Explorer
Web Crawler

Carlos Henrique P. da Silva, Lucas de Moraes Martins

Resumo

Web crawler desenvolvido para obter meta-dados (conteúdo HTML) do website desejado.

1 Introdução

Os também conhecidos como "indexadores", *web crawlers* foram desenvolvidos para percorrer a *web* em busca da extração de dados ou meta-dados, uma vez que a sua mesma coleta de informação pode ser utilizada para milhares de situações, desde portais de busca tais como *Google* e *Bing* a estatísticas representadas por algum tipo de visualização de dados.

Ao entrar em detalhes do funcionamento e de definição de um *web crawler*, podemos dizer que o princípio básico seria: o *software* deve navegar entre endereços *web* de forma automática depois de ser executado com seus parâmetros iniciais. Além de ter um propósito em específico, seja coletar e/ou indexar conteúdo.

Basicamente, seu ciclo de navegação pode ser descrito da seguinte maneira:

- Ser executado com parâmetros iniciais (*url* inicial e objetivo de busca na navegação);
- Rastrear/procurar por seu objetivo;
- Encontrar meios de navegação para outro endereço (*links*);
- Indexar os *links* encontrados.
- Entrar em cada *link* encontrado e repetir o processo até que o parâmetro de finalização da busca seja ativado, mesmo que seja algum tipo de erro.

2 Objetivos

O objetivo proposto é de entendimento simples: trazer meta-dados (conteúdo HTML) juntamente com suas respectivas referências, as chamadas de *hypertext references* ou de *sources*. Assim, entre os mesta-dados escolhidos para a visualização estão:

1. Links;
   Sendo toda e qualquer referência de um objeto da *internet*, ou seja, atributos *src* de imagens, *urls* das páginas encontradas, *links* externos e âncoras.

2. Vídeos;
   Referências extraídas de *iframes* e que estão contidas em:

   - www.youtube.com/watch?
   - player.vimeo.com/video

3. Imagens;
   Referências de *tags* IMG.

4. Documentos:
   Referências extraídas de *iframes* e que estão contidas em:

   - www.slideshare.net

3 Metodologia

3.1 Linguagem Python

Na criação do Web Crawler, foi realizado com a linguagem Python em sua versão 2.7 e com o sistema de gerenciamente de banco de dados SQLite, também foi utilizado no projeto o framework CherryPy como servidor web com o objetivo de fornecer objetos estáticos como HTML e CSS para o client. Para o parse do HTML foi usado o BeautifulSoup. Uma das técnicas foi utilizar SSE (Server Sent Events), com fins de tornar a comunicação de server-client rápida e por sua vez a visualização de dados.

O Python é uma linguagem de programação que possue estruturas de dados de alto nível, a linguagem se torna ideal para desenvolvimento rápido de aplicações de diversas áreas e na maioria das plataformas por ser de fácil aprendizado e totalmente expressiva.

3.2 Comunicação

Ao decorrer dos anos já foram abordados diversas técnicas e protocolos de comunicação entre cliente e servidor, uma das mais famosas técnicas de comunicação em tempo real com o cliente foi a Websocket API, a mesma providência boa comunicação bi-direcional entre cliente e servidor.

No modelo de crawler proposto, pouco agregaria valor o cliente enviar informações para o servidor e poolings são muito custosos, esse modelo geralmente é utilizado em games que existe extrema comunicação de ambas as partes, a alternativa para o problema foi procurar algo que fizesse o mesmo tipo de trabalho mas, funcionasse de forma unidirecional, isso dado o fato que o modelo somente o servidor precisava avisar ao cliente que existia novas informações, e a fim de reduzir o numero de requesições a solução foi baseada em uma api que trabalhasse com eventos, então somente havendo um objeto para sincronizar com o

cliente que existiria custo propriamente dito, então foi utilizado uma API mais simples e tão eficiente quanto, para trabalhar a comunicação unidirecional e baseada em eventos, SSE (Server Sent Events) API, para o entendimento da comunicação só é necessário a indicação de um protocolo de Event-Stream nos headers de resposta.

A comunicação é feita quando o crawler acha alguma url e após a inserção no banco, o cliente recebe uma informação de que um novo evento foi disparado, dado isso as informações são passadas para o front-end em real-time.

3.3 Front-end

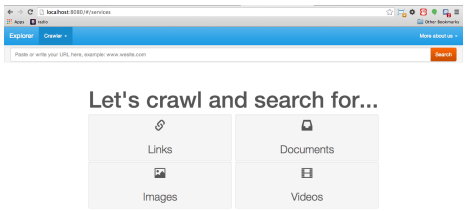
Para a realização do front-end foi utilizado a linguagem CoffeeScript/JavaScript, estilização de páginas com SASS/CSS/CSS3, markup com HTML/HTML5, com a utilização Yeoman para scaffolding, modern workflow e gerenciamento de pacotes juntamente com o Bower, além da incorporação de frameworks como: AngularJS, UnderscoreJS, GruntJS, Bootstrap (Twitter), Compass e JQuery.

3.4 Workflow

A aplicação é baseada em cinco passos, sendo:

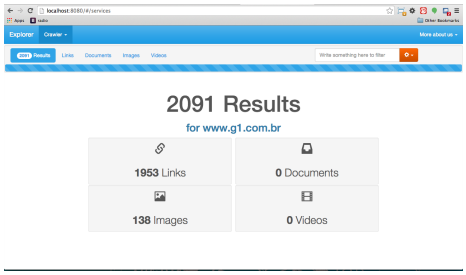
- Receber o site onde se deseja efetuar a busca dos meta-dados;
- Disparar o crawler no site desejado;
- Armazenar em lotes os meta-dados encontrados pelo crawler no banco de dados;
- Recuperar os items salvos no banco de dados;
- Repassar os items para o *client.*

4 Resultados e Discussão

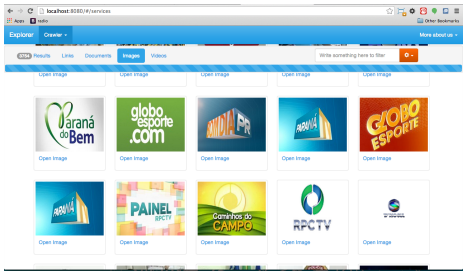


Baseado nas experiências dos usuários as informações apresentadas em tempo real foram um sucesso, a cada quantidade de informações recolhidas e apresentadas, as pessoas envolvidas realmente tiveram a percepção de que estavam vendo informações do site qual o crawler estava fazendo a busca, considerado por todos

uma falha foi a escolha da forma que os links dos sites foram apresentados, a falta de mais dados também foi requestionado, a quantidade itens que foram apresentadas foi considerada boa.



Buscas em sites ou portais de grande porte podem ser indexados em tempo real e o usuário não precisa esperar o crawler terminar toda a busca para ter algo palpável a sua frente, pois consegue obter resultados por demanda.



A arquitetura do crawler foi considerado boa, apesar de algumas ressalvas que foram levantadas perante a modelagem de dados no banco e também na sincronização de envio de informações.

5 Conclusão

Conseguimos chegar aos objetivos de mostrar as informações em tempo real para o usuário, algumas considerações ficaram para a apresentação de links, que deve ser melhor estudada para uma segunda versão, além disso adicionar mais tipos de meta-dados como documentos e videos foram propostos.

Referências

[1] **Python Software Foundation, DocumentacaoPython**. *Disponível em:* http://www.python.org.br/wiki/DocumentacaoPython, Acesso em: 20 de out. 2013.

[2] **SQLite Documentation**. *Disponível em:* http://www.sqlite.org/docs.html, Acesso em: 23 de out. 2013.

[3] **CoffeScript Documentation**. *Disponível em:* http://coffeescript.org, Acesso em: 23 de out. 2013.

[4] **Yeoman Documentation**. *Disponível em:* https://github.com/yeoman/yeoman/wiki, Acesso em: 27 de out. 2013.