Universidade Federal do Amazonas
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Programa de Pós-Graduação em Informática

Geração Automática de Padrões de Navegação para Web Sites de Conteúdo Dinâmico

Márcio Luiz Assis Vidal

Manaus – Amazonas
Março de 2006

Márcio Luiz Assis Vidal

Geração Automática de Padrões de Navegação para Web Sites de Conteúdo Dinâmico

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Departamento de Ciência da Computação da Universidade Federal do Amazonas, como requisito parcial para obtenção do Título de Mestre em Informática.

Orientador: Prof. Dr. Altigran Soares da Silva

Márcio Luiz Assis Vidal

Geração Automática de Padrões de Navegação para Web Sites de Conteúdo Dinâmico

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Departamento de Ciência da Computação da Universidade Federal do Amazonas, como requisito parcial para obtenção do Título de Mestre em Informática.

Banca Examinadora

Prof. Dr. Altigran Soares da Silva – Orientador
Departamento de Ciência da Computação – UFAM/PPGI

Prof. Marcos André Gonçalves, Ph.D.
Departamento de Ciência da Computação – UFMG

Prof. Dr. Edleno Silva de Moura
Departamento de Ciência da Computação – UFAM/PPGI

Prof. João Marcos Bastos Cavalcanti, Ph.D.
Departamento de Ciência da Computação – UFAM/PPGI

Manaus – Amazonas
Março de 2006

Aos meus pais, irmãs, Keyth e Gabriel.

Agradecimentos

Ao chefe supremo, acima de tudo.

Aos meus pais Antônio Vidal e Araci Vidal, pelo apoio incondicional.

As minhas irmãs que dentro do possível sempre me ajudaram.

A minha namorada Keyth Pina, por ter estado sempre presente, pela paciência e compreensão.

Ao meu orientador, Altigran Soares, pelo incentivo, credibilidade na minha capacidade, insistência, paciência e todas as horas gastas nas as orientações.

Aos grandes amigos que nos momentos difíceis sempre estiveram presentes dando apoio, incentivo e suporte: Areliam Maia, Célia Francisca, Daniel Oliveira, Péricles Oliveira, Roberto Oliveira e Ruam Belém.

A amiga Patrícia Peres, por toda a ajuda oferecida, paciência e tempo gasto.

A Fundação de Amparo à Pesquisa do Estado do Amazonas - FAPEAM, pelo suporte financeiro.

A todos aqueles que ajudaram de alguma forma na realizaçãao deste trabalho, o meu mais profundo agradecimento.

A sabedoria é uma construção sólida e única, na qual cada parte tem seu lugar e deixa sua marca.

Michel Eyquem de Montaigne

Resumo

Um crescente número de aplecações para Web necessitam processar coleções de páginas similares obtidas de Web sites. O objetivo final destas aplicações é tirar proveito de informações valiosas que estas páginas implicitamente contêm para realizar tarefas como consulta, busca, extração de dados, mineração de dados e análise de características de uso e popularidade. Para algumas destas aplicações os critérios para determinar quando uma página deve estar presente na coleção estão relacionados a características do conteúdo da página. Contudo, exitem muitas outras importantes situações em que características inerentes à estrutura das páginas, ao invés de seu conteúdo, provêm um critério melhor para guiar a coleta de páginas. Motivados por este problema, propomos nesta dissertação uma nova abordagem para geração de coletores guiados por estrutura que requer um esforço mínimo do usuário, pois são necessário apenas um exemplo das páginas a coletar e um ponto de entrada no Web site. Uma outra característica importante de nossa abordagem, é o fato de ser capaz de lidar com sitesonde as páginas a serem coletadas são geradas dinamicamente através do preenchimento de formulários. Ao contrário dos métodos existentes na literatura, no nosso caso não é necessária a existência de um banco de dados de amostra para auxiliar no processo de preenchimento do formulário, nem tão pouco é necessária grande iteração com o usuário. Resultados obtidos em experimento com nossa abordagem demonstraram um valor de 100% de precisão em coletas realizadas sobre 17 Web sites reais de conteúdo estático e dinâmico, e pelo menos 95% de revocação para 11 sites estáticos utilizados nos experimentos.

Abstract

A growing number of Web applications need to process collection of similar pages obtained from Web sites. These applications have the ultimate goal of taking advantage of the valuable information implicitly available in these pages to perform such tasks as querying, searching, data extraction and mining. For some of these applications, the criteria to determine when a Web page must be present in a collection are related to features of the content of the page. However, there are many other important applications in which the inherent structure of the pages, instead of its content, provides a better criterion for gathering the pages. Motivated by this problem, we propose in this work a new approach for generating structure-driven crawlers that requires a minimum effort from the user, since it only require an example of the page to be crawled and an entry point to the Web site. Another important feature in our approach is that it is capable of dealing with Web sites in which the pages to be collected are dynamically generated through the filling of forms. Contrary to existing methods in the literature, our approach does not require a sample database to help in the process of filling out forms and it also does not demand a great interaction with users. Results obtained in experiments with our approach demonstrate a 100% value of precision in craws performed over 17 real Web sites with static and dynamic contents and at least 95% of recall in all 11 static Web sites.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Um crescente número de aplicações para Web necessitam processar coleções de páginas similares obtidas de Web sites. O objetivo final destas aplicações é obter informações valiosas que estas páginas implicitamente contêm para realizar tarefas como consulta, busca, extração de dados, mineração de dados e análise de características de uso e popularidade. Para algumas destas aplicações, particularmente para busca, os critérios para determinar quando uma página deve estar presente na coleção estão relacionados ao conteúdo da página, por exemplo, palavras, frases, etc. Uma abordagem muito popular para se obter coleções de páginas de interesse específico em Web sites é o uso dos chamados coletores especializados (*focused crawlers*) [Chakrabarti et al., 1999].

Coletores especializados são usualmente empregados em situações onde é necessário gerar coleções de páginas relacionadas a um determinado tópico ou assunto. Este tipo de coletor é geralmente empregado em sistemas de busca de domínios específicos [McCallum et al., 1999] e bibliotecas digitais [Bergmark, 2002, Calado et al., 2002]. Nestes casos, todas as páginas a serem coletadas pelo coletor especializado compartilham o mesmo tópico [Chakrabarti, 1999, Chakrabarti et al., 2002, Liu et al., 2004]. Chamamos este tipo de coletor especializado de *coletor guiado por conteúdo*. A abordagem dos coletores guiados por conteúdo obtém bons resultados em muitas situações práticas. Contudo existem muitas outras importantes situações em que características inerentes à estrutura da páginas, ao invés de seu conteúdo, provêm um critério melhor para guiar o coletor em processo de coleta especializada.

Por exemplo, considere uma aplicação que requeira coleções de páginas contendo informações

sobre artistas de jazz do Web site *E-Jazz*[1]. Este site contém diversas páginas sobre artistas de jazz, discos, estilos musicais e etc. Definir um conjunto de características relacionadas ao conteúdo que abrange todos os artistas é uma tarefa difícil, pois os artistas estão usualmente relacionados a algum estilo de jazz ou a um instrumento musical, e existem vários estilos distintos e instrumentos a serem considerados. Por outro lado, também existem páginas não relacionadas a artistas que compartilhas o mesmo assunto com várias páginas de artistas. Por exemplo, a página da Figura 1.1(a) é uma página de artista que compartilha o mesmo assunto com a página da Figura 1.1(b), que é uma página de estilo de jazz. Além disso, a página da Figura 1.1(a) não compartilha o mesmo assunto com a páginas da Figura 1.1(c), apesar do fato de ambas serem páginas de artista. Nesta situação, utilizar características relacionadas ao conteúdo para caracterizar as páginas tem uma grande chance de falhar.

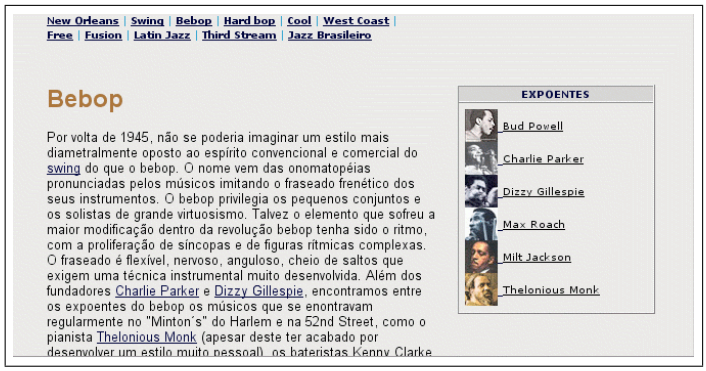
Motivados por este problema, propomos uma nova técnica de coleta baseada na estrutura das páginas ao invés de seu conteúdo. Enquanto muitos trabalhos na literatura têm direcionado seus esforços a coleta guiada por conteúdo [Rennie and McCallum, 1999, Chakrabarti et al., 1999, Tanaka and Tanaka, 1988, Chakrabarti et al., 2002, Bergmark et al., 2002, Liu et al., 2004, Qin et al., 2004], o uso da estrutura das páginas como critério para coleta de páginas tem sido quase totalmente negligenciado. Todavia, esta é uma alternativa interessante para solucionar problemas práticos em várias aplicações importantes de gerenciamento de dados na Web.

Em nosso trabalho, a principal aplicação considerada é automaticamente prover páginas ricas em dados para extratores de dados de páginas Web (*wrappers*), que geralmente utilizam padrões estruturais para realizar a extração de dados implícitos [Arasu and Garcia-Molina, 2003, Crescenzi et al., 2001, de Castro Reis et al., 2004, Laender et al., 2002a, Zhai and Liu, 2005]. Contudo outras aplicações também requerem usualmente coleções de páginas com estrutura similar. Exemplo destas aplicações são consultas e buscas baseadas na estrutura da páginas Web [Ahnizeret et al., 2004, Davulcu et al., 1999]; construção de bibliotecas digitais [Calado et al., 2002, Qin et al., 2004]; e mineração de dados na Web [Cooley, 2003].

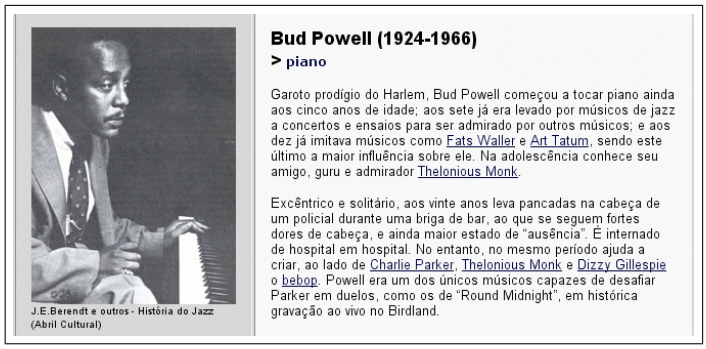
Nesta dissertação apresentamos uma abordagem para geração de coletores guiados por estrutura que requer um esforço mínimo do usuário, pois são necessários apenas um exemplo das páginas a coletar e um ponto de entrada no Web site. A partir daí é feita uma navegação exaustiva através do site à procura de *páginas-alvo*, que são todas as páginas estruturalmente similares

[1]Disponível em `http://www.ejazz.com.br`

(a)



(b)



(c)

Figura 1.1: Exemplos de páginas do Web site e-jazz.

a página de exemplo dada. Para determinar a similaridade estimada entre páginas, utilizamos uma medida conhecida como *distância de edição entre árvores* (*DEA*). Para isso utilizamos as árvores DOM [2] subjacentes às páginas HTML e aplicamos uma variação do algoritmo de DEA chamado RTDM [de Castro Reis et al., 2004] que foi proposto em [de Moura, 2004].

Em seguida, todos os caminhos que levam a páginas-alvo são guardados e é gerado um *padrão de navegação* [Lage et al., 2004], que é composto por uma sequência de padrões de links que um coletor deve seguir para atingir as páginas-alvo. Finalmente, é gerado um coletor baseado

[2] Disponível em `http://www.w3.org/TR/REC-DOM-Level-1/`

nestes padrões. Deste ponto em diante, o coletor pode ser utilizado para recuperar páginas que são estruturalmente similares à página exemplo, mesmo que novas páginas similares sejam adicionadas após a sua criação.

Na geração automática de coletores, a grande maioria das abordagens existentes na literatura ignora o conteúdo dinâmico da Web, ou seja as páginas Web ricas em dados geradas dinamicamente através do preenchimento de formulários. Por exemplo os coletores gerados pelos métodos propostos em [Golgher et al., 2000b] são capazes de alcançar estas páginas, mas sua geração demandam intensa interação com o usuário. Outros trabalhos como [Bergmark et al., 2002], [Liu et al., 2004] e [Lage et al., 2004] precisam de um banco de dados previamente criado para auxiliar no processo de preenchimento de formulários. Nesta dissertação, propomos uma abordagem onde a interação do usuário é mínima e não se faz necessária a criação de um banco de dados prévio.

Um ponto forte da abordagem guiada por estrutura é que os critérios estruturais são usualmente mais precisos e seguros que os critério de conteúdo. Assim, enquanto coletores guiados por conteúdo quase nunca atingem altos níveis de qualidade, a abordagem guiada por estrutura tende a ser extremamente precisa, sem perda de informação. De fato, em nossos experimentos, nos quais foram realizadas várias sessões de coleta sobre 17 sites reais, os coletores guiados por estrutura gerados por nossa abordagem foram capazes de coletar a quase totalidade das páginas similares à página de exemplo dada, incluindo páginas adicionadas após a sua geração. Os resultados obtidos demonstram um valor de 100% de precisão para todos os 17 sites e pelo menos 95% de revocação para todos os 11 sites estáticos, nos quais é possível medi-la.

1.1 Trabalhos Relacionados

O trabalho que apresentamos nesta dissertação é motivado por trabalhos anteriores sobre coleta automática para alimentar extratores com páginas ricas em dados. O problema de extração de dados de páginas Web é abordado em vários trabalhos recentes na literatura [Arasu and Garcia-Molina, 2003, Crescenzi et al., 2001, de Castro Reis et al., 2004, Laender et al., 2002a].

Nosso trabalho pode ser visto como uma evolução da ferramenta ASByE [Golgher et al., 2000a]. A ASByE é uma ferramenta baseada em exemplos onde

o usuário interage com os Web sites de interesse para que um *plano de coleta*, que especifica como um coletor deve navegar pelo site, seja gerado. A maior limitação dessa abordagem está no alto grau de dependência do usuário para a criação do plano de coleta, já que este depende da interação do usuário com o site.

Uma outra abordagem semelhante é o WebVCR [Anupam et al., 2000] que permite que o usuário crie atalhos para as páginas Web que requerem múltiplos passos para serem alcançadas. Basicamente, o WebVCR "grava" os passos necessários a partir de uma determinada páginas de entrada em um Web site até à página de interesse a ser coletada e os repete posteriormente. Nota-se claramente que a esta abordagem necessita de uma grande interação do usuário para atingir as páginas de interesse.

O uso de padrões de navegação para guiar coletores foi proposto em [Lage et al., 2004]. Neste trabalho, a partir de um conjunto pré-definido de padrões de navegação, coletores de páginas são automaticamente gerados. A abordagem, no entanto, necessita que o usuário forneça um banco de dados relativo ao domínio de interesse, como CDs ou livros, contendo os valores que serão usados no preenchimento de formulários de busca no site. O padrão de navegação em conjunto com o uso de heurísticas, permite a identificação, dentro do banco de dados do domínio, de que valores usar e que formulários preencher. Um coletor é então gerado para o site, através da seleção de que padrão de navegação, dentre os possíveis, deve ser aplicado. Neste trabalho estendemos a abordagem proposta em [Lage et al., 2004], no sentido de que ao invés de usarmos padrões de navegação previamente definidos e fixos, eles são gerados automaticamente a partir dos parâmetros fornecidos pelo usuário.

A estrutura interna da árvore DOM associada a páginas Web foi previamente utilizada em [Crescenzi et al., 2004] e [de Castro Reis et al., 2004] para criar agrupamento de páginas que são estruturalmente similares. Nestes trabalhos, a principal motivação também é organizar as páginas para alimentar processos de extração dos dados destas páginas. Contudo, em nosso trabalho atacamos um problema distinto, porém relacionado: como coletar páginas de uma única classe de páginas estruturalmente similares de acordo com uma página de exemplo dada como entrada.

Em [Crescenzi et al., 2004] os autores utilizam as características das coleções de links disponíveis nas páginas. Mais precisamente, eles utilizam o conjunto de caminhos entre a raiz da página HTML e as tags <a> para caracterizar a estrutura. Páginas que compartilham os

mesmos conjuntos de caminhos são consideradas estruturalmente similares.

O trabalho aqui apresentado é também relacionado aos trabalhos sobre os chamados coletores especializados *focused crawlers*) [Chakrabarti et al., 1999, Diligenti et al., 2000, Rennie and McCallum, 1999]. Nestes trabalhos, a principal idéia é gerar coletores que recuperam apenas páginas que são relevantes a um dado tópico. Para especificar este tópico, um conjunto de páginas de exemplo é fornecido. Para guiar o coletor através da Web, um classificador é utilizado para avaliar a relevância das páginas encontradas com respeito ao tópico. Nos referimos a este tipo de coletor especializado como *coletor orientado por conteúdo*, onde o coletor considera a similaridade entre o conteúdo das páginas encontradas durante a coleta e o conteúdo das páginas dadas como exemplo. Os coletores que abordamos no presente trabalho são *coletores orientados por estrutura*, pois consideram a similaridade estrutural da paginas.

O uso de outras características além do conteúdo das páginas em si tem sido considerado em trabalhos recentes sobre coletores guiados por conteúdo como [Aggarwal et al., 2001] e [Liu et al., 2004]. Dentre as características consideradas como evidências para determinar quão relevante uma página é para então ser coletada estão os fragmentos (*tokens*) presentes nas URLs, a natureza da página Web que se refere a uma dada página, e a estrutura dos links do Web site. Em nenhum destes trabalhos a estrutura interna das páginas é considerada como uma evidência.

1.2 Organização da Dissertação

Esta dissertação está organizada em seis capítulos. No Capítulo 2 serão apresentados conceitos básicos necessários à compreensão deste trabalho.

Os Capítulo 3 e 4 detalham respectivamente os métodos propostos para a geração automática dos coletores especializados e as técnicas desenvolvidas para o preenchimento automático dos formulários Web.

O Capítulo 5 apresenta os experimentos realizados, para analisar os métodos propostos utilizamos sites reais da Web. Finalmente, no Capítulo 6 apresentamos nossas conclusões, as contribuições do trabalho desenvolvido e apresentamos sugestões para trabalhos futuros.

Capítulo 2

Conceitos Básicos

Neste capítulo serão apresentados e discutidos de forma sucinta alguns conceitos básicos necessários a compreensão do método proposto neste trabalho.

2.1 Coletores de Páginas Web

Um dos componentes fundamentais de aplicações que processam grandes volumes de informações oriundas da Web (por exemplo, máquinas de busca) são os coletores de páginas Web ou *Web crawlers* [Pinkerton, 1994, da Silva et al., 1999, Heydon and Najork, 1999, Brin and Page, 1998, Arasu et al., 2001].

Coletores são aplicações que sistematicamente navegam pela Internet para realizar alguma tarefa específica, em geral coletar automaticamente páginas Web (sua principal tarefa) e armazená-las localmente para futuro processamento, como por exemplo, indexação, utilizada nas máquinas de busca, ou extração de dados, realizada por extratores.

O funcionamento de um coletor típico pode ser descrito pelo Algoritmo 1. Neste algoritmo as operações ***Enfileira()***, ***Desenfileira()*** e ***Vazio()*** são operações normais sobre uma fila F, com as seguintes modificações. A operação ***Enfileira()*** apenas adiciona um nova URL à fila se esta ainda não tiver sido adicionada. A operação ***Desenfileira()*** apenas marca a URL à frente da fila como "removida", ao invés de realmente removê-la. Como consequência ***Vazio()*** é verdadeiro quando todas as URLs na fila estiverem marcadas como "removida".

Para determinadas aplicações, os coletores tradicionais, utilizados em máquinas de busca, são muito abrangentes, pois coletam indiscriminadamente quaisquer tipos de páginas. Assim, como

```
1  Coletor;
2  Seja I uma lista de URLs inciais;
3  Seja F uma fila;
4  foreach URL i em I do
5  |  Enfileira(i, F);
6  end
7  while ¬ Vazio(F) do
8  |  u ← Desenfileira(F);
9  |  d ← Get(u); //requisita o documento apontado por u ;
10 |  Armazena d;
11 |  Extrai os lynks de d;
12 |  Seja U o conjunto de URLs apontadas por estes lynks;
13 |  foreach u ∈ U do
14 |  |  Enfileira(u, F);
15 |  end
16 end
```

Algoritmo 1: Descrição simplificada do processo de coleta automática na Web

uma alternativa a esses coletores, Chakrabarti propôs o uso dos chamados *coletores especializados* ou *Focused crawlers* [Chakrabarti et al., 1999].

O objetivo de um coletor especializado é procurar seletivamente páginas que são relevantes a um conjunto de tópicos pré-definidos. Os tópicos são especificados utilizando documentos de exemplo e não palavras chave (*keywords*). Coletar e indexar todo o conteúdo acessível da Web para ser capaz de responder todas as consultas não estruturadas possíveis, o coletor especializado analiza cuidadosamente as páginas procurando por links que são relevantes para a coleta e evitando regiões irrelevantes da Web.

Nosso trabalho pode ser visto como um método para geração de coletores especializados. No entanto, ao invés de utilizarmos o conteúdo como evidência para decidir que páginas coletar, é utilizada a similaridade entre as estruturas das páginas. Esta similaridade é calculada utilizando distância de edição entre árvores. Este conceito é revisado na seção seguinte, juntamente com o algoritmo que usamos em nosso trabalho para avaliar a similaridade.

2.2 Similaridade Estrutural de Páginas Web

Nesta seção revisaremos o conceito de Distância de Edição entre Árvores (DEA) [Selkow, 1977], que é um conceito chave em nossa abordagem. Discutiremos como a DEA é utilizada para realizar a comparação entre duas árvores dadas de acordo com sua estrutura. Iremos, também

apresentar o algoritmo RTDM, um algoritmo eficiente para computar a DEA entre duas árvores DOM.

Intuitivamente, a DEA entre duas árvores T_A e T_B é o custo associado ao menor conjunto de operações necessárias para transformar T_A em T_B. Assim, em geral, consideramos que a similaridade entre T_A e T_B é o inverso de sua distância de edição.

Aqui assumimos que a estrutura da página Web pode ser descrita, como um *árvore rotulada ordenada de raíz fixa*. Uma árvore de raíz fixa é a árvore cujo vértice raíz é fixo. Árvores ordenadas de raíz fixa são árvores de raíz fixa nas quais a ordem relativa de seus filhos é fixada para cada vértice. Uma árvore rotulada ordenada de raíz fixa tem um rótulo associado a cada um de seus vértices. A Figure 2.1 mostra um exemplo de duas árvores rotuladas ordenadas de raíz fixa T1 e T2. De agora em diante, iremos nos referenciar a árvores rotuladas ordenadas de raíz fixa simplesmente como árvores, exceto em situações explicitamente indicados.

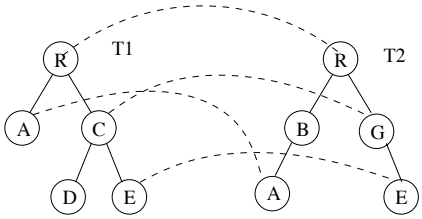


Figura 2.1: Duas árvores ordenadas rotuladas de raíz fixa e um exemplo de mapeamento.

Nesta formulação tradicional, o problema de distância de edição entre árvores consiste de três operações básicas: (a) remoção de vértices, (b) inserção de vértices e (c) substituição de vértices. Para cada uma destas operações é atribuído um custo. A solução deste problema consiste em determinar o menor conjunto de operações necessárias para transformar uma árvore em outra. Outra formulação equivalente (e possivelmente mais intuitiva) deste problema é descobrir um *mapeamento* com o menor custo entre as duas árvores. O conceito de mapeamento introduzido em [Tai, 1979], é formalmente definido a seguir.

Definição 1 *Seja T_x uma árvore e seja $T_x[i]$ o i-ésimo vértice da árvore T_x em um caminhamento em pré-ordem na árvore. O* mapeamento *entre uma árvore T_1 de tamanho n_1 e uma árvore T_2 de tamanho n_2 é o conjunto M de pares ordenados (i, j), que satisfaçam as seguintes condições para todos $(i_1, j_1), (i_2, j_2) \in M$*

- $i_1 = i_2$ *sse* $j_1 = j_2$*;*

- $T_1[i_1]$ *está à esquerda de* $T_1[i_2]$ *sse* $T_2[j_1]$ *está à esquerda de* $T_2[j_2]$*;*
- $T_1[i_1]$ *é um ancestral de* $T_1[i_2]$ *sse* $T_2[j_1]$ *é um ancestral de* $T_2[j_2]$*.*

Na Definição 1, a primeira condição estabelece que cada vértice não pode aparecer mais de uma vez no mapeamento, a segunda impõe a preservação da ordem entre nodos irmão na árvore e a terceira reforça a relação hierárquica entre os nodos em um árvore. A Figura 2.1 ilustra o mapeamento entre duas árvores.

Intutitivamente, um mapeamento é descrito como uma seqüêencia de operações de edição para transformar uma árvore em outra, ignorando a ordem com que estas operações são aplicadas. Na Figura 2.1, a linha pontilhada dos vértices de T_1 para os vértices T_2 indica que o vértice de T_1 deve ser alterado se os vértices forem diferentes, caso contrário deve permanecer inalterado. Vértices de T_1 não tocados pelas linha pontilhadas devem ser removidos, e vértices de T_2 não tocados devem ser inseridos.

Como havíamos mencionado, estimar a DEA é equivalente a encontrar o custo mínimo de mapeamento. Seja M um mapeamento entre as árvores T_1 e T_2, seja S o subconjunto de pares $(i, j) \in M$ com rótulos distintos, seja D um conjunto de nodos em T_1 que não ocorrem em qualquer $(i, j) \in M$ e seja I o conjunto de nodos em T_2 que não ocorrem em qualquer $(i, j) \in M$. O custo de mapeamento é dado por $c = |S|p + |I|q + |D|r$, onde p, q e r são custos atribuídos as operações de substituição, inserção e remoção respectivamente. Isto é comum para associar um custo unitário para todas as operações, contudo aplicações específicas podem requerer a especificação de casos distintos para cada tipo de operações distintas.

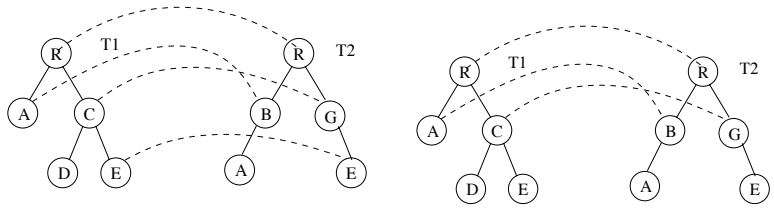
O problema de DEA é um problema de difícil solução, e vários algoritmos, com diferentes características, têm sido propostos. Todas as formulações possuem complexidade acima da quadrática [Chen, 1999]. No entanto, foi provado que, se as árvore não são ordenadas, o problema é NP-completo [Zhang et al., 1992]. O primeiro algoritmo para o problema de mapeamento foi apresentado em [Tai, 1979], a sua complexidade é $O(n_1 n_2 h_1 h_2)$, onde n_1 e n_2 são o tamanho das árvores e h_1 e h_2 são suas alturas. Este algoritmo utiliza programação dinâmica e recursivamente calcula a distância de edição entre as cadeias de caracteres formada pelo conjunto de vértices filhos para cada vértice interno da árvore. Em [Wang et al., 1998], um novo algoritmo foi apresentado com custo $O(d^2 n_1 n_2 min(h_1, l_1) min(h_2, l_2))$, onde d é a DEA e l_1 e l_2 são os números de níveis de cada árvore. Note que este custo depende do algoritmo de saída. O limite

superior mais conhecido para este problema é o do no algoritmo apresentado em [Chen, 1999], com complexidade $O(n_1 n_2 + l_1^2 + l_1^{2.5} l_2)$.

Apesar da complexidade inerente do problema de mapeamento em formulações genéricas, existem várias aplicações práticas que podem ser modeladas usando formulações restritas deste problema. Pela imposição de condições para as operações básicas correspondentes à formulação na Definição 1 (por exemplo, substituição, inserção e remoção), são obtidas quatro formulações clássicas e restritas: *alinhamento*, *distância entre árvores isoladas*, *distância top-down*, e distância *bottom-up*, por isso algoritmos mais rápidos e convenientes têm sido propostos [Valiente, 2001, Wang and Zhang., 2001].

Detalhar cada uma dessas formulações está fora do escopo deste trabalho, no entanto como nossa abordagem utiliza uma versão adaptada e restrita do problema de *mapeamento top-down* iremos brevemente revisá-lo e ilustrá-lo. Informalmente o mapeamento *top-down* é restrito as operações de remoção e inserção apenas nas folhas das árvores. A Figure 2.2(a) ilustra o mapeamento *top-down* que é formalmente definido como segue.

Definição 2 *Um mapeamento M entre a árvore T_1 e a árvore T_2 é dito* top-down *se, e somente se, para cada par $(i_1, i_2) \in M$ existe também um par(pai(i_1),pai(i_2)) $\in M$, onde i_1 e i_2 não são nodos raíz de T_1 e T_2 respectivamente.*



(a) (b)

Figura 2.2: Exemplo de um mapeamento *top-down* genérico (a) e restrito (b).

O primeiro algoritmo para o problema de distância de edição *top-down* foi proposto por Selkow [Selkow, 1977]. Em [W.Yang, 1991], Yang apresenta um algoritmo recursivo de programação dinâmica com custo $O(n_1 n_2)$ para o problema, onde n_1 a n_2 são os tamanhos de T_1 e T_2, respectivamente.

Um dos algoritmos mais populares para o problema é apresentado em [Chawavate, 1999] também com custo $O(n_1 n_2)$. Este algoritmo, contudo, não é recursivo e o problema é solucionado com uma única instância de programação dinâmica.

Mapeamento *top-down* tem sido aplicado com sucesso em várias aplicações a para Web como categorização de documentos. Por exemplo, Nierman e Jagadish [Nierman and Jagadish, 2002] utilizam o algoritmo de distância *top-down* para agrupar documentos XML.

Em nosso caso, estamos interessado no problema de avaliação da similaridade estrutural entre páginas Web. A maioria das páginas Web são estruturadas com formato como HTML e XML. Como mencionado anteriormente, estas são árvores rotuladas ordenadas e de raíz fixa [Crescenzi et al., 2001]. Desta forma, comumente utiliza-se para a manipulação de páginas Web sua representação DOM subjacente às páginas HTML.

Em nosso trabalho determinamos a similaridade estrutural entre duas páginas Web pela geração do mapeamento *top-down* entre as árvores subjacentes. Isto é realizado através do uso do algoritmo chamado RTDM, que foi inicialmente apresentado em [de Castro Reis et al., 2004] e é brevemente descrito a seguir.

O algoritmo RTDM

O algoritmo RTDM determina uma forma restrita de mapeamento *top-down* entre duas páginas. Além disso as operações de inserção e remoção, e substituição de diferente vértices também é restrita as folhas da árvore. Mais formalmente, temos a seguinte definição.

Definição 3 *Um mapeamento* top-down M *entre a árvore* T_1 *e a árvore* T_2 *é dito* top-down restrito *apenas se para cada par* $(i_1, i_2) \in M$*, tal que* $t_1[i_1] \neq t_2[i_2]$*, não existam descendente* i_1 *ou* i_2 *em* M*, onde* i_1 *e* i_2 *não são nodos raíz* T_1 *e* T_2 *respectivamente.*

A Figura 2.2(b) mostra um mapeamento *top-down* restrito. Podemos definir a *distância de edição top-down restrita* entre duas árvores T_1 e T_2 como sendo o custo do mapeamento top-down restrito entre duas árvores. O algoritmo *RTDM* combina as idéias apresentadas em [W.Yang, 1991] e [Valiente, 2001]. Para determinar o mapeamento *top-dow* restrito entre duas árvores T_1 e T_2, o algoritmo *RTDM* primeiro encontra todas as sub-árvores idênticas que ocorram em um mesmo nível da árvore de entrada. Este passo é realizado em tempo linear usando classes de equivalência de grafos, de maneira similar ao que foi realizado em [Valiente, 2001]. Contudo, o *RTDM* é besado no caminhamento pós-ordem de árvore. Embora muito simples, esta abordagem e aplicável porque olhamos apenas para as sub-árvores idênticas de um *mesmo nível*. Uma vez que os vértices em cada árvore tenham sido agrupados em classes equivalen-

tes, uma adaptação do algoritmo de Yang [W.Yang, 1991] é aplicado para obter o mapeamento *top-down* restrito mínimo entre as árvores.

Como mencionamos anteriormente, o algoritmo tradicional de distância de edição *top-down* de Chawathe [Chawavate, 1999] teve a complexidade de $O(n_1 n_2)$ para todos os casos. O algoritmo *RTDM* também teve para o pior caso a complexidade $O(n_1 n_2)$, mas na prática, este realiza o mapeamento muito melhor que apenas o mapeamento *top-down*.

Como já mencionado, propomos em nossa abordagem técnicas para automaticamente alcançar páginas estruturalmente similares a uma páginas Web dada como exemplo. Outra contribuição de nossa proposta é o fato de que nosso método é capaz de alcançar páginas Web geradas dinamicamente, ou seja páginas da chamada Web invisível, que será detalhada na seção seguinte.

2.3 A Web invisível

Páginas Web geradas dinamicamente e obtidas como resposta às consultas submetidas via formulários HTML possuem conteúdo altamente rico em dados de alta relevância. Sites com estas características formam a chamada *Web invisível* (*Hidden Web*) [Florescu et al., 1998, Lawrence and Giles, 1998].

Considera-se esta parte da Web particularmente importante devido ao fato de que organizações e empresas que dispõe de grande quantidade de informações de alta qualidade, como por exemplo lojas e jornais eletrônicos, instituições de pesquisa e empresas de entretenimento estão cada vez mais disponibilizando seu conteúdo em Web sites dinâmicos. Segundo os estudos de Lawrence e Giles [Lawrence and Giles, 1998], estima-se que 80% do conteúdo da Web invisível é gerado dinamicamente e que este número vem crescendo a cada dia.

Devido a grande quantidade latente de conteúdo rico em informação ainda não indexado e passivo de estruturação disponível nesta parte da Web, o interesse em se coletar tal conteúdo é cada vez maior. No entanto, devido ao fato de que os formulário Web são construídos para serem utilizados por seres humanos, a tarefa de se gerar automaticamente coletores especializados que atendam a qualquer tipo de consulta, é uma tarefa desafiadora.

Como já mencionado, a quantidade de informação disponível na Web Invisível vem crescendo com o passar do anos. É sabido que esse grande volume de informação é formado por páginas

ricas em dados. Consideram-se como páginas ricas em dados aquelas que possuem informações relevantes na forma de *constantes identificáveis* [Embley et al., 1999] e que permitam a criação de banco de dados estruturados, possibilitando desta forma a realização de consultas sofisticadas. Na Figura 2.3 apresentamos uma página com informação sobre livro obtida na livraria *Barnes and Noble*[1]. Podemos observar a existência de informações como o preço do livro, seu título, seu autor, editora, ISBN que exemplificam bem o conceito de páginas ricas em dados. Um banco de dados com informações sobre os livros comercializados por essa livraria eletrônica pode ser construído a partir da extração dos dados desta e de outras páginas similares.

Figura 2.3: Exemplo de página rica em dados

A estruturação da informação disponível nas páginas ricas em dados é realizada por extratores ou *wrappers* [Laender et al., 2002b], programas que extraem dados não estruturados de páginas Web e os armazenas em formatos apropriados como por exemplo XML ou tabelas relacionais. Para realizar esta tarefa os extratores devem receber como entrada um conjunto de páginas previamente coletadas. A construção semi-automática de coletores para obtenção des-

[1]Disponível em `http://www.barnesandnoble.com`

tes conjuntos de páginas é a principal motivação deste trabalho. No Capítulo 4 (Considerando Web Sites Dinâmicos) os métodos utilizados para a geração semi-automática de coletores serão detalhadamente descritos.

Capítulo 3

Geração Automática de Coletores Orientados por Estrutura

Neste capítulo apresentaremos nossa abordagem de geração semi-automática de coletores orientados por estrutura. Iniciaremos ilustrando a abordagem usando um exemplo simples e a seguir discutiremos os detalhes dos dois principais passos envolvidos no processo: *mapeamento do site* e a *geração do padrão de navegação*. Buscando uma melhor compreensão da abordagem, nos limitamos neste capítulo a considerar somente Web sites de conteúdo estático, ou seja, sem formulários. Discutimos os aspectos relacionados ao tratamento deste outro tipo de site no Capítulo 4.

3.1 Visão Geral da Abordagem e Exemplo Motivador

Nesta seção ilustraremos alguns dos conceitos utilizados em nosso trabalho através de um exemplo tirado do Web site E-Jazz[1]. Este site apresenta informações sobre estilos de jazz, instrumentos musicais, discos e artistas de jazz. Neste exemplo enfocaremos a porção do site relacionada a artistas. Uma visão geral do Web site é apresentada de maneira simplificada na Figura 3.1

Nesta figura, as páginas foram rotuladas para futura referência na discussão que se segue. Assuma que todas as páginas rotuladas $0/2/i$ $(1 \leq i \leq k)$ são similares à página $0/2/0$. Todas elas contêm links para páginas de artistas. Chamamos estas páginas de *listas de artistas*. Assuma

[1]Disponível em `http://www.ejazz.com.br`

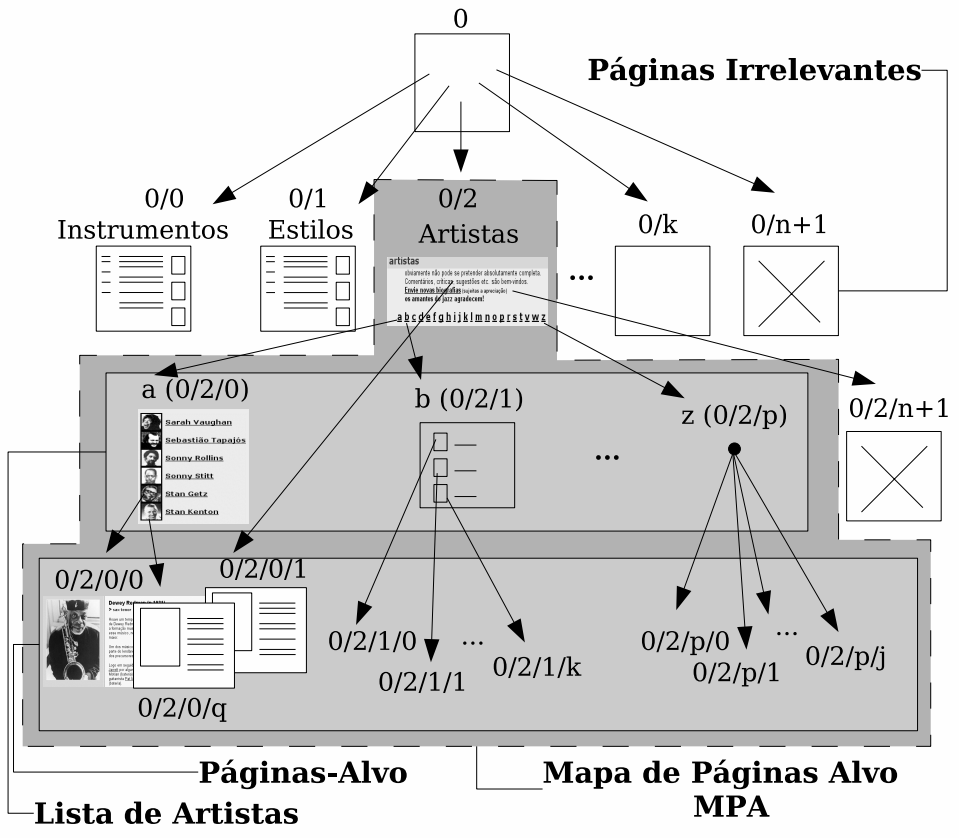


Figura 3.1: Estrutura geral do *Web site* E-Jazz.

também que as páginas rotuladas $0/2/i/j$ $(1 \leq i \leq k, 1 \leq j \leq k_i)$ são similares à página $0/2/0/0$, que é uma página de artista.

Agora, suponha que queiramos alcançar todas as páginas de artistas disponíveis no Web site E-Jazz. Para realizar esta tarefa, um coletor pode iniciar navegação na página $0/2$. Em seguida, deve acessar todas as listas de artista para então finalmente alcançar todas as páginas de artistas do Web site.

Note que novas páginas de artistas podem ser adicionadas ao Web site. Isto implica que algumas listas de artistas são freqüentemente atualizadas com links para novas páginas sendo incluídos ou links de páginas excluídas sendo removidos. Assim, estas alterações deverão ser considerados em futuros processos de coleta de páginas de artistas.

Para gerar um coletor capaz de realizar tal tarefa, a nossa abordagem requer dois parâmetros como entrada: uma URL para indicar uma *página exemplo* para as páginas a serem coletadas e outra URL indicando um *ponto de entrada* no Web site onde estas páginas devem ser encontradas. Em nosso exemplo, a página $0/2/0/0$ servirá como a página de exemplo, enquanto a

página 0/2 será o ponto de entrada no Web site.

A geração automática de um coletor, de acordo com a nossa, é realizada em duas fases distintas: *mapeamento do site* e *geração do padrão de navegação*. Na fase de mapeamento do site, é feito um caminhamento em largura por todo o site iniciando no ponto de entrada a procura por quaisquer *páginas-alvo* em sua trajetória. Uma página é considerada como uma página-alvo quando esta é estruturalmente similar à página dada como exemplo. Note que, a ação do coletor é limitada a um mesmo Web site. Caso seja encontrada durante este caminhamento alguma página que possua formulários, são utilizadas heurísticas e procedimentos especiais que serão discutidos no Capítulo 4. Cada caminho a partir do *ponto de entrada* que leva a uma página-alvo é guardado. Assim, se considerarmos o Web site como um grafo, a fase de mapeamento do site gera como saída uma árvore de amplitude mínima (*minimum spanning tree*) [Cormen et al., 2002] deste grafo onde todas as folhas são páginas alvo. Esta árvore é chamada de *Mapa de Páginas Alvo (MPA)*. Na Figura 3.1 o MPA corresponde à árvore delimitada por uma linha tracejada.

Na segunda fase, geração do padrão de navegação, o objetivo é criar uma representação genérica da MPA. Esta representação genérica corresponde a uma lista de expressões regulares, onde cada expressão representa os links em um página que o coletor gerado deverá seguir para alcançar as páginas-alvo. Esta representação genérica é chamada de *Padrão de Navegação (PN)*. Um PN funciona como um "guia" utilizado pelo coletor em sua navegação pelo site à procura de páginas-alvo. Desta forma, ele deve corresponder a um único caminho dentro da MPA. Contudo, como podem existir muitos caminhos que levam a páginas-alvo, escolhemos aquele que potencialmente leva ao maior número de páginas-alvo. Note que, se corretamente geradas, as expressões regulares no PN irão identificar novos links para páginas no Web site, mesmo depois da fase de mapeamento do site ter sido concluída, desde que se utilize as expressões regulares para identificar os links que levam às páginas-alvo.

Como um exemplo, a Tabela 3.1 mostra um padrão de navegação bem simples, porém real, para o Web site E-Jazz. As expressões regulares neste exemplo usam a sintaxe da linguagem Perl. Nesta tabela, a expressão E_1 é aplicada para o ponto de entrada. Tal expressão casa apenas com os links que levam às listas de artistas. Em seguida, a expressão E_2 é aplicada para cada lista de artista e casa com todos os links nestas páginas que levam a todas as páginas de artista, e apenas a estas páginas. Para gerar um padrão de navegação, o nosso método não requer qualquer intervenção do usuário, além da definição do ponto de entrada e da página-alvo.

Nas seções seguintes iremos detalhar as duas fases necessárias ao metódo para a geração de coletores orientados por estrutura.

Exp. Id	Expressão Regular	Aplicada em
E_1	`http://www.ejazz.com.br/artistas/default_^[a-zA-Z]+$`	Ponto de entrada
E_2	`http://www.ejazz.com.br/detalhes-artistas.asp?cd=^d+$`	lista de artistas

Tabela 3.1: Exemplo de padrão de navegação.

3.2 Mapeamento do Web site

Na fase de mapeamento, consideramos o Web site como sendo um grafo direcionado $S = \langle P, L \rangle$, onde P é o conjunto de paginas no Web site S e L é o conjunto de pares $\langle x, y \rangle$ tal que existe um link da página x para a página y em S. O objetivo final desta fase é gerar uma árvore chamada *Mapa de Páginas-Alvo (MPA)* definida como segue.

Definição 4 *Seja $S = \langle P, L \rangle$ um Web site como definido acima. Sejam $\boldsymbol{e}$,$\boldsymbol{p} \in \boldsymbol{P}$ respectivamente o ponto de entrada e a página exemplo fornecida pelo usuário. Definimos um* ***Mapa de Páginas Alvo (MPA)*** *como uma árvore $\boldsymbol{T}$ que é um subconjunto da árvore de amplitude mínima de $\boldsymbol{S}$ onde $\boldsymbol{e}$ é a raiz e o conjunto de nodos folhas é formado exatamente pelo conjunto de nodos p_i de $\boldsymbol{S}$ que são estruturalmente similares a $\boldsymbol{p}$.*

Para gerar a MPA, simplesmente coletamos o Web site iniciando do ponto de entrada usando caminhamento em largura (*breadth-first traversal*). Este procedimento é descrito no Algoritmo 2. O algoritmo recebe um ponto de entrada $\boldsymbol{e}$ de um Web site e uma página de exemplo $\boldsymbol{p}$ que será utilizada para comparação. Inicialmente, na Linha 4 todos os links em $\boldsymbol{e}$ são enfileirados em Q. Em seguida, o processo de coleta itera sobre essa fila no laço das Linhas de 5 a 12. Observamos que o caminhamento em largura é a melhor escolha para mapear o Web site, uma vez que em [Najork and Wiener, 2001] esta estratégia é citada como a melhor para realizar a cobertura abrangente de Web sites.

Esta iteração termina quando a fila está vazia ou a função ***Para()*** retorna o valor verdadeiro. Esta função codifica um conjunto de restrições para o coletor, por exemplo, o número máximo de páginas alcançadas, o número máximo de níveis a serem alcançados, etc. Em nossa implementação estas restrições são facilmente estimadas pelo usuário na forma de parâmetros.

```
1  MAPEAMENTO
   Entrada: Um ponto de entrada e do Web site S e uma página exemplo p
   Saída: A MPA T
2  Início
3  seja X um conjunto de páginas inicialmente vazio
4  seja Q uma fila
5  Q ← links extraídos de e
6  while Q ≠ ∅ || Para() do
7  |  x ← Desenfileira(Q)
8  |  if RTDMSim(p, x) then
9  |  |  X ← X ∪ {x}
10 |  else
11 |  |  Q ← links etraídos de {x}
12 |  end
13 end
14 foreach x ∈ X do
15 |  seja C = ⟨e, v1, ..., vk, x⟩ o caminho de e para x
16 |  Adiciona nós e arcos C a T
17 end
```

Algoritmo 2: Procedimento utilizado na fase de mapeamento do site.

Note que estes parâmetros são utilizados apenas na fase de mapeamento e não são necessários após a geração do coletor.

Nas Linhas 6 e 7 o elemento do início da fila é comparado com $\boldsymbol{p}$ usando a função de cálculo de similaridade ***RTDMSim()***, descrita adiante. Se a página corrente x e a página de exemplo $\boldsymbol{p}$ são consideradas estruturalmente similares, adicionamos x ao conjunto X de páginas-alvo (Linha 8) que devem ser coletadas. Se x e p não são similares, os links em x são enfileirados (Linha 10) e a navegação continua. No final da iteração, o conjunto X conterá todas as páginas-alvo encontradas.

A seguir, na iteração das Linhas de 13-16, todos os nodos e arcos presentes no caminho da página de entrada até as páginas-alvo em X serão adicionadas ao mapa de páginas-alvo (MPA) T. Observa-se que o procedimento de mapeamento obtém uma sub-árvore da árvore de amplitude mínima do Web site. Em nosso exemplo, a MPA gerada deve corresponder a árvore com linha tracejada na Figura 3.1.

Note que a página rotulada 0/2/0/1 é diretamente conectada ao ponto de entrada 0/2, enquanto todas as outras páginas similares próximas são conectadas à lista de artistas rotulada 0/2/0. Por outro lado, existe um link no Web site da página 0/2/0 para a página 0/2/0/1, sendo que este não foi considerado, neste caso, porque o coletor encontrou um link de 0/2 à 0/2/0/1

primeiro. Tais atalhos são comuns na maioria dos Web sites, e constituem anomalias para o mapeamento. Explicaremos como tratá-los mais adiante.

Na seção seguinte, discutiremos a segunda fase de nossa abordagem para construir o coletor orientado por estrutura: a fase de geração do padrão de navegação.

3.3 Geração de Padrão de Navegação

O objetivo desta fase é construir um *Padrão de Navegação (PN)* a partir da MPA gerada na fase de mapeamento. Como já havíamos mencionado, o PN consiste de uma lista de expressões regulares que representam os links em uma página que o coletor gerado deverá seguir para alcançar as páginas-alvo. Para gerá-lo, temos de generalizar as URLs das páginas nos caminhos do MPA usando expressões regulares, e selecionando entre todos os possíveis caminhos a partir do ponto de entrada, o melhor dentre eles que levam às páginas-alvo desejadas.

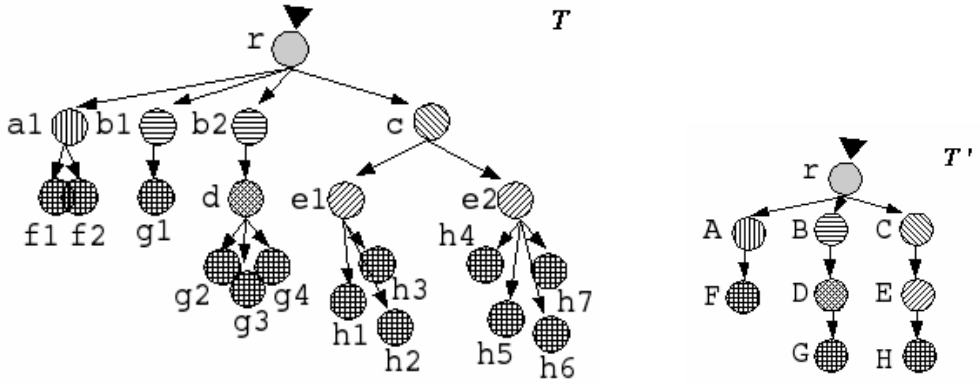
O primeiro passo desta fase consiste do agrupamento dos nodos na MPA que representam páginas com URLs que são consideradas *similares*. Adiaremos por enquanto a discussão sobre o conceito de similaridade de URL. O procedimento para realizar este agrupamento é apresentado no Algoritmo 3.

O procedimento **Group** é inicialmente aplicado à raiz da MPA e recursivamente aplicado aos seus nodos filhos. Neste procedimento, $\mu(x)$ denota a URL do nodo x, e $C(x)$ denota o conjunto de nodos filhos do nodo x. Para um dado nodo r, as Linhas de 3 até 6 definem um particionamento no conjunto de nodos filhos de r, onde cada partição G_i, contem apenas nodos correspondentes às páginas com URL similares. Para avaliar a similaridade entre duas URLs utilizamos a função ***URLsim()*** (Linha 6). Esta função será detalhada adiante.

Em seguida, na Linha 9, um novo nodo n_i é criado a partir dos nodos em G_i da seguinte forma: a URL em n_i corresponde à expressão regular π que generaliza o conjunto de URL dos nodos em G_i de acordo com a função ***URLpattern***, descrita no Algoritmo 4. Esta função será detalhada mais a frente. O conjunto de filhos de n_i é composto por todos os filhos dos nodos em G_i. Assim, todos os nodos em G_i são removidos do conjunto r (Linha 10), e substituídos pelos nodos n_i (Linha 11). Em resumo, substituímos todos os nodos correspondentes às páginas com URLs similares por um único nodo que os representa. Finalmente, na Linha 12, o procedimento **Group** é recursivamente invocado para o novo nó n_i.

```
1  Group(r)
2  seja G = {G_1, G_2, ..., G_k}, k ≤ |C(r)| tal que:
3        G_1 ∪ ... ∪ G_k = Cr e
4        G_i ∩ G_j = ∅ e
5        c_ℓ ∈ G_i sse c_m ∈ G_i e URLsim(μ(c_ℓ), μ(c_m))
6  foreach G_i = {c_i1 ... c_ik} do
7  |     π = URLpattern
8  |     n_i ← ⟨π, C(c_i1) ∪ ... ∪ C(c_ik)⟩
9  |     Remover todo c_ij ∈ G_i como filho de r
10 |     Adicionar n_i como filho de r
11 |     chama rotina Group(n_i)
12 end
```

Algoritmo 3: Procedimento de agrupamento de nós.



(a) (b)

Figura 3.2: Ilustração do procedimento **Group**.

A execução do procedimento **Group** é ilustrada na Figura 3.2, onde o lado esquerdo (a), ilustra uma MPA T e o lado direito (b) ilustra a árvore T' gerada como resultado da aplicação do procedimento. Considere que os nodos com o mesmo padrão de preenchimento correspondem a páginas com URLs similares. Cada nodo representado por uma letra maiúscula (`A,B,C,...`) em T', exceto pela raiz r, agrupa todos os nodos de T representados por letras minúsculas (`a1,b1,b2,...`). Note que os nós rotulados `A,C` e `D` representam os nodos `a,c,d,` respectivamente.

Após a execução do procedimento de agrupamento, é possível que tenhamos mais de um caminho que leva às páginas-alvo. Isto é ilustrado na Figura 3.2 onde três possíveis caminhos foram encontrados: r até F, r até G e r até H. Destes três caminhos, escolhemos o que leva ao maior número de paginas-alvo. Em nosso exemplo, observamos que esta condição é satisfeita através do caminho de r até H. Assim, este caminho irá finalmente ser escolhido como o padrão

de navegação NP. Este conceito é formalizado na Definição 5

Definição 5 *Seja* ***S*** *um Web site e seja* ***T*** *a MPA para* ***S*** *obtida quando* ***e,p*** $\in$ ***S*** *são respectivamente o ponto de entrada e a página exemplo fornecida pelo usuário. Um* ***NP*** *é o caminho na árvore* T' *obtida após a aplicação do procedimento* ***Group*** *sobre* ***T****, que inicia em* ***e*** *e termina no nodo de* T' *que corresponde ao maior número de páginas-alvo em* ***T*** *.*

A seguir discutiremos como medimos a similaridade entre duas URLs utilizando a função ***URLsim()*** (Linha 6 do Algoritmo 3) e como geramos um padrão para representar o conjunto de URL similares utilizando a função URLpattern (Linha 8 do mesmo algoritmo).

Avaliação de Similaridade Entre URLs

Consideramos uma URL como sendo uma cadeia de caracteres formada por várias subcadeias separadas pelo caracter "/". Chamamos estas subcadeia de *níveis*. Assim, seja $u = u_1/u_2/.../u_n$ $(n \geq r)$ e $v = v_1/v_2/.../v_m$ $(m \geq 1)$ duas URLs. Estas são consideradas similares se, e somente se:

(1) elas têm o mesmo número de níveis, $(n = m)$;

(2) o primeiro nível em u a v são iguais $(u_1 = v_1)$; e

(3) existem no máximo K níveis na mesma posição em u e v (exceto o primeiro) que não são iguais, ou seja, para $\delta = \{\langle u_i, v_i\rangle \mid u_i \neq v_i\}$, então $|\delta| \leq K$.

Em experimentos preliminares com várias URLs, observamos que o valor de $K = 2$ resulta em uma avaliação de similaridade precisa e segura. Este resultado foi confirmado em nossos experimentos finais.

Assim a função **URLsim**, chamada na Linha 5 do Algoritmo 3, retorna verdadeiro para as URLs u e v se estas satisfizerem as três condições acima.

Geração do Padrão de URLs

Detalharemos a seguir o procedimento utilizado para gerar o padrão que representa o conjunto de URLs, utilizada no Algoritmo 4. Consideramos aqui que a similaridade de URL é determinada da mesma forma descrita acima e que essas URLs diferem no máximo em K níveis. Assim, nossa estratégia para a geração do padrão de URLs consiste em construir expressões regulares

que casam com todas as cadeias que diferem em cada nível nas URLs. Para isto introduzimos a noção da árvore de expressões regulares ou **Árvore Regex**.

Definição 6 *Uma* ***Árvore Regex*** *R é uma hierarquia em que cada nodo* $\boldsymbol{n}$ *é associado com uma expressão regular* e_n *sobre um alfabeto* Σ *que denota uma linguagem* L_n*, tal que, para cada nodo interno* $\boldsymbol{a}$ *em R cujos filhos são* $\{c_1, \ldots, c_k\}$*, temos* $L_{c_1} \cup \ldots \cup L_{c_k} = L_a$ *e* $L_{c_1} \cap \ldots \cap L_{c_k} = \emptyset$*.*

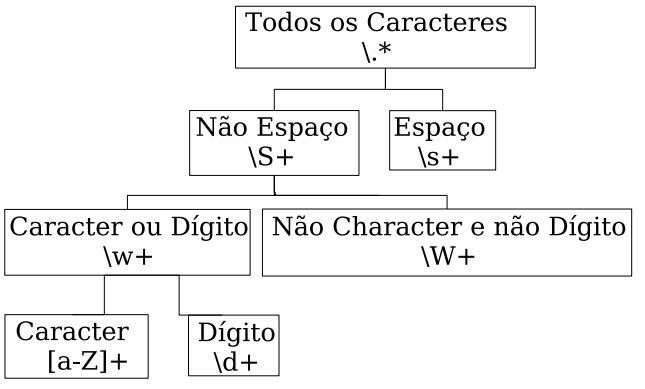


Figura 3.3: Árvore de expressões regulares.

A árvore de expressões regulares adotada em nosso trabalho é apresentada na Figura 3.3. Esta árvore é uma adaptação da chamada *Delimiters Tree* definida em [de Castro Reis et al., 2002]. De acordo com a Definição 6, a árvore de expressões regulares é construída de tal forma que cada nodo define uma linguagem que é disjunta às linguagens dos demais nodos de um mesmo nível da árvore. Assim, quando aplicamos os fragmentos de uma URL, um dado fragmento irá casar com apenas um nodo folha. Esta propriedade nos permite usar um nodo folha em uma árvore de expressões para fragmentar uma URL ou um nível da URL.

Por outro lado, cada nodo interno define uma linguagem que é formada pela união das linguagens dos seus nodos filhos. Esta propriedade permite-nos encontrar uma única expressão regular que case todos os fragmentos em uma mesma posição que ocorre em um conjunto de URLs. Sejam l_t e l_s dois nodos filhos que casam fragmentos distintos t e s. O nodo a que é o ancestral comum de l_t e l_s define uma expressão regular que casa t e s.

O procedimento completo para a gerar um padrão de URL, é descrito no Algoritmo 4, onde o símbolo "$\bullet$" é usado para denotar a operação de concatenação de cadeias de caracteres. Para explicar este procedimento, iremos utilizar um conjunto de URLs exemplo apresentadas

na Figura 3.4. Quando recebe como entrada estas URLs, o procedimento ***URLPattern*** itera no laço das Linha de 5 a 16 sobre os níveis nas URLs que são distintos. Como ilustrado na Figura 3.4(a), apenas os níveis 6 e 7 serão processados neste laço. Detalharemos o nível 7, pois este ilustra melhor a funcionalidade do procedimento. Este nível é apresentado na Figura 3.4(b).

	$u_i[1]$	$u_i[2]$	$u_i[3]$	$u_i[4]$	$u_i[5]$	$u_i[6]$	$u_i[7]$
u_1	informatik.uni-trier.de	∼ley	db	indices	a-tree	c	Chaves:Silvio_8=.html
u_2	informatik.uni-trier.de	∼ley	db	indices	a-tree	u	Ugher:Mangabeira_D=.html
u_3	informatik.uni-trier.de	∼ley	db	indices	a-tree	m	Mendes:Sergio.html
π	informatik.uni-trier.de	∼ley	db	indices	a-tree	[a-Z]+	[a-Z]+:[a-Z]+_*w*.html

(a)

	$\hat{f}_i[7][1]$	$\hat{f}_i[7][2]$	$\hat{f}_i[7][3]$	$\hat{f}_i[7][4]$	$\hat{f}_i[7][5]$	s
$u_1[7]$	Chaves	:	Silvio	_	8	.html
$u_2[7]$	Ugher	:	Mangabeira	_	D	.html
$u_3[7]$	Mendes	:	Sergio	λ	λ	.html
π	$[a-Z]+$	:	$[a-Z]+$	_*	∼w*	.html

(b)

Figura 3.4: Exemplo de um Padrão de navegação.

1 **URLPattern**
  Entrada: $U = \{u_1, u_2, \ldots, u_n\}$, um conjunto de URLs
  Saída: u_π, um padrão de URL
2 início
3 $u_\pi \leftarrow u_1$
4 seja $u_i[k]$ o $k-$ésimo nível da URL u_i
5 **foreach** *k tal que* $\{\langle u_1[k], \ldots, u_n[k]\rangle \mid u_i[k] \neq u_j[k]$ *para algum* $1 \leq i, j \leq n\}$ **do**
6 seja $u_i[k] = p \bullet f_i[k] \bullet s$, onde p (s) é o prefixo comum entre todo $u_i[k]$
7 **for** $i \leftarrow 1$***até*** n **do**
8 $\langle f_i[k][1], f_i[k][2], \ldots, f_i[k][m_{i,k}]\rangle \leftarrow$ **Tokenize**$(f_i[k])$
9 **end**
10 **for** $j \leftarrow 1$***até*** MAX $\{m_{1,k}, m_{2,k}, \ldots, m_{n,k}\}$ **do**
11 $x \leftarrow$ **TokenRegex**$(\hat{f}_1[k][j], \hat{f}_2[k][j], \ldots, \hat{f}_n[k][j])$
12 onde $\hat{f}_i[k][j] = f_1[k][j]$ se $j \leq m_{k,i}$ ou $tf_1[k][j] = \lambda$
13 $\pi \leftarrow \pi \bullet x$
14 **end**
15 $u_\pi[k] \leftarrow s \bullet \pi \bullet p$
16 **end**

Algoritmo 4: Procedimento de geração de padrões de URL

De acordo com a Linha 6, não existe um prefixo comum entre todos $u_i[7]$, mas existe um sufixo comum "`.html`" entre eles. No laço das Linhas de 7 a 9, é chamada a rotina **Tokenize**, descrita no Algoritmo 5, responsável pela fragmentar cada infixo $f_i[7]$ gerando os fragmentos presentes na Figura 3.4(b). Após isso, no laço da Linhas de 10 a 14, cada conjunto de fragmentos

em uma mesma posição dos infixos é passado como argumento para a chamada da função **TokenRegex**, descrita no Algoritmo 6. Isto significa que na Figura 3.4(b), cada coluna rotulada $\hat{f}_i[7][j]$ irá receber um argumento de **TokenRegex**. O resultado de cada chamada aparece na linha rotulada de π. Para coluna $\hat{f}_i[7][1]$ todos os fragmentos irão casar com alguma folha na árvore de expressões da Figura 3.3. Assim, a expressão regular é usada neste nó. Para a coluna $\hat{f}_i[7][2]$, como o mesmo fragmento se repete para todas as linhas, é simples de se obter a expressão regular, bastando repetir o valor da coluna. Na coluna $\hat{f}_i[7][5]$, observe que os fragmentos "8" e "D" casam com folhas distintas na árvore de expressões e o ancestral comum a ambas corresponde ao nodo cuja expressão regular é "$\backslash w$".

```
1  Tokenize
   Entrada: Uma cadeia de caracteres
   Saída: T = {s_1, s_2, ..., s_n}, um conjunto de fragmentos
2  início
3  seja R a árvore de expressões regulares
4  i ← 0
5  while s ≠ "" do
6  |  i++
7  |  s_i ← a maior subcadeia da caracteres mais a esquerda de s que casa com a folha em R
8  |  s ← s − s_i
9  end
```

Algoritmo 5: Rotina **Tokenize**, responsável por *tokenizar* as URL de entrada

```
1  TokenRegex
   Entrada: T = {t_1, t_2, ..., t_n}, um conjunto de tokens
   Saída: p_T, um expressão regular que casa com os tokens em T
2  início
3  if t_i = t_j  para todo  t_i, t_j ≠ λ then
4  |  p_T ← t_i
5  else
6  |  seja R be a Regex tree
7  |  seja η(t_i) o nó folha que casa com o fragmento t_i ≠ λ
8  |  a ← o ancestral comum de todos η(t_i)
9  |  p_T ← e_a
10 end
11 if existe um t_i = λ then
12 |  p_T ← p_t • *
13 else
14 end
```

Algoritmo 6: Rotina **TokenRegex**, responsável por verificar com qual expressão regular a cada token deve ser casado

3.4 Coletor Orientado por Estrutura

Nas seções anteriores descrevemos como criar um padrão de navegação (PN) para alcançar páginas alvo em um Web site. Nesta seção descreveremos como utilizar o PN gerado para guiar o coletor, este procedimento é apresentado no Algoritmo 7. Basicamente, este algoritmo consiste de um caminhamento em largura em um Web site onde os links a serem seguidos são selecionados de acordo com as expressões regulares em NP.

O algoritmo utiliza um fila Q cujos elementos são pares $\langle u, j\rangle$, onde u é uma URL e j indica o nível corrente sendo navegado no Web site em um dado momento, No laço das Linhas 9-11 os links da página de entrada que casam com a primeira expressão regular π no padrão de navegação são enfileirados. A partir deste ponto, enquanto a fila não estiver vazia, as páginas no site serão visitadas usando caminhamento em largura. Se a condição na Linha 15 falhar, isto significa que a página-alvo no último nível (n) foi alcançada. Assim, na Linha 18, esta página é armazenada. Ao final da execução, todas as páginas-alvo do Web site serão alcançadas.

```
1  Coletor guiado por estrutura
   Entrada: Um ponto de entrada e de um site S; Uma página exemplo p; O padrão de
            navegação
   Saída: O conjunto P de páginas estruturalmente similares a p
2  início
3  seja Q uma fila
4  NP = ⟨π1, ..., πn⟩
5  foreach link l de e que casa com π1 do
6  |  Q ← Q ∪ {⟨l, 2⟩}
7  end
8  while Q ≠ ∅ do
9  |  ⟨u, j⟩ ← Desenfileira(Q)
10 |  g ← Download(u)
11 |  if j ≠ n then
12 |  |  foreach link l de g que casa πj do
13 |  |  |  Q ← Q ∪ {⟨l, j + 1⟩}
14 |  |  end
15 |  else
16 |  |  P ← P ∪ {g}
17 |  end
18 end
```

Algoritmo 7: Coletor orientado por estrutura

Capítulo 4

Considerando Web Sites Dinâmicos

Até o momento nossa discussão teve como foco principal os sites estáticos. No entanto, como um volume muito grande de páginas de interesse para as aplicações está na chamada Web invisível ou *Hidden Web*, é necessário que os coletores gerados sejam capazes de tratar os Web sites dinâmicos. Desta forma, caso páginas que contenham formulários sejam encontradas durante a fase de mapeamento (Seção 3.2) em um dado Web site, deve-se decidir se o formulário será ou não incluído no mapa de páginas-alvo, ou seja, se este é capaz de gerar dinamicamente páginas-alvo. Caso seja, deve-se também determinar quais são os parâmetros necessários ao preenchimento do formulário para atingir estas páginas-alvo, e como estes devem ser preenchidos.

Neste capítulo discutiremos as questões relacionadas ao tratamento de sites dinâmicos em nossa abordagem. Iniciaremos esta discussão apresentando um exemplo de formulário codificado em HTML como forma de detalhar as características básicas destes formulários. Em seguida, apresentamos heurísticas que foram desenvolvidas para o preenchimento automático de formulários, incluindo uma nova estratégia para o preenchimento de campos de texto. Finalmente, descrevemos nossa estratégia para a submissão das consultas e análise das páginas obtidas como resposta.

4.1 Características dos Formulário HTML

Nesta seção serão descritas as características básicas de um formulário HTML. Nas discussões que seguem iremos considerar como um formulário o trecho de código HTML contido entre as tags <FORM> e </FORM> das páginas Web, como mostra a Figura 4.1. Esta figura consiste de um

formulário que obtém páginas com informações de filmes. Pode-se observar a existência de um menu de seleção (*drop down*) codificado entre as tags <SELECT name='select'> e </SELECT>; uma caixa de texto codificada com a tag <INPUT type='text' name='for' size='14'>; e as informações necessárias para a submissão do formulário que estão contidas na tag <FORM action='/find.cgi' method='get' NAME='QSFORM'>.

Diferentemente do trabalho de [Chen et al., 2004] que modela formulários como pares ⟨nome,valor⟩ e do trabalho de [Raghavan and Garcia-Molina, 2001] que utiliza pares ⟨domínio, valor⟩ para referenciar as informações extraídas dos formulários, utilizamos em nosso trabalho, a seguinte modelagem: Um formulário F é um conjunto de pares ⟨tipo,valor⟩, $F = \{(t_1, v_1), (t_2, v_2), ..., (t_k, v_k)\}$ onde a cada t_i é associado o nome do tipo padrão de entrada no formulário (*selction lists, text box, checkboxes, radio buttons, etc.*) para cada campo extraído deste, e v é o seu valor único. Essa modelagem foi adotada por possuir maior flexibilidade que as propostas anteriormente.

De acordo com essa modelagem, teremos para o formulário da Figura 4.1 a seguinte modelagem: $F = \{$⟨select, all⟩, ⟨select, title⟩, ⟨select, director⟩, ⟨select, cast⟩, ⟨for,"*texto*"⟩$\}$

```
<FORM action="/find.cgi" method="get" NAME="QSFORM">
  <SELECT name="select">
    <OPTION value="all"     > All      </OPTION>
    <OPTION value="title"   > Titles   </OPTION>
    <OPTION value="director"> Director </OPTION>
    <OPTION value="cast"    > Casting  </OPTION>
  </SELECT>
    <INPUT type="text" name="for" size=" 14">
    <SUBMIT name="submit">
</FORM>
```

Figura 4.1: Exemplo de um formulário HTML.

Ainda na Figura 4.1, na tag <FORM>, podemos observar o método de submissão do formulário descrito pelo atributo método, que neste caso em particular é GET. Porém, nada impede que durante a implementação do formulário este seja codificado como POST. A principal diferença entre estes dois métodos consiste na forma como o navegador (*browser*) codifica os dados a serem enviados a um determinado servidor Web, pois enquanto no método GET a submissão é enviada diretamente na URL, no POST os dados são codificados no cabeçalho HTTP.

No método GET as consultas são geradas concatenando-se a URL onde se encontra o formulário com valor definido para o atributo `action` e com o caracter especial "?" seguido de t_1, que é o nome do primeiro tipo de entrada do formulário, concatenado com "=", que por sua

vez é concatenado com o valor v_1. Caso existam outros campos no formulário, estes também farão parte da submissão concatenando a primeira parte gerada com o caracter "`&`" seguido da concatenação dos demais tipos $(t_2, ..., t_k)$.

Na tag <`FORM`> da Figura 4.1 o valor do atributo `action` é `find.cgi`. Considerando a URL que contém esse formulário como sendo `http://www.meuformulariohtml.com.br/`, teríamos, `http://www.meuformulariohtml.com.br/find.cgi?select=all&for=meutexto` ou `http://www.meuformulariohtml.com.br/find.cgi?select=title&for=meutexto`, entre outras, como possíveis submissões GET. Já no processo de submissão do método POST, as requisições geradas do formulário F como *select=all*, for=*meutexto*, etc., são submetidas juntamente com o cabeçalho HTTP.

Considerando as características descritas acima, desenvolvemos estratégias para automaticamente preencher os formulários encontrados durante a fase de mapeamento dos Web sites. Assim, descreveremos a seguir na Seção 4.2 como são geradas as consultas a serem submetidas para navegar através dos formulários e, em seguida, na Seção 4.3, discutiremos como essas consultas são submetidas e suas respostas analisadas.

4.2 Preenchimento Automático de Formulários Web

Nesta seção trataremos especificamente da estratégia utilizada neste trabalho para o preenchimento automático de formulários Web. Detalharemos os passos envolvidos no processo, a escolha dos formulário, seu tratamento, a obtenção de seus dados e a geração das consultas.

A maioria dos trabalhos na literatura que tratam do preenchimento automático de formulários tem como objetivo coletar todas ou a grande maioria das páginas geradas dinamicamente [Barbosa and Freire, 2004, Chen et al., 2004, Fontes and Silva, 2004, Raghavan and Garcia-Molina, 2001]. Em nosso trabalho, temos como objetivo principal gerar um padrão de navegação para coletar automaticamente as páginas estruturalmente similares à página dada como exemplo de interesse do usuário. Desta forma, nos interessa somente determinar se um formulário encontrado durante a fase de mapeamento deverá ou não fazer parte do mapa de páginas-alvo, ou seja se ele é capaz de gerar páginas-alvo.

Uma vez que se tenha realizado esta verificação e o formulário encontrado tenha sido tratado, devem ser registrados os parâmetros necessários ao preenchimento do formulário, por exemplo,

autor ou título de um livro, para que sejam passados os argumentos necessários a futura realização de coletas. Desta forma, como demonstrado nos experimentos, é possível atingir com precisão as páginas de interesse do usuário. A seguir dicutiremos este processo.

A nossa estratégia consiste inicialmente em verificar se o formulário é um formulário inválido. Consideramos como formulário inválido todo aquele que necessite de parâmetros como e-mail, senha, identificação do usuário, ou que possua código Java Script em seu conteúdo. Caso pelo menos um destes itens seja encontrado no formulário, este será desconsiderado.

Uma vez que foi verificado que o formulário encontrado pode ser manuseado, extraímos todos os campos deste formulário e criamos o conjunto P de pares ⟨tipo, valor⟩ onde para cada tipo específico podem existir valores codificados no formulário que serão utilizados futuramente para a geração automática de consultas. De acordo com o tipo encontrado em um determinado campo do formulário, pode existir um conjunto de possíveis valores que durante a formação destes pares serão selecionados um a um. Por exemplo, na Figura 4.1 o tipo "`SELECT`" de nome "`select`" possui quatro valores que são `all, title, director e cast`. Assim o conjunto de pares formados, para este tipo em especial será {⟨select, all⟩, ⟨select, title⟩, ⟨select, director⟩, ⟨select, cast⟩}.

Para o caso de formulários que não possuam campo de texto, são realizadas todas as possíveis combinações de tipos e valores e então é gerado um vetor de consultas. Porém, se no conjunto P for detectada a existência de pelo menos um campo do tipo texto é realizado um tratamento especial.

O tratamento consiste em associar a cada campo de texto termos que, quando usados para o seu preenchimento, possam gerar páginas de resposta válidas. Como não sabemos à priori quais são estes termos, partimos do princípio de que estes devem ser representativos do tópico ou assunto do formulário do site. Assim, extraimos da página de exemplo e da página do formulário cadeias de caracteres que consideramos representativos deste tópico e usamos para o preenchimento. Mais precisamente nossa estratégia para tratamento de campo de texto consiste em:

1. Extrair os termos menos frequentes da página de exemplo e da página que contém o formulário.

2. Extrair os rótulos dos campos de texto e gerar um conjunto P' de pares ⟨rótulo,valor⟩.

Para o passo dois, utilizamos uma série de heurísticas de extração de rótulos adaptadas de [Fontes and Silva, 2004], [Lage et al., 2004] e [Zhang et al., 2004]. Diferente das abordagem de [Chakrabarti et al., 2002], [Bergmark et al., 2002], [Lage et al., 2004] e [Liu et al., 2004], nós não utilizamos aqui banco de dados previamente fornecidos para fazer o preenchimento automático de campos de texto de formulários. Em nossa abordagem consideramos apenas os 15 termos menos frequentes da página de exemplo e da página que contém o formulário. Este valor foi estipulado, e considerado como adequado, após a realização de diversos experimentos onde observou-se que dentre estes 15 termos sempre haviam termos adequados para as consultas.

Ao final do processo temos, para cada campo do formulário, um conjunto de valores de preenchimento que são plausíveis. A seguir, são gerados diversas combinações de preenchimento dos campos, com cada combinação correspondente a uma consulta codificada no formulário preenchido.

Como um exemplo considere o formulário apresentado na Figura 4.1 e um conjunto dos 15 termos manos frequente ocorrendo na página Web dada como exemplo e na página de contém o formulário, como sendo (*Harry*, *Potter*, *Joanne*, *Kathleen*, *Rowling*, *filosofal*, *quadribol*, *Voldemort*, *Hogwarts*, *Azkaban*, *Rocco*, *Nárnia*, *Lewis*, *York*, *engalfinhando*). Teríamos como alguma das possíveis consultas para este formulário as seguintes:

- `select=all&for=Harry`
- `select=all&for=Potter`
- `select=all&for=Joanne`
- `select=title&for=Harry`

Uma vez que tenhamos as possíveis combinações geradas para cada campo do formulário passamos para a etapa de submissão e coleta das páginas resposta, que será discutido na seção seguinte.

4.3 Plano de Submissão e Coleta de Páginas Respostas

Nesta seção discutiremos os procedimentos utilizados para a submissão das consultas e obtenção e análise das páginas de resposta.

De posse do conjunto de consultas geradas, podemos realizar as submissões, seguindo as devidas particularidades dos métodos GET e POST.

Inicialmente é realizada a submissão e obtenção das páginas resposta para todas as consultas geradas. Em seguida, as páginas são analisadas. Para esta análise, consideramos somente os 30% das páginas obtidas que possuam o maior tamanho em bytes. Este valor foi estimado após exaustivos experimentos prévios onde verificamos que dentre estas consultas sempre existem consultas válidas.

Para análise, dentre as páginas coletadas podemos encontrar os seguintes casos:

1. Páginas-alvo: Se a página coletada é estruturalmente similar à página de exemplo, ela é guardada e o formulário em questão será posteriormente adicionado ao mapa de páginas-alvo.

2. Páginas de formulários: Neste caso para tratar esse novo formulário o procedimento descrito na Seção 4.2 é executado.

3. Páginas com links: Este é o caso mais comum e será detalhado na Seção 4.3.1.

4. Páginas de erro: Se consulta submetida é inválida, uma página de erro é gerada. Descrevemos o tratamento destas páginas na Seção 4.3.2.

4.3.1 Seguindo Links nas Páginas Resposta

Uma maneira simples de tratar esse tipo de página de resposta seria simplesmente seguir cada link da maneira usual descrita no Capítulo 3. No entanto observamos em experimentos preliminares, que o número de links é em geral muito grande. Portanto, para reduzir o número de links visitados, estes são agrupados de forma que em cada agrupamento U_i de links existam somente URLs que são similares entre si, conforme descrito na Seção 3.3. A seguir, tomamos subconjuntos aleatórios U_i' da cada U_i. Então, seguimos as URLs em U_i', e , se estas levam à páginas-alvo, adicionamos a página de resposta ao mapa de páginas-alvo. Em experimentos preliminares, determinamos que U_i' deve conter pelo menos 30% das URLs de U_i.

4.3.2 Tratamento de Páginas de Erro

O tratamento das páginas de erro empregado em [Doorenbos et al., 1997] e [Barbosa and Freire, 2004], utiliza um conceito simples onde consideram como páginas de

erro aquelas que casam com os exemplos de páginas previamente fornecidas. Por exemplo páginas que contenham palavras como *"ERROR"*, *"Internal Server Error"*, *"400 Bad Request"*, ou as frases de erro especificadas no Protocolo HTTP. Diferentemente, em nossa abordagem, simplesmente assumimos que os 30% das páginas consideradas que têm o maior tamanho em bytes são sempre originadas de consultas válidas. Por este motivo, não temos tratamento específico para os casos em que são retornadas páginas de erro. Em casos excepcionais em que o formulário retorna para todas as consultas páginas de erro, o processamento continua seguindo os links destas páginas. Quando a lista de links extraída destas páginas está vazia e nenhuma página alvo foi atingida, então o formulário que gerou estas consultas é descartado.

4.4 Incorporando Formulários ao MPA

Após a submissão das consultas geradas e posterior análise das páginas obtidas como resposta, temos como resultado um conjunto de consultas que levam à páginas-alvo. De posse destas consultas, é observado que existem parâmetros específicos para cada tipo de campo de formulário em particular.

Como exemplo, nas consultas geradas:

- `http://www.meuformulariohtml.com.br/find.cgi?select=all&for=Harry`
- `http://www.meuformulariohtml.com.br/find.cgi?select=title&for=Potter`,

onde observa-se que para o campo `select` existem os valores usados `title` e `all` e para o campo `for` os valores usados são `Harry` e `Potter`.

Note que os valores para cada tipo `select` e para cada tipo `for` são diferentes, estes são campos que necessitam de parâmetros para o preenchimento do formulário. O trecho restante das consultas que continua inalterado é a estrutura básica da consulta.

Assim ao final do processo é possível definir quais são os parâmetros necessários para o preenchimento adequado do formulário para se obter páginas-alvo.

Como visto no Capítulo 3, onde foi descrita a geração do Mapa de Páginas-Alvo (MPA), nossa proposta consiste em caminhar pelo Web site a partir de um ponto de entrada à procura de páginas-alvo, e guardar cada caminho que efetivamente atinja uma página estruturalmente similar à página dada como exemplo.

Porém, se durante este caminhamento for encontrado algum formulário e seu devido tratamento for realizado como verificado na Seção 4.2, deve-se considerá-lo como a parte do MPA. Para tanto, é incorporada ao MPA uma URL genérica que codifica a estrutura básica da consulta e os parâmetros necessários para a sua execução, simulando o preenchimento do formulário. Um exemplo desta URL é mostrado abaixo:

```
http://www.meuformulariohtml.com.br/find.cgi?select=*&for=*
```

Esta URL será posteriormente incorporada ao padrão de navegação, e utilizada pelo coletor gerado. Assim é possível definir antes da execução deste coletor quai os parâmetros que devem ser usados para a execução da consulta. Essa tarefa é deixada para o usuário quando da execução deste coletor.

Capítulo 5

Experimentos

Neste capítulo apresentaremos os resultados dos experimentos realizados com a abordagem proposta neste trabalho.

5.1 Ambiente de Experimentação

Para a realização dos experimentos em sites estáticos utilizamos 11 Web sites reais, que possuem coleções de páginas ricas em dados caracterizadas por possuirem uma estrutura comum. Para os experimentos em sites dinâmicos foram utilizados 6 Web sites reais extraídos da coleção TEL-8[1] com diferentes domínios de aplicação, que também possuem páginas ricas em dados com uma estrutura comum. Estes sites foram utilizados em outros trabalhos realizados sobre a Web invisível tais como [Zhang et al., 2004] e [Davulcu et al., 1999]. A lista dos sites utilizados, com uma breve descrição e definição das páginas-alvo a serem alcançadas é mostrada na Tabela 5.1, onde Na coluna "E/D" as linhas com a letra "E" correspondem aos sites estáticos e as linhas com a letra "D" correspondem aos sites dinâmicos.

Os experimentos foram realizados em um computador rodando sistema operacional Linux da distribuição GENTOO Kernel 2.4, com 1Gb de memória RAM, processador de 2.8Ghz e disco rígido de 80Gb. O protótipo de nossa abordagem foi desenvolvida na Linguagem Java.

Nossos experimentos consistem em primeiro gerar o padrão de navegação, utilizando para isso, a página de exemplo e o ponto de entrada fornecida pelo usuário. Em seguida realizar a coleta orientada por estrutura utilizando o coletor gerado.

[1]Disponível em `http://metaquerier.cs.uiuc.edu/repository/datasets/tel-8/`

E/D	Site	Descrição	Páginas-Alvo
E	www.ejazz.com.br	Páginas de estilos de jazz, artistas, instrumentos, etc.	Artistas
E	informatik.uni-trier.de/∼ley/	Conferência VLDB secção do DBLP	Conferência VLDB
E	www.olympic.org	Comite olimpico internacional	Heróis olímpicos
E	www1.folha.uol.com.br/folha/turismo	Seção de viajens da "Folha de São Paulo"	Notícias
E	www1.folha.uol.com.br/folha/dinheiro	Seção de economia da "Folha de São Paulo"	Notícias
E	www.dot.kde.org	Seção de lançamentos do KDE Desktop Manager	Lançamento de pacotes
E	www.amazon.com	Amazon Essential CDs section	CD
E	www.wallstreetandtech.com	WallStreet and Technology News	Últimas notícias
E	www.cnn.com/weather	Clima em várias cidades do mundo	Clima
E	sports.yahoo.com	Seção de esportes do YAHOO	Times de futebol Europeu
E	www.nasa.gov/home	Site oficial da Nasa	Notícias
D	www.bestwebbuys.com	Comparação de produtos online como: livros, DVDs, Eletrônicos	Livros cujo campo título possua o termo "Harry Potter"
D	www.chapters.indigo.ca	Venda on-line de diversos produtos como: livros, DVDs, Eletrônicos	CDs cujo campo título possua o termo "Right Now"
D	www.dvdempire.com	Venda on-line de DVDs	DVDs cujo campo título possua o termo "Final Fantasy"
D	www.gracenote.com	Venda on-line de CDs	CDs cujo campo cantor possua o termo "Eminem Show"
D	www.talkingbooks.com	Venda on-line de livros	Livros cujo campo título possua o termo "Discover"
D	www.zevelekakis.gr	Livraria on-line especializada em livros médicos	Livros cujo campo título possua o termo "Heart Disease"

Tabela 5.1: Lista de web sites usados no experimentos.

5.2 Resultados Para Sites Estáticos

A Tabela 5.2 apresenta os resultados de nossos experimentos para os 11 sites estáticos utilizados.

Site	Página Alvo		Links Navegados	
	Manual	Automática	Geração	Coleta
E-jazz	149	149(100%)	2213	199
VLDB	30	30(100%)	70	32
OLYMPIC	335	328(98%)	395	379
Traveling	301	301(100%)	348	335
Money	470	468(99%)	550	528
KDE	30	30(100%)	120	31
CDs	416	398(96%)	440	426
WallStreet	261	253(97%)	1579	283
CNN	51	49(96%)	485	65
Yahoo Sports	38	37(97%)	1307	45
NASA	339	325(95%)	687	389

Tabela 5.2: Resultado dos experimentos com cada um dos coletores gerados.

Na Tabela 5.2, a coluna "Manual" mostra o número de páginas alvo encontradas em cada site. Este número foi obtido através da navegação manual nas áreas de cada site onde as páginas-

alvo são encontradas. A coluna "Automática" mostra o número de páginas automaticamente identificadas e alcançadas para a geração do coletor. Seu percentual com relação ao total de páginas é também mostrado nesta coluna. É importante observar que, em todos os casos, alcançamos 100% de precisão, pois nenhuma página que não fosse uma página-alvo foi recuperada. A coluna "Links Navegados" mostra o número de links percorridos, respectivamente, para gerar o coletor durante a fase de mapeamento e para recuperar as páginas alvo durante a coleta. Note que, em todos os casos, os números na coluna "Coleta" são menores que os números na coluna "Geração". Isto ocorre porque, durante a coleta, apenas links que casam com as expressões regulares no padrão de navegação são percorridos.

Executamos o coletor gerado sobre cada Web site da Tabela 5.1 duas outras vezes. Para alguns deles, foi detectado que novas páginas foram adicionadas após a geração do coletor. A Tabela 5.3 mostra o resultado destes dois processos adicionais de coleta sobre cada site.

Site	Coleta	Páginas recuperadas		Links	Novas Páginas
		Manual	Automática		
Viagem	1	314	308(98%)	335	7
	2	303	291(96%)	310	11
Dinheiro	1	486	478(98%)	497	84
	2	482	476(99%)	497	80
KDE	1	29	29(100%)	31	14
	2	29	29(100%)	34	19
CDs	1	409	394(96%)	492	4
	2	418	412(98%)	487	18
WallStreet	1	267	257(96%)	271	17
	2	272	267(98%)	273	25
NASA	1	334	320(95%)	339	12
	2	337	323(95%)	341	13

Tabela 5.3: Resultado da coleta após a adição de novas páginas alvo.

Nota-se que a coluna "Automática" se mantém similar àquela da Tabela 5.2. O mesmo pode ser dito com respeito ao número de links percorridos durante a coleta, como é apresentado na coluna "Links Navegados". A coluna "Novas Páginas" enumera as novas páginas-alvo encontradas na coleta. Isto ocorre com todas as novas páginas encontradas pelo coletor que possuam uma estrutura similar a estrutura da página dada como exemplo durante a geração do coletor. Note que, caso a estrutura do site seja alterada ou se tenha interesse de coletar um novo tipo de página, um novo coletor pode ser facilmente gerado.

5.3 Resultados Para Sites Dinâmicos

Diferente dos experimentos realizados para Web sites estáticos onde queríamos demonstrar a flexibilidade do padrão de navegação gerado, aqui queremos demonstrar a autonomia da abordagem no preenchimento automático de formulários, visto que as únicas informações necessárias para o bom funcionamento do método são a página exemplo e o ponto de entrada. A seguir discutiremos os procedimentos utilizados nos experimentos.

Site	**Páginas Coletadas Manualmente**	**Coleta Automática**	
		Treinamento	Coleta final
www.bestwebbuys.com	334	240	330(98%)
www.chapters.indigo.ca	13	140	13(100%)
www.dvdempire.com	14	80	14(100%)
www.gracenote.com	20	15	20(100%)
www.talkingbooks.com	23	80	23(100%)
www.zevelekakis.gr	282	15	280(99%)

Tabela 5.4: Resumo das coletas sobre sites dinâmicos.

Site	**Parâmetros**	**Valor**
www.bestwebbuys.com	searchfor	title
	title	Harry Potter
www.chapters.indigo.ca	section	music
	keyword	Rigth Now
www.dvdempire.com	search_type	title
	media	DVD
	search_String	Final Fantasy
www.gracenote.com	q_artist	Eminem Show
www.talkingbooks.com	search_by	title
	search_for	Discover
www.zevelekakis.gr	title	Heart Disease

Tabela 5.5: Parâmetros e valores utilizados nos sites dinâmicos.

Inicialmente coletamos as páginas-alvo dos Web sites dinâmicos preenchendo manualmente os campos de seus formulários utilizando para isso a especificação apresentada na coluna "Páginas-Alvo" desta mesma tabela. Estes resultados podem ser verificados na coluna "Páginas Coletadas Manualmente" da Tabela 5.4.

Os valores encontrados na coluna "Treinamento", ainda na Tabela 5.4, correspondem ao número de páginas coletadas oriundas das submissões automaticamente geradas a partir dos formulários encontrados para cada site, ou seja, correspondem ao número de consultas geradas.

Note-se que, para o caso particular destes 6 Web sites, as páginas de resposta dos formulários eram sempre páginas-alvo.

Os coletores gerados foram executados sobre os Web sites e, para cada coleta, passamos como valor dos parâmetros que nos foram solicitados pelo coletor gerado, os termos utilizados são mostrados na Tabela 5.5.

Após a realização da coleta, utilizando estes parâmetros, obtivemos os resultados observados na coluna "Coleta Final" da Tabela 5.4. Note que em todos os Web sites obtivemos 100% de precisão pois nenhuma página que não fosse página-alvo foi coletada.

Chamamos atenção para o fato de que o percentual na coluna "Coluna Final" da Tabela 5.4, corresponde ao percentual das páginas coletadas automaticamente em comparação com o número de páginas coletadas manualmente quando os mesmo argumentos de consulta (apresentados na Tabela 5.5) são usados.

Este percentual, pode ser interpretado como sendo o nível de revocação alcançado pelos coletores em sites dinâmicos, que são da mesma ordem daqueles alcançados nos sites estáticos (Tabela 5.3).

No entanto, esta métrica faz pouco sentido para sites dinâmicos, uma vez que o número de páginas à coletar não é absoluto, mas depende dos argumentos usados nas consultas ou no preenchimento dos formulários.

Capítulo 6

Conclusão e Trabalhos Futuros

Nesta dissertação, propusemos e avaliamos uma nova abordagem para o desenvolvimento de coletores especializados. Esta nova abordagem utiliza a estrutura das páginas Web em vez de seu conteúdo para determinar que páginas devem ou não ser coletadas. Nossos experimentos indicam que a nova abordagem orientada por conteúdo pode ser extremamente eficiente, resultando em alta precisão quando coleta páginas ricas em dados em Web sites de um domínio específico, incluindo nos casos em que as páginas-alvo são geradas dinamicamente através de formulários HTML e que compõem a chamada Web invisível (*Hidden Web*).

O método proposto tem a vantagem de denotar uma quantidade mínima de exemplos para identificar o conjunto de páginas que devem ser recuperadas. De fato, utilizamos apenas uma página exemplo em nossos experimentos, enquanto na abordagem de coletores orientados por conteúdo geralmente é necessário um conjunto maior de páginas de exemplo [Chakrabarti et al., 2002, Liu et al., 2004, Qin et al., 2004].

Uma outra características importante do nosso método, é o fato de ser capaz de lidar com sites onde as páginas-alvo são geradas dinamicamente através do preenchimento de formulários. Em nossa abordagem, não é necessário a existência de um banco de dados inicial para auxiliar no processo de preenchimento do formulário, como em [Chakrabarti et al., 2002], [Raghavan and Garcia-Molina, 2001] e [Lage et al., 2004], nem tão pouco é necessária grande interação com o usuário como em [Golgher et al., 2000b], [Anupam et al., 2000] e [Raghavan and Garcia-Molina, 2001]. Ao contrário, utilizamos termos frequentes disponíveis na páginas de exemplo e na página de formulário e construimos consultas amostrais para preencher automaticamente os formulários em busca de páginas-alvo.

A abordagem orientada por estrutura, que propomos neste trabalho, complementa a abordagem tradicional orientada por conteúdo, no sentido de que ela é mais adequada para Web sites que possuem uma grande concentração de páginas ricas em dados e, ao mesmo tempo, apresentam estrutura regular. Isso significa que o método proposto é a melhor opção para tarefas de coleta restrita a Web sites, para gerar entradas para métodos de extração de dados e outras aplicações que tiram proveito da regularidade estrutural das páginas. É importante observar que este tipo de site está se tornando cada vez mais popular na Web.

Como trabalho futuro temos a intenção de combinar as abordagens de coleta guiada por estrutura e guiada por conteúdo, criando uma estratégia híbrida que combine as vantagens dos dois métodos. A idéia é produzir métodos que sejam mais flexíveis que a abordagem guiada por estrutura apresentada neste trabalho, mantendo os altos níveis de precisão alcançados.

Os métodos aqui apresentados estão sendo atualmente incorporados em um ferramenta que permitirá o seu uso por usuários finais. Pretendemos então utilizar a ferramenta em diversas aplicações reais como forma de avaliar sua eficácia e validar os métodos aqui propostos. Para isso, algumas restrições práticas que não discutimos aqui, devem ser observadas, como é o caso do uso dos *COOKIES*, ou marcadores de estado, por diversos sites de conteúdo dinâmico.

Referências Bibliográficas

[Aggarwal et al., 2001] Aggarwal, C. C., Al-Garawi, F., and Yu, P. S. (2001). On the design of a learning crawler for topical resource discovery. *Transactions on Information Systems*, 19(3):286–309.

[Ahnizeret et al., 2004] Ahnizeret, K., Fernandes, D., Cavalcanti, J. M. B., de Moura, E. S., and da Silva, A. S. (2004). Information retrieval aware web site modelling and generation. In *Proceedings of the 23rd International Conference on Conceptual Modeling*, pages 402–419.

[Anupam et al., 2000] Anupam, V., Freire, J., Kumar, B., and Lieuwen, D. F. (2000). Automating web navigation with the WebVCR. *Computer Networks*, 33(1-6):503–517.

[Arasu et al., 2001] Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A., and Raghavan, S. (2001). Searching the Web. *Transactions on Internet Technology*, 1(1):2–43.

[Arasu and Garcia-Molina, 2003] Arasu, A. and Garcia-Molina, H. (2003). Extracting structured data from web pages. In *Proceedings of the 19th International Conference on Management of Data*, pages 337–348.

[Barbosa and Freire, 2004] Barbosa, L. and Freire, J. (2004). Siphoning hidden-web data through keyword-based interfaces. In *Anais do XIX Simpósio Brasileiro de Banco de Dados*, pages 309–321.

[Bergmark, 2002] Bergmark, D. (2002). Collection synthesis. In *Proceedings of the 2nd International Conference on Digital Libraries*, pages 253–262.

[Bergmark et al., 2002] Bergmark, D., Lagoze, C., and Sbityakov, A. (2002). Focused crawls, tunneling, and digital libraries. In *Proceedings of the 6th European Conference on Digital Libraries*, pages 91–106.

[Brin and Page, 1998] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117.

[Calado et al., 2002] Calado, P., da Silva, A. S., Ribeiro-Neto, B. A., Laender, A. H. F., Lage, J. P., de Castro Reis, D., Roberto, P. A., Vieira, M. V., Gonçalves, M. A., and Fox, E. A. (2002). Web-DL: an experience in building digital libraries from the web. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 675–677.

[Chakrabarti, 1999] Chakrabarti, S. (1999). Recents results in automatic web resource discovery. *ACM Computing Surveys*, 31(4):17.

[Chakrabarti et al., 2002] Chakrabarti, S., Punera, K., and Subramanyam, M. (2002). Accelerated focused crawling through online relevance feedback. In *Proceedings of the 11th International Conference on World Wide Web*, pages 148–159.

[Chakrabarti et al., 1999] Chakrabarti, S., van den Berg, M., and Dom, B. (1999). Focused crawling: A new approach to topic-specific web resource discovery. *Computer Networks*, 31(11-16):1623–1640.

[Chawavate, 1999] Chawavate, S. S. (1999). Comparing hierarchical data in external memory. In *Proceedings of the 25th International conference on Very Large Data Base*, pages 90–101.

[Chen, 1999] Chen, W. (1999). New algorithm for tree-to-tree correction problem. *Journal of Algorithms*, 29(1):135–158.

[Chen et al., 2004] Chen, X. H., Embley, D. W., and Liddle, S. W. (2004). Query rewriting for extracting data behind html forms. In *Proceedings of the 23th International Workshop on Conceptual Model-Directed Web Information Integration and Mining*, pages 335–348.

[Cooley, 2003] Cooley, R. (2003). The use of web structure and content to identify subjectively interesting web usage patterns. *Transactions on Internet Technology*, 3(2):93–116.

[Cormen et al., 2002] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2002). *Algoritmos Teoria e Prática*. Editora Campos.

[Crescenzi et al., 2001] Crescenzi, V., Mecca, G., and Merialdo, P. (2001). Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 12th International Conference on Very Large Data Bases*, pages 109–118.

[Crescenzi et al., 2004] Crescenzi, V., Merialdo, P., and Missier, P. (2004). Clustering web pages based on their structure. *Data an Knowledge Engineering*, 54(3):277–393.

[da Silva et al., 1999] da Silva, A. S., Veloso, E. A., Golgher, P. B., Ribeiro-Neto, B. A., Laender, A. H. F., and Ziviani, N. (1999). Cobweb - a crawler for the brazilian web. In *Proceedings of the 6th Symposium on String Processing and Information Retrieval*, pages 184–191.

[Davulcu et al., 1999] Davulcu, H., Freire, J., Kifer, M., and Ramakrishnan, I. V. (1999). A layered architecture for querying dynamic Web content. In *Proceedings of the 25th International Conference on Management of Data*, pages 491–502.

[de Castro Reis et al., 2002] de Castro Reis, D., Araújo, R. B., da Silva, A. S., and Ribeiro-Neto, B. A. (2002). A framework for generating attribute extractors for web data sources. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval*, pages 210–226.

[de Castro Reis et al., 2004] de Castro Reis, D., Golgher, P. B., da Silva, A. S., and Laender, A. H. F. (2004). Automatic web news extraction using tree edit distance. In *Proceedings of the 13th International conference on World Wide Web*, pages 502–511.

[de Moura, 2004] de Moura, J. J. B. (2004). Agrupamento de páginas web baseado em estrutura usando distância de edição em árvores. Master's thesis, Universidade Federal do Amazonas - UFAM.

[Diligenti et al., 2000] Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., and Gori, M. (2000). Focused crawling using context graphs. In *Proceedings of the 26th International Conference on Very Large Databases*, pages 527–534.

[Doorenbos et al., 1997] Doorenbos, R. B., Ezioti, O., and Weld, D. S. (1997). A scalable comparison-shopping agent for the world-wide web. In *Proceedings of the 1st International Conference on Autonomous Agents*, pages 39–48.

[Embley et al., 1999] Embley, D. W., Campbell, D. M., Jiang, Y. S., Liddle, S. W., Ng, Y., Quass, D., and Smith, R. D. (1999). Conceptual-model-based data extraction from multiple-record. *Data and Knowledge Engineering*, 31(3):227–251.

[Florescu et al., 1998] Florescu, D., Levy, A. Y., and Mendelzon, A. O. (1998). Database techniques for the world-wide web. *A survey. SIGMOD Record*, 27(3):59–74.

[Fontes and Silva, 2004] Fontes, A. C. and Silva, F. S. (2004). SmartCrawl: a new strategy for the exploration of the hidden web. In *Proceedings of the 6th International Workshop on Web Information and Data Management*, pages 9–15.

[Golgher et al., 2000a] Golgher, P. B., Laender, A. H. F., da Silva, A. S., and Ribeiro-Neto, B. (2000a). An example-based environment for wrapper generation. In *Proceedings of the 19th Symposium on String Processing and Information Retrieval*, pages 152–164.

[Golgher et al., 2000b] Golgher, P. B., Laender, A. H. F., da Silva, A. S., and Ribeiro-Neto, B. A. (2000b). ASByE: uma ferramenta baseada em exemplos para especificação de agentes para coleta de documentos web. In *Anais do XV Simpósio Brasileiro de Banco de Dados*, pages 217–231.

[Heydon and Najork, 1999] Heydon, A. and Najork, M. (1999). Mercator: A scalable, extensible web crawler. In *Proceedings of the 8th International conference on World Wide Web*, pages 219–229.

[Laender et al., 2002a] Laender, A. H. F., Ribeiro-Neto, B. A., and da Silva, A. S. (2002a). Debye: Data extraction by example. *Data Knowledge Engineering*, 40(2):121–154.

[Laender et al., 2002b] Laender, A. H. F., Ribeiro-Neto, B. A., da Silva, A. S., and Teixeira, J. S. (2002b). A brief survey of web data extraction tools. In *Proceedings of the 28th International Conference on Management of Data*, pages 84–93.

[Lage et al., 2004] Lage, J. P., da Silva, A. S., Golgher, P. B., and Laender, A. H. F. (2004). Automatic generation of agents for collecting hidden web pages for data extraction. *Data an Knowledge Engineering*, 49(2):177–196.

[Lawrence and Giles, 1998] Lawrence, S. and Giles, C. L. (1998). Searching the Word Wide Web. *Science*, 6(2):98–100.

[Liu et al., 2004] Liu, H., Milios, E. E., and Janssen, J. (2004). Probabilistic models for focused web crawling. In *Proceedings of the 6th International Workshop on Web Information and Data Management*, pages 16–22.

[McCallum et al., 1999] McCallum, A., Nigam, K., Rennie, J., and Seymore, K. (1999). A machine learning approach to building domain-specific search engines. In *Proceedings of the 16th Symposium on Intelligent Agents in Cyberspace*, pages 662–667.

[Najork and Wiener, 2001] Najork, M. and Wiener, J. L. (2001). Breadth-first crawling yields high-quality pages. In *Proceedings of the 10th International conference World Wide Web*, pages 114–118.

[Nierman and Jagadish, 2002] Nierman, A. and Jagadish, H. V. (2002). Evaluating structural similarity in xml documents. In *Proceedings of the 5th International Workshop on the Web and Database*, pages 61–66.

[Pinkerton, 1994] Pinkerton, B. (1994). Finding what people want: Experiences with the WebCrawler. In *Proceedings of the 1st International conference World Wide Web*.

[Qin et al., 2004] Qin, J., Zhou, Y., and Chau, M. (2004). Building domain-specific web collections for scientific digital libraries: a meta-search enhanced focused crawling method. In *Proceedings of the 4th Conference on Digital Libraries*, pages 135–141.

[Raghavan and Garcia-Molina, 2001] Raghavan, S. and Garcia-Molina, H. (2001). Crawling the hidden web. In *Proceedings of the 27th International conference on Very Large Data Base*, pages 129–138.

[Rennie and McCallum, 1999] Rennie, J. and McCallum, A. (1999). Using reinforcement learning to spider the web efficiently. In *Proceedings of the 16th International Conference on Machine Learning*, pages 335–343.

[Selkow, 1977] Selkow, S. M. (1977). The tree-to-tree editing problem. *Information Processing Letters*, 6(6):184–186.

[Tai, 1979] Tai, K. C. (1979). The tree-to-tree correction problem. *Journal of the Association for Computing Machinery*, 26(3):422–433.

[Tanaka and Tanaka, 1988] Tanaka, E. and Tanaka, K. (1988). An interactive www search engine for user-defined collections. In *Proceedings of the 1st International Pattern Recognition and Artificial Intelligence*, pages 221–240.

[Valiente, 2001] Valiente, G. (2001). An efficient bottom-up distance between trees. In *Proceedings of the 8th International Symposium on String Processing and Information Retrival*, pages 212–219.

[Wang et al., 1998] Wang, J. T. L., Shapiro, B. A., D.Shasha, Zhang, K., and Currey, K. M. (1998). An algorithm for find the largest approximately common substructure of two trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):889–895.

[Wang and Zhang., 2001] Wang, J. T. L. and Zhang., K. (2001). Finding similar consensus between trees: an algorithm and distance hierarchy. *Pattern Recognition*, 34(1):127–137.

[W.Yang, 1991] W.Yang (1991). Identifying syntatic differences between two programs. *Software - Practice and Experience*, pages 739–755.

[Zhai and Liu, 2005] Zhai, Y. and Liu, B. (2005). Web data extraction based on partial tree alignment. In *Proceedings of the 14th International conference on World Wide Web*, pages 76–85.

[Zhang et al., 1992] Zhang, K., Statman, R., and Shasha, D. (1992). On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(1):133–139.

[Zhang et al., 2004] Zhang, Z., He, B., and Chang, K. C.-C. (2004). Understanding web query interfaces: Best-effort parsing with hidden syntax. In *Proceedings of the 30th International Conference on Management of Data*, pages 107–118.