición para el modelo de surfista aleatorio. El elemento T_{ij} representa la probabilidad de pasar de la página I a la página j, y:

$$T_{ij} = \frac{\lambda}{norte} + (1 - \lambda) \frac{1}{L_I}$$

Aquellos de ustedes familiarizados con el álgebra lineal pueden reconocer que la solución R es un *vector propio* de la matriz T.

La figura 4.11 muestra un pseudocódigo para calcular el PageRank. El algoritmo toma una gráfica *GRAMO* como entrada. Los gráficos se componen de vértices y aristas, por lo que $G = (V, E)$. En este caso, los vértices son páginas web y los bordes son enlaces, por lo que el pseudocódigo usa las letras *PAG* y *L* en lugar de. Un enlace se representa como un par (p, q), dónde *pag* y *q* son las páginas de origen y destino. Enlaces colgantes, que son enlaces donde la página *q* no existe, se supone que se eliminan. Las páginas sin enlaces salientes son *fregaderos de rango,* porque acumulan PageRank pero no lo distribuyen. En este algoritmo, asumimos que estas páginas enlazan con todas las demás páginas de la colección.

El primer paso es adivinar el valor de PageRank de cada página. Sin mejor información, asumimos que el PageRank es el mismo para cada página. Dado que los valores de PageRank deben sumar 1 para todas las páginas, asignamos un PageRank de $1 / |P|$ a cada página en el vector de entrada I. Una alternativa que puede producir una convergencia más rápida sería utilizar un valor basado en el número de inlinks.

```
1: procedimiento P R (GRAMO)
2:                  ◃ G es el gráfico web, que consta de vértices (páginas) y bordes (enlaces).
3:     (P, L) ← GRAMO                          ◃ Divida el gráfico en páginas y enlaces
4:     I ←un vector de longitud | P | ◃La estimación actual de PageRank
5:     R ←un vector de longitud | P | ◃La mejor estimación de PageRank resultante
6:     para todos entradas I_I ∈ I hacer
7:         I_I ← 1 / | P |                     ◃ Comience con cada página siendo igualmente probable
8:     final para
9:     tiempo R no ha convergido hacer
10:        para todos entradas R_I ∈ R hacer
11:            R_I ← λ / | P |                 ◃ Cada página tiene un λ / | P | posibilidad de selección aleatoria
12:        final para
13:        para todos paginas pag ∈ PAG hacer
14:            Q ← el conjunto de páginas tal que (p, q) ∈ L y q ∈ PAG
15:            si | Q | >0 luego
dieciséis:         para todos paginas q ∈ Q hacer
17:                R_q ← R_q + (1 -λ) yo_p / | Q | ◃Probabilidad I_pag de estar en la pagina pag
18:            final para
19:        demás
20:            para todos paginas q ∈ PAG hacer
21:                R_q ← R_q + (1 - λ) yo_p / | P |
22:            final para
23:        terminara si
24:            I ← R                           ◃ Actualice nuestra estimación actual de PageRank
25:        final para
26:    terminar mientras
27:    regreso R
28: procedimiento final
```

Figura 4.11. Pseudocódigo para el algoritmo iterativo de PageRank

En cada iteración, comenzamos creando un vector de resultado, R, y almacenando $\lambda / | P |$ en cada entrada. Esta es la probabilidad de aterrizar en una página en particular debido a un salto aleatorio. El siguiente paso es calcular la probabilidad de aterrizar en una página debido a un enlace en el que se hace clic. Lo hacemos iterando sobre cada página web en PAG. A cada página, recuperamos la probabilidad estimada de estar en esa página, I_{pag}. Desde esa página, el usuario tiene un λ posibilidad de saltar al azar, o $1 - \lambda$ de hacer clic en un Enlace. Hay $| Q |$ enlaces para elegir, por lo que la probabilidad de saltar a una página $q \in Q$ es $(1-\lambda) yo_{p / | Q |}$. Agregamos esta cantidad a cada entrada R_q. En caso de que

no hay enlaces salientes utilizables, asumimos que el usuario salta al azar, y por lo tanto, la probabilidad (1 - λ) yo_{pag} se distribuye uniformemente entre todos $|P|$ páginas.

En resumen, PageRank es un ejemplo importante de metadatos independientes de consultas datos que pueden mejorar la clasificación para la búsqueda web. Las páginas web tienen el mismo valor de PageRank independientemente de la consulta que se esté procesando. Los motores de búsqueda que utilizan PageRank preferirán páginas con valores de PageRank altos en lugar de asumir que todas las páginas web tienen la misma probabilidad de satisfacer una consulta. Sin embargo, el PageRank no es tan importante en la búsqueda web como sostiene la sabiduría convencional. Es solo una de las muchas características que se utilizan en la clasificación. Sin embargo, tiende a tener el mayor impacto en las consultas populares, lo cual es una propiedad útil.

La *GOLPES*[24] El algoritmo (Kleinberg, 1999) para el análisis de enlaces se desarrolló aproximadamente al mismo tiempo que PageRank y también ha sido muy influyente. Este algoritmo estima el valor del contenido de una página (el*autoridad* valor) y el valor de los enlaces a otras páginas (el *centro* valor). Ambos valores se calculan utilizando un algoritmo iterativo basado únicamente en la estructura del enlace, similar a PageRank. El algoritmo HITS, a diferencia de PageRank, calcula los valores de autoridad y de concentrador para un subconjunto de páginas recuperadas por una consulta determinada.[25] Esto puede ser una ventaja en términos del impacto de los metadatos HITS en la clasificación, pero puede ser computacionalmente inviable para motores de búsqueda con alto tráfico de consultas. En el Capítulo 10, discutimos la aplicación del algoritmo HITS para encontrar comunidades web.

4.5.3 Calidad del enlace

Es bien sabido que técnicas como PageRank y extracción de texto de anclaje se utilizan en motores de búsqueda comerciales, por lo que los diseñadores de páginas web sin escrúpulos pueden intentar crear enlaces inútiles solo para mejorar la ubicación del motor de búsqueda de una de sus páginas web. Se llama *enlace de spam.* Sin embargo, incluso los usuarios habituales pueden engañar sin saberlo a las técnicas sencillas de los motores de búsqueda. Un buen ejemplo de esto son los blogs.

Muchas publicaciones de blog son comentarios sobre otras publicaciones de blog. Supongamos que el autor*A* lee una publicación llamada *B* en autor *B*blog de. Autor*A* podría escribir una nueva publicación de blog, llamada *a,* que contiene un enlace para publicar *B.* En proceso de publicación, autor *A*puede publicar un *rastrear* publicar *B* en autor *B*blog de. Un trackback es un tipo especial de comentario que alerta al autor*B* que se ha publicado una respuesta en el autor *A*blog de.

---

[24] Búsqueda de temas inducida por hipertexto

[25] Las versiones independientes de consultas de HITS y las versiones dependientes del tema de PageRank tienen también se ha definido.

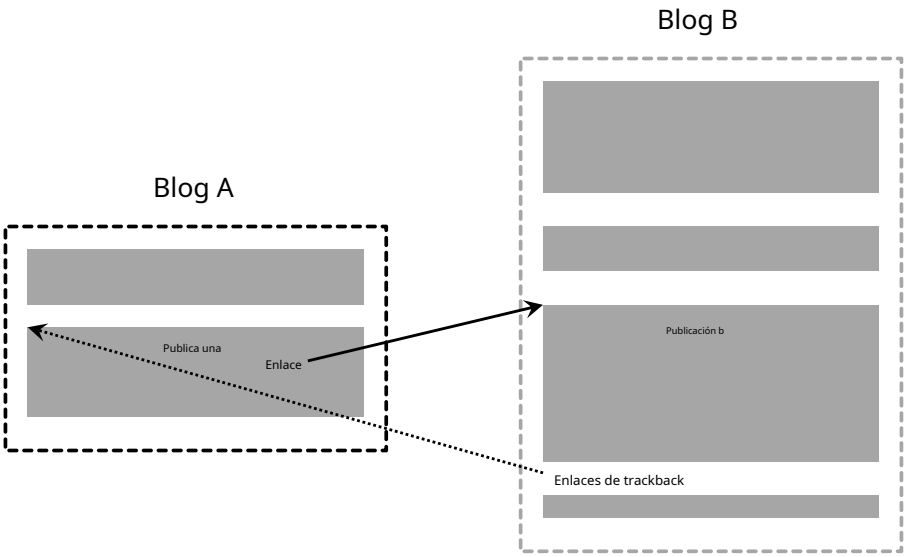


Figura 4.12. Enlaces de seguimiento en publicaciones de blogs

Como muestra la Figura 4.12, se ha desarrollado un ciclo entre *a* y publicar *B.* Correo *a* enlaces para publicar *B,* y publicar *B* contiene un enlace de trackback para publicar *una.* Intuitivamente diríamos que post *B* es influyente, porque el autor *A* ha decidido escribir sobre ello. Sin embargo, desde la perspectiva de PageRank,*a* y *B* tienen vínculos entre sí y, por lo tanto, ninguno es más influyente que el otro. El problema aquí es que un trackback es un tipo de enlace fundamentalmente diferente al que aparece en una publicación.

La sección de comentarios de un blog también puede ser una fuente de spam de enlaces. Los autores de páginas pueden intentar promocionar sus propios sitios web publicando enlaces a ellos en la sección de comentarios de blogs populares. Basándonos en nuestra discusión sobre PageRank, sabemos que un enlace de un sitio web popular puede hacer que otro sitio web parezca mucho más importante. Por lo tanto, esta sección de comentarios es un objetivo atractivo para los spammers.

En este caso, una solución es que las empresas de motores de búsqueda detecten automáticamente estas secciones de comentarios e ignoren efectivamente los enlaces durante la indexación. Una forma aún más sencilla de hacerlo es pedir a los propietarios de sitios web que modifiquen los enlaces sin importancia para que los motores de búsqueda puedan detectarlos. Este es el propósito detrás de larel = nofollow atributo de enlace.

La mayor parte del software de blogs está ahora diseñado para modificar cualquier enlace en un comentario de blog para que contenga la rel = nofollow atributo. Por lo tanto, una publicación como esta:

Ven a visitar mi página web .

se convierte en algo como esto:

Ven a visitar mi página web .

El enlace sigue apareciendo en el blog, pero los motores de búsqueda están diseñados para ignorar todos los enlaces marcados. rel = nofollow. Esto ayuda a preservar la integridad del cálculo de PageRank y la recolección de texto de anclaje.

4.6 Extracción de información

Extracción de información es una tecnología del lenguaje que se enfoca en extraer la estructura del texto. La extracción de información se utiliza en una serie de aplicaciones, y en particular para*minería de datos de texto.* Para las aplicaciones de búsqueda, el uso principal de la extracción de información es identificar las características que el motor de búsqueda puede utilizar para mejorar la clasificación. Algunas personas han especulado que las técnicas de extracción de información podrían eventualmente transformar la búsqueda de texto en un problema de base de datos al extraer toda la información importante del texto y almacenarla en forma estructurada, pero las aplicaciones actuales de estas técnicas están muy lejos de lograrlo. objetivo.

Algunos de los pasos de procesamiento de texto que ya hemos discutido podrían considerarse extracción de información. Identificar frases nominales, títulos o incluso texto en negrita son ejemplos. En cada uno de estos casos, se ha reconocido que una parte del texto tiene alguna propiedad especial, y esa propiedad puede describirse utilizando un lenguaje amarkup, como XML. Si un documento ya se describe mediante HTML o XML, el reconocimiento de algunas de las características estructurales (como los títulos) es sencillo, pero otras, como las frases, requieren un procesamiento adicional antes de que la característica pueda ser*anotado* utilizando el lenguaje de marcado. En algunas aplicaciones, como cuando los documentos de la colección se ingresan a través de OCR, el documento no tiene marcado e incluso las estructuras simples como los títulos deben reconocerse y anotarse.

Estos tipos de características son muy generales, pero la mayor parte de la investigación reciente sobre extracción de información se ha centrado en características que tienen un contenido semántico específico, como *entidades nombradas, relaciones,* y *eventos.* Aunque todas estas funciones contienen información importante, *reconocimiento de entidad nombrada* se ha utilizado con mayor frecuencia en aplicaciones de búsqueda. Una entidad nombrada es una palabra o secuencia de palabras que se usa para referirse a algo de interés en una aplicación en particular. Lo mas

ejemplos comunes son nombres de personas, nombres de empresas u organizaciones, ubicaciones, expresiones de fecha y hora, cantidades y valores monetarios. Es fácil encontrar otras "entidades" que serían importantes para aplicaciones específicas. Para una aplicación de comercio electrónico, por ejemplo, el reconocimiento de nombres de productos y números de modelo en páginas web y reseñas sería esencial. En una aplicación farmacéutica, el reconocimiento de los nombres de los medicamentos, las dosis y las condiciones médicas puede ser importante. Dada la naturaleza más específica de estas características, el proceso de reconocerlas y etiquetarlas en el texto a veces se denomina *anotación semántica.*
Algunas de estas entidades reconocidas se incorporarían directamente a la búsqueda utilizando, por ejemplo, *facetas* consulte el Capítulo 6), mientras que otros pueden utilizarse como parte de la exploración de los resultados de la búsqueda. Un ejemplo de esto último es la función del motor de búsqueda que reconoce las direcciones en las páginas y proporciona enlaces al mapa apropiado.

Fred Smith, que vive en 10 Water Street, Springfield, MA, es un coleccionista de **pez tropical.**

 <PersonName> <GivenName> Fred </GivenName> <Sn> Smith </Sn> </ PersonName>, que vive en <address> <Street> 10 Water Street </Street>, <City> Springfield </ City>, <State>MA</State> </address>, es un recolector desde hace mucho tiempo de peces tropicales.

Figura 4.13. Texto etiquetado por extracción de información

La figura 4.13 muestra una oración y el marcado XML correspondiente después de utilizar la extracción de información. En este caso, la extracción se realizó mediante un conocido programa de procesamiento de textos.[26] Además del marcado de estructura habitual (y), Se han agregado varias etiquetas que indican qué palabras forman parte de entidades nombradas. Muestra, por ejemplo, que una dirección que consta de una calle ("10 Water Street"), una ciudad ("Springfield") y un estado ("MA") fue reconocida en el texto.

Se han utilizado enfoques de dos dominios para construir reconocedores de entidades con nombre: basados en reglas y estadísticos. Un reconocedor basado en reglas usa uno o más *léxicos* listas de palabras y frases) que categorizan nombres. Algunas categorías de ejemplo serían ubicaciones (por ejemplo, pueblos, ciudades, estados, países, lugares de interés), nombres de personas (nombres de pila, apellidos) y organizaciones (por ejemplo, empresas, gobierno

[26] Microsoft Word

agencias, grupos internacionales). Si estas listas son lo suficientemente completas, gran parte de la extracción se puede realizar simplemente mediante una búsqueda. En muchos casos, sin embargo, se utilizan reglas o patrones para verificar el nombre de una entidad o para encontrar entidades que no están en las listas. Por ejemplo, un patrón como "<número> <palabra> calle"Podría usarse para identificar direcciones de calles. Patrones como "<dirección postal>, <ciudad>" o "en <ciudad>"Podría usarse para verificar que el nombre que se encuentra en el léxico de ubicación como ciudad era efectivamente una ciudad. Del mismo modo, un patrón como "<dirección postal>, <ciudad>, <estado>"También podría usarse para identificar nuevas ciudades o pueblos que no estaban en el léxico. Los nuevos nombres de personas pueden reconocerse mediante reglas como "<título> <nombre>", Donde <título> incluiría palabras como "presidente", "señor" y "director ejecutivo". Los nombres son generalmente más fáciles de extraer en texto en mayúsculas y minúsculas, porque las mayúsculas a menudo indican un nombre, pero muchos patrones se aplicarán también a todo el texto en minúsculas o mayúsculas. Las reglas que incorporan patrones se desarrollan manualmente, a menudo por ensayo y error, aunque un conjunto inicial de reglas también se puede utilizar como *semillas* en un proceso de aprendizaje automatizado que puede descubrir nuevas reglas.[27]

Un reconocedor de entidad estadística utiliza un modelo probabilístico de las palabras dentro y alrededor de una entidad. Se han utilizado varios enfoques diferentes para construir estos modelos, pero debido a su importancia, describiremos brevemente los *Modelo HiddenMarkov (*HMM) enfoque. Los HMM se utilizan para muchas aplicaciones en el habla y el lenguaje. Por ejemplo, los etiquetadores POS se pueden implementar utilizando este enfoque.

4.6.1 Modelos de Markov ocultos para extracción

Una de las partes más difíciles de la extracción de entidades es que las palabras pueden tener muchos significados diferentes. La palabra "arbusto", por ejemplo, puede describir una planta o una persona. Del mismo modo, "Maratón" podría ser el nombre de una carrera o un lugar en Grecia. La gente dice la diferencia entre estos diferentes significados basados en el *contexto* de la palabra, es decir, las palabras que la rodean. Por ejemplo, si "Maratón" va precedido de "Boston", es casi seguro que el texto describa una carrera. Podemos describir el contexto de una palabra matemáticamente modelando el *Generacion*[28] de la secuencia de palabras en un texto como un proceso con el *Propiedad de Markov,* lo que significa que la siguiente palabra en la secuencia depende solo de un pequeño número de las palabras anteriores.

---

[27] GATE (http://gate.ac.uk) es un ejemplo de un conjunto de herramientas de código abierto que proporciona tanto un componente de extracción de información y un entorno para personalizar la extracción para una aplicación específica.

[28] Discutimos *modelos generativos* con más detalle en el Capítulo 7.

Más formalmente, un *MarkovModelo* describe un proceso como una colección de *estados* con *transiciones* entre ellos. Cada una de las transiciones tiene una probabilidad asociada. El siguiente estado del proceso depende únicamente del estado actual y las probabilidades de transición. En un HiddenMarkovModel, cada estado tiene un conjunto de posibles salidas que se pueden generar. Al igual que con las transiciones, cada salida también tiene una probabilidad asociada.

La figura 4.14 muestra un *diagrama de estado* que representa un modelo muy simple para la generación de frases que podría ser utilizado por un reconocedor de entidad con nombre. En este modelo, se supone que las palabras de una oración son parte del nombre de una entidad (en este caso, una persona, organización o ubicación) o no son parte de una. Cada una de estas categorías de entidad está representada por un estado, y después de cada palabra, el sistema puede permanecer en ese estado (representado por las flechas) o pasar a otro estado. Hay dos estados especiales que representan el inicio y el final de la oración. Asociado con cada estado que representa una categoría de entidad, hay una distribución de probabilidad de las probables secuencias de palabras para esa categoría.

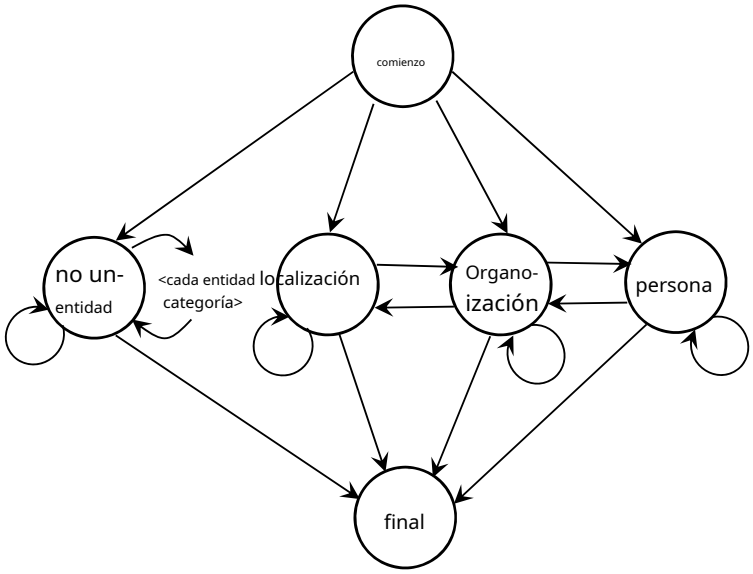


Figura 4.14. Modelo de oración para extractor de entidades estadísticas

Un posible uso de este modelo es construir nuevas oraciones. Supongamos que comenzamos en el estado de inicio, y luego el siguiente estado se elige al azar de acuerdo con

a la tabla de probabilidad de transición del estado inicial. Por ejemplo, podemos hacer la transición al estado de persona. Una vez que hemos entrado en el estado de la persona, completamos la transición eligiendo una salida de acuerdo con la distribución de probabilidad de salida del estado de la persona. Un resultado de ejemplo puede ser la palabra "Thomas". Este proceso continuaría, con la transición a un nuevo estado y la generación de una salida durante cada paso del proceso. El resultado final es un conjunto de estados y sus salidas asociadas.

Aunque estos modelos se pueden usar para generar nuevas oraciones, se usan más comúnmente para reconocer entidades en una oración. Para hacer esto para una oración dada, se encuentra una secuencia de categorías de entidad que da la mayor probabilidad para las palabras en esa oración. Solo las salidas generadas por las transiciones de estado son visibles (es decir, pueden observarse); los estados subyacentes están "ocultos". Para la oración de la figura 4.13, por ejemplo, el reconocedor encontraría la secuencia de estados

<start><name><not-an-entity><location><not-an-entity> <end>

tener la probabilidad más alta para ese modelo. Las palabras que estaban asociadas con las categorías de entidades en esta secuencia se etiquetarían. El problema de encontrar la secuencia de estados más probable en un HMM se resuelve mediante el *Viterbi algoritmo,*[29] el cual es un *programación dinámica* algoritmo.

El aspecto clave de este enfoque para el reconocimiento de entidades es que las probabilidades en el modelo de oración deben estimarse a partir de los datos de entrenamiento. Para estimar las probabilidades de transición y salida, generamos datos de entrenamiento que consisten en texto anotado manualmente con las etiquetas de entidad correctas. A partir de estos datos de entrenamiento, podemos estimar directamente la probabilidad de palabras asociadas con una categoría dada (es decir, probabilidades de salida) y la probabilidad de transiciones entre categorías. Para construir un reconocedor más preciso, las características que están altamente asociadas con entidades nombradas, como palabras en mayúscula y palabras que son todos dígitos, se incluirían en el modelo. Además, las probabilidades de transición podrían depender tanto de la palabra anterior como de la categoría anterior.[30] Por ejemplo, la aparición de la palabra "Sr." aumenta la probabilidad de que la siguiente categoría seaPersona.

Aunque estos datos de entrenamiento pueden ser útiles para construir HMM precisos, recopilarlos requiere una gran cantidad de esfuerzo humano. Para generar aproximadamente un millón de palabras de texto anotado, que es el tamaño aproximado de los datos de entrenamiento necesarios para obtener estimaciones precisas, las personas tendrían que anotar el equivalente de

[29] El nombre del ingeniero eléctrico Andrew Viterbi.

[30] Bikel y col. (1997) describen uno de los primeros reconocedores de entidades con nombre basado en la Enfoque HMM.

más de 1.500 noticias. Esto puede requerir un esfuerzo considerablemente mayor que desarrollar reglas para un conjunto simple de características. Tanto el enfoque estadístico como el basado en reglas tienen una efectividad de reconocimiento de alrededor del 90%.[31] para entidades como nombre, organización y ubicación, aunque los reconocedores estadísticos son generalmente los mejores. Otras categorías de entidades, como los nombres de productos, son considerablemente más difíciles. La elección de qué enfoque de reconocimiento de entidades utilizar dependerá de la aplicación, la disponibilidad del software y la disponibilidad de los anotadores.

Curiosamente, hay poca evidencia de que las entidades con nombre sean características útiles para aplicaciones de búsqueda generales. El reconocimiento de entidades con nombre es una parte fundamental de los sistemas de respuesta a preguntas (sección 11.5) y puede ser importante en motores de búsqueda verticales o específicos de un dominio para reconocer e indexar con precisión los términos del dominio. El reconocimiento de entidades con nombre también puede ser útil para el análisis de consultas en aplicaciones como la búsqueda local y como herramienta para comprender y examinar los resultados de la búsqueda.

4.7 Internacionalización

La Web se usa en todo el mundo, y no solo por hablantes de inglés. Aunque entre el 65% y el 70% de la Web está escrito en inglés, ese porcentaje sigue disminuyendo. Más de la mitad de las personas que utilizan la Web y, por lo tanto, realizan búsquedas en la Web, no utilizan el inglés como idioma principal. Otras aplicaciones de búsqueda, como la búsqueda en el escritorio y la búsqueda corporativa, se utilizan en muchos idiomas diferentes todos los días. Incluso una aplicación diseñada para un entorno que tiene principalmente usuarios de habla inglesa puede tener muchos documentos que no estén en inglés en la colección. Intente utilizar "poissons tropicaux" (peces tropicales) como una consulta para su motor de búsqueda web favorito y vea cuántas páginas web francesas se recuperan.[32]

A *monolingüe* motor de búsqueda es, como su nombre indica, un motor de búsqueda diseñado para un idioma en particular.[33] Muchas de las técnicas de indexación y modelos de recuperación que discutimos en este libro funcionan bien para cualquier idioma. Las diferencias entre los idiomas que tienen el mayor impacto en el diseño de los motores de búsqueda están relacionadas con los pasos de procesamiento del texto que producen los términos del índice para la búsqueda.

---

[31] Con esto queremos decir que aproximadamente 9 de cada 10 de las entidades encontradas están identificadas con precisión, y se encuentran 9 de cada 10 de las entidades existentes. Consulte el Capítulo 8 para obtener detalles sobre las medidas de evaluación.

[32] Encontraría muchas más páginas web en francés, por supuesto, si utilizara una versión en francés del motor de búsqueda, como http://fr.yahoo.com.

[33] Discutimos *lenguaje cruzado* motores de búsqueda en la sección 6.4.

Como mencionamos en el capítulo anterior, la codificación de caracteres es un tema crucial para los motores de búsqueda que se ocupan de idiomas distintos del inglés, y Unicode se ha convertido en el estándar de codificación de caracteres predominante para la internacionalización del software.

También es necesario personalizar otros pasos de procesamiento de texto para diferentes idiomas. Ya se ha mencionado la importancia de la derivación para lenguajes con muchas inflexiones, pero cada idioma requiere una derivación personalizada. La creación de tokens también es importante para muchos idiomas, especialmente para la familia de idiomas CJK. Para estos idiomas, el problema clave es*segmentación de palabras,* donde las rupturas correspondientes a palabras o términos de índice deben identificarse en la secuencia continua de caracteres (generalmente no se usan espacios). Una alternativa a la segmentación es indexar bigramas de caracteres superpuestos (pares de caracteres, consulte la sección 4.3.5). Figura
4.15 muestra la segmentación de palabras y bigramas para el texto "impacto de las sequías en China". Aunque la eficacia de clasificación de la búsqueda basada en bigramas es bastante buena, en muchas aplicaciones se prefiere la segmentación de palabras porque muchos de los bigramas no se corresponden con palabras reales. Se puede implementar una técnica de segmentación basada en enfoques estadísticos, como aHiddenMarkovModel, con suficientes datos de entrenamiento. La segmentación también puede ser un problema en otros idiomas. El alemán, por ejemplo, tiene muchas palabras compuestas (como “fischzuchttechniken” para “técnicas de cultivo de peces”) que deben segmentarse para su indexación.

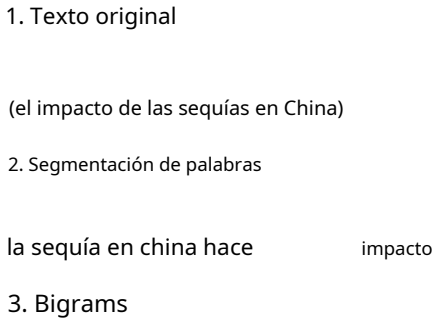


Figura 4.15. Segmentación china y bigramas

En general, dadas las herramientas disponibles, no es difícil construir un motor de búsqueda para los principales idiomas. La misma afirmación es válida para cualquier idioma que

tiene una cantidad significativa de texto en línea disponible en la Web, ya que esto se puede usar como un recurso para construir y probar los componentes del motor de búsqueda. Sin embargo, hay una gran cantidad de otros idiomas denominados de “baja densidad” que pueden tener muchos hablantes pero pocos recursos en línea. Crear motores de búsqueda eficaces para estos idiomas es un desafío mayor.

Referencias y lecturas adicionales

Las propiedades y estadísticas de las colecciones de textos y documentos se han estudiado durante algún tiempo bajo el título de *bibliometría,* que es parte del campo de *bibliotecas y ciencias de la información.* Revistas de ciencias de la información como la *Diario de la Sociedad Estadounidense de Ciencia y Tecnología de la Información (*JASISTA) o *Información Procesamiento y Gestión (*IPM) contienen muchos artículos en esta área general. La recuperación de información ha enfatizado, desde el principio, una visión estadística del texto, y los investigadores de las RI y las ciencias de la información siempre han trabajado en estrecha colaboración. Belew (2000) contiene una buena discusión de los aspectos cognitivos de la ley de Zipf y otras propiedades del texto en relación con las RI. Con el cambio a los métodos estadísticos en la década de 1990, *procesamiento natural del lenguaje* Los investigadores también se interesaron en estudiar las propiedades estadísticas del texto. Manning y Schütze (1999) es un buen resumen de las estadísticas de texto desde esta perspectiva. Ha et al. (2002) dan un resultado interesante que muestra que las frases (o n-gramas) también generalmente siguen la ley de Zipf, y que la combinación de frases y palabras da como resultado mejores predicciones para las frecuencias en rangos bajos.

El artículo de Anagnostopoulos et al. (2005) describe una técnica para estimar el tamaño del resultado de la consulta y también señala gran parte de la literatura relevante en esta área. Del mismo modo, Broder et al. (2006) muestran cómo estimar el tamaño del corpus y comparar su estimación con técnicas anteriores.

No se ha escrito mucho sobre tokenizar o detener. Ambos se consideran lo suficientemente “bien conocidos” como para que apenas se mencionen en los artículos. Sin embargo, como hemos señalado, hacer bien estos pasos básicos es crucial para la efectividad del sistema en general. Durante muchos años, los investigadores utilizaron la lista de palabras vacías publicada en van Rijsbergen (1979). Cuando quedó claro que esto no era suficiente para las colecciones más grandes de TREC, con frecuencia se utilizó una lista de palabras vacías desarrollada en la Universidad de Massachusetts y distribuida con el juego de herramientas Lemur. Como se mencionó anteriormente, esta lista contiene más de 400 palabras, lo que será demasiado largo para muchas aplicaciones.

El artículo original que describe el tallador de Porter fue escrito en 1979, pero fue reimpreso en Porter (1997). El artículo de Krovetz (1993) describe su algoritmo de derivación, pero también adopta un enfoque más detallado para estudiar el papel de la morfología en una derivación.[34] El tallador Krovetz está disponible en el sitio web de Lemur. Hay varios sitios web disponibles en Stemmers para otros idiomas (incluido el sitio web de Lemur y el sitio web de Porter Stemmer). Se puede encontrar una descripción de las técnicas de derivación árabe en Larkey et al. (2002).

La investigación sobre el uso de frases en la búsqueda tiene una larga historia. Croft y col. (1991) describen experimentos de recuperación con frases derivadas tanto del procesamiento sintáctico como estadístico de la consulta, y mostraron que la efectividad era similar a las frases seleccionadas manualmente. Muchos grupos que han participado en las evaluaciones TREC han utilizado frases como parte de sus algoritmos de búsqueda (Voorhees & Harman). , 2005).

Church (1988) describió un enfoque para construir una estadística (o *estocástico)* etiquetador de parte del discurso que es la base de muchos etiquetadores actuales. Este enfoque utiliza datos de entrenamiento etiquetados manualmente para entrenar un modelo probabilístico de secuencias de partes del habla, así como la probabilidad de una parte del habla para una palabra específica. Para una oración dada, se usa el etiquetado de la parte del discurso que da la mayor probabilidad para la oración completa. Este método es esencialmente el mismo que el utilizado por un extractor de entidad estadística, siendo los estados partes del discurso en lugar de categorías de entidad. El etiquetador Brill (Brill, 1994) es un enfoque alternativo popular que utiliza reglas que se aprenden automáticamente a partir de datos etiquetados. Manning y Schütze (1999) proporcionan una buena descripción general de los métodos de etiquetado de partes del discurso.

Se pueden encontrar muchas variaciones de PageRank en la literatura. Muchas de estas variaciones están diseñadas para ser más eficientes en el cálculo o se utilizan en diferentes aplicaciones. La versión dependiente del tema de PageRank se describe en Haveliwala (2002). Tanto PageRank como HITS tienen sus raíces en los algoritmos de análisis de citas desarrollados en el campo de la bibliometría.

La idea de mejorar la representación de un documento de hipertexto (es decir, una página web) utilizando el contenido de los documentos que apuntan a él existe desde hace algún tiempo. Por ejemplo, Croft y Turtle (1989) describen un modelo de recuperación basado en la incorporación de texto de documentos de hipertexto relacionados, y Dunlop y van Rijsbergen (1993) describen cómo los documentos con poco contenido de texto (como los que contienen imágenes) podrían recuperarse utilizando el texto en enlaces enlazados. documentos. Re-

[34] La morfología es el estudio de la estructura interna de las palabras, y la derivación es una forma de *procesamiento morfológico.*

McBryan (1994) mencionó por primera vez la restricción del texto que se incorpora al texto de anclaje asociado con los enlaces de entrada. Se ha demostrado que el texto de anclaje es esencial para algunas categorías de búsqueda web en las evaluaciones de TREC, como en Ogilvie y Callan (2003).

Se han desarrollado técnicas para aplicar el análisis de enlaces en colecciones sin una estructura de enlace explícita (Kurland y Lee, 2005). En este caso, los vínculos se basan en similitudes entre el contenido de los documentos, calculadas mediante una medida de similitud como la correlación del coseno (ver Capítulo 7).

Las técnicas de extracción de información se desarrollaron principalmente en programas de investigación como TIPSTER y MUC (Cowie & Lehnert, 1996). El uso de la extracción de entidades nombradas para proporcionar características adicionales para la búsqueda también se estudió al principio de las evaluaciones de TREC (Callan et al., 1992, 1995). Uno de los sistemas de extracción de información basados en reglas más conocidos es FASTUS (Hobbs et al., 1997). El sistema BBN Identifinder (Bikel et al., 1999), que se basa en un HMM, se ha utilizado en muchos proyectos.

Se puede encontrar una descripción detallada de los HMM y el algoritmo de Viterbi en Manning y Schütze (1999). McCallum (2005) proporciona una descripción general de la extracción de información, con referencias a avances más recientes en el campo. Modelos estadísticos que incorporan características más complejas que los HMM, como *campos aleatorios condicionales,* se han vuelto cada vez más populares para la extracción (Sutton & McCal- lum, 2007).

En Wikipedia se pueden encontrar descripciones detalladas de todos los esquemas de codificación principales. Fujii y Croft (1993) fue uno de los primeros artículos que discutió los problemas del procesamiento de texto para búsquedas con lenguajes CJK. Un diario completo *ACMTransactions onAsianLanguage Information Processing,*[35] ahora ha sido votó a favor de esta cuestión. Peng y col. (2004) describen un modelo estadístico para la segmentación de palabras chinas y dan referencias a otros enfoques.

Ejercicios

4.1. Trace curvas de frecuencia de rango (usando un gráfico log-log) para palabras y bigramas en la colección de Wikipedia disponible a través del sitio web del libro (http://www.searchengines-book.com). Trace una curva para la combinación de los dos. ¿Cuáles son los mejores valores para el parámetro? *C* para cada curva?

[35] http://talip.acm.org/

4.2. Trace el crecimiento del vocabulario para la colección de Wikipedia y estime los parámetros de la ley de Heaps. ¿Debe marcar alguna diferencia el orden en que se procesan los documentos?

4.3. Intente estimar el número de páginas web indexadas por dos motores de búsqueda diferentes utilizando la técnica descrita en este capítulo. Compare las estimaciones de tamaño de una variedad de consultas y discuta la consistencia (o la falta de ella) de estas estimaciones.

4.4. Modifique el tokenizador Galago para manejar apóstrofos o puntos de una manera diferente. Describe las nuevas reglas que implementa tu tokenizador. Dé ejemplos de dónde el nuevo tokenizador hace un mejor trabajo (en su opinión) y ejemplos de dónde no lo hace.

4.5. Examine la lista de palabras vacías de Lemur y enumere 10 palabras que crea que causarían problemas para algunas consultas. Dé ejemplos de estos problemas.

4.6. Procese cinco documentos de Wikipedia con la lectora Porter y la lectora Krovetz. Compare el número de tallos producidos y encuentre 10 ejemplos de diferencias en el derivado que podrían tener un impacto en la clasificación.

4.7. Utilice el etiquetador GATE POS para etiquetar un documento de Wikipedia. Defina una regla o reglas para identificar frases y mostrar las 10 frases más frecuentes. Ahora use el etiquetador POS en las consultas de Wikipedia. ¿Hay algún problema con las frases identificadas?

4.8. Encuentre los 10 documentos de Wikipedia con más inlinks. Muestre la colección de texto de anclaje para esas páginas.

4.9. Calcule PageRank para los documentos de Wikipedia. Enumere los 20 documentos con los valores de PageRank más altos junto con los valores.

4.10. La figura 4.11 muestra un algoritmo para calcular PageRank. Demuestre que las entradas del vector I suma a 1 cada vez que el algoritmo ingresa al ciclo en línea 9.

4.11. Implemente un reconocedor basado en reglas para ciudades (puede elegir un subconjunto de ciudades para facilitar esto). Cree una colección de prueba que pueda escanear manualmente para encontrar las ciudades mencionadas en el texto y evaluar su reconocedor. Resuma el desempeño del reconocedor y discuta ejemplos de fallas.

4.12. Cree una pequeña colección de pruebas en algún idioma que no sea el inglés utilizando páginas web. Realice los pasos básicos de procesamiento de texto de tokenizar, derivar y dejar de usar herramientas del sitio web de libros y de otros sitios web. Muestre ejemplos de la representación de términos de índice de los documentos.