

Universidade do Minho
Conselho de Cursos de Engenharia
Licenciatura em Engenharia Informática
Disciplina de LI IV - Projecto
Ano Lectivo de 2007/08



escola de engenharia



departamento de
informática

O Filatélico

João Branco 47124, José Durães 48318, Sérgio Carvalho 47035

Julho, 2008

Data de Recepção
Responsável
Avaliação
Observações

O Filatélico

João Branco 47124

José Durães 48318

Sérgio Carvalho 47035

Julho, 2008

Resumo

Área de Aplicação: Web Crawling , parser e Bases de dados.

Palavras-Chave: Web Crawler, Parser, Java, Bases de dados, SQL Server, C#

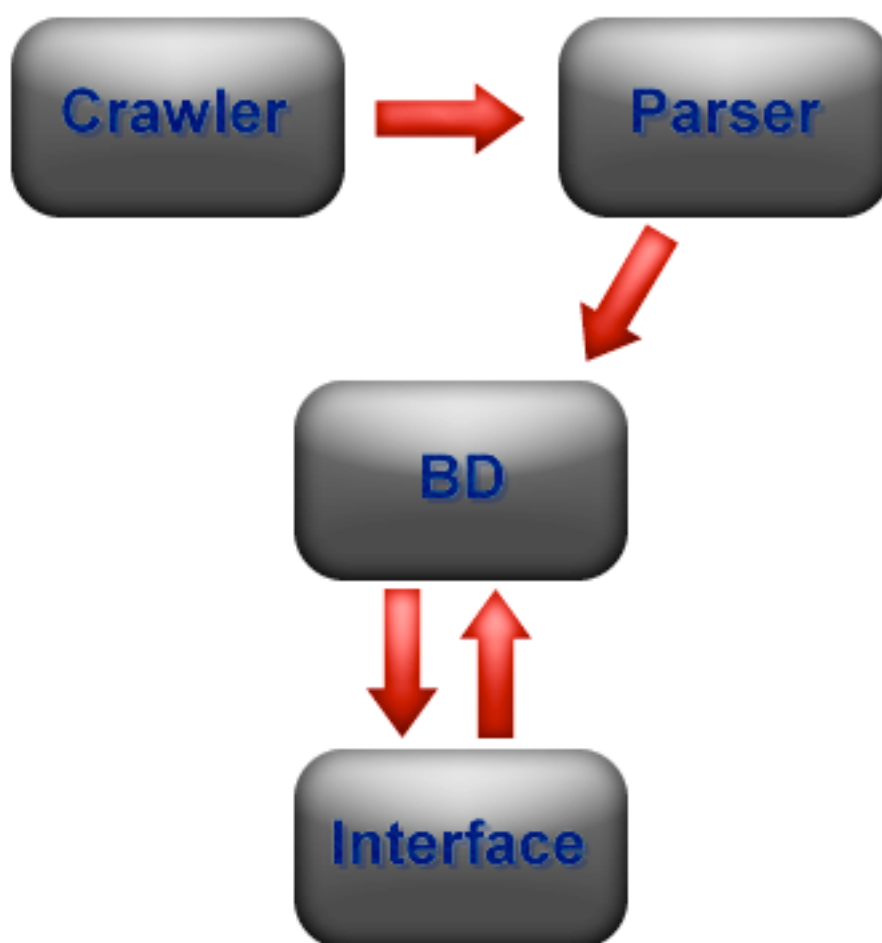


Fig. 1 : Diagrama geral do trabalho

Conteúdo

Resumo	3
Conteúdo	4
1 Introdução	5
1.1 Contextualização	5
1.2 Apresentação do Caso de Estudo	5
1.3 Motivação e Objectivos	5
1.4 Estrutura do Relatório	5
2 Recolha e Tratamento de Dados	6
2.1 Crawler	6
2.1.1 Definição	6
2.1.2 Testes.....	6
2.1.2 Aplicação Escolhida.....	7
2.1.3 Configuração.....	7
2.1.4 Sementes	8
2.2 Parser HTML	8
2.3 Analisador de Texto	9
3 Base de Dados	10
4 Interface	11
5 Conclusões e Trabalho Futuro	12
Referências WWW	13
Lista de Acrónimos	14

1 Introdução

1.1 Contextualização

O coleccionismo filatélico português surgiu no século XIX, altura em que começam a ser editadas as primeiras publicações periódicas, com o lançamento de “O Filatelista” em 1888. Desde então o número de selos e colecções cresceu consideravelmente, tornando bastante complexa a catalogação de toda a informação filatélica existente.

1.2 Apresentação do Caso de Estudo

Fazendo uso de um crawler open source, o nosso intento é preencher uma base de dados, que permita indexar vários URL's de índole filatélica. Assim poderíamos construir um ranking de URL's, sendo possível responder com uma taxa de maior sucesso a uma pesquisa deste tipo de conteúdos.

1.3 Motivação e Objectivos

Este projecto surge na necessidade comum a qualquer coleccionador em identificar num selo certas características, tais como o ano em que foi publicado, o preço a que está cotado, o significado histórico inerente, etc.

Tendo em conta que a oferta de software que permita obter facilmente este tipo de conhecimento filatélico não abunda, decidimos criar um motor de busca orientado unicamente a esta matéria.

1.4 Estrutura do Relatório

A abordagem a este projecto pode ser feita identificando três partes estruturais principais: a recolha e tratamento de dados a partir da internet, o armazenamento dos mesmos numa base de dados e a interface gráfica que permite fazer as pesquisas baseadas na informação da base de dados.

2 Recolha e Tratamento de Dados

2.1 Crawler

2.1.1 Definição

Um Web Crawler define-se como sendo um programa que percorre a internet, de uma forma metódica e autónoma, recolhendo e processando informação. Previamente é-lhe fornecida uma lista de sementes (*seeds*) a partir da qual é iniciado o seu percurso, prosseguindo pelas ligações recolhidas nestas sementes.

2.1.2 Testes

De forma a compreendermos melhor o funcionamento de cada crawler em particular, procedemos ao teste de algum código open source. Na execução desta tarefa debruçamo-nos sobre aplicações em java, vista esta ser uma linguagem que dominámos bastante bem, permitindo-nos, se necessário fazer algumas modificações ao código fonte. Os programas testados foram os seguintes: *Websphinx*, *Webharvest*, *Heritrix*, *Nutch*...

O *Websphinx*, é um crawler bastante interessante, muito simples de usar e com o essencial das funcionalidades pretendidas neste tipo de aplicação. Permite configurar nº de threads, tempo de crawling, tamanho máximo total de dados, profundidade da procura, etc. Possui funcionalidades bastante atractiva como é a capacidade de guardar em ficheiros *html* o conteúdo analisado, concatenar num só ficheiro *html* as várias páginas percorridas, ou extrair apenas pedaços de código *html* definido pelo utilizador. Acompanhar o processo de crawling é possível também visualizar um grafo que nos dá a percepção das ligações existentes entre os vários sites.

No entanto este crawler não consegue lidar com grandes quantidades de informação, deste modo para um trabalho que envolva mais de algumas centenas de páginas esta não é uma opção viável.

Testamos também o programa *WebHarvest*, onde as configurações do mesmo são feitas em ficheiros XML. Desta forma é possível extrair do crawler diversas potencialidades de acordo com o trabalho que se pretendesse executar sobre a Web.

O *Heritrix* apresentou-se como o crawler mais poderoso e robusto, podendo ser configurado segundo diversos parâmetros. Neste programa para além das opções usualmente disponíveis de configuração é também possível, através da utilização de diferentes classes, escolher:

- a maneira como é feito o pré-processamento de cada página (verificar se pertence ao domínio, etc...);
- o modo como a informação da URL é processada;
- como são recolhidos os links encontrados;
- em que formato guardar o conteúdo processado;
- como actua o pós-processador (módulo que acrescenta os links encontrados à lista de seeds a precorrer);

Este crawler apresenta ainda, por defeito, diversos gráficos sobre o trabalho executado, com os valores percentuais por tipo de ficheiros encontrados, a taxa de URL's bem sucedidos, percentagem de dados por seed, etc...

Tentamos também testar o *Nutch*, no entanto esta acção não foi muito bem sucedida, pois ao carregar o programa para o servidor *Apache Tomcat* este apresentava alguns erros, os quais não conseguimos deslindar recorrendo à ajuda disponibilizada.

2.1.2 Aplicação Escolhida

Nesta fase foi necessário tomar uma decisão em relação ao método que iríamos adoptar para a recolha de informação. Assim surgiram duas hipóteses principais entre as quais teríamos de escolher.

- utilizar um crawler que recolhesse para o disco ficheiros html dos sites por onde passava, e de seguida utilizar um parser offline para tratar estes dados e colocá-los na base de dados;
- ou fazer um crawling que devolvesse um log com os URL's dos sites por onde passou, e utilizar um parser que conseguisse processar as páginas online, colocando de seguida a informação na base de dados.

Tendo em conta que a segunda opção despenderia mais recursos Web, pois para cada site seriam feitos dois descarregamentos (uma pelo crawler, outra pelo parser), decidimos optar pela primeira escolha. Para isso, a aplicação seleccionada foi o *Heritrix*, pois apresentou-se como sendo a mais opção completa e robusta, permitindo lidar com as quantidades de dados consideráveis que pretendemos.

2.1.3 Configuração

A configuração definida para este caso, baseou-se no *default* disponibilizado.

Em termos da profundidade do crawling, esta foi definida para se limitar ao domínio das seeds fornecidas, o que definido recorrendo à classe *org.archive.crawler.scope.DomainScope* na secção "Modules" em "Select Crawl Scope". Para configurar o modo como o conteúdo processado era armazenado foi alterado também na secção "Modules" o campo "Select Writers". Como

pretendíamos guardar os sites no disco em formato html , para posteriormente serem analisados por um parser, seleccionámos aqui a classe *org.archive.crawler.writer.MirrorWriterProcessor*.

2.1.4 Sementes

As “seeds” fornecidas ao crawler basearam-se nas dez primeiras referências encontradas pelo Google, em pesquisas dos termos:

- filatelia
- selos
- filatélico
- stamps
- philatelic

2.2 Parser HTML

Depois do trabalho efectuado pelo crawler na recolha de páginas, será necessário avaliá-las, de modo a preenchermos a base de dados, com dados que nos possibilitem a sua pesquisa futura.

Assim, primeiramente é necessário analisar o html em tags específicas que nos interessem, como por exemplo o título, ou o rodapé. Depois desta análise geral, retira-se o texto que compõe a página, sendo este analisando em busca das palavras que são mais frequentes.

É possível encontrar-se facilmente, na Web, diversos parsers html.

O primeiro que analisámos foi o *Noviway HTML To XML Converter*, escrito em C#, possui classes específicas para fazer parser de ficheiros html, por tags definidas pelo utilizador, podendo aceder directamente à web para consultar a página. A documentação sobre a API é muito pobre, e estávamos a obter erros estranhos, que nos obrigaram a não o escolher.

De seguida, após mais alguma pesquisa encontramos o *HTML Parser*, que apresenta-se construído desta feita em Java. Este disponibiliza alguns “templates” de pesquisa, como retirar texto respeitante a certas tags (título, imagens,...) ou retirar todo o texto da página, com exemplos bastante bem explicados e documentados. Desta forma optamos por utilizar este segundo parser.

Utilizamos um *TagNameFilter* aplicado a tag Title, juntamente com um *FilterBean* para retirar o título da página, e um *StringBean*, com o método *getStrings* para retirar o texto. Tudo de modo bastante fácil e intuitivo.

2.3 Analisador de Texto

Após este parsing inicial, obtemos o “sumo” da página, o texto. Para não sobrecarregar a BD com informação desnecessária, e como não faria sentido para as pesquisas introduzir o texto em bruto, aplicamos mais um filtro para analisá-lo. Assim, nas nossas pesquisas, encontramos uma Java applet, com um parser que fazia exactamente aquilo que procurávamos. Percorre o texto, separa em frases, depois em palavras, e conta a ocorrência de cada uma delas. Necessitamos apenas de transformar um pouco o código

3 Base de Dados

Depois de processada pelo parser a informação irá ser armazenada numa base de dados. Tivemos de definir a informação que é relevante ter na BD, considerando que não vamos armazenar páginas Web completas mas sim referências a estas.

Como sistema de gestão de base de dados resolvemos escolher uma solução usada optámos pelo *Microsoft SQL Server*, que oferece vantagens ao nível da integração com a aplicação de interface com o utilizador, desenvolvida em C#, além de ser bastante eficiente em termos de gestão de recursos devido a implementar um sistema próprio de gestão de memória. Este sgbd integra-se na plataforma .NET Framework permitindo que procedimentos e triggers sejam escritos numa linguagem compatível, o que é o caso do C#.

Interessa antes de mais definir a estrutura da BD e as tabelas que a vão constituir. Numa primeira fase de normalização, teríamos uma tabela com os seguintes campos: url, título, descrição, tags e data de modificação, que normalizada resulta em duas tabelas, de seguida representadas:

Páginas (#Código Url, Url, Título, Descrição, Data de Modificação)

Tags (#Código Url, #Código Tag, Tag)

A data de modificação será útil para o crawler decidir se durante a análise de uma página já existente na BD ela carece ou não de actualização.

O SQL Server possui funcionalidades como “SQL Server Full Text Search” que permitirão indexar e efectuar queries ao texto da BD, deixando de parte expressões desnecessárias, e permitindo desta forma pesquisas mais rápidas. As palavras serão agregadas numa tabela indicando a sua localização e número de ocorrências, o que será útil para o “ranking” das pesquisas.

As tags referem-se as tags html que com informação textual relevante. Apesar de não se guardadas imagens na BD, dentro do conteúdo das tags html das imagens o parser retira o campo correspondente à respectiva descrição.

Através da interface de pesquisa o utilizador poderá efectuar consultas e obter os resultados ordenados de acordo com a sua relevância. A ordem das páginas depende de factores como a data de modificação (páginas mais recentes têm um maior peso) e o número de ocorrências das chaves passadas como parâmetros de pesquisa presentes nos campos título, descrição e tags.

4 Interface

A interface será minimal, ao estilo de um motor de busca online, com um campo para o utilizador introduzir a expressão que deseja pesquisar e um botão para validar. O programa, após efectuar a consulta à base de dados, retorna os resultados sob a forma de uma lista em que cada item conterà o respectivo url, título da página e descrição. Em cada item o utilizador poderá seguir a hiperligação e ser conduzido à página web pretendida.

5 Conclusões e Trabalho Futuro

Concluída esta fase do trabalho, em que o nosso foco foi principalmente de análise, e apesar do início complicado devido a complexidade da instalação e teste de alguns crawlers, conseguimos obter uma ideia geral e concreta do trabalho a executar, tomando algumas decisões fundamentais.

Apresentamos claramente a divisão das fases necessárias para a implementação da aplicação pretendida, já com ideias relativamente elaboradas de como o fazer. Assim, o nosso trabalho futuro será apenas o de terminar a implementação da aplicação de acordo com os dados aqui apresentados.

Referências WWW

1 - <http://www.manageability.org/blog/stuff/open-source-web-crawlers-java>

Site que reúne vários links de crawlers open source feitos em java, acompanhados de uma breve descrição.

2 - <http://crawler.archive.org/>

Página oficial do crawler Heritrix onde podemos fazer o download e recolher informação sobre o mesmo.

3 - <http://www.noviway.com/Code/HTML-To-XML.aspx>- Noviway HTML to Xml converter

Página do parser HTML escrito em C#, onde se pode obter o programa e algumas informações sobre ele.

4 - <http://www.mste.uiuc.edu/pavel/java/text/>- Analisador de texto

Local onde é possível encontrar o analisador de texto e informação relacionada.

5 - <http://htmlparser.sourceforge.net/> - HTML parser

Página oficial do htmlPARSER onde podemos fazer o download do programa, aceder a manuais, etc...

6 - <http://www.csharp-station.com/Tutorial.aspx>- Tutoriais C#

Página com tutoriais básicos de C#, para a introdução na linguagem.

Lista de Acrónimos

URL	Uniform Resource Locator
BD	Base de Datos
XML	eXtensible Markup Language
SQL	Structured Query Language