UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO
INE5644 - DATA MINING

RELATÓRIO WEB CRAWLERS

Augusto Verzbickas
Eduardo Ferreira Mocelin
Milton Bittencourt de Souza Neto
Renata Tomaz Siega

FLORIANÓPOLIS, 2013

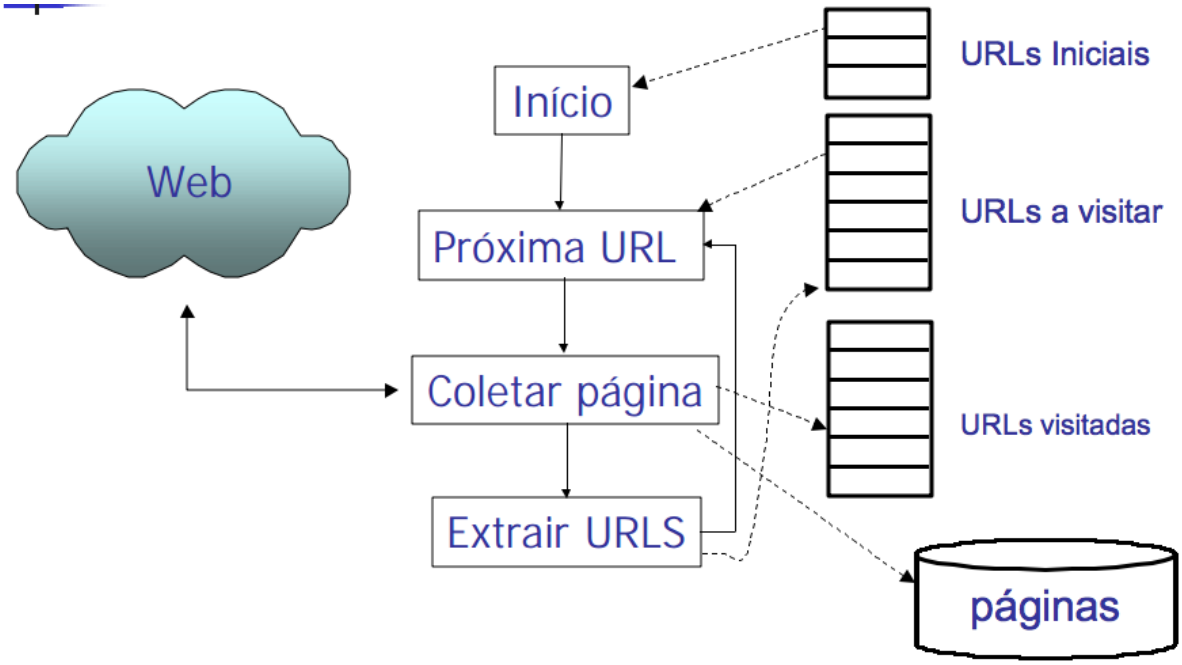
SUMÁRIO

1. INTRODUÇÃO

Hoje a internet possui cerca de 3.77 bilhões de páginas indexadas, esse grande número de informações é originado pelo crescente avanço nas tecnologias da informação e comunicação. Além das páginas *web*, é crescente o número de *e-mails*, *e-books*, *logs* de sistema, currículos, manuais dos mais diversos tipos, relatórios, entre outros tipos de informação.

Diante disso surge o desafio e a oportunidade de coletar e integrar todas essas informações para produzir vantagem competitiva para as organizações. Para atuar nesse sentido, pode ser utilizado um *Web Crawler*, que é basicamente um software que navega pela *World Wide Web* de forma metódica e automatizada.

Um *Web Crawler*, que pode ser chamado também de *Web Spider*, *ant*, *automatic indexer* ou *Web scutter*, é um programa que faz *downloads* de páginas *Web* para criar índices ou cache. Pode ser aplicado em diversas áreas com o intuito de obter informações e até mesmo mapear uma determinada gama de sites. Muito utilizado por empresas que possuem motores de busca como o Google, devido a necessidade de obter dados e indexar páginas ou empresas que utilizem essas informações para conhecimento estratégico.



Um *Web Crawler* inicia o seu processo com um conjunto de URLs chamados 'sementes', ao visitar cada URL faz *download* da página, coleta os *links* existentes nessa página e coloca eles na fila junto com as demais URLs. Esse processo se repete até que alguma condição de parada seja imposta, normalmente o número de URLs visitadas.

2. DESAFIOS

Os desafios ao implementar um *Web Crawler* são muitos, dado que a internet possui tantas paginas que é dificil estimar quantas, onde 3.77 bilhões estão indexadas por motores de busca, totalizando cerca 5 milhões de terabytes em conteúdo, e que crescem cada dia mais. A expectativa é que em 2014 77% das pessoas de países globalizados utilizem a internet.

Páginas novas continuam a surgir e as que já existem são continuamente atualizadas. Em torno disso alguns desafios devem ser levados em consideração:

a. Que páginas o *Web Crawler* deve fazer *download*? Não é possível para o *crawler* fazer download de todas as páginas na *Web*, por isso é importante que o *Web Crawler* cuidadosamente selecione primeiramente as páginas "importantes" para visitar primeiro, para que ao menos essas páginas visitadas sejam significantes.

b. Como escolher quais páginas revisitar para atualizar seus *downloads*? Uma vez que o *Web Crawler* fez *download* de diversas páginas é necessário que ele começe a revisitá-las para detectar atualizações a fim de manter seus *downloads* atualizados. Nem todas as páginas na internet são atualizadas com uma mesma frequência sendo assim é necessário que o *Web Crawler* decida de maneira inteligente quais páginas revisitar, com mais ou com menos frequência.

c. Como minimizar o acesso aos sites visitados? Cada vez que o *Web Crawler* acessa um site e coleta páginas consome recursos de terceiros. Para evitar que terceiros sintam esse impacto de uma forma negativa, é necessário que o projeto do *Web Crawler* seja feito de tal forma a minimizar o impacto sobre tais recursos.

d. Como os processos do *Web Crawler* irão executar em paralelo? Dado o tamanho da *Web* é necessário que *Web Crawlers* executem em paralelo em diferentes máquinas para fazer o maior número de *downloads* no menor tempo possível. Surgiu assim a necessidade da criação de técnicas de escalonamento com o intuito de uma busca mais eficiente, essas técnias coordenam as ações dos *Web Crawlers* e garantem que não haverá *downloads* repetidos. As formas de escalonamento podem ser: em profundidade ou LIFO, em largura ou FIFO, baseadas em *ranking* de URLs e comparativo entre as técnicas de escalonamento. No entanto o desafio nesse ponto é que para estabelecer a coordenação entre os *Web Crawlers* é gerada uma sobrecarga que acabar por limitar o número de processos executando simultâneamente.

e. Como integrar dados estruturados e dados não estruturados? Uma forma é utilizando *linked data*, que pode ser entendido como uma maneira de publicar, interligar e estruturar dados na *Web*. Através das técnicas do *linked data*, podem

ser conectados dados de qualquer domínio a outros dados sejam estes de domínios relacionados ou não.

f. Como adicionar semântica aos dados? Uma forma de definir semântica aos dados é criando uma ontologia, a qual pode ser utilizado tanto por humanos quanto por agentes de *software*, a fim de estabelecer um entendimento comum sobre os conceitos e relacionamentos desse domínio de conhecimento.

3. POLÍTICAS

Para garantir que o *Web Crawler* funcione de acordo, é necessária a implementação de algumas políticas.

a. Seleção

Com a quantidade enorme de páginas existentes o *Web Crawler* deve decidir quais páginas são mais relevantes, o que requer uma métrica de priorização que podem ser usadas separadamente ou combinadas.

Para isso, algumas métricas são utilizadas, para classificar as páginas por nível de importância de cada página, de acordo com o interesse de quem usa o processo de *crawling*. As métricas são as seguintes:

i. Similarity to a Driven Query

Para cada página, a importância da mesma é dada pela similaridade dos termos citados na página com um ou mais termos de busca inseridos no *crawler*. Por exemplo, se utilizarmos um *crawler* para buscar páginas de esportes, os termos de busca serão esportes e outros que tangem este universo, como futebol, basquete, etc. Logo, as páginas que contiverem uma ou mais destas palavras, terão maior nível de importância, proporcional também ao número de ocorrências por página.

ii. Backlink Count

Esta métrica calcula a importância de cada página conforme a quantidade de links por toda a *web* que a referenciam. Quanto maior a quantidade de *links* que apontam para determinada página, maior a sua importância. Por exemplo, se a página www.uol.com.br é citada 300 vezes e a página www.terra.com.br é citada 250 vezes, a página www.uol.com.br tem um maior nível de importância.

iii. PageRank

Esta métrica possui uma metodologia de calcular a importância de uma página

similar ao ***Backlink Count***, aumentando a importância de cada página conforme a quantidade de *links* na web que redirecionam para ela. Só que no caso do ***PageRank***, é levado em conta também a importância da página que possui o *link*. Por exemplo, aumenta mais a importância de uma página se ela tiver 1 *link* na página da Apple apontando para ela do que se tiver 1 *link* apontando para ela em uma página pessoal de um aluno comum de uma faculdade.

iv. Forward Link Count

Esta métrica calcula a importância de uma página conforme a quantidade de links para outras páginas contidas nela. Quanto mais links esta página tiver para outras páginas, maior nível de importância ela tem. A lógica dessa métrica é que uma página com muitos links seria uma espécie de "Diretório *Web*". Por exemplo, o Google, que faz referência a bilhões de outras páginas, é provavelmente a página com maior nível de importância para essa métrica hoje na web.

v. Location Metric

A métrica *Location Metric* calcula a importância de uma página conforme o seu endereço. Não é considerado o conteúdo da página, mas sim se o seu endereço é relevante para o *crawler*. Esta métrica pode levar em conta o domínio, um domínio **.br** pode ser mais relevante que um domínio **.is** para um *crawler*. Um endereço com alguma das palavras chave do *crawler* também se torna mais relevante. Podemos considerar também um endereço com mais caraceres "/" menos relevante do que um com menos.

Por exemplo, caso tenhamos um crawler buscando por páginas brasileiras sobre esportes, a página **www.jornaldeesportes.com.br/futebol** é mais relevante do que a página com endereço **www.jornalvolei.com.pt/times/cimed**.

As métricas citadas podem ser combinadas, a fim de especializar o *crawler* para os interesses de quem o utiliza.

b. Revisita

O processo de *crawling* pode demorar muito tempo, e no decorrer desse tempo é possível que as páginas já visitadas tenham sido atualizadas, e possuir informações desatualizadas tem um custo. Para que as informações de posse do cliente não se tornem desatualizadas o *Web Crawler* deve periodicamente acessá-las novamente em busca de atualizações. As medidas Freshness e Age indicam o quanto uma página é atual e nova, como mostram as funções a seguir:

i. Freshness

$$F(e_i;t) = \begin{cases} 1 & \text{if } e_i \text{ is up-to-date at time } t \\ 0 & \text{otherwise.} \end{cases}$$

ii. Age

$$A(e_i;t) = \begin{cases} 0 & \text{if } e_i \text{ is up-to-date at time } t \\ t - \text{modification time of } e_i & \text{otherwise.} \end{cases}$$

As páginas podem ser revisitadas com uma mesma frequência independentemente do quanto elas são atualizadas no decorrer do tempo ou revisitada com uma frequência proporcional à atualização de seu conteúdo. No entanto, deve-se adicionar mais informações quanto à qualidade do conteúdo de cada página para atingir uma política mais realista, pois num cenário real o conteúdo de cada página tem diferentes valores agregados.

c. Cortesia (Politeness)

Um *Web Crawler* faz inúmeras requisições o que pode impactar na performance do site que está acessando. O que acontece é que inúmeros *Web Crawlers* podem estar executando seus processos sobre o mesmo servidor, o que claramente tornará dificil para o servidor atender à essas diversas requisições. Os prejuízos de se usar *Web Crawlers* são:

i. Recursos de rede, pois o crawler consome muita largura banda e opera com um alto grau de paralelismo durante um longo período de tempo;

ii. Sobrecarga do servidor, especialmente se a frequência de acesso a um servidor é muito alta;

iii. Código mal escrito, que podem quebrar servidores ou roteadores, ou até mesmo o download de páginas com as quais não conseguem lidar;

iv. Interrupção de redes e servidores *web*, ocasionados por muitos *deploys* de *Web Crawler* pessoais em um servidor.

Uma solução para esses problemas é a convenção do uso de um protocolo de exclusão localizado em um arquivo chamado *robots.txt* na raíz do servidor. Ao acessar esse arquivo o *Web Crawler* saberá quais páginas não devem ser acessadas durante o processamento.

d. Paralelização

Dado o grande número de dados na *Web*, é necessário que o *Web Crawler* execute em múltiplas máquinas e façam os *downloads* das páginas em processos paralelos. Essa paralelização é necessária para baixar um grande número de páginas em um tempo relativamente baixo. Logicamente, deve haver um cuidado na programação destes *Crawlers* para garantir que trabalhem de forma coordenada e não visitem o mesmo site múltiplas vezes em paralelo. Algumas questões devem ser levadas em consideração no desenvolvimento de um *parallel crawler*:

i. Overlap

Um processo de um *parallel crawler* pode não estar informado de que outro processo já fez *download* da página em questão, sendo assim é necessário que os processos sejam coordenados para evitar *Overlap*.

ii. Quality

Quando os processos de um *parallel crawler* fazem *downloads* das páginas é necessário que decidam quando uma página é "importante" para ser salva, no entanto, cada processo tem apenas uma visão geral das páginas que ele percorreu e não das dos demais. Nesse contexto é necessário que um *parallel crawler* tenha condições de fazer boas decisões sobre quais páginas são "importantes" tanto quanto um Web Crawler de um único processo.

iii. *Comunication bandwith*

Para implantar técnicas que evitem o *Overlap* e garantam *Quality* é necessário que haja comunicação entre os processos do *parallel crawler*. No entanto, é necessário minimizar a comunicação mantendo a eficácea dos processos pois o número de processos se comunicando pode gerar um *overhead* de comunicação.

iv. Scalability

Dado o enorme tamanho da Web, pode ser mandatório executar um *parallel crawler* para alcançar uma alta taxa de *downloads* em alguns casos.

v. *Network-load dispersion*

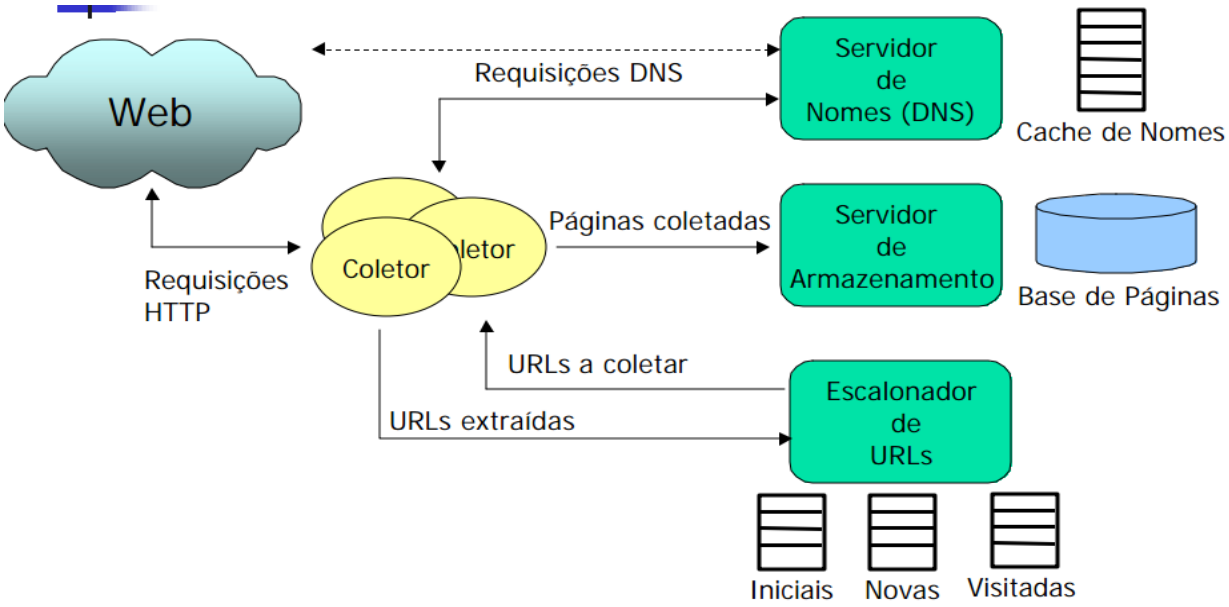
Um *parallel crawler* em larga escala pode ter processos executando em diferentes locais geograficamente distantes a fim de dispersar a carga sobre a rede. Assim é possível que os processos façam download de páginas localizadas em diferentes lugares do mundo.

vi. *Network-load reduction*

Ao dispersar geograficamente a carga sobre a rede os processos de um *parallel crawler* acaba por diminuir a carga sobre a rede global pois as páginas apenas atravessam as redes locais.

4. ARQUITETURA

Um *Web Crawler* genérico segue a seguinte arquitetura:

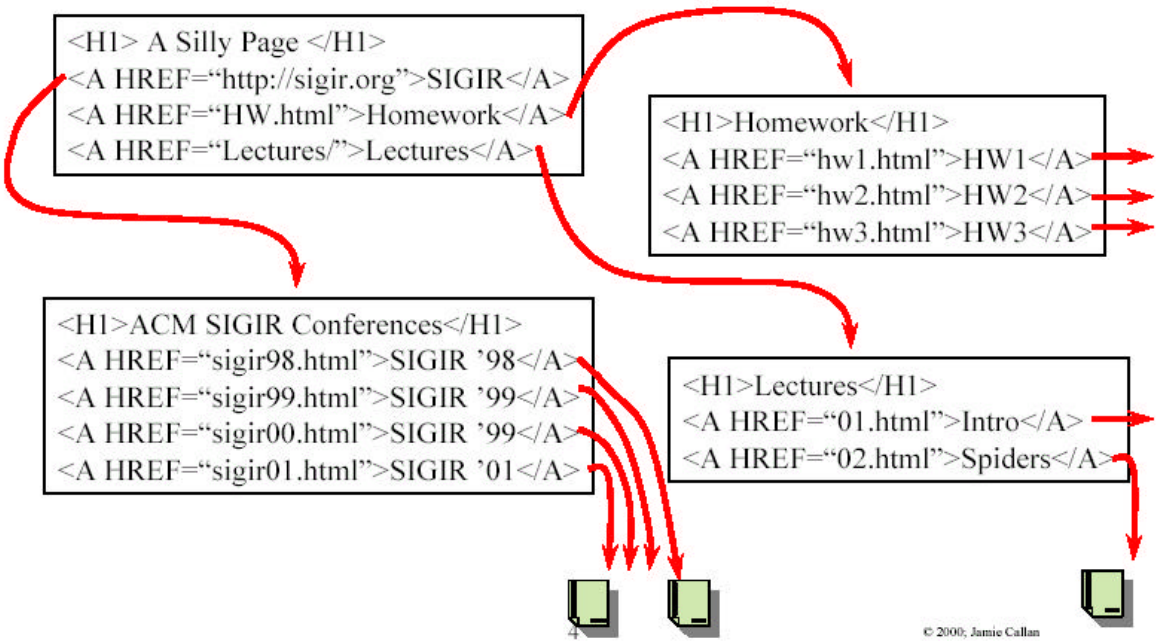


As informações disponíveis em Páginas Web, RDBMS, Wikis, E-mails, Search Logs e Documentos são coletadas e indexadas. Também são criadas ontologias para entender o significado dessas informações, e desta forma indexa-lás da melhor maneira possível. Mais genericamente, pode ser entendido como o processo de coleta de informação a partir de várias fontes e formatos. Devido ao alto volume de informação cada vez mais crescente e evolução das mídias, caracteriza-se como um processo complexo.

a. Coletores

Os coletores tem como principais funções:

- Fazer requisições de páginas aos servidores HTTP;
- Extrair os links das páginas recebidas e enviar ao escalonador;
- Requisitar do escalonador uma ou mais URLs a serem coletadas;
- Realizar um escalonamento local (*short term scheduling*).



b. Servidor de Armazenamento

O Servidor de Armazenamento desempenha as seguintes funções:
- Receber as páginas ou outros objetos coletados e armazenar em uma base local;
- Fazer a extração (*parsing*) de texto;
- Tratar formatos variados como: Postscript, PDF, Word, Powerpoint, etc.

c. Servidor de Nomes

O Servidor de Nomes é responsável por:
- Atender requisições DNS dos coletores;
- Manter um cache de identificadores DNS (nomes) resolvidos;
- Evitar que cada coletor faça requisições DNS remotas;

Além disso, o Servidor de Nomes é um potencial ponto de gargalo do *Web Crawler*.

d. Escalonador

O Escalonador tem como função:
- Decidir qual a próxima URL a ser coletada;
- Coordenar as ações dos coletores;
- Garantir protocolo de exclusão;
- Garantir o retardo mínimo entre requisições a um mesmo servidor HTTP;
- Garantir que não haverão coletas repetidas.

Assim como o Servidor de Nomes, o Escalonador é um potencial ponto de gargalo do

Crawler.

5. IDENTIFICAÇÃO

Web Crawlers tipicamente se identificam com o *Web Server* usando o campo User agent de um HTTP Request. Com isso, administradores checam a quantidade e periodicidade de visitas de *Web Crawlers* checando no cabeçalho das requisições HTTP se a identificação é de um *User Agent*. O *User Agent* pode incluir também uma URL onde o administrador do site pode encontrar informações sobre o *crawler*. Os logs de acesso permitem descobrir o endereço Ip,o dia e hora do acesso, o tipo de pedido(GET,POST),o que foi pedido e o User-Agent. Examinar logs de servidores é uma tarefa extremamente exaustiva e repetitiva, portanto os administradores utilizam ferramentas para auxiliar a identificar, rastrear e verificar *web crawlers*. Há a possibilidade de não enviar a identificação ou então mandar uma identificação falsa.

Spambots e outros *crawlers* maliciosos tentam mascarar sua função identificando-se como *browsers* ou outros *web crawlers* conhecidos. É importante que exista essa identificação, caso seja necessário que os administradores precisem entrar em contato. Em alguns casos, os *crawlers* podem ficar presos nas chamadas *crawler traps* (armadilhas contra *crawlers*) do servidor ou estar sobrecarregando o servidor, desta forma o responsável deve parar o *crawler*. A identificação também é útil para administradores que estão interessados em saber quando eles podem esperar visitas de *Web Crawlers* de uma ferramenta de busca específica.

6. RECOMENDAÇÕES

Principais dicas, recomendações e boas práticas relacionadas ao uso de *Web Crawlers*:

- Evitar “Rapid Fire” Usar retardo mínimo entre requisições a um mesmo servidor HTTP
  - Tipicamente 60 segundos
- Usar header “Usent Agent” Prover as informações necessárias para os administradores de site
  - Nome do robô, e-mail, responsável, instituição, etc.
- Evitar coleta maciça em horas de tráfego alto
- Tratar problema da duplicação de páginas
  - Inconveniente para coleta
  - Esforço de coleta inútil
  - Espaço de armazenamento desnecessário
  - Inconveniente para processamento de consultas
  - Maior demanda de processamento
  - Prejuízo na avaliação de similaridade
  - Inconveniente para os usuários
- Cuidar de sites gigantes
  - Limitar o número de páginas coletadas uma vez que alguns sites contém um número excessivamente grande páginas

- Tipos de páginas
  - Não devem ser coletados objetos que não podem ser indexados: .exe, .jpg, .gif, .ram, .au, …
- Utilizar regras de exclusão para página
  - Uso de metatags em páginas HTML
  - Não coletar, seguir links *<meta name="ROBOTS" content="NOINDEX">*
  - Coletar, não seguir links *<meta name="ROBOTS" content="NOFOLLOW">*
  - Não indexar, não seguir links *<meta name="ROBOTS" content= "NOINDEX,NOFOLLOW">*

7. EXEMPLOS

Grandes empresas que possuem motores de busca utilizam *Web Crawlers* para auxiliarem na indexação das páginas. O Google possui o **Googlebot**. O Yahoo! possui o **Yahoo! Slurp**. Já o Bing possui o **Msnbot**.

Existem também alguns exemplos de *Web Crawlers open source*. O **Methabot** é um *Web Crawler* desenvolvido em C, com foco em velocidade e flexibilidade. O **HTTrack** é um navegador *offline* distribuído como software livre de código aberto que permite o download de páginas web, desde que não tenha sido bloqueado o download pelo arquivo *robots.txt*. Além destes existem vários outros *web crawlers* famosos, como o arachnode.net, Goutte, Wget, Datapark Search, Crawljax, entre outros.

Para demonstração em sala de aula, contruímos dois crawlers baseados na api java crawler4j. Os códigos estão disponíveis no seguinte endereço:

Webcrawlercolector:

https://github.com/miltonbsn/webcrawlercolector.git

Webcrawlernavigation:

https://github.com/miltonbsn/webcrawlernavigation.git

8. REFERÊNCIAS

SILVA, Altigran; MOURA, Edleno. Web Crawling. Acessado em Outubro/2013. http://www.eicstes.org/EICSTES_PDF/PRESENTATIONS/Web%20crawling%20-%20Coleta%20automática%20na%20web%20(Silva-Moura).PDF.

http://www.seomarketing.com.br/robots.txt.html. Acessado em Outubro/2013.

http://www.worldwidewebsize.com/ Acessado em Outubro/2013

https://oak.cs.ucla.edu/~cho/papers/cho-thesis.pdf Acessado em Outubro/2013

http://www.oficinadanet.com.br/artigo/otimizacao__seo/qual-a-diferenca-entre-robospider-e-crawler. Acessado em Outubro/2013.

BERNERS-LEE, T. The Internet Engineering Task Force, Janeiro 2005. Disponível em: <http://www.ietf.org/rfc/rfc3986.txt>. Acessado em Outubro/2013.

KOBAYASHI, M.; TAKEDA, K. Information retrieval on the web. ACM Computing Surveys (CSUR), v. 32, n. 2, p. 144-173, 2000. ISSN 0360-0300.

http://www.worldwidewebsize.com/ Acessado em Outubro/2013.

http://www.wisegeek.org/how-big-is-the-internet.htm Acessado em Outubro/2013.

http://en.wikipedia.org/wiki/File:Internet_users_per_100_inhabitants_ITU.svg Acessado em Outubro/2013.