Crawler CVM: Aspectos técnicos da captura de dados financeiros de empresas de capital aberto disponíveis no sistema de consulta da Comissão de Valores Mobiliários

Vitor Arins
Programador @ Neoway

10 de Julho de 2012

Resumo

Este documento tem como objetivo apresentar as idéias utilizadas para implementar o crawler [2] que extrai informações sobre a receita de empresas de capital aberto. Esse programa foi desenvolvido com Python, Mechanize, CookieLib e BeautifulSoup. A intenção da ferramenta é capturar dados como patrimônio líquido, ativo, receita líquida, resultado bruto, resultado líquido, número de ações, valor por ação e resultado líquido por ação. Estas informações são relevante para agregar a base de dados de empresas formada na Neoway.

1 Introdução

De acordo com a diretoria da Neoway, foi detectada a necessidade de agregar informações sobre a receita de empresas. Para isso foi solicitada a implementação de um bot que pegue informações financeiras de empresas de capital aberto. A aplicação on-line encontrada para extrair essas informações se encontra no site www.cvm.gov.br.

Este documento começa apresentando a idéia principal e as ferramentas utilizadas [1.1]. Na seção 2 encontra-se uma breve explicação de Web Crawlers e suas tarefas. Também é discutido nessa seção, o modo com que cada tarefa se aplica no Crawler CVM. Finalmente na seção 3 são discutidas as considerações finais, implementações que ainda podem ser feitas e as lições aprendidas com o desenvolvimento do programa.

1.1 Ferramentas

Para implementar o crawler desejado, foram usadas as seguintes ferramentas:

Google Chrome: Browser(Navegador Web) para inspecionar o site. Em geral foram utilizadas as ferramentas de desenvolvedor do Chrome, facilmente encontradas no menu de configuração

Python 2.7.3: Linguagem de programação e seu respectivo interpretador.

Mechanize: Biblioteca implementada em Python para automatizar teste de aplicações Web. Também imita as ações de um Browser

CookieLib: Biblioteca implementada em Python para manter e renovar cookies de sessão. Necessário para validação de algumas URLs presentes que teriam que ser exploradas. Cookies serão discutidos na seção 2.2.1.

BeautifulSoup: Biblioteca implementada em Python para fazer o parsing das páginas. O parsing será discutido futuramente.

Python-CSV: Biblioteca implementada em Python para salvar os dados em disco.

2 Crawler

O que é: Um programa que varre um ou diversos sites de uma maneira metódica e automatizada extraindo informações. Também pode ser chamado de Web crawler, Bot e Web spider.

Um ponto crucial no desenvolvimento de um Crawler é a análise completa da aplicação on-line. O mapeamento e análise do modo em que os dados são exibidos, são tarefas essenciais para o desenvolvimento de um crawler, e devem ser concentradas na pessoa que irá desenvolver o produto final. Em geral o programador deve ter alguma experiência no desenvolvimento de aplicações web para poder descobrir possíveis vulnerabilidades do site que será utilizado.

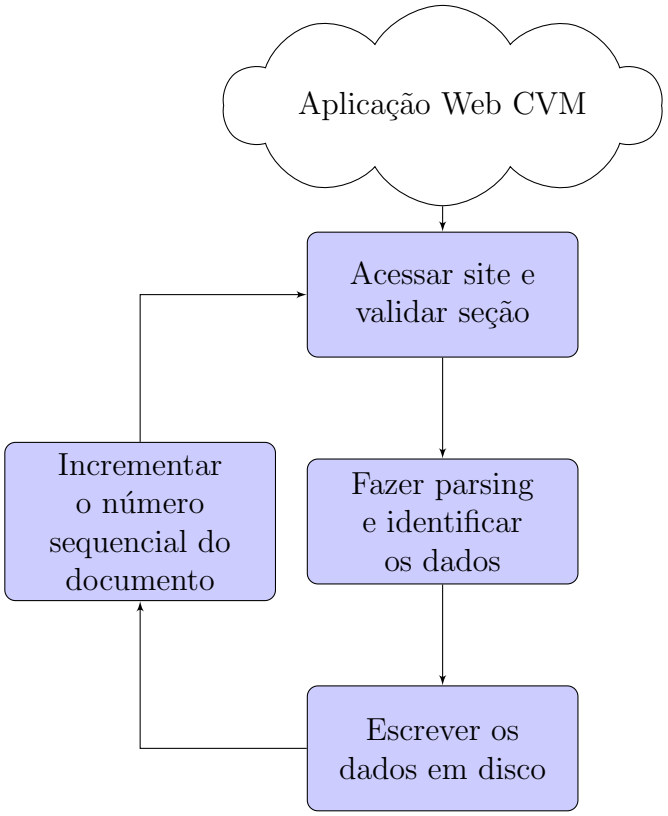
O desenvolvimento de um crawler pode ser divido em 4 tarefas principais:

1. Mapeamento da aplicação
2. Acesso ao site e código-fonte

3. Parsing do código-fonte

4. Escrita dos dados em disco

O fluxograma do funcionamento básico de um crawler:



2.1 Mapeamento

Para mapear um site é preciso acessar todos os links que possam levar aos dados desejados. As aplicações podem variar o modo que os dados são exibidos. Na maioria das vezes é necessário preencher um formulário para acessar os dados. Todas as páginas acessadas possuem um código-fonte, e este precisa ser analisado minuciosamente pelo desenvolvedor já que muitas vezes é possível detectar outros links que acessem direto os dados. Em aplicações que usam JavaScript para mostrar os dados, a análise do código-fonte se faz indispensável já que muitas dessas aplicações redirecionam a visão dos dados para outra página em outro endereço. Em geral o código pode mostrar falhas e vulnerabilidades que não são vistas na primeira instância.

No caso do Crawler CVM, é possível ver que a página que traz as informações financeiras tem um número sequencial na própria URL do site:

http://www.rad.cvm.gov.br/enetconsulta/frmGerenciaPaginaFRE.aspx?**NumeroSequencialDocumento=17053**

Como a página usa JavaScript para mostrar o conteúdo, foi feita análise do código-fonte e inspeção dos elementos html com as ferramentas de desenvolvedor do chrome [1.1]. Após essa análise foi constatado que o crawler deveria acessar primeiro a URL que contem o número sequencial para validar a sessão e extrair os cookies [2.2.1]. Em seguida acessar outras URLs, cada uma contendo uma página de dados específicos como dados gerais, endereço, demonstrações financeiras abrangentes e informações financeiras padronizadas. Depois de saber onde estão os dados, o próximo passo é acessá-los.

2.2 Acesso ao site

O acesso à aplicação significa fazer o download do código-fonte da página. Há dois métodos principais no protocolo HTTP que são usados para acessar as aplicações. O método GET é usado para baixar a página direto de uma URL. Com o método POST é possível passar parametros que serão usados pela aplicação, em um formulário html por exemplo. É preciso ter certeza que o código possui as informações que serão extraídas em primeiro lugar, portanto uma ótima prática de programação seria certificar-se que a cada interação com a página, a resposta desta seja relevante.

A abordagem utilizada no Crawler CVM foi o uso de uma biblioteca (Mechanize[1.1]) que submetesse os métodos à rede com facilidade, só foi utilizado o método GET para acessar e baixar as páginas. Todos os passos a seguir foram realizados recursivamente, iterando por todos os documentos atráves do número sequencial encontrado, começando pelo número 1 e indo até o número 20000. Dependendo da empresa e do número do documento, a aplicação exibe os dados de forma diferente. Uma forma é o menu de Informações Financeiras que possui todas as informações desejadas em uma só página, a outra forma é o menu de Demonstrações Financeiras (DFs), que distribui as informações em diversas páginas diferentes. o crawler acessa as seguintes páginas:

- Página de validação para salvar os cookies[2.2.1] da sessão.
- Página de dados gerais, nessa página conseguiremos dados como o nome, o nome anterior (caso a empresa tenha mudado) e CNPJ.
- Página de endereço, além do endereço contém telefone, fax e e-mail para contato com a empresa ou responsável financeiro da empresa.

- Se o documento for do tipo Informações Financeiras gerais, só é preciso acessar a página que contém todas as informações.
- No caso do documento for do tipo DFs, é preciso acessar as seguintes:
  - Página do Balanço Patrimonial Ativo, para extrair o Ativo Total.
  - Página de Demonstração do Resultado, extrai a Receita Líquida.
  - Página de Demonstração do Resultado Abrangente, extrai o Resultado Líquido.
  - Página da Demonstração das Mutações do Patrimônio Líquido, para extrair o saldo final do Patrimônio Líquido para cada ano.

Em alguns casos, só foi possível acessar e extrair dados da página de dados gerais, já que o cadastro da empresa provavelmente era desatualizado ou o documento era antigo. Outra dificuldade foi o tipo de documento que continha DFs Assim que as páginas possuiam várias informações que não se mostravam relevantes para o requisitado, além de falhar em mostrar dados como a quantidade e valor de ações, resultado bruto e o resultado líquido por ação.

2.2.1 Cookies

O que são: São pequenos dados guardados no Browser. Os cookies são guardados, acessados e gerados por uma aplicação Web. Cada cookie tem um certo dado, tipo de dado e domínio (site da aplicação que o utiliza). Também pode ser chamado de HTTP cookie, Web cookie ou Browser cookie.

O cookie de sessão só dura enquanto o usuário está usando o Website e a aplicação Web precisa desse cookie para validar a sessão do usuário. Após fechar o site, o cookie é apagado automaticamente.

Como a finalidade do Crawler é imitar um Browser, para cada documento que é acessado, é preciso gerar e salvar o cookie em memória. Com o cookie salvo, validamos a sessão para acessar todas as páginas de dados. Para manejar melhor todos os cookies que possam ser gerados pelo site, utilizamos a Biblioteca chamada CookieLib[1.1], que adicionada ao Mechanize[1.1] faz o trabalho automaticamente.

2.3 Parsing

O que é: Parsing consiste no processo de analisar um texto feito de uma sequência de símbolos (por exemplo, palavravas), para determinar a estrutura

gramatical e separar os símbolos relevantes. O parsing significa fazer uma análise sintática do texto.

Em conjunto com um crawler, o parsing serve como um tradutor quando se extrai informações do código-fonte de uma página. O código é analisado sintaticamente e os dados são gravados em estruturas de dados da linguagem de programação utilizada.

Tratando do Crawler CVM, as funções de parsing se encontram em um módulo separado, com o auxílio da biblioteca BeautifulSoup [1.1]. É necessário fazer o parsing de cada página acessada, para a primeira página há uma função que descobre o tipo de documento que está sendo tratado. Porém, para qualquer tipo de documento, é preciso pegar os dados gerais e o endereço da empresa. Os dados gerais, assim como o endereço, se encontram células específicas. Portanto para pegar o CNPJ procura-se na célula 'txtCnpj', o telefone na célula 'txtTel' e assim por diante. Aí então no caso de um documento que possui Demonstrações Financeiras (DFs) padronizadas o procedimento é o seguinte:

- Descobrir o Patrimônio Ativo Total, para isso extraímos a primeira linha de uma tabela da página Balanço Patrimonial Ativo. Guardamos também o ano em que a informação foi desponibilizada.

- Receita Líquida, vem da página Demonstração de Resultado. A informação também se encontra na primeira linha da tabela de informações. Aqui não é necessário se preocupar com o ano da informação pois isso já foi agregado pela função que pega o Ativo Total.

- O Resultado Líquido segue o mesmo processo de captura que o da Receita Líquida, exceto que os dados vem da página Demonstração de Resultado Abrangente.

- Para extrair O Patrimônio Líquido, o método mais viável encontrado foi a inversão das tabelas encontradas no código-fonte. Foi percebido que as informações de patrimônio líquido dos últimos anos viriam sempre como os últimos dados da tabela. Essa é a última informação possível para se encontar no documento de DFs.

No caso do documento que mostra Informações Financeiras gerais, os dados são capturados da tabela encontrada na página que contém todos os dados relevantes para o crawler. Na primeira coluna e linha verifica-se o tipo e o ano, respectivamente, do dado que está sendo extraído. Em seguida em cada linha, o dado específico é capturado.

Os dados são guardados em uma estrutura de dados de Python chamada 'dictionaire' ou dicionário em português. Um dicionário mapeia um texto à um dado qualquer. Assim teremos uma lista de dicionários em que cada dicionário se refere à uma empresa. No caso do crawler, o texto será o tipo da informação, por exemplo se é CNPJ, telefone ou patrimônio ativo, e os dados serão os valores de cada um, por exemplo '01.234.567-0001-00', '222-3333' e '12.000.000' respectivamente. Isso fará com que os dados sejam guardados em disco mais facilmente.

2.4 Escrita em disco

Para disponibilizar os dados que foram capturados, é necessário guardá-los em algum lugar. A escrita dos dados geralmente é feita em um banco de dados com SQL. O banco de dados em SQL oferece uma interface mais computacional e programável para desenvolvedores que trabalham com aplicações Web.

Porém o banco de dados SQL não foi utilizado neste projeto. Por se tratar de um projeto individual e um número de dados agregados não tão grande comparado à escala da empresa em geral, não foi sentida a necessidade de um banco de dados. Outro problema seria o consumo desses dados no futuro, dependendo do administrador de banco que recebesse os dados, a visão deste poderia não sincronizar com a visão que o programador teve da disposição dos dados.

Portanto foi escolhido guardar os dados em uma tabela com os dados separados por vírgula (CSV), para isso foi utilizada uma biblioteca [1.1] que escrevesse cada dicionário[2.3], referente à empresa correspondente, em uma linha da tabela. Essa foi a forma mais modulável e dinâmica encontrada para se trabalhar, consequentemente o os dados ficarão mais acessáveis por parte do consumidor e poderão ser facilmente modificados.

A tabela que contém os dados possui 60 colunas, a primeira coluna mostra o número sequencial de documento utilizado para fazer a consulta no site. Depois se encontram as colunas com dados gerais e endereço das empresas, nada de especial. Em seguida estam os dados de receita de cada empresa, que é o mais relevante. Os headers para cada coluna dos dados de receita, foram dispostos no seguinte formato:

$$\text{tipo} + \text{ano}$$

Onde o tipo seria a identificação do dado que estamos pegando, portanto os expemplos são algo como: "ativo2011", "resultado_bruto2008" e "valor_acoes2010". Como cada ano é uma coluna na tabela, fica justificado o motivo de tantas colunas.

3 Conclusão

É importante salientar que a maioria dos programas implementados **sempre** podem passar por um processo de melhoria. Durante o desenvolvimento do bot CVM, boas práticas de programação foram utilizadas, mas sempre é possível adicionar e modificar elementos que amplificam a eficiência de um código. O Crawler CVM não é nenhum desafio que teste esta eficiência por se tratar de uma quantidade relativamente pequena de dados e também por não apresentar muita dificuldade para a captura de dados. Porém, como todo projeto, há recomendações para trabalhos futuros a ser implementados na ferramenta.

3.1 ToDos

Os "To Do's", ou afazeres, são mudanças de código que ainda podem ser feitas para melhorar a forma com que o programa se comporta realizando a tarefa que é exigida. Como recomendações futuras, há alguns pontos importantes que foram deixados de lado neste crawler, por não apresentar uma dinâmica muito grande de dados, ou seja, as informações no site aparentemente são atualizadas anualmente apenas. Os pontos a levar em consideração são:

Multithreading: A idéia básica do multithread é realizar várias tarefas ao mesmo tempo. Após criado o código e gerado funções que realizam a tarefa desejada, é possível criar várias instâncias ou processos para cada função e fazer com que realizem a tarefa da função de forma sincronizada, aproveitando assim todo o potencial do processador e memória do computador.

Message passing: Utilizando-se do modelo de "Atores" baseado em Message passing, é possível que cada processo envolvido no programa aja de forma independente. Nesse modelo cada tarefa principal será um ator, portanto teremos o papel de acesso a página, papel de parsing e o de escrita dos dados. A assim os atores se comunicam através de mensagens que ativam cada processo e especificam quais dados devem ser trabalhados.

Banco de dados: Um banco de dados é o melhor jeito de salvar dados hoje em dia. É possível usar SQL como linguagem de administração do banco de dados, fazendo com que as informações sejam facilmente manipuladas e manejadas para futuramente disponibilizar uma aplicação

contendo os dados desejados. No crawler em questão é possível utilizar uma biblioteca chamada "sqlite3" embutida em Python [1.1], ou também é possível usar diversas bibliotecas disponíveis na internet para postgreSQL e MySQL.

3.2 Lições

Ao desenvolver o Crawler CVM algumas lições foram tiradas. É certo que ao se aprofundar em qualquer projeto, o programador ganha muita experiência de tentativas e erros que são feitos ao longo do desenvolvimento. É possível dizer que esses aprendizados são as consequencias mais importantes do projeto, afinal é assim que o programador melhora e se empenha para realizar projetos de uma maneira melhor.

O principal fator no começo do projeto é a análise da situação. Quando se captura dados de uma aplicação, é preciso saber como os dados são exibidos, como foi implementada a aplicação, o tempo estimado para download, as estruturas de dados utilizadas e a política de acesso ao site. Assim é possível escolher as bibliotecas que serão utilizadas para desenvolver o produto final e compor a estrutura de dados interna da ferramenta de captura.

Como já foi dito uma das tarefas mais importantes no desenvolvimento foi o mapeamento do site. O trabalho de captura se tornou mais fácil depois de análisar todas as páginas da aplicação e pensar na melhor abordagem na hora de extrair os dados do site.

Como considerações finais, fica a dica do tipo de profissional que deve ser encarregado da tarefa, de preferência alguém com alguma experiência em programação Web ou algo parecido. Outro fator que deve ser levado em consideração é a dificuldade de extrair dados de cada site. O jeito que os dados são dispostos obviamente varia de aplicação para aplicação, portanto deve-se lembrar que nem sempre é possível capturar todos os dados desejados. Assim como é impossível que o mesmo Crawler, sendo esse com o mesmo parser e acesso a dados, funcione para aplicações diferentes em sites diferentes.

Com os projetos de código aberto disponíveis hoje na internet, não é preciso criar funções ou frameworks para ser reutilizados, porque no final essas funções vão acabar sendo um retrabalho das ferramentas existentes. Geralmente é mais prático ter um leque dessas ferramentas conhecidas e reutilizá-las para cada projeto de captura iniciado.

Referências

[Crawler] *Web Crawler at Wikipedia*