Web Robots

ما تور خود را در معماری HTTP با نگاهی دقیق به Self-Animating User Agents (عوامل کاربر خود متحرک) به نام Web Robots ادامه می‌دهیم.

Web Robotها برنامه‌های نرم افزاری هستند که مجموعه‌ای از تراکنش‌های وب را بدون تعامل انسانی خودکار می‌کنند. بسیاری از ربات‌ها از وب‌سایتی به وب‌سایت دیگر سرگردان هستند، محتوا را واکشی می‌کنند، لینک‌ها را دنبال می‌کنند و داده‌هایی را که پیدا می‌کنند پردازش می‌نمایند. به این نوع ربات‌ها نام‌های رنگارنگی مانند «crawlers»، «spiders»، «worms» و «bots» داده می‌شود، زیرا به‌طور خودکار وب‌سایت‌ها را کاوش می‌کنند.

در اینجا چند نمونه از ربات‌های وب آورده شده است:

ربات‌های Stock-graphing، HTTP GETها را هر چند دقیقه به سرورهای بازار سهام صادر می‌کنند و از داده‌ها برای ساخت نمودارهای روند قیمت سهام استفاده می‌کنند.

ربات‌های Web-census، اطلاعات «سرشماری» را در مورد مقیاس و تکامل شبکه جهانی وب جمع‌آوری می‌کنند. آن‌ها در وب پرسه می‌زنند و تعداد صفحات را می‌شمارند و اندازه، زبان و نوع رسانه هر صفحه را ثبت می‌کنند.

ربات‌های Search-engine، تمام اسنادی را که پیدا می‌کنند جمع آوری می‌کنند تا پایگاه داده‌های جستجو را ایجاد کنند.

ربات‌های Comparison-shopping صفحات وب را از کاتالوگ فروشگاه‌های آنلاین جمع‌آوری می‌کنند تا پایگاه داده‌ای از محصولات و قیمت‌های آن‌ها بسازند.

Crawlers and Crawling

Web Crawler ها ربات‌هایی هستند که به صورت بازگشتی از وب‌ها عبور می‌کنند، ابتدا یک صفحه وب، سپس تمام صفحات وب را که آن صفحه به آن‌ها اشاره می‌کند، سپس تمام صفحات وب را که آن صفحات به آن‌ها اشاره می‌کنند، واکشی می‌کنند. هنگامی که یک ربات به صورت بازگشتی پیوندهای وب را دنبال می‌کند، به آن Crawler یا Spider می‌گویند.

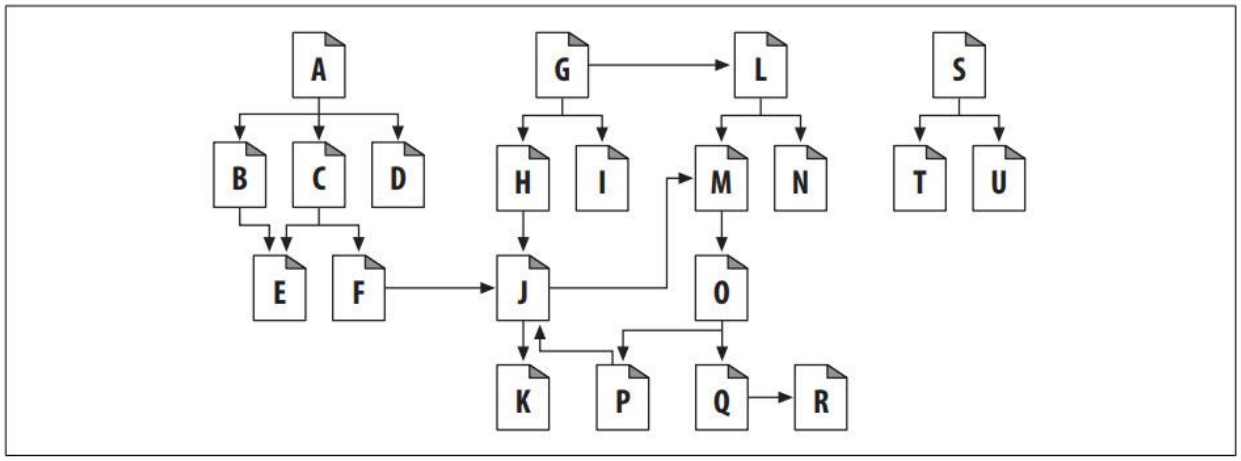
موتورهای جستجوی اینترنتی از Crawler ها برای پرسه زدن در وب استفاده می‌کنند و تمام اسنادی را که با آن‌ها مواجه می‌شوند پس می‌کشند. سپس این اسناد برای ایجاد یک پایگاه داده قابل جستجو

پردازش می‌شوند و به کاربران امکان می‌دهد اسنادی را که حاوی کلمات خاصی هستند پیدا کنند. با وجود میلیاردها صفحه وب برای یافتن و بازگرداندن، این Spider های موتور جستجو لزوماً برخی از پیچیده‌ترین ربات‌ها هستند. بیایید با جزئیات بیشتری به نحوه کار Crawler ها نگاه کنیم.

Where to Start: The Root Set

قبل از اینکه بتوانید Crawler گرسنه خود را آزاد کنید، باید به آن نقطه شروع بدهید. مجموعه اولیه URL هایی که Crawler شروع به بازدید از آن‌ها می کند، Root Set نامیده می‌شود. هنگام انتخاب یک Root Set، باید URL ها را از مکان‌های مختلف انتخاب کنید که همه پیوندها را Crawl نموده و در نهایت شما را به بیشتر صفحات وب مورد علاقه تان برساند.

یک Root Set خوب برای Crawl در وب در شکل زیر چیست؟



در وب واقعی، هیچ سند واحدی وجود ندارد که در نهایت به هر سند پیوند دهد. اگر با سند A در شکل بالا شروع کنید، می‌توانید به B، C، و D، سپس به E و F، سپس به J، و سپس به K برسید. اما هیچ زنجیره‌ای از پیوندها از A به G یا از A به N وجود ندارد.

برخی از صفحات وب در این وب، مانند S، T، و U، تقریباً جدا شده اند، بدون اینکه هیچ پیوندی به آن‌ها اشاره کند. شاید این صفحات تنهایی جدید باشند و هنوز کسی آن‌ها را پیدا نکرده باشد. یا شاید واقعا قدیمی یا مبهم هستند.

به طور کلی، برای پوشش دادن بخش بزرگی از وب، به صفحات زیادی در Root Set نیاز ندارید. در شکل بالا، برای دسترسی به تمام صفحات فقط به A، G و S در Root Set نیاز دارید.

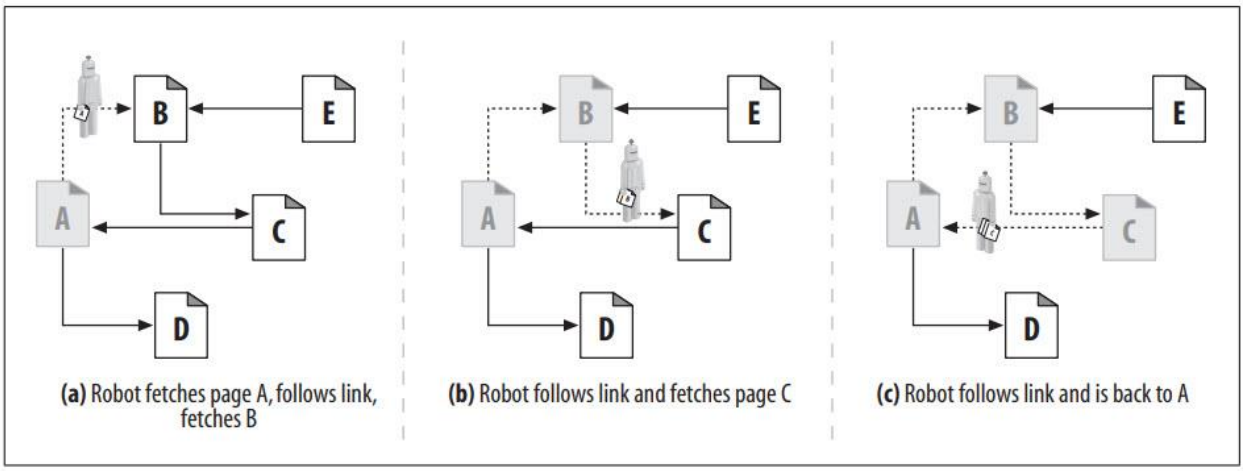
به طور معمول، یک Root Set خوب شامل وب سایت‌های بزرگ و محبوب (به عنوان مثال، http://www.yahoo.com)، لیستی از صفحات تازه ایجاد شده و لیستی از صفحات مبهم است که اغلب به آن‌ها پیوند داده نمی‌شود. بسیاری از Crawler های تولیدی در مقیاس بزرگ، مانند آن‌هایی که توسط موتورهای جستجوی اینترنتی استفاده می‌شوند، راهی برای کاربران دارند تا صفحات جدید یا مبهم را در مجموعه اصلی ارسال کنند. این Root Set در طول زمان رشد می‌کند و لیست دانه (Sedd List) برای هر خزیدن تازه است.

Extracting Links and Normalizing Relative Links

هنگامی که یک Crawler در وب حرکت می‌کند، دائماً صفحات HTML را بازیابی می‌کند. باید لینک‌های URL را در هر صفحه ای که بازیابی می‌کند تجزیه نموده و آن‌ها را به لیست صفحاتی که باید Crawl شوند اضافه کند. در حالی که Crawling در حال پیشرفت است، این لیست اغلب به سرعت گسترش می‌یابد، زیرا Crawler لینک‌های جدیدی را کشف می‌کند که باید کاوش شوند.

Cycle Avoidance

هنگامی که یک ربات یک وب را Crawl می‌کند، باید بسیار مراقب باشد که در یک حلقه یا چرخه گیر نکند. به Crawler در شکل زیر نگاه کنید:



در بخش a، ربات صفحه A را واکشی می‌کند، می‌بیند که A به B لینک دارد و صفحه B را واکشی می‌کند.

در بخش b، ربات صفحه B را واکشی می‌کند، می‌بیند که B به C لینک دارد و صفحه C را واکشی می‌کند.

در بخش c، ربات صفحه C را واکشی می‌کند و می‌بیند که C به A لینک دارد. اگر ربات دوباره صفحه A را واکشی کند، در یک چرخه به پایان می‌رسد و A، B، C، A، B، C، A را واکشی می‌کند.

ربات‌ها باید بدانند کجا بوده اند تا از چرخه دوری کنند. چرخه‌ها می‌توانند به تله‌های رباتی منجر شوند که می‌توانند پیشرفت ربات را متوقف یا کند نماید.

Loops and Dups

چرخه‌ها حداقل به سه دلیل برای Crawler ها بد هستند:

آن‌ها Crawler را وارد حلقه‌ای می‌کنند که می‌تواند در آن جا گیر کند. یک حلقه می‌تواند باعث شود یک Crawler با طراحی ضعیف به دور خود بچرخد و تمام وقت خود را صرف بارها و بارها واکشی صفحات مشابه کند. Crawler می‌تواند پهنای باند شبکه زیادی را مصرف نموده و ممکن است به طور کامل نتواند هیچ صفحه دیگری را واکشی نماید.

در حالی که Crawler زنده به طور مکرر صفحات مشابه را واکشی می‌کند، سرور وب در طرف دیگر در حال ضربه خوردن است. اگر Crawler به خوبی متصل باشد، می‌تواند وب سایت را تحت تأثیر قرار دهد و از دسترسی هر کاربر واقعی به سایت جلوگیری کند. چنین انکار سرویسی می‌تواند دلیلی برای دعاوی حقوقی باشد.

حتی اگر حلقه کردن به خودی خود مشکلی نداشته باشد، Crawler تعداد زیادی صفحه تکراری (اغلب "dups" نامیده شده که با "loops" هم قافیه می‌شود) واکشی می‌کند. برنامه Crawler مملو از محتوای تکراری می‌شود که ممکن است برنامه را بی فایده کند. نمونه‌ای از این مورد یک موتور جستجوی اینترنتی است که صدها مورد مشابه را دقیقاً در همان صفحه برمی‌گرداند.

Trails of Breadcrumbs

متأسفانه، ردیابی جایی که بوده‌اید همیشه آسان نیست. در زمان نگارش این مقاله، میلیاردها صفحه وب مجزا در اینترنت وجود دارد، بدون احتساب محتوای تولید شده از Gateway های پویا.

اگر می‌خواهید بخش بزرگی از محتوای وب جهان را Crawl کنید، باید برای بازدید از میلیاردها URL آماده باشید. ردیابی آدرس‌هایی که بازدید شده‌اند می‌تواند بسیار چالش برانگیز باشد. به دلیل تعداد زیاد URL ها، باید از ساختارهای داده پیچیده استفاده کنید تا به سرعت مشخص کنید کدام URL ها را بازدید کرده‌اید. ساختارهای داده باید در سرعت و استفاده از حافظه کارآمد باشند.

سرعت مهم است زیرا صدها میلیون URL به ساختارهای جستجوی سریع نیاز دارند. جستجوی جامع در لیست‌های URL قابل بحث نیست. حداقل، یک ربات باید از درخت جستجو یا جدول هش استفاده کند تا بتواند به سرعت تشخیص دهد که آیا URL بازدید شده است یا خیر.

صدها میلیون URL نیز فضای زیادی را اشغال می‌کنند. اگر میانگین URL برابر ۴۰ کاراکتر باشد و یک ربات وب ۵۰۰ میلیون URL (فقط بخش کوچکی از وب) را جستجو کند، یک ساختار داده جستجو می‌تواند به ۲۰ گیگابایت یا بیشتر حافظه فقط برای نگهداری URLها نیاز داشته باشد.

در اینجا چند تکنیک مفید وجود دارد که Crawler های وب در مقیاس بزرگ از آن‌ها برای مدیریت مکان بازدیدشان استفاده می‌کنند:

- Trees and hash tables

ربات‌های پیچیده ممکن است از درخت جستجو یا جدول هش برای پیگیری URL های بازدید شده استفاده کنند. این‌ها ساختارهای داده نرم افزاری هستند که جستجوی URL را بسیار سریعتر می‌کنند.

- Lossy presence bit maps

برای به حداقل رساندن فضا، برخی از Crawler های مقیاس بزرگ از ساختارهای داده با تلفات مانند آرایه‌های بیت حضوری (Presence Bit Arrays) استفاده می‌کنند. هر URL توسط یک تابع هش به یک عدد اندازه ثابت تبدیل می‌شود و این عدد دارای یک "presence bit" مرتبط در یک آرایه است. هنگامی که یک URL، Crawl شد، بیت حضور مربوطه تنظیم می‌شود. اگر بیت حضور از قبل تنظیم شده باشد، Crawler فرض می‌کند که URL قبلاً Crawl شده است.

- Checkpoints

مطمئن شوید که لیست URL های بازدید شده را در دیسک ذخیره کرده اید. (در صورتی که برنامه ربات خراب شود)

- Partitioning

همانطور که وب رشد می‌کند، ممکن است تکمیل Crawl با یک ربات منفرد روی یک کامپیوتر غیر عملی شود. آن رایانه ممکن است حافظه، فضای دیسک، قدرت محاسباتی یا پهنای باند شبکه کافی برای تکمیل Crawl نداشته باشد. برخی از ربات‌های وب در مقیاس بزرگ از "مزرعه" ربات‌ها استفاده می‌کنند که هر کدام یک کامپیوتر جداگانه هستند و پشت سر هم کار می‌کنند. به هر ربات یک "slice" خاص از URL ها اختصاص داده می‌شود که مسئولیت آن را بر عهده دارد. ربات ها با هم برای Crawl در وب کار می‌کنند. آن‌ها ممکن است نیاز به برقراری ارتباط داشته باشند تا URLها را به عقب و جلو ارسال کنند.

یک کتاب مرجع خوب برای پیاده سازی ساختارهای داده عظیم، Managing Gigabytes: Compressing and Indexing Documents and Images ، نوشته Witten و همکاران است. این کتاب پر از ترفندها و تکنیک‌های مدیریت حجم زیاد داده است.

Aliases and Robot Cycles

حتی با ساختارهای داده‌ای مناسب، گاهی اوقات تشخیص اینکه آیا قبلاً از صفحه‌ای بازدید کرده‌اید یا خیر، دشوار است، و این به دلیل URL Aliasing است. اگر URL ها متفاوت به نظر برسند اما واقعاً به یک منبع اشاره کنند، آن گاه دو URL اصطلاحاً Aliases هستند.

جدول زیر چند راه ساده را نشان می‌دهد که URL های مختلف می‌توانند به یک منبع اشاره کنند:

	First URL	Second URL	When aliased
a	*http://www.foo.com/bar.html*	*http://www.foo.com:80/bar.html*	Port is 80 by default
b	*http://www.foo.com/~fred*	*http://www.foo.com/%7Ffred*	%7F is same as ~
c	*http://www.foo.com/x.html#early*	*http://www.foo.com/x.html#middle*	Tags don't change the page
d	*http://www.foo.com/readme.htm*	*http://www.foo.com/README.HTM*	Case-insensitive server
e	*http://www.foo.com/*	*http://www.foo.com/index.html*	Default page is *index.html*
f	*http://www.foo.com/index.html*	*http://209.231.87.45/index.html*	*www.foo.com* has this IP address

Canonicalizing URLs

اکثر ربات‌های وب سعی می‌کنند تا نام مستعار آشکار را با «Canonicalizing» یا متعارف نمودن URLها به شکل استاندارد حذف کنند. یک ربات ممکن است ابتدا هر URL را صورت زیر به یک فرم متعارف تبدیل کند:

- اگر پورت مشخص نشده باشد، ":80" را به نام میزبان اضافه کنند.
- تبدیل همه %xx کاراکترهای Escape شده به معادل کاراکترهایشان
- حذف تگ‌های #

این مراحل می‌توانند مشکلات همخوانی نشان داده شده در بخش a-c جدول بالا را حذف کنند. اما، بدون دانستن اطلاعات مربوط به وب سرور خاص، ربات هیچ راه خوبی برای جلوگیری از تکرارهای بخش‌های d-f جدول ندارد:

- ربات باید بداند که آیا وب سرور به حروف بزرگ و کوچک حساس نیست تا از نام مستعار در بخش d جدول جلوگیری کند.
- ربات باید پیکربندی صفحه فهرست وب سرور را برای این دایرکتوری بداند تا بداند آیا URL های بخش e جدول نام مستعار هستند یا خیر.

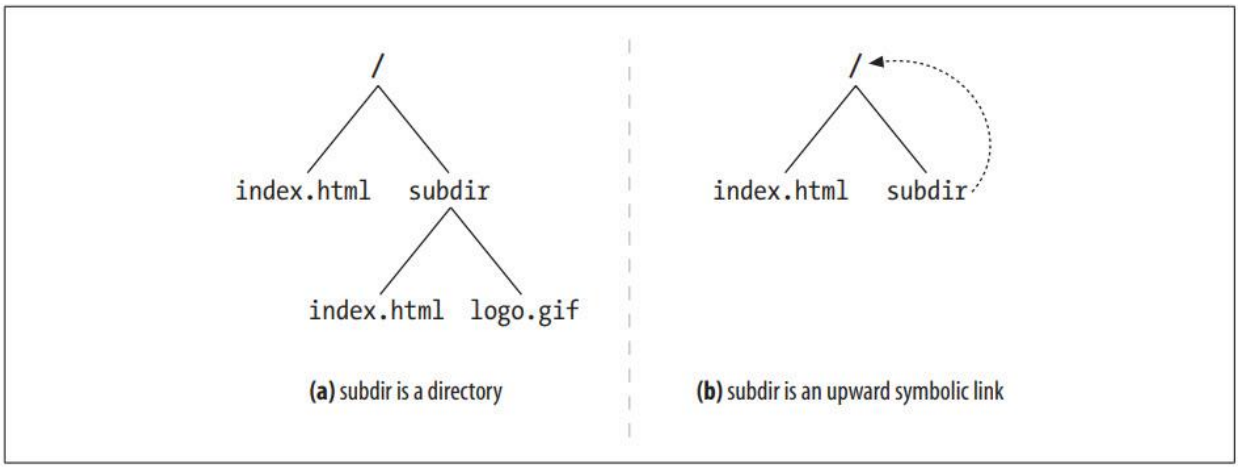
- ربات باید بداند که آیا وب سرور برای انجام Virtual Hosting پیکربندی شده است تا بداند آیا URL های بخش f جدول نام مستعار هستند، حتی اگر نام میزبان و آدرس IP مربوط به همان رایانه فیزیکی را بداند.

متعارف سازی URL می‌تواند Basic Syntactic Aliases را حذف کند، اما ربات‌ها با نام مستعار URL دیگری روبرو می‌شوند که از طریق تبدیل URL ها به فرم‌های استاندارد قابل حذف نیستند.

Filesystem Link Cycles

پیوندهای نمادین (Symbolic Links) در یک سیستم فایل می‌توانند نوعی چرخه به خصوص موذیانه ایجاد کنند، زیرا آن‌ها می‌توانند توهم یک سلسله مراتب دایرکتوری عمیق بی نهایت را ایجاد کنند که هیچ کدام وجود ندارد. چرخه‌های پیوند نمادین معمولاً نتیجه یک خطای ناخواسته توسط مدیر سرور هستند، اما می‌توانند توسط "وب مسترهای شرور" به عنوان یک دام مخرب برای روبات‌ها ایجاد شوند.

شکل زیر دو فایل سیستم را نشان می‌دهد.



در بخش a تصویر، subdir یک دایرکتوری معمولی است. در بخش b تصویر، subdir یک پیوند نمادین است که به / اشاره دارد. در هر دو شکل، فرض کنید فایل /index.html حاوی یک پیوند به فایل subdir/index.html است.

با استفاده از سیستم فایل بخش a تصویر، یک Crawler وب ممکن است اقدامات زیر را انجام دهد:

1 .GET http://www.foo.com/index.html

Get /index.html, find link to subdir/index.html.

2 .GET http://www.foo.com/subdir/index.html

Get subdir/index.html, find link to subdir/logo.gif.

3 .GET http://www.foo.com/subdir/logo.gif

Get subdir/logo.gif, no more links, all done

اما در فایل سیستم بخش b تصویر، موارد زیر ممکن است رخ دهد:

1. GET http://www.foo.com/index.html

Get /index.html, find link to subdir/index.html.

2. GET http://www.foo.com/subdir/index.html

Get subdir/index.html, but get back same index.html.

3. GET http://www.foo.com/subdir/subdir/index.html

Get subdir/subdir/index.html.

4. GET http://www.foo.com/subdir/subdir/subdir/index.html

Get subdir/subdir/subdir/index.html.

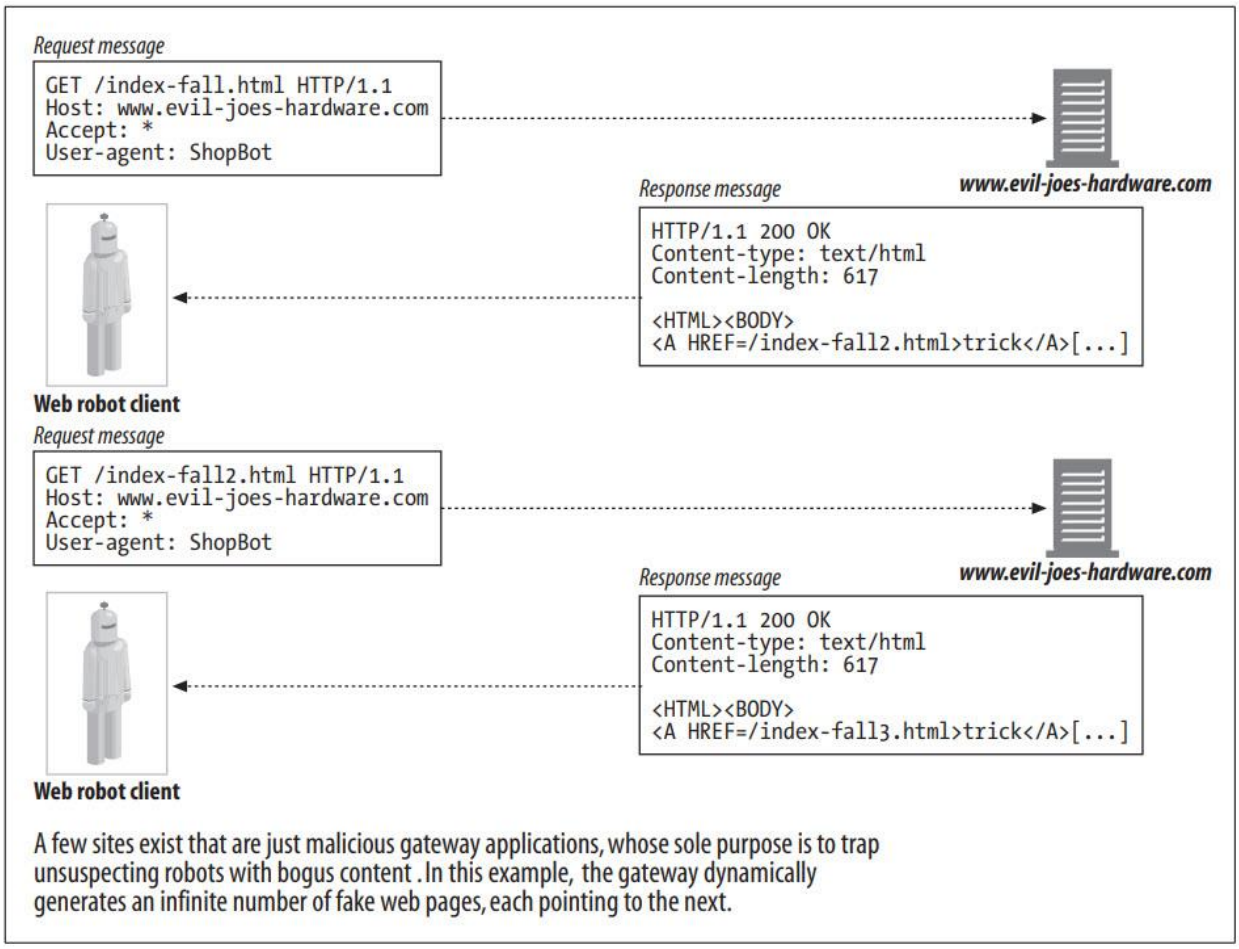
مشکل بخش b تصویر این است که /subdir چرخه بازگشت به / است، اما از آنجا که URL ها متفاوت به نظر می‌رسند، ربات تنها از طریق URL نمی‌داند که اسناد یکسان هستند. ربات ناآگاه خطر ورود به یک حلقه را دارد. بدون نوعی تشخیص حلقه، این چرخه اغلب تا زمانی که طول URL از محدودیت‌های ربات یا سرور تجاوز کند ادامه خواهد داشت.

Dynamic Virtual Web Spaces

این امکان برای وب مسترهای مخرب وجود دارد که عمداً حلقه‌های Crawler پیچیده‌ای را برای به دام انداختن ربات‌های بی گناه و بی خبر ایجاد کنند. به طور خاص، انتشار URL که شبیه یک فایل معمولی بوده اما واقعاً یک برنامه Gateway می‌باشد، آسان است. این برنامه می‌تواند HTML را که حاوی لینک‌هایی به URL های خیالی در همان سرور است، به سرعت بالا ببرد. هنگامی که این URL های خیالی درخواست می‌شوند، سرور بد یک صفحه HTML جدید با URL های خیالی جدید می‌سازد.

وب سرور مخرب می‌تواند ربات ضعیف را به سفر آلیس در سرزمین عجایب در یک فضای وب مجازی بی نهایت ببرد، حتی اگر وب سرور واقعاً حاوی هیچ فایلی نباشد. حتی بدتر از آن، تشخیص چرخه برای

ربات بسیار دشوار است، زیرا URL ها و HTML هر بار می‌توانند بسیار متفاوت به نظر برسند. شکل زیر نمونه‌ای از یک وب سرور مخرب را نشان می‌دهد که محتوای جعلی تولید می‌کند.



معمولاً، وب مسترهای خوش نیت ممکن است ناخواسته یک تله Crawler از طریق لینک‌های نمادین یا محتوای پویا ایجاد کنند. به عنوان مثال، یک برنامه تقویم مبتنی بر CGI را در نظر بگیرید که یک تقویم ماهانه و یک لینک به ماه بعد ایجاد می‌کند. یک کاربر واقعی برای همیشه به درخواست لینک ماه آینده ادامه نخواهد داد، اما روباتی که از ماهیت پویا محتوا آگاه نیست ممکن است به طور نامحدود به درخواست این منابع ادامه دهد.

Avoiding Loops and Dups

هیچ روشی برای اجتناب از تمام چرخه‌ها وجود ندارد. در عمل، ربات‌هایی که به خوبی طراحی شده‌اند، باید مجموعه‌ای از اکتشافات را برای جلوگیری از چرخه‌ها شامل شوند.

به طور کلی، هرچه یک Crawler مستقل‌تر باشد (نظارت انسانی کمتر)، احتمال بروز مشکل بیشتر است. مقداری Trade-off وجود دارد که پیاده‌کنندگان ربات باید آن را انجام دهند - این Heuristicها

می‌توانند به جلوگیری از مشکلات کمک کنند، اما تا حدودی زیان بار (lossy) هستند، زیرا در نهایت می‌توانید محتوای معتبری را که مشکوک به نظر می‌رسد نادیده بگیرید.

برخی از تکنیک‌هایی که ربات‌ها برای رفتار بهتر (در یک وب پر از خطرات ربات) استفاده می‌کنند عبارتند از:

Canonicalizing URLs

با تبدیل URL ها به فرم استاندارد از نام مستعار نحوی (Syntactic Aliases) اجتناب کنید.

Breadth-first crawling

Crawler ها مجموعه بزرگی از URL های بالقوه برای Crawl در هر زمان دارند. با برنامه ریزی URL ها برای بازدید به روشی گسترده، در سراسر وب سایت‌ها، می‌توانید تأثیر چرخه‌ها را به حداقل برسانید. حتی اگر به تله ربات برخورد کنید، باز هم می‌توانید صدها هزار صفحه را از وب سایت‌های دیگر واکشی کنید قبل از اینکه برای واکشی یک صفحه از چرخه بازگردید.

Throttling

تعداد صفحاتی را که ربات می‌تواند از یک وب سایت در یک دوره زمانی دریافت کند محدود کنید. اگر ربات به یک چرخه برخورد کند و به طور مداوم سعی کند به نام مستعار از یک سایت دسترسی پیدا کند، می‌توانید تعداد کل تکرارهای تولید شده و تعداد کل دسترسی‌ها به سرور را با Throttling محدود کنید.

Limit URL size

ربات ممکن است از Crawl نمودن URL های بیش از یک طول معین خودداری کند (۱ کیلوبایت رایج است). اگر یک چرخه باعث افزایش اندازه URL شود، محدودیت طول در نهایت چرخه را متوقف می‌کند. برخی از سرورهای وب با دادن URL طولانی از کار می‌افتند و ربات‌هایی که در چرخه افزایش URL گرفتار می‌شوند می‌توانند باعث از کار افتادن برخی از سرورهای وب شوند. این ممکن است باعث شود مدیران وب سایت ربات را به عنوان یک مهاجم انکار سرویس تعبیر کنند.

به عنوان یک احتیاط، این تکنیک مطمئناً می‌تواند منجر به از دست رفتن محتوا شود. امروزه بسیاری از سایت‌ها از URL ها برای کمک به مدیریت وضعیت کاربر استفاده می‌کنند (به عنوان مثال، ذخیره شناسه‌های کاربر در URL های ارجاع شده در یک صفحه). اندازه URL می‌تواند راهی دشوار برای محدود کردن Crawling باشد. با این حال، می‌تواند یک Flag عالی برای کاربر فراهم کند تا آنچه را که در یک سایت خاص اتفاق می‌افتد بررسی کندکه این کار با ثبت یک خطا هر زمان که URL های درخواستی به اندازه معینی رسیدند، انجام می‌شود.

URL/site blacklist

فهرستی از سایت‌ها و آدرس‌های اینترنتی شناخته شده را که با چرخه‌ها و تله‌های ربات مطابقت دارند، حفظ کنید و مانند طاعون از آن‌ها دوری کنید. با پیدا شدن مشکلات جدید، آن‌ها را به لیست سیاه اضافه کنید.

این مستلزم یک اکشن انسانی است. با این حال، اکثر Crawlerهای مقیاس بزرگ که امروزه تولید می‌شوند، نوعی لیست سیاه دارند همچنین می‌توان از لیست سیاه برای جلوگیری از برخی سایت‌هایی که در مورد Crawling سر و صدا ایجاد کرده‌اند، استفاده کرد.

Pattern detection

چرخه‌های ناشی از Symlink های سیستم فایل و پیکربندی‌های نادرست مشابه، از برخی الگوها پیروی می‌کنند. به عنوان مثال، URL ممکن است با اجزای تکراری رشد کند. برخی از ربات‌ها URL های دارای اجزای تکرار شونده را به عنوان چرخه‌های بالقوه می‌بینند و از Crawl نمودن URL هایی با بیش از دو یا سه جزء تکراری خودداری می‌کنند.

همه تکرارها فوری نیستند (به عنوان مثال، "/subdir/subdir/subdir/..."). ممکن است چرخه‌های دوره ۲ یا فواصل دیگر مانند "/images/subdir/ images/subdir/images/..." وجود داشته باشد. برخی از ربات‌ها به دنبال الگوهای تکرار شونده از چند دوره مختلف هستند.

Content fingerprinting

Fingerprinting روشی مستقیم‌تر برای شناسایی موارد تکراری است که توسط برخی از Crawlerهای وب پیچیده‌تر استفاده می‌شود. ربات‌هایی که از Fingerprinting محتوا استفاده می‌کنند، بایت‌های موجود در صفحه را می‌گیرند و یک Checksum را محاسبه می‌کنند. این Checksum نمایشی فشرده از محتوای صفحه است. اگر یک ربات صفحه‌ای را واکشی کند که Checksum آن را قبلاً دیده است، پیوندهای صفحه Crawl نمی‌شوند—اگر ربات قبلاً محتوای صفحه را دیده باشد، Crawl پیوندهای صفحه را قبلاً آغاز کرده است.

تابع Checksum باید به گونه‌ای انتخاب شود تا احتمال اینکه دو صفحه مختلف دارای Checksum یکسان باشند، کاهش پیدا کند. توابع Message Digest مانند MD5 برای Fingerprinting محبوب هستند.

از آنجایی که برخی از سرورهای وب به صورت پویا صفحات را در لحظه تغییر می‌دهند، ربات‌ها گاهی اوقات بخش‌های خاصی از محتوای صفحه وب، مانند لینک‌های تعبیه‌شده (Embedded Links) را از محاسبه Checksum حذف می‌کنند. با این حال، سمت سرور پویا شامل سفارشی کردن محتوای صفحه دلخواه (افزودن تاریخ، شمارنده‌های دسترسی و غیره) ممکن است از تشخیص تکراری جلوگیری کند.

Human monitoring

وب یک مکان وحشی است. ربات شجاع شما در نهایت به مشکلی برخورد خواهد کرد که هیچ یک از تکنیک‌های شما آن را حل نمی‌کند. تمام ربات‌های با کیفیت باید با تشخیص و لاگ طراحی شوند، بنابراین انسان‌ها می‌توانند به راحتی پیشرفت ربات را زیر نظر بگیرند و در صورت وقوع اتفاق غیرعادی به سرعت به آن‌ها هشدار داده شود. در برخی موارد، شهروندان شبکه خشمگین با ارسال ایمیل‌های ناخوشایند مشکل را برای شما برجسته می‌کنند.

اکتشافات عنکبوتی خوب برای Crawling مجموعه داده‌هایی به وسعت وب همیشه در حال انجام است. قوانین در طول زمان ساخته می‌شوند و با اضافه شدن انواع جدیدی از منابع به وب سازگار می‌شوند. لازم به ذکر است که قوانین خوب همیشه در حال تغییر هستند.

بسیاری از Crawlerهای کوچک‌تر و سفارشی‌تر برخی از این مسائل را کنار می‌گذارند، زیرا منابع (سرورها، پهنای باند شبکه، و غیره) که توسط یک Crawler خطاکار تحت تأثیر قرار می‌گیرند، قابل مدیریت هستند، یا حتی احتمالاً تحت کنترل شخصی هستند که Crawling را انجام می‌دهند (مانند یک سایت اینترانت). این Crawlerها برای جلوگیری از مشکلات به نظارت بیشتر انسانی متکی هستند.

Robotic HTTP

ربات‌ها هیچ تفاوتی با سایر برنامه‌های سرویس گیرنده HTTP ندارند. آن‌ها نیز باید از قوانین مشخصات HTTP پیروی کنند. رباتی که درخواست‌های HTTP را ایجاد نمود و خود را به‌عنوان کلاینت HTTP/1.1 تبلیغ می‌کند، باید از هدرهای درخواست HTTP مناسب استفاده کند.

بسیاری از ربات‌ها سعی می‌کنند حداقل مقدار HTTP مورد نیاز برای درخواست محتوای مورد نظر خود را پیاده سازی کنند. این می‌تواند منجر به مشکلاتی شود. با این حال، بعید است که این رفتار به این زودی‌ها تغییر کند. در نتیجه، بسیاری از ربات‌ها درخواست‌های HTTP/1.0 را ایجاد می‌کنند، زیرا آن پروتکل الزامات کمی دارد.

Identifying Request Headers

علیرغم حداقل مقدار HTTP که ربات‌ها تمایل دارند از آن پشتیبانی کنند، اکثر آن‌ها برخی از هدرهای شناسایی را پیاده‌سازی و ارسال می‌کنند – به ویژه هدر HTTP مربوط به User-Agent. توصیه می‌شود که پیاده‌کننده‌های ربات برخی از اطلاعات اولیه هدر را ارسال کنند تا سایت را از قابلیت‌های ربات، هویت ربات و محل پیدایش آن مطلع کنند.

این اطلاعات مفیدی هم برای ردیابی صاحب Crawler خطاکار و هم برای دادن اطلاعاتی به سرور در مورد نوع محتوایی است که ربات می‌تواند مدیریت کند. برخی از هدرهای شناسایی اولیه که پیاده‌کنندگان ربات تشویق به پیاده‌سازی می‌شوند عبارتند از:

User-Agent: نام رباتی را که درخواست می‌کند به سرور می گوید.

From: آدرس ایمیل کاربر/مدیر ربات را ارائه می‌دهد.

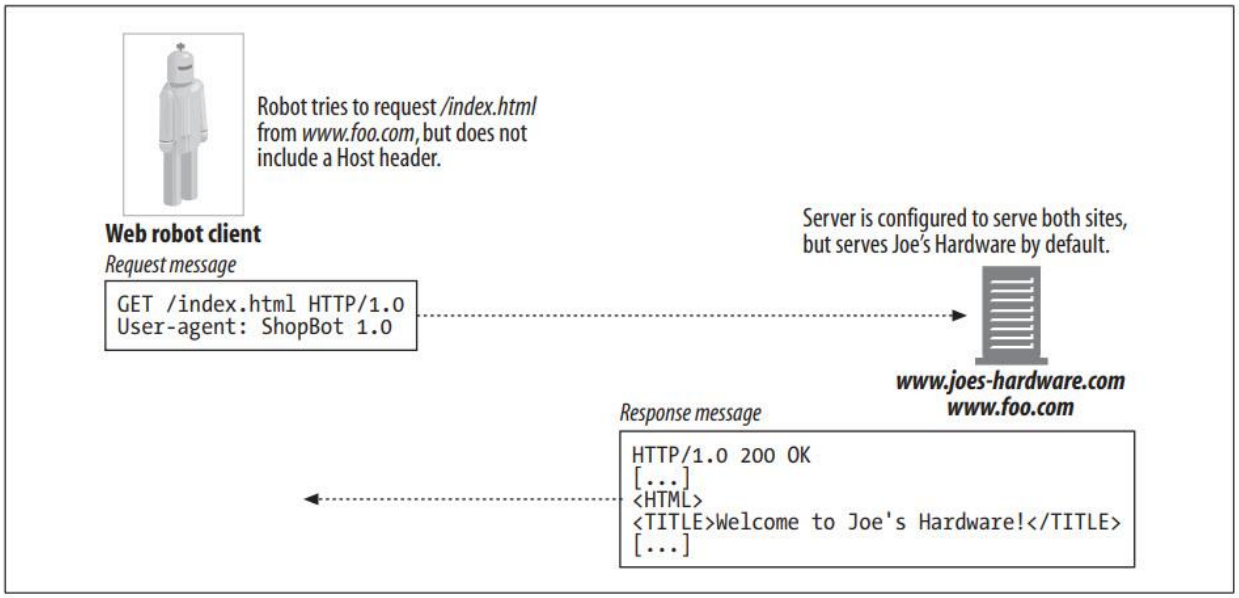
Accept: به سرور می‌گوید که چه نوع رسانه ای برای ارسال مناسب است. در این بخش اطمینان حاصل می شود که ربات فقط محتوای مورد علاقه خود را دریافت می‌کند (متن، تصاویر و غیره).

Referer: نشانی اینترنتی سندی را ارائه می‌کند که حاوی URL درخواست فعلی است.

Virtual Hosting

پیاده‌کننده‌های ربات باید از هدر Host پشتیبانی کنند. با توجه به رایج شدن میزبانی مجازی، عدم درج Host Header در درخواست‌ها می‌تواند منجر به شناسایی محتوای اشتباه توسط ربات‌ها با یک URL خاص شود. HTTP/1.1 به این دلیل نیاز به استفاده از هدر Host دارد.

اکثر سرورها به صورت پیش فرض برای سرویس دهی به یک سایت خاص پیکربندی شده اند. بنابراین، یک Crawler بدون هدر Host می‌تواند به سروری که دو سایت را ارائه می‌کند، مانند موارد موجود در شکل زیر (www.joes-hardware.com و www.foo.com) درخواست بدهد و اگر سرور به‌طور پیش‌فرض برای سرویس www.joes-hardware.com پیکربندی شده باشد (و به هدر Host نیاز ندارد)، منجر به دریافت محتوا از سایت Joe’s Hardware می‌شود. بدتر از آن، Crawler در واقع فکر می‌کند که محتوای Joe’s Hardware از www.foo.com است. مطمئنم اگر اسنادی از دو سایت با دیدگاه‌های سیاسی قطبی یا سایر دیدگاه‌ها از یک سرور ارائه شود، می‌توانید به موقعیت‌های ناگوارتری فکر کنید.



Conditional Requests

با توجه به عظمت برخی از تلاش‌های رباتیک، اغلب معقول است که مقدار محتوایی را که یک ربات بازیابی می‌کند به حداقل برسانیم. همانطور که در مورد ربات‌های موتور جستجوی اینترنتی، با میلیاردها صفحه وب برای دانلود بالقوه، بازیابی مجدد محتوا تنها در صورتی منطقی است که تغییر کرده باشد.

برخی از این ربات‌ها درخواست‌های HTTP مشروط را پیاده‌سازی می‌کنند، برچسب‌های زمانی یا موجودیت را مقایسه می‌کنند تا ببینند آیا آخرین نسخه‌ای که بازیابی کرده‌اند به‌روزرسانی شده است یا خیر. این بسیار شبیه به روشی است که یک HTTP Cache اعتبار کپی محلی منبعی که قبلاً واکشی شده را بررسی می‌کند.

Response Handling

از آنجایی که بسیاری از ربات‌ها در درجه اول به دریافت محتوای درخواستی از طریق متدهای ساده GET علاقمند هستند، اغلب در نحوه رسیدگی به پاسخ کار زیادی انجام نمی‌دهند. با این حال، ربات‌هایی که از برخی ویژگی‌های HTTP (مانند درخواست‌های شرطی) استفاده می‌کنند و همچنین آن‌هایی که می‌خواهند بهتر کاوش کنند و با سرورها تعامل داشته باشند، باید بتوانند انواع مختلف پاسخ‌های HTTP را مدیریت کنند.

Status codes

به طور کلی، ربات‌ها باید بتوانند حداقل کدهای وضعیت رایج یا مورد انتظار را مدیریت کنند. همه ربات‌ها باید کدهای وضعیت HTTP مانند 200 OK و 404 Not Found را درک کنند. آن‌ها همچنین باید بتوانند با کدهای وضعیتی که به صراحت بر اساس دسته بندی کلی پاسخ درک نمی‌کنند، مقابله کنند.

توجه به این نکته ضروری است که برخی از سرورها همیشه کدهای خطای مناسب را برنمی‌گردانند. برخی از سرورها حتی کد 200 OK را با متن پیام که خطا را توصیف می‌کند، برمی‌گردانند! انجام کارهای زیادی در این مورد سخت است - این فقط چیزی است که اجراکنندگان باید از آن آگاه باشند.

Entities

همراه با اطلاعات جاسازی شده در هدرهای HTTP، ربات‌ها می‌توانند به دنبال اطلاعات در خود موجودیت بگردند. تگ‌های متا HTML، مانند تگ متا http-equiv، ابزاری برای نویسندگان محتوا برای جاسازی اطلاعات اضافی درباره منابع هستند.

```
<meta http-equiv="Refresh" content="1;URL=index.html">
```

این تگ به گیرنده دستور می‌دهد تا با سند به گونه‌ای رفتار کند که گویی هدر پاسخ HTTP آن حاوی یک هدر Refresh HTTP با مقدار "1, URL=index.html" است.

برخی از سرورها در واقع محتویات صفحات HTML را قبل از ارسال آن‌ها تجزیه می‌کنند و دستورالعمل‌های http-equiv را به عنوان هدر درج می‌کنند. با این حال، برخی هم این کار را نمی‌کنند. پیاده‌کننده‌های ربات ممکن است بخواهند عناصر HEAD اسناد HTML را برای جستجوی اطلاعات httpequiv اسکن کنند.

User-Agent Targeting

مدیران وب باید در نظر داشته باشند که بسیاری از ربات‌ها از سایت‌های آن‌ها بازدید می‌کنند و بنابراین باید از آن‌ها انتظار درخواست را داشته باشند. بسیاری از سایت‌ها، محتوا را برای User-Agentهای مختلف بهینه می‌کنند و سعی می‌کنند انواع مرورگرها را شناسایی کنند تا از پشتیبانی از ویژگی‌های مختلف سایت اطمینان حاصل کنند. با انجام این کار، سایت‌ها به جای محتوا، صفحات خطا را در اختیار ربات‌ها قرار می‌دهند. انجام جستجوی متنی برای عبارت «مرورگر شما از فریم‌ها پشتیبانی نمی‌کند» در برخی از موتورهای جستجو، فهرستی از نتایج را برای صفحات خطای حاوی این عبارت به دست می‌دهد، در حالی که در واقع کلاینت HTTP اصلا یک مرورگر نبود، بلکه یک ربات بوده است.

مدیران سایت باید یک استراتژی برای رسیدگی به درخواست‌های ربات برنامه ریزی کنند. به عنوان مثال، به جای محدود کردن توسعه محتوای خود به پشتیبانی مرورگر خاص، می‌توانند صفحاتی را برای مرورگرها و ربات‌های بدون ویژگی (non–feature) ایجاد کنند. حداقل، آن‌ها باید انتظار داشته باشند که ربات‌ها از سایت‌های آن‌ها بازدید نموده و در هنگام بازدید از آن‌ها غافل نشوند.

Misbehaving Robots

راه‌های زیادی وجود دارد که ربات‌های سرکش می‌توانند باعث آشفتگی شوند. در اینجا چند اشتباهی که ربات‌ها می‌توانند مرتکب شوند و تأثیر اعمال نادرست آن‌ها آورده شده است:

Runaway robots

ربات‌ها درخواست‌های HTTP را بسیار سریع‌تر از وب‌گردهای انسانی ارسال می‌کنند و معمولاً روی رایانه‌های سریع با لینک‌های شبکه سریع اجرا می‌شوند. اگر یک ربات دارای یک خطای منطقی برنامه نویسی باشد، یا در چرخه‌ای گیر بیفتد، می‌تواند بار شدیدی را بر روی یک وب سرور پرتاب کند - احتمالاً به اندازه ای که منجر به Overload سرور و عدم دسترسی سرویس برای دیگران شود. همه نویسندگان ربات باید در طراحی ربات‌های خود به این موضوع توجه زیادی داشته باشند.

Stale URLs

برخی از ربات‌ها از لیست URL ها بازدید می‌کنند. این لیست‌ها می‌توانند قدیمی باشند. اگر یک وب‌سایت تغییر بزرگی در محتوای خود ایجاد کند، ربات‌ها ممکن است تعداد زیادی URL غیرموجود را درخواست کنند. این امر برخی از مدیران وب سایت را آزار می‌دهد، که دوست ندارند گزارش‌های خطای آن‌ها با درخواست‌های دسترسی برای اسناد موجود پر شود و دوست ندارند که ظرفیت وب سرور آن‌ها با هزینه‌های سربار صفحات خطا کاهش یابد.

Long, wrong URLs

در نتیجه چرخه‌ها و خطاهای برنامه نویسی، ربات‌ها ممکن است URL های بزرگ و بی معنی را از وب‌سایت‌ها درخواست کنند. اگر URL به اندازه کافی طولانی باشد، ممکن است عملکرد وب سرور را کاهش دهد، گزارش‌های دسترسی به سرور وب را به هم ریخته و حتی باعث از کار افتادن سرورهای وب شکننده شود.

Nosy robots

برخی از روبات‌ها ممکن است URLهایی دریافت کنند که به داده‌های خصوصی اشاره می‌کنند و آن داده‌ها را به راحتی از طریق موتورهای جستجوی اینترنتی و سایر برنامه‌ها، در دسترس قرار می‌دهند. اگر صاحب داده‌ها به طور فعال صفحات وب را تبلیغ نکرده باشد، ممکن است انتشار رباتیک را در بهترین حالت یک مزاحم و در بدترین حالت تجاوز به حریم خصوصی بداند.

معمولاً این اتفاق می‌افتد زیرا یک لینک به محتوای «خصوصی» که ربات دنبال می‌کرد از قبل وجود دارد (یعنی محتوا آن‌قدر که مالک فکر می‌کرد مخفی نیست، یا مالک فراموش کرده است که یک لینک قبلی را حذف کند).

پیاده‌کننده‌های رباتی که حجم زیادی از داده‌ها را از وب بازیابی می‌کنند باید بدانند که روبات‌های آن‌ها احتمالاً در نقطه‌ای می‌توانند داده‌های حساس را بازیابی کنند - داده‌هایی که پیاده‌کننده سایت هرگز قصد نداشت از طریق اینترنت قابل دسترسی باشد. این داده‌های حساس می‌تواند شامل فایل‌های رمز عبور یا حتی اطلاعات کارت اعتباری باشد.

واضح است که مکانیزمی برای نادیده گرفتن محتوا پس از اشاره به این موضوع (و حذف آن از هر فهرست جستجو یا آرشیو) مهم است. کاربران موتورهای جستجوی مخرب و آرشیو، از توانایی‌های Web Crawler ها در مقیاس بزرگ برای یافتن محتوا استفاده می‌کنند—بعضی از موتورهای جستجو، مانند Google، در واقع نمایش‌هایی از صفحاتی را که Crawl کرده اند بایگانی می‌کنند، بنابراین حتی اگر محتوا حذف شود، هنوز برای مدتی می‌توان محتوا را یافته و به آن دسترسی داشت.

Dynamic gateway access

ربات‌ها همیشه نمی‌دانند به چه چیزی دسترسی دارند. یک ربات ممکن است یک URL را واکشی کند که محتوای آن از یک برنامه Gateway می‌آید. در این مورد، داده‌های به‌دست‌آمده ممکن است برای هدف خاص باشند و ممکن است محاسبه آن گران باشد. بسیاری از مدیران وب‌سایت‌ها از ربات‌های ساده لوح درخواست اسناد که از Gateway ها می‌آیند، خوششان نمی‌آید.

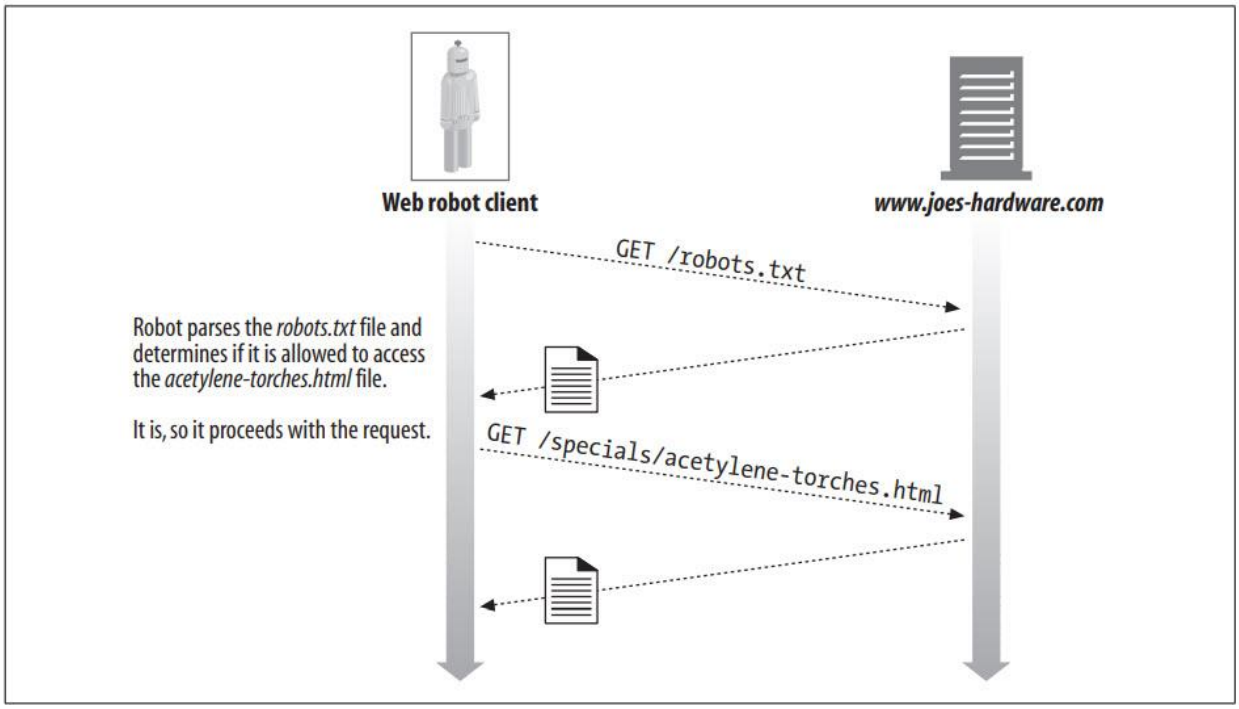
Excluding Robots

جامعه ربات مشکلاتی را که دسترسی به وب سایت رباتیک می‌تواند ایجاد کند، درک کرده است. در سال ۱۹۹۴، یک تکنیک ساده و داوطلبانه پیشنهاد شد تا ربات‌ها را از جایی که به آن‌ها تعلق ندارند دور نگه دارد و مکانیزمی برای کنترل بهتر رفتارشان برای مدیران وب‌سایت فراهم کند. این استاندارد "Robots Exclusion Standard" یا استاندارد حذف ربات‌ها نام داشت، اما اغلب فقط robots.txt نامیده می‌شود، فایلی که اطلاعات کنترل دسترسی در آن ذخیره می‌شود.

ایده robots.txt ساده است. هر وب سرور می‌تواند یک فایل اختیاری به نام robots.txt را در مسیر root وب سرور یا وب اپلیکیشن ایجاد کند. این فایل حاوی اطلاعاتی در مورد این است که چه رباتی

می‌تواند به چه بخش‌هایی از سرور دسترسی داشته باشد. اگر روباتی از این استاندارد داوطلبانه پیروی کند، قبل از دسترسی به هر منبع دیگری از آن سایت، فایل robots.txt را از وب سایت درخواست می‌کند.

به عنوان مثال، ربات موجود در شکل زیر می‌خواهد acetylene-torches.html را با آدرس مشخص، از Joe's Hardware دانلود کند. قبل از اینکه ربات بتواند صفحه را درخواست کند، باید فایل robots.txt را بررسی کند تا ببیند آیا مجوز واکشی این صفحه را دارد یا خیر. در این مثال، فایل robots.txt ربات را مسدود نمی‌کند، بنابراین ربات به صفحه دسترسی خواهد داشت و امکان دریافت آن را دارد.



The Robots Exclusion Standard

استاندارد حذف روبات‌ها یک استاندارد موقت است. در زمان نگارش این مقاله، هیچ نهاد رسمی استاندارد، مالک این استاندارد نیست و فروشندگان زیر مجموعه‌های مختلفی از استاندارد را پیاده سازی می‌کنند. با این حال، برخی از توانایی‌های مدیریت دسترسی ربات‌ها به وب‌سایت‌ها، حتی اگر ناقص باشد، بهتر از هیچ چیز است و اکثر فروشندگان اصلی و Crawler های موتور جستجو از استاندارد Exclusion پشتیبانی می‌کنند.

سه نسخه از استاندارد حذف روبات‌ها وجود دارد، اگرچه که نام این نسخه‌ها به خوبی تعریف نشده است. ما شماره‌گذاری نسخه نشان داده شده در جدول زیر را اتخاذ می‌کنیم.

Version	Title and description	Date
0.0	A Standard for Robot Exclusion—Martijn Koster's original *robots.txt* mechanism with Disallow directive	June 1994
1.0	A Method for Web Robots Control—Martijn Koster's IETF draft with additional support for Allow	Nov. 1996
2.0	An Extended Standard for Robot Exclusion—Sean Conner's extension including regex and timing information; not widely supported	Nov. 1996

امروزه اکثر ربات‌ها از استانداردهای v0.0 یا v1.0 استفاده می‌کنند. استاندارد v2.0 بسیار پیچیده‌تر است و به طور گسترده مورد استفاده قرار نگرفته است و ممکن است هرگز مورد استفاده قرار نگیرد. ما در اینجا بر روی استاندارد v1.0 تمرکز خواهیم کرد، زیرا استفاده گسترده ای دارد و کاملاً با نسخه v0.0 سازگار است.

Web Sites and robots.txt Files

قبل از بازدید از هر URL در یک وب سایت، یک ربات باید فایل robots.txt در وب سایت را در صورت وجود بازیابی و پردازش کند. یک منبع robots.txt برای کل وب سایت وجود دارد که با نام میزبان و پورت تعریف شده است. اگر سایت به صورت مجازی میزبانی شود، می‌تواند مانند هر فایل دیگری برای هر docroot مجازی یک فایل robots.txt متفاوت وجود داشته باشد.

در حال حاضر، هیچ راهی برای نصب فایل‌های robots.txt «محلی» در زیر شاخه‌های جداگانه یک وب‌سایت وجود ندارد. مدیر وب سایت مسئول ایجاد یک فایل robots.txt انبوه است که قوانین حذف را برای تمام محتوای وب سایت شرح می‌دهد.

Fetching robots.txt

روبات‌ها منبع robots.txt را با استفاده از متد HTTP GET، مانند هر فایل دیگری در سرور وب، واکشی می‌کنند. سرور فایل robots.txt را، در صورت وجود، به صورت متن/متن ساده برمی‌گرداند. اگر سرور با کد وضعیت 404 Not Found پاسخ دهد، ربات می‌تواند فرض کند که هیچ محدودیتی برای دسترسی رباتیک وجود ندارد و می‌تواند هر فایلی را درخواست کند.

روبات‌ها باید اطلاعات شناسایی را در هدرهای From و User-Agent ارسال کنند تا به مدیران سایت کمک کنند دسترسی‌های رباتیک را ردیابی کنند و در صورت نیاز مدیر سایت به درخواست یا شکایت از ربات، اطلاعات تماس را ارائه دهند. در اینجا یک نمونه درخواست HTTP Crawler از یک ربات وب تجاری وجود دارد:

```
GET /robots.txt HTTP/1.0
Host: www.joes-hardware.com
```

User-Agent: Slurp/2.0

Date: Wed Oct 3 20:22:48 EST 2001

Response codes

بسیاری از وب سایت‌ها منبع robots.txt ندارند، اما ربات این را نمی‌داند. باید سعی کند منبع robots.txt را از هر سایتی دریافت کند. ربات بسته به نتیجه بازیابی robots.txt اقدامات مختلفی انجام می‌دهد:

- اگر سرور با وضعیت موفقیت آمیز پاسخ دهد (کد وضعیت HTTP 2XX)، ربات باید محتوا را تجزیه کند و قوانین حذف را برای واکشی از آن سایت اعمال کند.
- اگر پاسخ سرور نشان دهد که منبع وجود ندارد (کد وضعیت HTTP 404)، ربات می‌تواند فرض کند که هیچ قانونی فعال نبوده و دسترسی به سایت توسط robots.txt محدود نشده است.
- اگر پاسخ سرور محدودیت دسترسی را نشان دهد (کد وضعیت HTTP 401 یا 403)، ربات باید دسترسی به سایت را کاملاً محدود در نظر بگیرد.
- اگر تلاش درخواست منجر به شکست موقت شود (کد وضعیت HTTP 503)، ربات باید بازدید از سایت را تا زمان بازیابی منبع به تعویق بیندازد.
- اگر پاسخ سرور نشان دهنده تغییر مسیر باشد (کد وضعیت HTTP 3XX)، ربات باید تغییر مسیرها را تا زمانی که منبع پیدا شود دنبال کند.

robots.txt File Format

فایل robots.txt یک Syntax بسیار ساده و line-oriented دارد. سه نوع خط در فایل robots.txt وجود دارد: خطوط خالی، خطوط Comment و خطوط قانون. خطوط قوانین شبیه هدرهای HTTP (<Field>:<Value>) هستند و برای تطبیق الگو استفاده می‌شوند. به عنوان مثال:

```
# this robots.txt file allows Slurp & Webcrawler to crawl
# the public parts of our site, but no other robots...

User-Agent: slurp
User-Agent: webcrawler
Disallow: /private

User-Agent: *
Disallow:
```

خطوط در یک فایل robots.txt به طور منطقی به ایجاد "رکوردها" منجر می‌شوند. هر رکورد مجموعه‌ای از قوانین حذف را برای مجموعه خاصی از روبات‌ها توصیف می‌کند. به این ترتیب، قوانین حذف متفاوتی را می‌توان برای روبات‌های مختلف اعمال کرد.

هر رکورد شامل مجموعه‌ای از خطوط قانون است که با یک خط خالی یا کاراکتر انتهای فایل خاتمه می‌یابد. یک رکورد با یک یا چند خط User-Agent شروع می‌شود که مشخص می‌کند کدام ربات تحت تأثیر این رکورد قرار می‌گیرد و به دنبال آن خطوط Disallow و Allow می‌گویند که این روبات‌ها به چه URL هایی می‌توانند دسترسی داشته باشند.

مثال قبلی یک فایل robots.txt را نشان می‌دهد که به روبات‌های Slurp و Webcrawler اجازه می‌دهد به هر فایلی به جز آن فایل‌های موجود در زیر شاخه خصوصی دسترسی داشته باشند. همین فایل همچنین از دسترسی ربات‌های دیگر به هر چیزی در سایت جلوگیری می‌کند.

بیایید به خطوط User-Agent، Disallow و Allow نگاه کنیم.

The User-Agent line

هر رکورد ربات با یک یا چند خط User-Agent به شکل زیر شروع می‌شود:

User-Agent: <robot-name> or User-Agent: *

نام ربات (انتخاب شده توسط طراح ربات) در سربرگ User-Agent درخواست HTTP GET ربات ارسال می‌شود.

هنگامی که یک ربات یک فایل robots.txt را پردازش می‌کند، باید از رکوردهای زیر پیروی کند:

- اولین نام ربات که شامل رشته است (عدم حساسیت به حروف کوچک و بزرگ)
- اولین نام ربات که به صورت * است.

اگر ربات نتواند یک خط User-Agent مطابق با نام خود پیدا کند و نتواند یک خط "User-Agent: *" پیدا کند، هیچ رکوردی مطابقت نداشته و دسترسی نامحدود است.

از آنجایی که نام ربات به حروف کوچک و بزرگ حساس نیست، مراقب تطابق‌های نادرست باشید. به عنوان مثال، "User-Agent: bot" با تمام ربات‌هایی به نام‌های Bot، Robot، Bottom-Feeder، Spambot و Dont-Bother-Me مطابقت دارد.

The Disallow and Allow lines

خطوط Disallow و Allow بلافاصله از خطوط User-Agent قرار می‌گیرند. آن‌ها توضیح می‌دهند که کدام مسیرهای URL به صراحت برای روبات‌های مشخص شده ممنوع یا به صراحت مجاز هستند.

ربات باید URL مورد نظر را به ترتیب با تمام قوانین Disallow و Allow مطابقت دهد. اولین مطابقت یافت شده استفاده می‌شود. اگر مطابقت پیدا نشد، URL مجاز است.

برای تطابق خط Allow/Disallow با URL، مسیر قانون باید پیشوند مسیر URL حساس به حروف بزرگ و کوچک باشد. به عنوان مثال، "Disallow: /tmp" با همه این URL ها مطابقت دارد:

http://www.joes-hardware.com/tmp

http://www.joes-hardware.com/tmp/

http://www.joes-hardware.com/tmp/pliers.html

http://www.joes-hardware.com/tmpspc/stuff.txt

Disallow/Allow prefix matching

در اینجا چند جزئیات بیشتر در مورد تطبیق پیشوند Allow/Disallow وجود دارد:

- قوانین Allow/Disallow به تطابق پیشوند حساس به حروف بزرگ و کوچک نیاز دارند. ستاره هیچ معنای خاصی ندارد (برخلاف خطوط User-Agent)، اما Universal Wildcarding Effect را می‌توان از رشته خالی به دست آورد.
- هر Escaped Charachter ای مانند %XX در مسیر قانون یا مسیر URL قبل از مقایسه به بایت‌ها بازگردانده می‌شود (Unescaped Back). به استثنای %2F که معادل فوردارد اسلش بوده و باید دقیقاً مطابقت داشته باشد.
- اگر مسیر قانون رشته خالی باشد، با همه چیز مطابقت دارد.

جدول زیر چندین نمونه از تطابق بین مسیرهای قانون و مسیرهای URL را فهرست می‌کند:

Rule path	URL path	Match?	Comments
/tmp	/tmp	✓	Rule path == URL path
/tmp	/tmpfile.html	✓	Rule path is a prefix of URL path
/tmp	/tmp/a.html	✓	Rule path is a prefix of URL path
/tmp/	/tmp	✗	/tmp/ is not a prefix of /tmp
	README.TXT	✓	Empty rule path matches everything
/~fred/hi.html	%7Efred/hi.html	✓	%7E is treated the same as ~
/%7Efred/hi.html	/~fred/hi.html	✓	%7E is treated the same as ~
/%7efred/hi.html	/%7Efred/hi.html	✓	Case isn't significant in escapes
/~fred/hi.html	~fred%2Fhi.html	✗	%2F is slash, but slash is a special case that must match exactly

تطبیق پیشوند معمولاً به خوبی کار می‌کند، اما چند جا وجود دارد که به اندازه کافی رسا نیست. اگر زیردایرکتوری‌های خاصی وجود دارد که می‌خواهید Crawling برای آن‌ها نیز ممنوع شود، صرف نظر از اینکه پیشوند مسیر چیست، robots.txt هیچ وسیله ای برای این کار ارائه نمی‌دهد. برای مثال، ممکن است بخواهید از Crawling زیرشاخه‌های کنترل نسخه RCS خودداری کنید. نسخه ۱,۰ طرح robots.txt هیچ راهی برای پشتیبانی از این ارائه نمی‌دهد، به جز برشمردن جداگانه هر مسیر به هر زیرشاخه RCS.

Other robots.txt Wisdom

در اینجا قوانین دیگری در رابطه با تجزیه فایل robots.txt وجود دارد:

فایل robots.txt ممکن است حاوی فیلدهایی غیر از User-Agent، Disallow و Allow باشد، زیرا مشخصات پیشرفت می‌کند. یک ربات باید هر زمینه ای را که نمی‌فهمد نادیده بگیرد.

برای سازگاری با قبل(Backward Compatibility)، شکستن خطوط مجاز نیست.

Comment در هر نقطه از فایل مجاز است. آن‌ها از فضای خالی اختیاری تشکیل شده‌اند و به دنبال آن یک کاراکتر (#) و سپس Comment مورد نظر تا به کاراکتر پایان خط برسیم.

نسخه 0.0 از Robots Exclusion Standard از خط Allow پشتیبانی نمی‌کند. برخی از ربات‌ها فقط مشخصات نسخه 0.0 را اجرا می‌کنند و خطوط مجاز را نادیده می‌گیرند. در این شرایط، یک ربات محافظه کارانه رفتار می‌کند و URL های مجاز را بازیابی نمی‌کند.

Caching and Expiration of robots.txt

اگر یک ربات مجبور شود قبل از هر دسترسی به فایل، یک فایل robots.txt را دوباره واکشی کند، بار روی سرورهای وب را دو برابر می‌کند و همچنین کارایی ربات را کاهش می‌دهد. در عوض، از روبات‌ها انتظار می‌رود که فایل robots.txt را به صورت دوره‌ای واکشی کرده و نتایج را در Cache نگه دارند. کپی Cache شده robots.txt باید توسط ربات استفاده شود تا زمانی که فایل robots.txt منقضی شود.

مکانیسم‌های استاندارد HTTP cache-control هم توسط سرور مبدا و هم ربات‌ها برای کنترل ذخیره‌سازی فایل robots.txt استفاده می‌شوند. روبات‌ها باید در پاسخ HTTP به هدرهای Cache-Control و Expires توجه داشته باشند.

امروزه بسیاری از Crawler های تولیدی با کلاینت HTTP/1.1 کار نمی‌کنند. مدیران وب‌سایت‌ها باید توجه داشته باشند که آن Crawler ها لزوماً دستورالعمل‌های Cache ارائه‌شده برای منبع robots.txt را درک نمی‌کنند.

اگر دستورالعمل‌های Cache-Control وجود نداشته باشد، مشخصات پیش‌نویس (Draft Specification) اجازه Caching به مدت هفت روز را می‌دهد. اما، در عمل، این اغلب خیلی طولانی است. اگر فایل robots.txt برای یک هفته در Cache ذخیره شود، فایل robots.txt جدید ایجاد شده هیچ تاثیری نخواهد داشت و مدیر سایت، مدیر ربات را به عدم رعایت Robots Exclusion Standard متهم می‌کند.

Robot Exclusion Perl Code

چند کتابخانه پرل در دسترس عموم برای تعامل با فایل‌های robots.txt وجود دارد. یک مثال ماژول WWW::RobotsRules است که برای آرشیو عمومی پرل CPAN موجود است.

فایل robots.txt تجزیه شده در شیء WWW::RobotRules نگهداری می‌شود، که روش‌هایی را برای بررسی اینکه آیا دسترسی به یک URL داده شده ممنوع است یا خیر، ارائه می‌کند. همان شی WWW:: RobotRules می‌تواند چندین فایل robots.txt را تجزیه کند.

در اینجا روش‌های اولیه در WWW::RobotRules API آمده است:

ایجاد یک شیء RobotRules

```
$rules = WWW::RobotRules->new($robot_name);
```

بارگزاری فایل robots.txt

```
$rules->parse($url, $content, $fresh_until);
```

بررسی اینکه آیا URL سایت قابل واکشی است یا خیر

```
$can_fetch = $rules->allowed($url);
```

در اینجا یک برنامه کوتاه Perl وجود دارد که استفاده از WWW::RobotRules را نشان می‌دهد:

```
require WWW::RobotRules;
# Create the RobotRules object, naming the robot "SuperRobot"
my $robotsrules = new WWW::RobotRules 'SuperRobot/1.0';
use LWP::Simple qw(get);

# Get and parse the robots.txt file for Joe's Hardware, accumulating the rules
$url = "http://www.joes-hardware.com/robots.txt";
my $robots_txt = get $url;
$robotsrules->parse($url, $robots_txt);

# Get and parse the robots.txt file for Mary's Antiques, accumulating the rules
$url = "http://www.marys-antiques.com/robots.txt";
my $robots_txt = get $url;
;robotsrules->parse($url, $robots_txt)$
# Now RobotRules contains the set of robot exclusion rules for several
# different sites. It keeps them all separate. Now we can use RobotRules
# to test if a robot is allowed to access various URLs.
if ($robotsrules->allowed($some_target_url))
{
        $c = get $url;
        ...
       }
```

فایل زیر یک فایل robots.txt فرضی برای www.marys-antiques.com است:

```
##########################################################################
# This is the robots.txt file for Mary's Antiques web site
##########################################################################
# Keep Suzy's robot out of all the dynamic URLs because it doesn't
# understand them, and out of all the private data, except for the
# small section Mary has reserved on the site for Suzy.

User-Agent: Suzy-Spider
Disallow: /dynamic
Allow: /private/suzy-stuff
Disallow: /private

# The Furniture-Finder robot was specially designed to understand
# Mary's antique store's furniture inventory program, so let it
# crawl that resource, but keep it out of all the other dynamic
# resources and out of all the private data.

User-Agent: Furniture-Finder
Allow: /dynamic/check-inventory
Disallow: /dynamic
Disallow: /private

# Keep everyone else out of the dynamic gateways and private data.

User-Agent: *
Disallow: /dynamic
Disallow: /private
```

این فایل robots.txt حاوی یک رکورد برای ربات به نام SuzySpider، یک رکورد برای ربات به نام FurnitureFinder و یک رکورد پیش فرض برای همه ربات‌های دیگر است. هر رکورد مجموعه متفاوتی از سیاست‌های دسترسی را برای روبات‌های مختلف اعمال می‌کند:

- Exclusion Record برای SuzySpider، ربات را از Crawling در آدرس‌های اینترنتی Gateway موجودی فروشگاه که با /dynamic شروع می‌شوند و از داده‌های خصوصی کاربر خارج می‌شوند، حفظ می‌کند، به‌جز قسمتی که برای Suzy رزرو شده است.
- رکورد ربات FurnitureFinder به ربات اجازه می‌دهد تا URL دروازه موجودی مبلمان را Crawl کند. شاید این ربات فرمت و قوانین دروازه Mary را درک کند.
- همه ربات‌های دیگر از تمام صفحات وب پویا و خصوصی دور نگه داشته می‌شوند، اگرچه می‌توانند بقیه URL ها را Crawl کنند.

جدول زیر نمونه‌هایی را برای دسترسی ربات‌های مختلف به وب سایت Mary's Antiques فهرست می‌کند.

URL	SuzySpider	FurnitureFinder	NosyBot
http://www.marys-antiques.com/	✓	✓	✓
http://www.marys-antiques.com/index.html	✓	✓	✓
http://www.marys-antiques.com/private/payroll.xls	✗	✗	✗
http://www.marys-antiques.com/private/suzy-stuff/taxes.txt	✓	✗	✗
http://www.marys-antiques.com/dynamic/buy-stuff?id=3546	✗	✗	✗
http://www.marys-antiques.com/dynamic/check-inventory?kitchen	✗	✓	✗

HTML Robot-Control META Tags

فایل robots.txt به مدیر سایت اجازه می‌دهد تا ربات‌ها را از برخی یا همه یک وب سایت حذف کند. یکی از معایب فایل robots.txt این است که متعلق به مدیر وب سایت است نه نویسنده محتوا.

نویسندگان صفحه HTML راه مستقیم‌تری برای محدود کردن روبات‌ها از صفحات جداگانه دارند. آن‌ها می‌توانند تگ‌های کنترل ربات را مستقیماً به اسناد HTML اضافه کنند. ربات‌هایی که به تگ‌های HTML کنترل ربات پایبند هستند همچنان می‌توانند اسناد را واکشی کنند، اما اگر تگ Robot Exclusion وجود داشته باشد، اسناد را نادیده می‌گیرند. به عنوان مثال، یک ربات موتور جستجوی اینترنتی سند را در فهرست جستجوی خود قرار نمی‌دهد. همانند استاندارد robots.txt، مشارکت تشویق می‌شود اما اجرا نمی‌شود.

تگ‌های Robot Exclusion با استفاده از تگ‌های HTML META، با استفاده از فرم پیاده سازی می‌شوند:

<META NAME="ROBOTS" CONTENT=directive-list>

Robot META directives

انواع مختلفی از دستورالعمل‌های META ربات وجود دارد و احتمالاً دستورالعمل‌های جدید به مرور زمان و با گسترش فعالیت‌ها و مجموعه ویژگی‌های موتورهای جستجو و روبات‌های آن‌ها اضافه خواهند شد. دو دستورالعمل META ربات که اغلب مورد استفاده قرار می‌گیرند عبارتند از:

NOINDEX

این تنظیم به یک ربات می‌گوید که محتوای صفحه را پردازش نکند و سند را نادیده بگیرد (یعنی محتوا را در هیچ فهرست یا پایگاه داده‌ای درج نکند).

<META NAME="ROBOTS" CONTENT="NOINDEX">

NOFOLLOW

این تنظیم به یک ربات می‌گوید که هیچ لینک خروجی را از صفحه Crawl نکند.

<META NAME="ROBOTS" CONTENT="NOFOLLOW">

علاوه بر NOINDEX و NOFOLLOW، دستورالعمل‌های مخالف INDEX و FOLLOW، دستورالعمل NOARCHIVE و دستورالعمل‌های ALL و NONE نیز وجود دارد. این دستورالعمل‌های META Tag ربات به شرح زیر خلاصه می‌شوند:

INDEX

به ربات می‌گوید که ممکن است محتویات صفحه را ایندکس کند.

FOLLOW

به یک ربات می‌گوید که ممکن است لینک‌های خروجی را در صفحه Crawl کند.

NOARCHIVE

به یک ربات می‌گوید که نباید یک Local Copy از صفحه را در Cache نگه دارد.

ALL

معادل INDEX، FOLLOW.

NONE

معادل NOINDEX، NOFOLLOW.

تگ‌های متا ربات، مانند تمام تگ‌های متا HTML، باید در بخش HEAD یک صفحه HTML ظاهر شوند:

```
<html>
<head>
    <meta name="robots" content="noindex,nofollow">
    <title>...</title>
</head>
<body>
    ...
</body>
</html>
```

توجه داشته باشید که نام "robots" تگ و محتوا به حروف بزرگ و کوچک حساس نیستند.

بدیهی است که نباید دستورالعمل‌های متناقض یا تکراری را مشخص کنید، مانند:

```
<meta name="robots" content="INDEX,NOINDEX,NOFOLLOW,FOLLOW,FOLLOW">
```

رفتاری که احتمالاً تعریف نشده است و مطمئناً از اجرای ربات به اجرای ربات متفاوت خواهد بود.

Search engine META tags

ما فقط تگ‌های متا ربات را مورد بحث قرار دادیم که برای کنترل فعالیت Crawling و Indexing ربات‌های وب استفاده می‌شود. تمام تگ‌های متا ربات‌ها حاوی ویژگی name=”robots” هستند.

بسیاری از انواع دیگر تگ‌های متا در دسترس هستند، از جمله مواردی که در جدول زیر نشان داده شده است. تگ‌های متا DESCRIPTION و KEYWORDS برای ربات‌های موتور جستجو فهرست‌کننده محتوا مفید هستند.

name=	content=	Description
DESCRIPTION	<text>	Allows an author to define a short text summary of the web page. Many search engines look at META DESCRIPTION tags, allowing page authors to specify appropriate short abstracts to describe their web pages. `<meta name="description" content="Welcome to Mary's Antiques web site">`
KEYWORDS	<comma list>	Associates a comma-separated list of words that describe the web page, to assist in keyword searches. `<meta name="keywords" content="antiques,mary,furniture,restoration">`
REVISIT-AFTER [a]	<no. days>	Instructs the robot or search engine that the page should be revisited, presumably because it is subject to change, after the specified number of days. `<meta name="revisit-after" content="10 days">`

Robot Etiquette

در سال ۱۹۹۳، مارتین کوستر، پیشگام در جامعه ربات‌های وب، فهرستی از دستورالعمل‌ها را برای نویسندگان ربات‌های وب نوشت. در حالی که برخی از توصیه‌ها قدیمی هستند، بسیاری از آن‌ها هنوز کاملا مفید هستند. رساله اصلی مارتین، "Guidelines for Robot Writers" را می‌توان در http://www.robotstxt.org/wc/guidelines.html یافت.

جدول زیر یک به روز رسانی مدرن برای طراحان و اپراتورهای ربات ارائه می‌دهد که عمدتاً بر اساس روح و محتوای فهرست اصلی است. بیشتر این دستورالعمل‌ها ربات‌های وب جهانی را هدف قرار داده‌اند. با این حال، آن‌ها برای Crawler های مقیاس کوچکتر نیز قابل استفاده هستند.

Guideline	Description
(1) Identification	
Identify Your Robot	Use the HTTP User-Agent field to tell web servers the name of your robot. This will help administrators understand what your robot is doing. Some robots also include a URL describing the purpose and policies of the robot in the User-Agent header.
Identify Your Machine	Make sure your robot runs from a machine with a DNS entry, so web sites can reverse-DNS the robot IP address into a hostname. This will help the administrator identify the organization responsible for the robot.
Identify a Contact	Use the HTTP From field to provide a contact email address.

Guideline	Description
(2) Operations	
Be Alert	Your robot will generate questions and complaints. Some of this is caused by robots that run astray. You must be cautious and watchful that your robot is behaving correctly. If your robot runs around the clock, you need to be extra careful. You may need to have operations people monitoring the robot 24 × 7 until your robot is well seasoned.
Be Prepared	When you begin a major robotic journey, be sure to notify people at your organization. Your organization will want to watch for network bandwidth consumption and be ready for any public inquiries.
Monitor and Log	Your robot should be richly equipped with diagnostics and logging, so you can track progress, identify any robot traps, and sanity check that everything is working right. We cannot stress enough the importance of monitoring and logging a robot's behavior. Problems and complaints will arise, and having detailed logs of a crawler's behavior can help a robot operator backtrack to what has happened. This is important not only for debugging your errant web crawler but also for defending its behavior against unjustified complaints.
Learn and Adapt	Each crawl, you will learn new things. Adapt your robot so it improves each time and avoids the common pitfalls.
(3) Limit Yourself	
Filter on URL	If a URL looks like it refers to data that you don't understand or are not interested in, you might want to skip it. For example, URLs ending in ".Z", ".gz", ".tar", or ".zip" are likely to be compressed files or archives. URLs ending in ".exe" are likely to be programs. URLs ending in ".gif", ".tif", ".jpg" are likely to be images. Make sure you get what you are after.
Filter Dynamic URLs	Usually, robots don't want to crawl content from dynamic gateways. The robot won't know how to properly format and post queries to gateways, and the results are likely to be erratic or transient. If a URL contains "cgi" or has a "?", the robot may want to avoid crawling the URL.
Filter with Accept Headers	Your robot should use HTTP Accept headers to tell servers what kind of content it understands.
Adhere to *robots.txt*	Your robot should adhere to the *robots.txt* controls on the site.
Throttle Yourself	Your robot should count the number of accesses to each site and when they occurred, and use this information to ensure that it doesn't visit any site too frequently. When a robot accesses a site more frequently than every few minutes, administrators get suspicious. When a robot accesses a site every few seconds, some administrators get angry. When a robot hammers a site as fast as it can, shutting out all other traffic, administrators will be furious. In general, you should limit your robot to a few requests per minute maximum, and ensure a few seconds between each request. You also should limit the total number of accesses to a site, to prevent loops.
(4) Tolerate Loops and Dups and Other Problems	
Handle All Return Codes	You must be prepared to handle all HTTP status codes, including redirects and errors. You should also log and monitor these codes. A large number of non-success results on a site should cause investigation. It may be that many URLs are stale, or the server refuses to serve documents to robots.
Canonicalize URLs	Try to remove common aliases by normalizing all URLs into a standard form.
Aggressively Avoid Cycles	Work very hard to detect and avoid cycles. Treat the process of operating a crawl as a feedback loop. The results of problems and their resolutions should be fed back into the next crawl, making your crawler better with each iteration.

Guideline	Description
Monitor for Traps	Some types of cycles are intentional and malicious. These may be intentionally hard to detect. Monitor for large numbers of accesses to a site with strange URLs. These may be traps.
Maintain a Blacklist	When you find traps, cycles, broken sites, and sites that want your robot to stay away, add them to a blacklist, and don't visit them again.
(5) Scalability	
Understand Space	Work out the math in advance for how large a problem you are solving. You may be surprised how much memory your application will require to complete a robotic task, because of the huge scale of the Web.
Understand Bandwidth	Understand how much network bandwidth you have available and how much you will need to complete your robotic task in the required time. Monitor the actual usage of network bandwidth. You probably will find that the outgoing bandwidth (requests) is much smaller than the incoming bandwidth (responses). By monitoring network usage, you also may find the potential to better optimize your robot, allowing it to take better advantage of the network bandwidth by better usage of its TCP connections.[a]
Understand Time	Understand how long it should take for your robot to complete its task, and sanity check that the progress matches your estimate. If your robot is way off your estimate, there probably is a problem worth investigating.
Divide and Conquer	For large-scale crawls, you will likely need to apply more hardware to get the job done, either using big multiprocessor servers with multiple network cards, or using multiple smaller computers working in unison.
(6) Reliability	
Test Thoroughly	Test your robot thoroughly internally before unleashing it on the world. When you are ready to test off-site, run a few, small, maiden voyages first. Collect lots of results and analyze your performance and memory use, estimating how they will scale up to the larger problem.
Checkpoint	Any serious robot will need to save a snapshot of its progress, from which it can restart on failure. There will be failures: you will find software bugs, and hardware will fail. Large-scale robots can't start from scratch each time this happens. Design in a checkpoint/restart feature from the beginning.
Fault Resiliency	Anticipate failures, and design your robot to be able to keep making progress when they occur.
(7) Public Relations	
Be Prepared	Your robot probably will upset a number of people. Be prepared to respond quickly to their enquiries. Make a web page policy statement describing your robot, and include detailed instructions on how to create a *robots.txt* file.
Be Understanding	Some of the people who contact you about your robot will be well informed and supportive; others will be naïve. A few will be unusually angry. Some may well seem insane. It's generally unproductive to argue the importance of your robotic endeavor. Explain the Robots Exclusion Standard, and if they are still unhappy, remove the complainant URLs immediately from your crawl and add them to the blacklist.
Be Responsive	Most unhappy webmasters are just unclear about robots. If you respond immediately and professionally, 90% of the complaints will disappear quickly. On the other hand, if you wait several days before responding, while your robot continues to visit a site, expect to find a very vocal, angry opponent.

Search Engines

گسترده‌ترین ربات‌های وب توسط موتورهای جستجوی اینترنتی استفاده می‌شود. موتورهای جستجوی اینترنتی به کاربران این امکان را می‌دهند که اسناد مربوط به هر موضوعی را در سراسر جهان پیدا کنند.

امروزه بسیاری از محبوب‌ترین سایت‌ها در وب موتورهای جستجو هستند. آن‌ها به عنوان نقطه شروع برای بسیاری از کاربران وب عمل می‌کنند و خدمات ارزشمندی را ارائه می‌دهند که به کاربران کمک می‌کند اطلاعات مورد علاقه خود را پیدا کنند.

Crawlerهای وب موتورهای جستجوی اینترنتی را با بازیابی اسناد موجود در وب تغذیه می‌کنند و به موتورهای جستجو اجازه می‌دهند تا Indexهایی از کلماتی که در چه اسنادی ظاهر می‌شوند را ایجاد کنند. موتورهای جستجو منبع اصلی ربات‌های وب هستند - بیایید نگاهی گذرا به نحوه کار آن‌ها بیندازیم.

Think Big

زمانی که وب در مراحل ابتدایی خود بود، موتورهای جستجو پایگاه داده‌های نسبتا ساده‌ای بودند که به کاربران کمک می‌کردند اسناد مورد نظر خود را در وب بیابند. امروزه با میلیاردها صفحه قابل دسترسی در وب، موتورهای جستجو برای کمک به کاربران اینترنت مجبور به یافتن اطلاعات ضروری شده و کاملاً پیچیده شده‌اند، چراکه باید برای مدیریت مقیاس وسیعی از وب تکامل یابند.

با میلیاردها صفحه وب و میلیون‌ها کاربر به دنبال اطلاعات، موتورهای جستجو باید Crawler های پیچیده‌ای را برای بازیابی این میلیاردها صفحه وب ایجاد نموده و همچنین می‌بایست برای مدیریت بار پرس و جو ای که میلیون‌ها کاربر ایجاد می‌کنند، ساختارهای پیچیده توازن بار را مستقر نمایند.

به وظیفه یک Crawler وب فکر کنید که باید میلیاردها پرس و جوی HTTP صادر نموده تا صفحات مورد نیاز Search Index را بازیابی کند. اگر تکمیل هر درخواست نیم ثانیه طول بکشد (که احتمالاً برای برخی از سرورها کند بوده و برای برخی دیگر سریع است)، هنوز هم (برای ۱ میلیارد سند) زمان زیادی طول خواهد کشید:

0.5 seconds × (1,000,000,000) / ((60 sec/day) × (60 min/hour) × (24 hour/day))

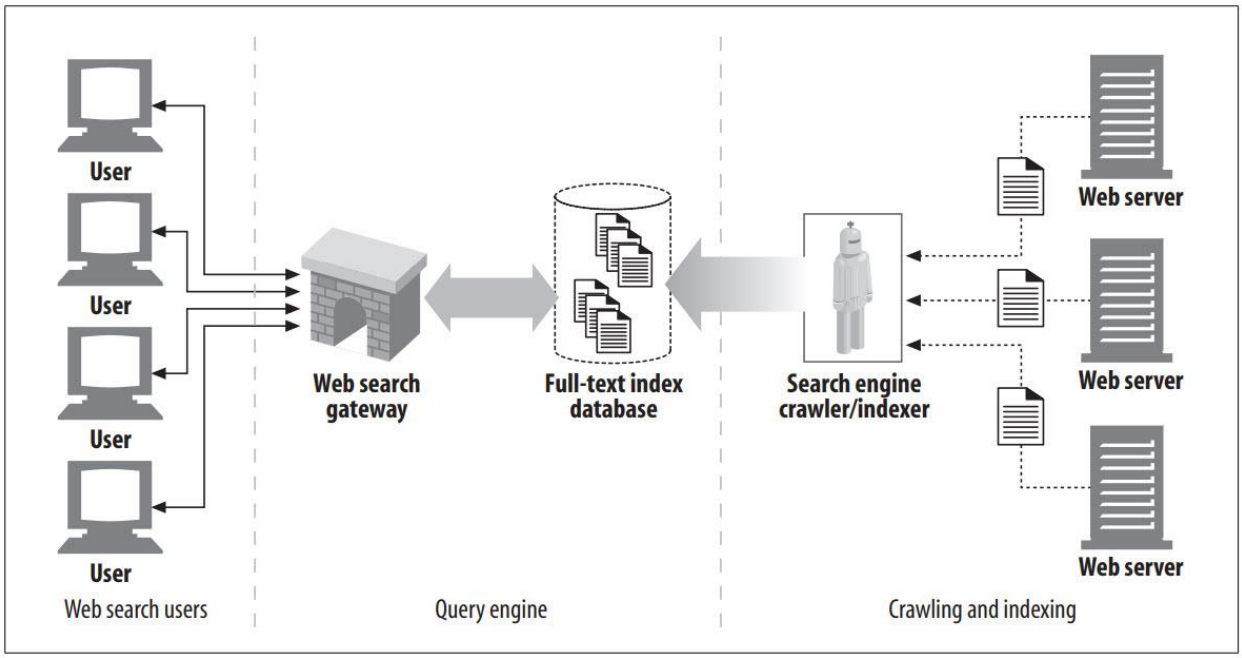
واضح است که Crawler های مقیاس بزرگ باید باهوش‌تر باشند، درخواست‌ها را موازی‌سازی کنند و از بانک‌های ماشین‌ها برای تکمیل کار استفاده کنند. با این حال، به دلیل مقیاس آن، تلاش برای Crawling در کل وب هنوز یک چالش دلهره آور است.

Modern Search Engine Architecture

موتورهای جستجوی امروزی پایگاه‌های اطلاعاتی محلی پیچیده‌ای به نام «full-text indexes» درباره صفحات وب در سراسر جهان و محتوای آن‌ها می‌سازند. این Index ها به عنوان نوعی کاتالوگ کارت برای تمام اسناد موجود در وب عمل می‌کنند.

Crawler های موتورهای جستجو، صفحات وب را جمع آوری کرده و به خانه می‌آورند و به full-text index اضافه می‌کنند. در همان زمان، کاربران موتورهای جستجو از طریق دروازه‌های جستجوی وب مانند HotBot (http://www.hotbot.com) یا گوگل (http://www.google.com) پرس و جوهایی را بر اساس full-text index صادر می‌کنند. از آنجایی که صفحات وب دائماً در حال تغییر هستند و به دلیل زمان زیادی که ممکن است برای Crawling بخش بزرگی از وب طول بکشد، full-text index در بهترین حالت یک snapshot از وب است.

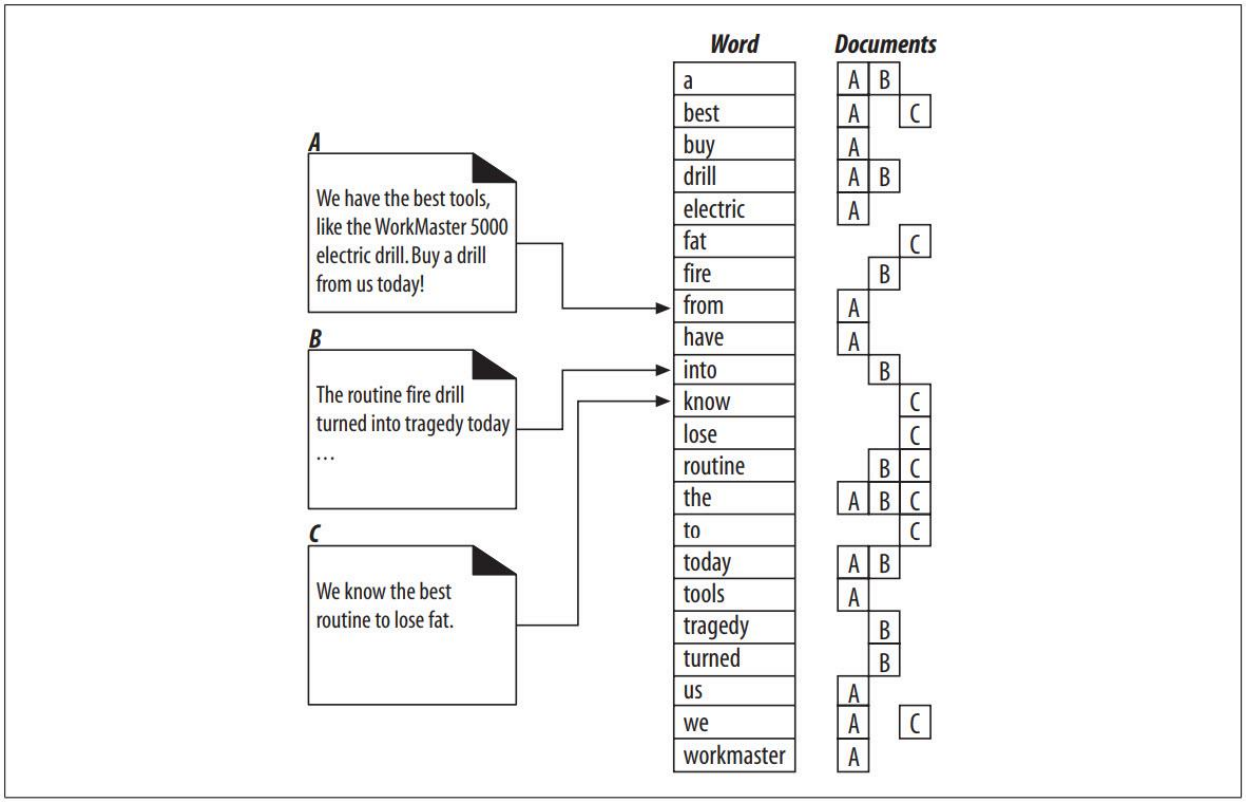
معماری سطح بالا یک موتور جستجوی مدرن در شکل زیر نشان داده شده است.



Full-Text Index

full-text index پایگاه داده ای است که یک کلمه را می‌گیرد و بلافاصله تمام اسناد حاوی آن کلمه را به شما می‌گوید. خود اسناد پس از ایجاد Index نیازی به اسکن ندارند.

شکل زیر سه سند و full-text index مربوطه را نشان می‌دهد. full-text index اسناد حاوی هر کلمه را فهرست می‌کند.



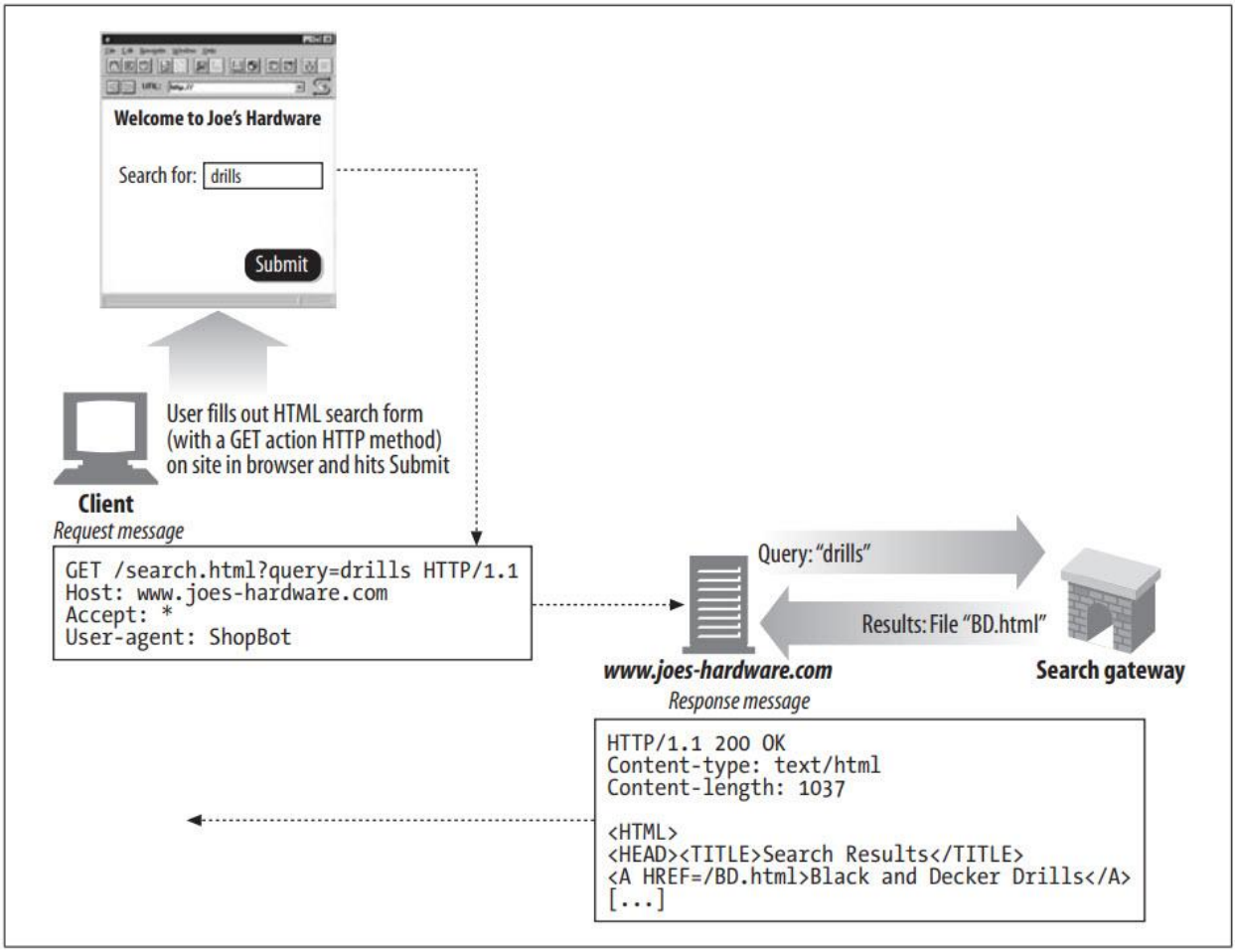
به عنوان مثال:

- کلمه a در اسناد A و B آمده است.
- کلمه best در اسناد A و C آمده است.
- کلمه drill در اسناد A و B آمده است.
- کلمه routine در اسناد B و C آمده است.
- کلمه the در تمامی اسناد A، B و C آمده است.

Posting the Query

هنگامی که کاربر یک پرس و جو را برای یک Gateway موتور جستجوی وب ارسال می‌کند، یک فرم HTML را پر می‌کند و مرورگر او فرم را با استفاده از یک درخواست HTTP GET یا POST به Gateway هدایت می‌کند. برنامه Gateway عبارت جستجو را استخراج می‌کند و پرس و جوی رابط کاربری وب را به عبارت مورد استفاده برای جستجوی full-text index تبدیل می‌کند.

شکل زیر یک پرس و جو ساده کاربر را در سایت www.joes-hardware.com نشان می‌دهد. کاربر عبارت "drills" را در فرم جعبه جستجو تایپ می‌کند و مرورگر آن را به یک درخواست GET با پارامتر query به عنوان بخشی از URL ترجمه می‌کند. وب سرور Joe's Hardware درخواست را دریافت می‌کند و آن را به برنامه Gateway جستجوی خود می‌دهد، که لیست حاصل از اسناد را به وب سرور برمی‌گرداند. در انتها نیز نتایج را به یک صفحه HTML برای کاربر قالب بندی می‌کند.



Sorting and Presenting the Results

هنگامی که یک موتور جستجو از فهرست خود برای تعیین نتایج یک پرس و جو استفاده می‌کند، برنامه Gateway نتایج را می‌گیرد و صفحه نتایج را برای کاربر نهایی می‌سازد.

از آنجایی که بسیاری از صفحات وب می‌توانند حاوی هر کلمه‌ای باشند، موتورهای جستجو از الگوریتم‌های هوشمندانه‌ای استفاده می‌کنند تا نتایج را رتبه بندی کنند. به عنوان مثال، در شکل پیشین کلمه "best" در چندین سند ظاهر می‌شود. موتورهای جستجو باید از ترتیبی که باید فهرست اسناد

نتیجه را ارائه کنند بدانند تا مرتبط‌ترین نتایج را به کاربران ارائه دهند. این موضوع Relevancy Ranking نامیده می‌شود که شامل فرآیند امتیازدهی و ترتیب فهرستی از نتایج جستجو است.

برای کمک بهتر به این فرآیند، بسیاری از موتورهای جستجوی بزرگتر در واقع از داده های سرشماری جمع آوری شده در طول Crawling در وب استفاده می‌کنند. برای مثال، شمارش تعداد لینک‌هایی که به یک صفحه معین اشاره می‌کنند می‌تواند به تعیین محبوبیت آن کمک کند و از این اطلاعات می‌توان برای وزن دادن به ترتیب ارائه نتایج استفاده کرد. الگوریتم‌ها، نکات مربوط به Crawling و سایر ترفندهای مورد استفاده موتورهای جستجو، برخی از محافظت‌شده‌ترین اسرار آن‌هاست.

Spoofing

از آنجایی که کاربران اغلب وقتی آنچه را که به دنبال آن هستند در چند نتیجه اول جستجوی جستجو نمی‌بینند، ناامید می‌شوند، ترتیب نتایج جستجو می‌تواند در یافتن یک سایت مهم باشد.

انگیزه زیادی برای وبمسترها وجود دارد که سعی کنند سایت‌های خود را در نزدیکی بالای بخش نتایج برای کلماتی که فکر می‌کنند بهترین توصیف سایتشان است، فهرست کنند، به خصوص اگر سایت‌ها دارای خدمات تجاری بوده و به کاربران برای یافتن آن‌ها و استفاده از آن‌ها متکی هستند.

این تمایل برای فهرست بهتر منجر به بازی‌های زیادی در سیستم جستجو شده است و یک کشمکش دائمی بین اجراکنندگان موتورهای جستجو و کسانی که به دنبال فهرست برجسته سایت‌های خود هستند ایجاد کرده است. بسیاری از وب مسترها هزاران کلمه کلیدی (بعضی نامربوط) را لیست می‌کنند و صفحات Fake یا جعلی را به کار می‌برند - حتی برنامه‌های دروازه ای که صفحات جعلی تولید می‌کنند که ممکن است الگوریتم‌های مرتبط موتورهای جستجو را برای کلمات خاص فریب دهند.

در نتیجه همه این‌ها، پیاده‌کنندگان موتورهای جستجو و ربات‌ها باید دائماً الگوریتم‌های مربوط به خود را تغییر دهند تا بهتر این جعل‌ها را شناسایی نمایند.

For More Information

http://www.robotstxt.org/wc/robots.html

http://www.searchengineworld.com

http://www.searchtools.com

http://search.cpan.org/doc/ILYAZ/perl_ste/WWW/RobotRules.pm

http://www.conman.org/people/spc/robots2.html

Managing Gigabytes: Compressing and Indexing Documents and Images