Davi de Castro Reis

Distância de Edição em Árvores Aplicada à Extração de Dados da Web

Belo Horizonte
19 de abril de 2005

Davi de Castro Reis

Distância de Edição em Árvores Aplicada à Extração de Dados da Web

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte

19 de abril de 2005

Resumo

A World Wide Web é hoje um dos maiores repositórios de informação existentes, com bilhões de páginas, tratando dos mais variados tópicos, ao alcance de pessoas das mais diferentes nacionalidades. Esse conteúdo, porém, é formatado para o consumo humano, e agentes computacionais têm grande dificuldade em acessar e manipular os dados contidos nas páginas da Web.

Uma das opções para contornar esse problema é escrever manualmente extratores para todas as páginas da Web das quais se deseja extrair dados, de modo a torná-las adequadas para o consumo por agentes computacionais. Porém, mesmo com o advento de novas ferramentas para geração semi-automática desses extratores, ainda assim não é possível fazer a extração de dados de um grande volume de páginas da Web pois, dada a necessidade de intervenção humana, essas ferramentas têm escalabilidade limitada.

Esta dissertação apresenta uma nova estratégia para construção de sistemas de extração da dados da Web. Os sistemas criados a partir da estratégia proposta são completamente automáticos e podem ser usados para extração de dados de grandes quantidades de páginas. Em nossos experimentos, realizamos a extração, de forma completamente automática, das notícias de 35 dos principais veículos de comunicação brasileiros, totalizando 4088 páginas, e atingimos um grau de precisão de 87,71%.

A chave para obtenção desse resultado é o uso da técnica de distância de edição em árvores. Uma vez que páginas da Web são árvores serializadas, pode-se usar essa técnica para obter as variações entre as páginas e então extrair os dados contidos nessas páginas. Além de uma revisão extensa da literatura relacionada ao problema de distância de edição em árvores, esta dissertação apresenta um novo algoritmo para o problema. O algoritmo, denominado Restricted Top-Down Mapping, ou simplesmente RTDM, é descrito em detalhes, incluindo pseudo-código, estudo dos limites assintóticos e análise empírica, o que nos levou à conclusão que o algoritmo supera todos os demais algoritmos existentes na literatura com aplicação ao problema de extração de dados da Web.

Abstract

The World Wide Web is the largest information repository nowadays, with billions of pages, dealing with several topics, avaible to people of different nationalities. The content of these pages, however, is formatted for human comsuption, and computer agents have a lot of difficulties to access and manipulate the data in these web pages.

One of the options to circumnvent this problem is to write, manually, extractors for all web pages one is interested in, and, therefore, make them suitable for computer agents. Recently, new semi-automatic extractor generation tools have been developed, but, even with these tools, it is still not possible to extract data from a large collection of web pages, due to the need of human intervention.

This dissertation presents a new strategy for the construction of web data extraction systems. The systems created using the proposed strategy are completely automatic and can be used for large extractions tasks. In our experiments, we extracted in a completely automatic fashion, the news found in the pages of 35 of the main Brazilian media vehicles present on the Web, summing up a total of 4088 pages, with correctness precision of 87.71%.

The key to achive this result is the use of the tree edit distance technique. Given that web pages are serialized trees, we can use this technique to find the differences between the trees and, then, extract the data from the pages. Besides an extensive revision of the tree edit distance problem, this dissertation presents a new algorithm for the problem. The algorithm, named Restricted Top-Down Mapping, or simply RTDM, is described in details, including pseudo-code, assyntotical limits and empirical analysis, which led to the conclusion that this algorithm surpasses all other algorithms, with applications to web data extraction, available in the literature.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A World Wide Web é hoje um dos maiores repositórios de informação existentes, com bilhões de páginas, tratando dos mais variados tópicos, ao alcance de pessoas das mais diferentes nacionalidades. Esse conteúdo, porém, é formatado para o consumo humano, e agentes computacionais têm grande dificuldade em acessar e manipular os dados contidos nas páginas da Web.

A comunidade científica tem investido grandes esforços para superar esse obstáculo e conseguir prover serviços mais complexos sobre a estrutura da Web. Grande parte desses esforços estão dentro da proposta da Web Semântica [4], uma nova organização da Web, onde tanto seres humanos quanto agentes computacionais serão capazes de analisar e manipular os dados disponíveis na Web. A Web Semântica, porém, é ainda um trabalho incipiente e dificilmente teremos ampla adoção de suas tecnologias nos próximos anos. Por sua vez, a demanda pelo acesso aos dados da Web por agentes computacionais é imediata.

A principal dificuldade para o consumo dos dados da Web por agentes computacionais é a falta de estrutura dos dados nas páginas da Web. Na verdade, dada a atual organização da Web, pode-se dizer que a maior parte dos dados possui estrutura, já que as suas páginas são geradas automaticamente, a partir de bancos de dados. Essa estrutura, porém, quando os dados são embutidos em uma página Web, torna-se implícita. Para que se possa, portanto, acessar e manipular esses dados é preciso ser capaz de inferir essa estrutura e extrair os dados contidos nas páginas Web, para apresentá-los de forma conveniente aos agentes computacionais.

O problema de extração de dados da Web é o principal tópico desta dissertação, e existem vários trabalhos na literatura, se valendo de diferentes abordagens, para a realização dessa tarefa. Não obstante, a comunidade científica ainda não foi capaz de obter uma solução geral e satisfatória para o problema. Como páginas da Web são essencialmente árvores serializadas, a *distância de edição em árvores* apresenta-se como uma alternativa extremamente promissora para a tarefa de extração de dados da Web. Nesta dissertação apresenta-se uma nova abordagem para extração de dados da Web, a primeira a utilizar distância de edição em árvores, até onde vai o conhecimento do autor. Os resultados obtidos comprovam a eficácia da abordagem adotada. Posteriormente, trabalhos independentes produziram resultados [21, 52] que corroboram ainda mais o acerto da estratégia utilizada.

1.1 Definição do Problema

O problema de extração de dados da Web consiste em, dado um conjunto de páginas Web, extrair os dados nelas contidos, e, possivelmente, a estrutura desses dados e a informação semântica associada aos mesmos. A motivação por trás do problema é a grande dificuldade de manipulação que esses dados oferecem a agentes computacionais, uma vez que páginas Web são construídas para consumo humano.

A existência de extratores possibilita uma visão da Web mais próxima à visão tradicional oferecida por bancos de dados, e a conveniência dessa visão já foi previamente mostrada [18]. A tarefa de um extrator é, minimamente, separar os *dados* em uma página Web das informações de apresentação, tais como formatação e elementos estilísticos. Além disso, é, geralmente, também oportuno que o extrator forneça a *estrutura* dos dados e informação *semântica* associada a eles.

Tomando o modelo relacional [17] como base para uma analogia, podemos dizer que as células de uma tabela constituem os dados, o esquema da tabela descreve a estrutura desses dados, e, finalmente, os rótulos das colunas provêem a semântica. Ironicamente, a maior parte dos dados disponíveis na Web já está armazenada em bancos relacionais. No entanto, quando esses dados são apresentados como páginas Web, as informações de estrutura e semântica se perdem, ou se tornam implícitas. Até os próprios dados tornam-se inacessíveis a agentes computacionais, pois os mesmos são codificados juntamente com a informação de apresentação.

A Figura 1.1 ilustra o problema de extração de dados da Web, explicitando cada um dos três sub-problemas, que são: (1) a extração dos dados propriamente ditos, (2) a extração da estrutura, e, finalmente, (3) a extração da informação semântica. Como podemos ver, só através de um intermediador, ao qual chamamos extrator, faz-se possível, a agentes computacionais, o acesso aos dados da Web.

O foco desta dissertação está voltado para o sub-problema (1). Porém, a solução apresentada para o sub-problema (1) envolve, ainda que implicitamente, a solução para o sub-problema (2). Ou seja, obtém-se como produto colateral da extração dos dados, uma aproximação da estrutura que os descreve. Para ambos sub-problemas, recorre-se à técnica de distância de edição em árvores. Como o agente computacional desenvolvido para comprovação da eficácia da estratégia aqui apresentada é um sistema agregador de notícias, fez-se também necessário obter alguma informação semântica associada aos dados, o que corresponde ao sub-problema (3). O Capítulo 4 descreve em detalhes como tratou-se cada um dos sub-problemas descritos.

Na próxima seção trataremos de alguns dos trabalhos sobre extração de dados da Web existentes na literatura.

1.2 Trabalhos Relacionados

Antes do advento das técnicas de distância de edição em árvores, as abordagens para extração de dados da Web eram, em sua maioria, basedas em aprendizagem supervisionada [30]. Em geral, o usuário precisava especificar um conjunto inicial de exemplos e, potencialmente, outras informações. A partir disso, o sistema gerava um extrator para aqueles exemplos e outros objetos semelhantes da

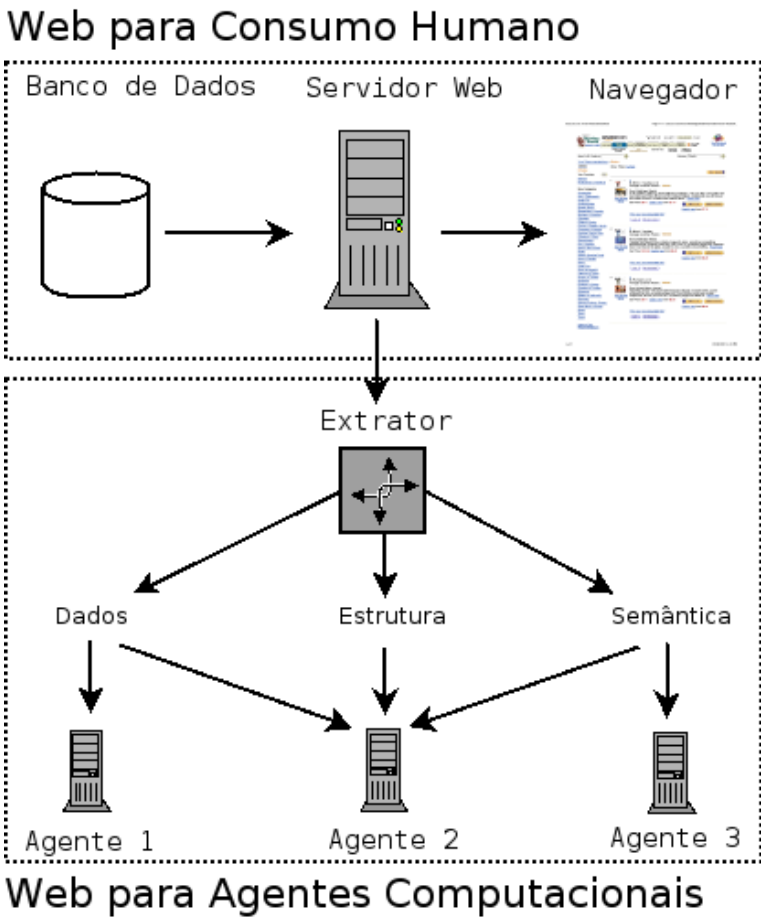

Figura 1.1: O papel dos extratores.

página. Cada abordagem definia uma linguagem própria para descrição de seus extratores. Uma breve revisão desses trabalhos pode ser encontrada em [30].

Kushmerick et al. [28] foram os primeiros a tentar formalizar a tarefa de geração de extratores. Desde então, outros trabalhos têm apresentado sistemas específicos para a tarefa de extração de dados da Web. Em [10], Cohen e Jensen propuseram uma arquitetura extensível para criação de sistemas de geração de extratores.

O sistema SoftMealy [22], por Hsu et al., usa *Tradutores de Estado Finito (TEF)*[1] para construção de extratores. Os tradutores de estado finito são máquinas de estado onde as arestas representam o que Hsu et al. definem como regras de contexto, que são pequenas regras capazes de identificar trechos de dados com algumas características. Para criação dessas regras, o sistema SoftMealy recorre a técnicas de cobertura de conjuntos (*set-covering*), aplicadas sobre o conjunto de exemplos fornecidos.

Muslea et al. mostraram o sistema Stalker em [33]. O sistema usa o conceito de *autômatos landscape* para descrição de seus extratores. Os extratores são criados a partir de um conjunto pré-determinado de heurísticas. Pequenos extratores são gerados e incrementalmente compostos para obtenção do extrator final.

[1] Finite-State Transducers ou FST no trabalho original.

Já em 2001, quando o estudo da extração de dados da Web ainda estava em sua infância, o trabalho de Baumgartner, Flesca e Gottlob [3], se tornou um produto comercial, de nome Lixto. Essa ferramenta tem uma interface gráfica e permite que usuários com alto grau de especialização criem extensões para a estratégia de geração de extratores, nos casos onde apenas as heurísticas nativas são incapazes de obter sucesso.

A ferramenta DEByE [14, 29] também conta com uma interface gráfica e, baseada em exemplos e no conceito de tabelas aninhadas para descrição da estrutura dos dados, é capaz de construir um extrator, utilizando uma estratégia pré-determinada. Enquanto a maioria das abordagens existentes na época tentava construir os objetos complexos tentando primeiro determinar a extração do objeto inteiro e depois a extração de suas propriedades, a ferramenta DEByE extraía primeiro as propriedades e depois utilizava uma estratégia para construir os objetos complexos usando o resultado dessa extração e a informação de estrutura provida pelo usuário.

Em um trabalho que antecede esta dissertação, antes de se adotar a distância de edição em árvores como principal pilar para solução do problema de extração de dados Web, construiu-se um sistema baseado nas idéias da ferramenta DEByE, porém com um conjunto flexível de heurísticas para construção de extratores [37]. Esse trabalho, embora limitado à extração dos dados, sem a pretensão de explicitar a sua estrutura, foi aplicado com sucesso em algumas tarefas reais [6].

Em [19], Grieser et al. mostraram uma formalização para o problema de geração de extratores baseada em exemplos. O trabalho apresenta o problema sob uma visão de aprendizagem de linguagem, o que resultou em um formalismo capaz de descrever a maior parte dos trabalhos existentes à época.

Crescenzi et al. foram os primeiros a utilizar a diferença entre o código HTML de diferentes páginas como principal evidência para extração de dados da Web [13]. Seu trabalho, porém, não interpreta o código HTML como uma árvore, mas como uma estrutura plana. Por outro lado, a idéia de alinhar diferentes páginas para encontrar os dados relevantes, apresentada por Crescenzi et al., compõe, juntamente com a técnica de distância de edição em árvores, o cerne da estratégia de geração de extratores apresentada nesta dissertação. A ferramenta desenvolvida por Crescenzi et al., chamada RoadRunner, se destacou por ser o primeiro sistema completamente automático para o problema de extração de dados da Web. Recentemente essa ferramenta foi publicada como *software livre*.

Embora, até a publicação do primeiro trabalho proveniente desta dissertação, não houvesse, na literatura, nenhum trabalho usando distância de edição em árvores para extração de dados Web, recentemente surgiram novos trabalhos [21, 52] que fazem uso dessa idéia e são descritos a seguir.

O sistema Thresher foi apresentado por Hogue e Karger em [21]. O objetivo do sistema é prover a usuários finais de navegadores a capacidade de construir extratores para objetos em páginas Web e, através da visão estruturada obtida a partir dos extratores, acrescentar *informação semântica* a esses objetos. O sistema Thresher está inserido em uma plataforma maior, chamada *Haystack* [36], que tem como objetivo permitir que usuários finais construam aplicações para a Web Semântica [4]. Ao contrário dos extratores tradicionais, um extrator gerado pelo sistema Thresher não tem como saída apenas os dados contidas no objeto extraído, mas também a marcação semântica do objeto, definida pelo padrão RDF. Essa marcação permite à plataforma Haystack lidar com os dados de forma poderosa, encontrando relações semânticas entre os objetos e possibilitando uma série de ações por parte

do usuário sobre esses dados. Se o objeto sendo extraído representa, por exemplo, as informações de um professor em uma universidade, é possível disparar ações como agendar um encontro com o professor, enviar uma mensagem de correio eletrônico, ou verificar a lista de publicações do mesmo, caso essa informação esteja disponível dentro do sistema. Hogue e Karger geram seus extratores aplicando a distância de edição em árvores para obter uma generalização dos exemplos fornecidos pelo usuário. O algoritmo usado é a formulação sem restrições da distância de edição em árvores [41], uma opção inferior, no que tange o custo computacional, ao algoritmo apresentado nesta dissertação.

Outro trabalho recente que faz uso de distância de edição em árvores para extração de dados da Web foi apresentado por Zhai, Liu e Grossman. A principal contribuição desse trabalho é o algoritmo MDR, inicialmente apresentado em [31], e posteriormente expandido com a técnica de distância de edição em árvores, como visto em [52]. O algoritmo MDR compartilha vários dos pressupostos existentes nesta dissertação. Existe, porém uma diferença central entre as duas abordagens. Enquanto os resultados aqui apresentados são voltados para páginas com um único objeto (ex., uma notícia), o algoritmo MDR foi construído para lidar exclusivamente com páginas contendo múltiplos objetos (ex., uma lista de livros), e portanto não pode ser usado para o problema específico de extração de notícias, abordado nesta dissertação. São três as principais idéias nos trabalhos de Zhai, Liu e Grossman. A primeira delas é a construção das árvores usando informação visual, seguida pelo uso de alinhamento parcial de árvores para a extração dos dados e, finalmente, a busca pelos registros de dados. As duas primeiras idéias são bastante gerais e podem ser usadas futuramente para melhorar o sistema de extração de notícias apresentado nesta dissertação, como veremos no Capítulo 5.

1.3 Principais Contribuições

A primeira contribuição desta dissertação é uma revisão ampla dos algoritmos de distância de edição em árvores existentes na literatura, classificados de acordo com o conjunto de restrições de cada um deles e acompanhados pelos respectivos limites assintóticos.

Uma peculiaridade das páginas Web, que é o fato dos dados relevantes se encontrarem nas folhas ou em subárvores que partilham todas as folhas com a árvores das páginas, nos levou à construção de um novo algoritmo, especialmente apropriado para extração de dados da Web, que denominamos Restricted Top-Down Mapping, ou simplesmente RTDM [37]. O algoritmo se restringe à formulação restrita da distância de edição em árvores, e tem amplas aplicações práticas e desempenho superior a todos os algoritmos com aplicação à extração de dados conhecidos até então. O algoritmo é apresentado em detalhes, incluindo pseudo-código, análise de complexidade e avaliação experimental.

Finalmente, mostramos uma nova estratégia, utilizando o algoritmo RTDM, para construção de sistemas de extração de dados da Web e, através de uma aplicação ao problema específico de extração de notícias de jornais e revistas em formato eletrônico, comprovamos a eficácia dessa estratégia. Nos experimentos que realizamos, envolvendo um total de 4088 páginas de 35 dos principais veículos de comunicação brasileiros, atingimos um grau de precisão de 87.71% na extração automática de notícias.

1.4 Organização da Dissertação

No Capítulo 2 formalizamos o conceito de distância de edição em árvores e apresentamos os principais algoritmos existentes na literatura. Em seguida, no Capítulo 3, descrevemos o algoritmo RTDM, base do trabalho de dissertação ora apresentado, juntamente com considerações teóricas e práticas de suas características. No Capítulo 4 discutimos como podemos usar a distância de edição em árvores, e, mais especificamente, o algoritmo RTDM, para construção de sistemas de extração de dados da Web. Ao longo desta discussão mostramos a experiência prática da construção de um sistema agregador de notícias. Finalmente, as conclusões provenientes deste trabalho podem ser encontradas no Capítulo 5.

Capítulo 2

Distância de Edição em Árvores

Neste capítulo, apresentamos uma ampla revisão do problema de distância de edição em árvores e dos trabalhos existentes na literatura. Primeiramente, definimos os conceitos necessários à discussão do tópico. Em seguida, descrevemos os diversos algoritmos existentes na literatura dentro de uma categorização baseada nas diferentes formulações do problema.

2.1 Conceitos

Um dos conceitos aplicados amplamente em representações planas de dados é o conceito de *distância de edição*. A distância de edição é a medida da diferença entre dois conjuntos de dados representados como *cadeias*. Em seu entendimento mais simples, a distância de edição é vista como a distância entre duas cadeias de caracteres, para alguma definição de distância. Em uma compreensão mais geral, podemos interpretar um caracter como qualquer *átomo informacional*, como um nome próprio, um evento ou um gene. Uma vez definido o componente de uma cadeia, é preciso definir formalmente o que é uma cadeia.

Definição 2.1. *Uma **cadeia** S é um conjunto ordenado de elementos, onde para cada elemento e temos:*

- *e é uma cadeia.*
- *Para uma cadeia e_1e_2, e_1 é antecessor de e_2.*
- *Para uma cadeia e_1e_2, e_2 é sucessor de e_1.*
- *$e \circ S$[1] é a cadeia eS.*

Da Definição 2.1 vemos que a única informação que a cadeia acrescenta aos seus elementos (átomos informacionais) é a noção de ordem. A distância de edição é uma medida do esforço de se transformar uma cadeia S em uma cadeia S', levados em consideração seus elementos e a ordem entre eles. Além da distância de edição, podemos também definir o **mapeamento** entre duas cadeias S e S', que é uma representação do conjunto de operações necessárias para transformar S em S'. A Figura 2.1 mostra um exemplo de mapeamento entre duas cadeias com a distância de edição associada.

[1] *A notação $\circ$ é comumente utilizada para representar um operador de composição.*

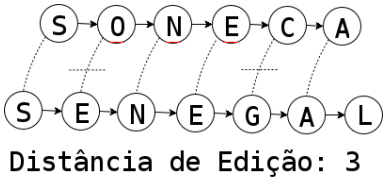

Figura 2.1: Mapeamento entre cadeias

A distância de edição entre cadeias é uma técnica bem estabelecida [46] com amplas aplicações. Dentre elas podemos citar agrupamento de consultas em mecanismos de busca [48], alinhamento de estruturas de DNA, e reconhecimento de fala, dentre outros [27].

Árvores são estruturas de dados mais ricas que cadeias e são definidas como digrafos simples e acíclicos [44]. Embora essa definição seja precisa, neste trabalho estamos interessados, principalmente, em um conjunto mais restrito de árvores, as chamadas *árvores ordenadas com raiz*. Árvores com raiz são aquelas nas quais um vértice é definido como raiz da árvore. Árvores ordenadas são aquelas cujo conjunto de filhos de um vértice qualquer forma uma cadeia. A partir desse momento iremos nos referir a uma árvore ordenada com raiz simplesmente como *árvore*, exceto quando for explicitamente dito o contrário.

Uma vez que toda cadeia pode ser considerada uma árvore de altura 2 (com a adição de uma raiz virtual), a distância de edição em árvores é a generalização do conceito de distância de edição em cadeias [34, 46]. Assim como ocorre com as cadeias, a distância de edição em árvores é a medida do esforço de se transformar uma árvore T em uma árvore T', levados em consideração os vértices que compõem a árvore e as arestas que os conectam. Porém, a árvore estabelece um conjunto significativamente maior de relações, entre seus elementos, do que uma cadeia, e, portanto, a definição de distância de edição é proporcionalmente mais complexa.

Assim como é possível definir a distância de edição entre cadeias, podemos também definir o conceito de distância de edição para árvores. Intuitivamente, a distância de edição entre duas árvores T e T' é o custo associado ao o conjunto mínimo de operações necessárias para transformar T em árvore T'.

Antes, porém, é necessário definir o conceito de árvore e estabelecer uma terminologia a ser adotada no restante do texto.

Definição 2.2. *Um digrafo conectado $G = (V, E)$ é uma árvore se o grafo não-direcionado subjacente é acíclico e existe um vértice distinto, chamado **raiz** da árvore, para o qual, para todos os vértices $v \in V$, existe um caminho em G da raiz r para o vértice v. Além disso:*

- *Para todo vértice $z \in V$ e $w \in V$, tal que a aresta $(z, w) \in E$, diz-se que w é **filho** de z e z é **pai** de w.*
- *Para todo vértice $z \in V$ e $w \in V$, tal que a aresta $(z, w) \in E$, se existe um caminho em G entre z e w, diz-se que z é **ancestral** de w e w é **descendente** de z.*
- *Para todo vértice a e b, tal que a aresta $(v, a) \in E$ e $(v, b) \in E$ diz-se que a e b são vértices **irmãos**.*

Em uma árvore ordenada, a posição relativa de vértices irmãos determina a ordem desses vértices. Dados dois vértices irmãos w e z, define-se que $w < z$ se e somente se w antecede z na ordem relativa definida. Uma definição intuitiva consiste em dizer que uma árvore ordenada é uma árvore embutida em um plano. A seguir apresentamos a definição da terminologia associada a árvores ordenadas.

Definição 2.3. *Seja $T = (V, E)$ uma árvore ordenada, e sejam as arestas $(v, w), (v, z) \subset E$, e dada uma operação de ordem relativa $w < z$ definida para vértices irmãos em T, diz-se que:*

- *O vértice $w \in V$ é o **primeiro filho** do vértice $v \in V$, denotado primeiro$[v]$, se não existe um vértice $x \in V$ tal que $(v, x) \in E$ e $x < w$.*
- *O vértice $z \in V$ é o **último filho de** $v \in V$, denotado último$[v]$, se não existe um vértice $x \in V$ tal que $(v, x) \in E$ e $z < x$.*
- *O vértice $z \in V$ é o **sucessor** de w se $\{(v, w), (v, z)\} \subset E$ e $w < z$.*
- *O vértice $w \in V$ é o **antecessor** de z se $\{(v, w), (v, z)\} \subset E$ e $w < z$.*

Também definiremos d como o grau máximo de uma árvore, l como o número de folhas de uma árvore e h como a profundidade ou altura de uma árvore. Usaremos i para nos referir ao indíce do i-ésimo vértice de uma árvore em um caminhamento pré-ordem [44], sendo esse vértice denominado por $t[i]$, e $T[i]$ para designar a sub-árvore com raiz em $t[i]$. Finalmente, $|T|$, ou n onde o contexto for claro, designa o número de vértices em uma árvore. A Figura 2.2 ilustra uma árvore ordenada e algumas de suas propriedades. Nas ilustrações restantes deste documento omitimos a direção das arestas das árvores, uma vez que sempre estará claro qual é raiz da árvore.

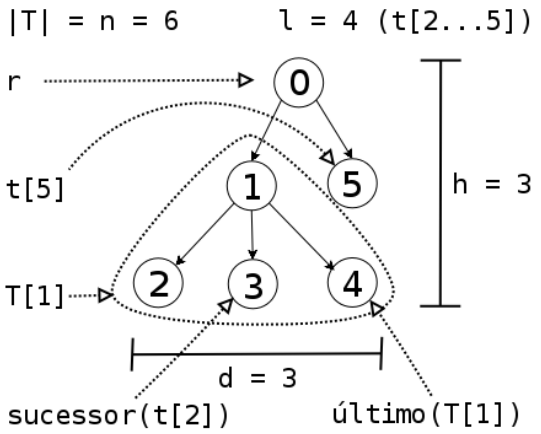

Figura 2.2: Uma árvore ordenada

Uma vez que introduzimos formalmente o conceito de árvore e a terminologia associada, vamos nos aprofundar no conceito de distância de edição em árvores. Em sua formulação mais comum, a distância de edição em árvores leva em consideração três operações: (a) remoção de um vértice, (b) inserção de um vértice e (c) substituição de um vértice. A cada uma dessas operações, existe um custo associado. A solução para o problema de distância de edição em árvores consiste em determinar o custo associado ao conjunto de operações de menor custo total capaz de transformar

uma árvore T em uma árvore T'. Embora o problema de distância de edição diga respeito apenas ao custo desse conjunto de operações, na maior parte das vezes as próprias operações são, também, de interesse. Tendo isso em vista, uma forma, possivelmente mais intuitiva, é apresentar o problema como a descoberta de um mapeamento de custo mínimo entre duas árvores. De fato, como mostrado em [56], os problemas são equivalentes. O conceito de mapeamento é formalmente definido como se segue.

Definição 2.4. *Um **mapeamento** entre uma árvore T e uma árvore T', $0 \leq i < |T|, 0 \leq j < |T'|$, é um conjunto M de pares ordenados (i, j), satisfazendo as seguintes condições para todo $(i_1, j_1), (i_2, j_2) \in M$:*

1. $i_1 = i_2$ *sse* $j_1 = j_2$.
2. $T[i_1] < T[i_2]$ *sse* $T'[j_1] < T'[j_2]$.
3. $T[i_1]$ *é um ancestral de* $T[i_2]$ *sse* $T'[j_1]$ *é um ancestral de* $T'[j_2]$.

Na Definição 2.4, a primeira condição estabelece que cada vértice pode aparecer, no máximo, uma vez em um mapeamento. Note que a segunda condição garante a preservação da ordem entre vértices irmãos e a terceira condição assegura a manutenção da relação hierárquica entre vértices nas árvores. É importante ressaltar que a segunda condição não se aplica a árvores não ordenadas. A Figura 2.3 ilustra um mapeamento entre duas árvores, enquanto a Figura 2.4 nos mostra um mapeamento inválido.

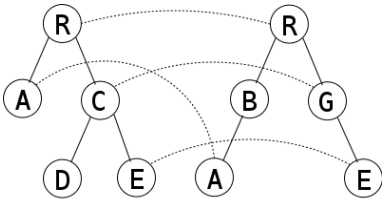

Figura 2.3: Mapeamento

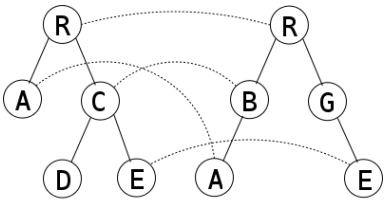

Figura 2.4: Mapeamento inválido

Podemos pensar em um mapeamento como a descrição do modo como uma seqüência de operações transforma uma árvore em outra, ignorando a ordem em que essas operações ocorrem. Na Figura 2.3, uma linha tracejada de um vértice de T_1 para um vértice de T_2 indica que o vértice de T_1 deve ser substítuido caso os vértices conectados sejam diferentes. Os vértices de T_1 que não são tocados por linhas tracejadas devem ser removidos e os vértices de T_2 não tocados devem ser inseridos. O conceito de igualdade ou diferença entre vértices, utilizado acima é definido de acordo com a aplicação, seguindo as semânticas usuais de igualdade[2].

A partir desse momento, iremos assumir que os vértices são rotulados com base em um alfabeto Σ e que a operação de igualdade definida para os vértices é a operação de igualdade entre os rótulos. Esse pressuposto não invalida a discussão, pois uma operação análoga poderia ser definida para tipos arbitrários de átomos informacionais associados aos vértices.

[2]Os algoritmos de distância de edição presumem que operações de igualdade definidas são reflexivas e transitivas.

Outra possível representação para um mapeamento é uma *seqüência de edição*. A seqüência de edição é uma ordenação do conjunto de operações necessárias para transformar uma árvore em outra. Embora qualquer ordenação possa caracterizar uma seqüência de edição, neste trabalho iremos adotar a definição a seguir.

Definição 2.5. *Uma* ***seqüência de edição*** $\mathcal{E}$ *para duas árvores* T *e* T' *é uma seqüência de pares* $(i \cup \lambda, j \cup \lambda) \setminus (\lambda, \lambda)$[3,4] *tal que a cada par* (i, j)*, é associado um custo* γ *e uma das seguintes operações:*

1. *remoção se* $j = \lambda$.
2. *inserção se* $i = \lambda$.
3. *substituição se* $t[i] \neq t'[j]$.

A Figura 2.5 nos mostra a seqüência de edição para duas árvores, com o mapeamento e os custos das operações associadas, para o modelo de custo unitário.

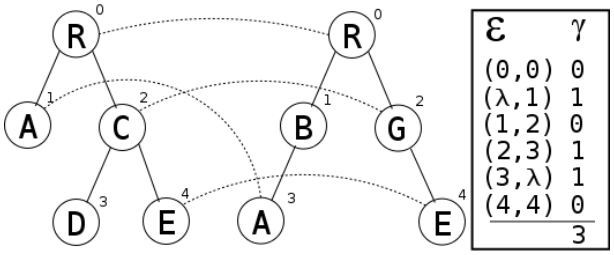

Figura 2.5: Um mapeamento e a seqüência de edição associada.

Embora as representações para o conjunto de operações necessárias para transformar duas árvores sejam equivalentes [54], cada uma delas possui pontos fortes e fracos em relação ao espaço ocupado e legibilidade. Em [8], são discutidas as representações acima, e uma nova representação, mais intuitiva e compacta, chamada *assinaturas representativas*, é proposta.

Encontrar a distância de edição entre duas árvores equivale a encontrar um mapeamento de custo mínimo entre duas árvores. Seja M um mapeamento entre as árvores T_1 e T_2, seja S o subconjunto de pares $(i, j) \in M$, seja D o conjunto de vértices em T_1 que não ocorre em nenhum $(i, j) \in M$ e seja I o conjunto de vértices em T_2 que não ocorre em nenhum $(i, j) \in M$. O custo associado ao mapeamento M é dado por $\gamma(M) = p|S| + q|I| + r|D|$, onde p, q e r são os custos associados às operações de substiuição, inserção e remoção respectivamente. É comum associar um custo unitário a cada uma dessas operações. Aplicações específicas, porém, podem exigir custos diferentes para cada tipo de operação, ou mesmo para cada operação. Mesmo o conjunto de operações pode ser modificado de acordo com cada aplicação. Para quaisquer duas árvores existem diversos mapeamentos possíveis. Um mapeamento M_{min} entre duas árvores T_1 e T_2 é dito de custo mínimo, se para todo mapeamento M entre T_1 e T_2, $\gamma(M_{min}) \leq \gamma(M)$. O custo associado a um mapeamento de custo mínimo entre duas árvores é a distância de edição entre as duas árvores.

[3] *A notação de conjuntos* $A \setminus B$ *é utilizada para indicar o conjunto resultante da subtração dos elementos do conjunto* B *do conjunto* A.

[4] *O símbolo* λ *denota o elemento nulo.*

Como já mencionado, algoritmos de distância de edição em árvores podem trabalhar com diferentes conjuntos de operações e diferentes custos associados a essas operações. Uma vez que o problema de mapeamento, como definido acima, tem um custo computacional alto, a maioria dos algoritmos na literatura restringe as operações possíveis. Com base nessas restrições, definiu-se categorias para os algoritmos de distância de edição em árvores. Na seção seguinte, veremos quais são essas categorias.

2.2 Categorias de Algoritmos

O problema de cálculo da distância de edição em árvores para árvores não ordenadas é provadamente NP-Completo [56] e, mesmo quando se trabalha com árvores ordenadas, ainda assim os algoritmos existentes têm custo computacional elevado. Restringindo-se o conjunto das operações permitidas, conseguiu-se criar algoritmos mais eficientes [38, 43, 47]. Nesta seção, iremos mostrar uma categorização dos algoritmos de acordo com o conjunto de restrições associadas ao mapeamento gerado. A categorização que definimos a seguir pode ser considerada como uma expansão dos trabalhos apresentados em [43] e [47].

2.2.1 Distância de Edição

O problema mais genérico de distância de edição em árvores consiste em encontrar um mapeamento de custo mínimo entre duas árvores. O primeiro algoritmo para esse problema foi proposto por Tai, em [41], juntamente com a formulação do conceito de mapeamento e a prova da equivalência de distância de edição e mapeamento de custo mínimo. Formalmente podemos definir a distância de edição entre as árvores conforme a seguir.

Definição 2.6. *A **distância de edição** entre duas árvores, T_1 e T_2, é dada por $\delta(T_1, T_2) = min(\gamma(M) \mid M$ é um mapeamento entre T_1 e $T_2)$.*

O algoritmo apresentado em [41] tem custo $O(n^2h^4)$, onde n e h são respectivamente o número de vértices e a altura das árvores sendo comparadas.

Outro algoritmo para esse problema foi proposto por Zhang e Shasha em [54], com custo $O(n_1n_2\text{min}(h_1, l_1)\text{min}(h_2, l_2))$, onde l_1 e l_2 são o número de folhas das árvores sendo comparadas. Como, no pior caso (árvore de altura 2), o número de folhas da árvore é $\Omega(n)$, então o pior caso do algoritmo de Zhang e Shasha é $O(n^4)$. Baseado no trabalho de Zhang e Shasha, Klein propôs um novo algoritmo [26] para o mesmo problema, com custo $O(n_1^2n_2 \log n_2)$. O novo algoritmo possui um limite inferior para o pior caso, mas não é necessáriamente melhor para todos os casos. O algoritmo, porém, é capaz de operar com o mesmo desempenho com árvores sem raiz. Finalmente, Dulucq e Touzet fizeram uma análise de ambos os algoritmos em [16] e propuseram um algoritmo também com pior caso $O(n_1^2n_2 \log n_2)$, porém ótimo em respeito à quantidade de *sub-árvores relevantes*[5] analisadas.

Posteriormente, Dulucq e Ticht obtiveram a ordem de complexidade do caso médio do algoritmo apresentado em [54], chegando ao custo $O(n_1^{\frac{3}{2}}n_2^{\frac{3}{2}})$. A análise de complexidade pode ser vista

[5] Veja [16] para a definição correspondente.

em [15]. O avanço mais recente no limite assintótico do problema é resultado do trabalho de Chen [9], onde um algoritmo baseado em multiplicação rápida de matrizes é apresentado. O algoritmo tem custo $O((n_1 + l_1^2)\text{min}(l_2 h_2) + n_2)$.

Nesse interim, Zhang, Shasha e Wang desenvolveram uma extensão de seu algoritmo [55] para casamento[6] de árvores na presença de um tipo específico de curinga[7], chamado VLDC[8], voltado para aplicações de processamento de linguagem natural. Outra variação do mesmo algoritmo também aparece em [23], aplicado à análise de estruturas secundárias de RNA. Finalmente, esse algoritmo foi, em momento posterior, expandido por Barnard, Clarke e Duncan para possibilitar operações em sub-árvores [2]. Neste mesmo trabalho podem ser encontradas implementações para os algoritmos descritos em [40], [41] e [54].

Uma análise detalhada dos algoritmos acima foi feita por Bille, em [5]. A Tabela 2.1 resume os resultados obtidos.

Algoritmo	Tempo	Espaço
Chen [9]	$O(n_1^2 n_2 + n_1 l_1^2 + l_1^{2.5} l_2)$	$O((n_1 + l_1^2)\text{min}(l_2, h_2) + n_2)$
Klein [26]	$O(n_1^2 n_2 \log(n_2))$	$O(n_1 n_2)$
Tai [41]	$O(n_1 n_2 h_1^2 h_2^2)$	$O(n_1 n_2 h_1^2 h_2^2)$
Zhang e Shasha [54]	$O(n_1 n_2 \text{min}(h_1, l_1)\text{min}(h_2, l_2)$	$O(n_1 n_2)$

Tabela 2.1: Algoritmos para distância de edição entre árvores.

2.2.2 Distância de Alinhamento

A *distância de alinhamento* equivale à distância de edição, com um conjunto adicional de restrições em sua formulação, e foi apresentada por Jiang et al. em [25], juntamente com uma prova que a obtenção da distância de alinhamento para árvores não ordenadas é MAX SNP-Difícil. A seguir formalizamos o conceito de *mapeamento de alinhamento*:

Definição 2.7. *Um mapeamento é um **mapeamento de alinhamento** se ele puder ser estendido para um isomorfismo[9] entre as duas árvores não rotuladas subjacentes após inserir um número mínimo de vértices nas duas árvores.*

A distância de alinhamento $\alpha(T, T')$ é definida como o custo associado a um mapeamento de alinhamento de custo mínimo entre T e T'. A Figura 2.6 ilustra um mapeamento de alinhamento de custo mínimo, enquanto a Figura 2.7 nos mostra que um mapeamento sem restrições pode ser obtido com menor custo. As operações de custo nulo foram omitidas da seqüências de edição por simplicidade.

[6] O casamento de duas árvores é análogo ao casamento de uma expressão regular à uma cadeia de caracteres. Veja [55] para definição precisa.

[7] Um curinga é um vértice que pode assumir qualquer rótulo.

[8] Variable Length Don't Care.

[9] *Um isomorfismo é um mapa bijetivo f tal que f e seu inverso f^{-1} são homeomorfismos, i.e. mapeamentos que preservam a estrutura. Em nosso contexto, podemos definir simplesmente que duas árvores isomórficas são duas árvores estruturalmente idênticas.*

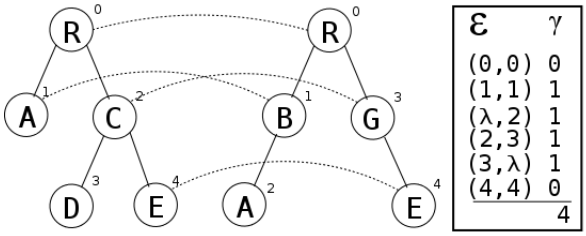

Figura 2.6: Um mapeamento de alinhamento.

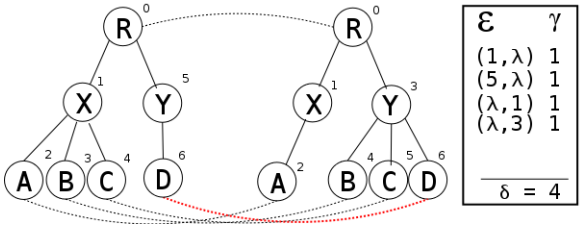

Figura 2.7: Um mapeamento sem restrições.

O algoritmo de Jiang et al. tem custo $O(n_1 n_2 (d_1 + d_2)^2$, onde d_1 e d_2 representam os graus máximos das árvores sendo comparadas. Um novo algoritmo, com tempo de execução sensível à saída, foi posteriormente proposto por Jansson e Lingas em [24]. Esse algoritmo tem custo $O((n_1 + n_2)\log(n_1 + n_2)(d_1 + d_2)^4 s^2)$, onde s é a soma do número de operações de inserção e remoção[10]. Caso os graus das árvores seja limitado, o custo se reduz a $O((n_1 + n_2)\log(n_1 + n_2)s^2)$.

A Tabela 2.2 sumariza os resultados da literatura para a distância de alinhamento.

Algoritmo	Tempo	Espaço
Jansson e Lingas [24]	$O(n^2 \log n^2 d^4 s^2)$	$O(n^2 \log n^2 d^4 s^2)$
Jiang et al. [25]	$O(n_1 n_2 (d_1 + d_2)^2)$	$O(n_1 n_2 (d_1 + d_2)^2)$

Tabela 2.2: Algoritmos para distância de alinhamento.

2.2.3 Distância de Sub-Árvores Isoladas

Mapeamentos de sub-árvores isoladas exigem que sub-árvores disjuntas sejam mapeadas para sub-árvores disjuntas. Formalmente, podemos definir um mapeamento de sub-árvores isoladas conforme se segue.

Definição 2.8. *Um mapeamento M entre duas árvores T e T' é um* ***mapeamento de sub-árvores isoladas*** *satisfazendo a condição último$(T[i_1]) < t[i_2] \leftrightarrow$ último$(T'[j_1]) < t'[j_2]$ para todo par $(i_1, j_1), (i_2, j_2) \in M$.*

[10]No trabalho original, os autores definem s como o número de vértices brancos, o que corresponde exatamente à soma do número de operações de inserção e remoção, em nossa terminologia.

Analogamente à distância de edição, a *distância de sub-árvores isoladas* $\beta(T, T')$ é definida como o custo de um mapeamento de sub-árvores isoladas de custo mínimo. Na Figura 2.8 podemos ver um mapeamento de sub-árvores isoladas de custo mínimo.

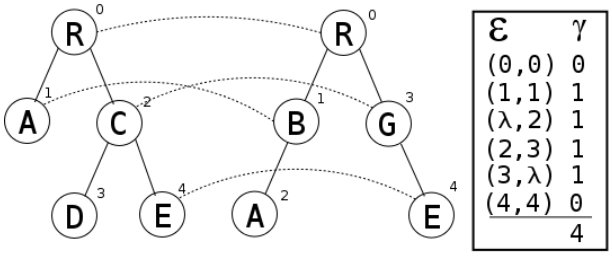

Figura 2.8: Um mapeamento de sub-árvores isoladas.

Tanaka e Tanaka são os responsáveis pelo primeiro algoritmo de distância de sub-árvores isoladas [42], com custo $O(n_1 n_2 \log n_1 \log n_2)$. Algoritmos para o mesmo problema podem ser encontrados em dois outros trabalhos, sob diferentes nomes. No primeiro deles, descrito em [53], a medida é chamada *distância de edição restrita*[11] e o algoritmo tem custo $O(n_1 n_2)$. No ano seguinte, Richter apresentou a mesma medida sob o nome *distância de edição respeitadora da estrutura*[12], juntamente com um algoritmo de custo $O(n_1 n_2 d_1 d_2)$, porém com menor ocupação de memória para árvores de menor grau [39].

A Tabela 2.3 mostra um resumo dos resultados existentes para a distância de sub-árvores isoladas.

Algoritmo	Tempo	Espaço
Richter [39]	$O(n_1 n_2 d_1 d_2)$	$O(n_1 h_2 d_2)$
Tanaka e Tanaka [42]	$O(n_1 n_2 \log n_1 \log n_2)$	
Zhang [53]	$O(n_1 n_2)$	$O(n_1 n_2)$

Tabela 2.3: Algoritmos para distância de sub-árvores isoladas.

2.2.4 Distância Top-Down

O primeiro algoritmo publicado sobre comparação de árvores descreve um algoritmo para *distância top-down*. Nesse trabalho, publicado em 1977, Selkow propõe um algoritmo quadrático para o problema [40]. Embora o conceito de mapeamento ainda não existisse naquele momento, posteriormente o trabalho foi inserido dentro da categorização de mapeamentos atualmente adotada pela literatura. Formalmente podemos definir um *mapeamento top-down* como se segue.

Definição 2.9. *Um mapeamento M entre duas árvores T e T' é um **mapeamento top-down** se ele satisfizer, para todo par $(i, j) \in M$, tal que $t[i]$ não é raiz de T e $t'[j]$ não é raiz de T', a seguinte condição: $(i, j) \in M \leftrightarrow (pai(i), pai(j)) \in M$.*

Intuitivamente, diz-se que a distância *top-down* restringe remoções e inserções às folhas das árvores. Por esse motivo, ela é também conhecida como *distância de edição de grau 1* [5]. Como

[11] Constrained edit distance no original.

[12] Structure respecting edit distance no original.

usualmente, define-se a distância *top-down*, $\mu(T, T')$, como o custo associado a um mapeamento de custo mínimo entre T e T'. Podemos ver um mapeamento *top-down* de custo mínimo na Figura 2.9. Já a Figura 2.10 nos mostra um mapeamento de sub-árvores isoladas que não é *top-down*, com custo menor para as mesmas árvores.

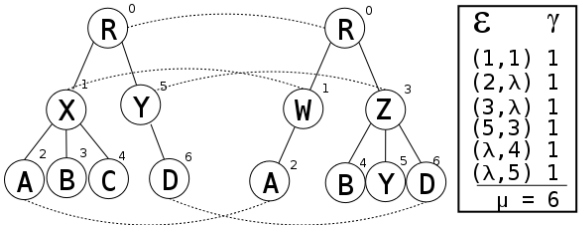

Figura 2.9: Um mapeamento top-down

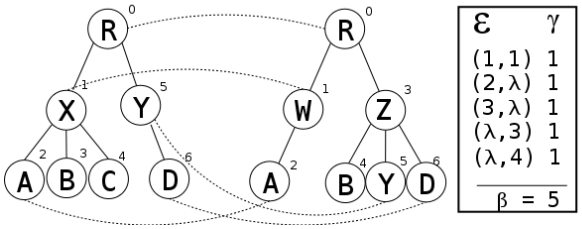

Figura 2.10: Um mapeamento de sub-árvores isoladas que não é *top-down*.

Posteriormente, Yang apresentou outro algoritmo quadrático para o problema, com aplicação para identificação de diferenças sintáticas entre dois programas [51]. Uma vez que programas de computador podem ser vistos como as árvores produzidas pelo processo de compilação, a aplicação da distância de edição em árvores é uma abordagem conveniente. Mais recentemente, Valiente mostrou que o problema de encontrar um mapeamento *top-down* é equivalente ao problema de *máximo isomorfismo de árvores* [45]. Valiente também dedicou todo um capítulo de um livro [44] para a distância de edição *top-down*, mostrando algoritmos baseados nas técnicas de *backtracking*, *branch-and-bound*, *divide-and-conquer* e programação dinâmica. Embora Valiente denomine o problema simplesmente como distância de edição em árvores em seu livro, sua formulação equivale ao que chamamos distância *top-down*. Finalmente, Chawathe propôs um algoritmo baseado em programação dinâmica, que pode ser facilmente adaptado para memória externa [7]. O algoritmo de Chawathe se diferencia por modelar o problema com uma única instância da matriz de programação dinâmica. É importante ressaltar que todos os algoritmos conhecidos para o problema têm custo quadrático no pior caso.

Na Tabela 2.4 podemos ver os resultados existentes para a distância *top-down*.

2.2.5 Distância Top-Down Restrita

A formulação mais recente para o problema de distância de edição em árvores, a *distância top-down restrita*, se assemelha à distância *top-down*, porém restringe também a operação de substituição às

Algoritmo	Tempo	Espaço
Chawathe [7]	$O(n_1 n_2)$	$O(n_1 n_2)$
Selkow [40]	$O(n_1 n_2)$	$O(n_1 n_2)$
Valiente [44]	$O(n_1 n_2)$	$O(n_1 n_2)$
Yang [51]	$O(n_1 n_2)$	$O(n_1 n_2)$

Tabela 2.4: Algoritmos para distância *top-down*.

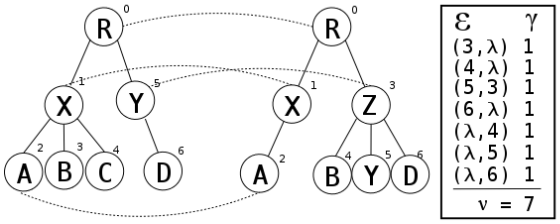

Figura 2.11: Um mapeamento top-down restrito

folhas das árvores. A distância *top-down* restrita foi apresentada em [38]. A Definição 2.10 formaliza o conceito de *mapeamento top-down restrito*.

Definição 2.10. *Um mapeamento top-down M entre duas árvores T e T' é um **mapeamento top-down restrito** se ele satisfizer, para todo par $(i, j) \in M$, as seguintes condições:*

1. $(i, j) \in M \leftrightarrow (i_x, j_x) \ni M \mid i_x \in \mathit{desc}(i), j_x \neq \lambda$

2. $(i, j) \in M \leftrightarrow (i_x, j_x) \ni M \mid j_x \in \mathit{desc}(j), i_x \neq \lambda$

A distância *top-down* restrita, denotada $\nu(T, T')$, é o custo associado a um mapeamento *top-down* restrito entre duas árvores T e T'. Na Figura 2.11 temos um mapeamento dessa espécie.

A Tabela 2.5 mostra os resultados conhecidos para distância de edição top-down restrita.

Algoritmo	Tempo	Espaço
Reis et al. [38]	$O(n_1 n_2)$	$O(d_1 d_2 \min(h_1, h_2))$

Tabela 2.5: Algoritmo para distância top-down restrita.

2.2.6 Distância Bottom-Up

A última formulação do problema de distância de edição dentro da categorização sendo apresentada é a *distância bottom-up*. Valiente introduziu a distância *bottom-up* em [43], onde pode-se encontrar um algoritmo linear para o problema. A seguir, define-se formalmente um *mapeamento bottom-up*.

Definição 2.11. *Um mapeamento M entre duas árvores T e T' é um **mapeamento bottom-up** se satisfizer, para todo par $(i, j) \in M$, a seguinte condição: $(i, j) \in M \leftrightarrow (i_k, j_k) \in M$, onde $t[i_k]$ e $t'[j_k]$ são filhos de $t[i]$ e $t'[j]$ respectivamente.*

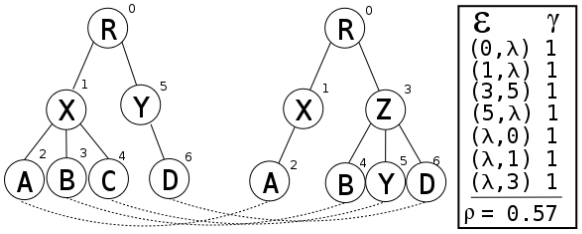

Figura 2.12: Um mapeamento *bottom-up*

Embora possamos definir a distância *bottom-up* de maneira análoga àquela feita com as outras medidas, Valiente definiu, em seu trabalho, a distância *bottom-up*, denotada aqui por $\rho(T, T')$, como $1 - \frac{f}{\max(|T|,|T'|)}$, onde f é o número de vértices da maior floresta comum a T e T'. Dessa forma, a distância *bottom-up* será sempre um número real no intervalo fechado $[0;\ 1]$. A Figura 2.12 nos mostra um mapeamento *bottom-up* e o valor de distância associado.

Na Tabela 2.6 temos a análise do algoritmo de Valiente para a distância *bottom-up*.

Algoritmo	Tempo	Espaço
Valiente [43]	$O(n_1 + n_2)$	$O(n_1 + n_2)$

Tabela 2.6: Algoritmo para distância *bottom-up*.

2.3 Hierarquia de Distâncias

Como visto, as várias formulações para o problema de distância de edição em árvores são definidas como restrições a partir de um problema geral. A partir dessas restrições, é possível definir uma hierarquia entre as diversas categorias de algoritmos. A Figura 2.13 ilustra tal hierarquia.

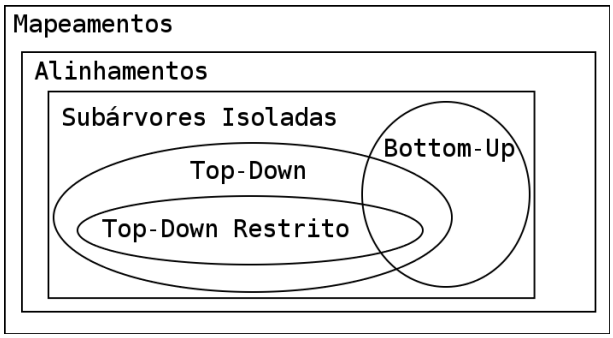

Figura 2.13: Hierarquia de distâncias

Pela Figura 2.13, podemos ver que todo alinhamento é um mapeamento, todo mapeamento de sub-árvores isoladas é um alinhamento e assim por diante. Em geral, algoritmos para formulações menos restritas podem ser modificados para obter soluções para as formulações mais restritas. Por outro lado, os algoritmos específicos para formulações mais restritas têm limites assintóticos mais baixos.

Apenas a relação entre mapeamentos *bottom-up* e mapeamentos *top-down* não é uma relação de continência. Embora existam pares de árvores para os quais ambas as formulações resultam em um mesmo mapeamento, árvores mais complexas irão resultar em mapeamentos diferentes. Em [43], Valiente prova que a interseção entre mapeamentos *top-down* e mapeamentos *bottom-up* dá-se apenas em árvores isomórficas. Como mapeamentos *top-down* restritos são sempre mapeamentos *top-down*, por conseqüência, só há interseção entre mapeamentos dessa classe e mapeamentos *bottom-up* quando se comparam árvores isomórficas.

Capítulo 3

O Algoritmo RTDM

Neste capítulo, descreveremos em detalhes o algoritmo *Restricted Top-Down Mapping*, ou RTDM [38]. Esse algoritmo é uma das principais contribuições desta dissertação e é o pilar da estratégia de extração de dados da Web a ser apresentada no próximo capítulo.

O algoritmo RTDM combina idéias apresentadas em [43] e [51]. Para determinar o mapeamento *top-down* restrito entre duas árvores T_1 e T_2, o algoritmo RTDM primeiro encontra todas as sub-árvores idênticas que ocorrem na mesma profundidade nas árvores de entrada. Nesta etapa, usa-se um grafo de classes de equivalência, de modo similar ao que é feito em [43]. Em seguida, é usada programação dinâmica para obter um mapeamento de custo mínimo entre as duas árvores. Iremos mostrar separadamente as duas etapas do algoritmo.

3.1 Identificação de Sub-Árvores Idênticas

A primeira etapa do algoritmo RTDM é responsável por encontrar todas as sub-árvores idênticas nas duas árvores de entrada. Esse resultado será utilizado como um corte do processamento da segunda etapa, responsável por calcular a distância *top-down* restrita propriamente dita. Ambas as etapas são independentes e a primeira etapa pode ser omitida, com penalidade de desempenho no tempo final de execução nos casos em que as árvores sendo comparadas compartilham um número significativo de sub-árvores idênticas. Mais a frente, o termo "significativo" é definido com precisão. Na Figura 3.1 pode-se ver o pseudo-código para a primeira etapa do algoritmo, denominada *Identical Subtrees*. Essa etapa é responsável por encontrar classes de equivalência para os vértices das árvores de entrada.

Cada árvore é percorrida em caminhamento pós-ordem [44]. Um caminhamento pós-ordem garante que os pais de um vértice serão visitados apenas após a visita aos seus filhos. Para cada vértice, verifica-se quais vértices de mesmo rótulo já foram visitado. Para cada um desses vértices, verifica-se se os filhos do vértice sendo visitado estão associados às mesmas classes de equivalência[1] dos filhos do vértice de mesmo rótulo, onde dois vértices são ditos equivalentes se são raízes de subárvores idênticas. Nessa hipótese, o vértice visitado é raiz de uma sub-árvore idêntica ao vértice de mesmo rótulo e recebe, por conseqüência, a mesma classe de equivalência. Caso contrário, uma nova classe

[1] Dado um conjunto X e uma relação de equivalência $\sim$, a *classe de equivalência* de um elemento a em X é o subconjunto de todos os elementos de X que são equivalentes a a.

```
1  IdenticalSubtrees(T_1, T_2)
2  begin
3    let m: o tamanho de T_1;
4    let n: o tamanho de T_2;
6    let K: um vetor de tamanho m + n;
7    let H: um mapa multi-associativo;
8    let nextClass: um inteiro;
9    let index: o indíce pós-ordem de um vértice, se ele pertence a T;
10   let index: o indíce pós-ordem de um vértice, somado a m ele pertence a T'.
12   nextClass ← 0
13   //O procedimento addTree adiciona a informação de vértices e arestas
14   //de uma árvore ao mapa multi-associativo e define a classe de
15   //equivalência de cada um de seus vértices.
16   addTree(T)
17   begin
18     for i ∈ postorder(T)
19       if label(i) ∋ H
20         H[label(i)] ← ∅
21         K[index(i)] ← nextClass
22         nextClass ← nextClass + 1
23       fi
24       for v ∈ H[label(i)]
25         k_1 ← ⋃_{a∈children(i)} K[index(a)]
26         k_2 ← ⋃_{b∈children(v)} K[index(b)]
27         if k_1 = k_2
28           K[index(i)] ← K[index(v)]
29           break
30         fi
31         K[index(i)] ← nextClass
32         nextClass ← nextClass + 1
33       end
34       H[label(i)] ← H[label(i)] ∪ (i)
35     end
36   end
37   addTree(T_1)
38   addTree(T_2)
39   return K
40 end
```

Figura 3.1: O algoritmo *IndenticalSubtrees*.

de equivalência é criada. Como, em um caminhamento pós-ordem, vértices-filho são visitados antes dos vértices-pai, as classes de equivalência dos filhos já foram sempre criadas antes de se visitar os pais.

A Figura 3.2 mostra de forma intuitiva a operação do algoritmo. Os vértices estão numerados de acordo com a função $index$ do algoritmo. Na primeira parte da figura, os vértices da primeira árvore são inseridos em um mapa multi-associativo[2] (H), e suas classes de equivalência são criadas (K). Esse

[2]Uma mapa multi-associativo é qualquer estrutura de dados que implemente, para um universo de chaves K e um

é o estado do algoritmo ao retornamos do procedimento *addTree*, definido na linha 16. Na segunda parte da figura vê-se o resultado da repetição do procedimento para a segunda árvore. A saída do algoritmo é o vetor K, contendo as classes de equivalências. As sub-árvores com a mesma classe de equivalência são idênticas.

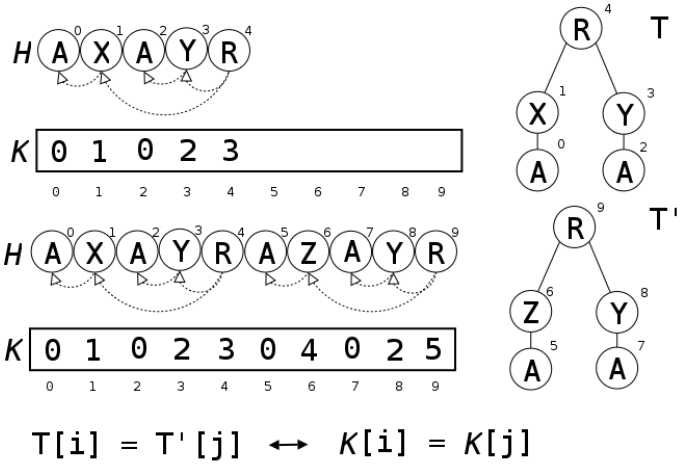

Figura 3.2: O algoritmo *IdenticalSubtrees* em operação.

Teorema 3.1. *O algoritmo* IdenticalSubtrees *tem custo de execução* $O(n)$ *e ocupa espaço* $O(n)$.

Prova. O algoritmo percorre cada vértice de cada árvore (linha 18) e para cada um desses vértices ele verifica o valor de K de cada um de seus filhos (linhas 25 a 27), ou seja, uma comparação adicional para cada aresta da árvore. Como o número de arestas em uma árvore de tamanho n é sabidamente $n-1$, temos um total de $n+n-1 = 2n-1$ comparações. Concluímos então que o algoritmo é linear.

Para a operação do algoritmo, são utilizadas duas estruturas cujos tamanhos são função da entrada, K e H. K é um vetor de tamanho $m+n$ e portanto ocupa espaço $O(n)$. O mapa multi-associativo H possui uma entrada para cada vértice das árvores ao final do algoritmo e, portanto, ocupa também espaço $O(n)$, completando assim a prova. □

O algoritmo *IdenticalSubtrees* pode ser modificado para usar um mapa associativo simples e diminuir a utilização de memória. O custo assintótico de espaço, porém, é dominado por K e se mantém.

3.2 Cálculo da Distância de Edição Top-Down Restrita

A segunda etapa do algoritmo é responsável por encontrar a distância de edição *top-down* restrita entre as árvores. A Figura 3.3 contém o pseudo-código referente a esta etapa do algoritmo.

universo de valores V, uma sobrejeção $F : k \mid k \subseteq K \rightarrow \{\{v \mid v \in V\} \cup \emptyset\}$. Tabelas *hash* e árvores de pesquisas são examplos de mapas associativos. A modificação necessária para se obter um mapa multi-associativo consiste simplesmente em associar um conjunto de valores a cada chave dessas estruturas.

Esta etapa é baseada no algoritmo de distância *top-down* proposto por Yang em [51]. A idéia central por trás do algoritmo é utilizar programação dinâmica [11] para obter o melhor mapeamento entre as sub-árvores de um mesmo nível. Como o custo de cada operação necessária para obtenção desse mapeamento depende do custo associado ao mapeamento das sub-árvores nos níveis inferiores, o algoritmo se chama recursivamente (linha 33), até que sejam atingidas as folhas das árvores sendo analisadas (linhas 27 e 29). O ganho de desempenho em relação ao algoritmo original de Yang se deve à formulação *top-down* restrita e ao uso do algoritmo *IdenticalSubtrees*, que possibilitam parar

```
1  RTDM(T1, T2, ε: threshold)
2  begin
3     let m: o número de filhos da raiz de T1
4     let n: o número de filhos da raiz de T2
7     M[0,0] ← 0
8     for i = 1 to m
9        M[i,0] ← M[i−1,0] + Σ_k^{t1[k]∈Ci} delete(t1[k])
10    end
11    for j = 1 to n
12       M[0,j] ← M[0,j−1] + Σ_k^{t2[k]∈Cj} insert(t2[k])
13    end
14    for i = 1 to m
15       for j = 1 to n
16          Ci ← descendentes próprios(t1[i])
17          Cj ← descendentes próprios(t2[j])
18          d ← M[i−1,j] + Σ_k^{t1[k]∈Ci} delete(t1[k])
19          i ← M[i,j−1] + Σ_k^{t2[k]∈Cj} insert(t2[k])
20          s ← M[i−1,j−1]
21          if M[i−1,j−1] > ε
22             s ← ∞
23          elsif t1[i] e t2[j] são sub-árvores idênticas
24                s ← s + 0
25          elsif t1[i] ≠ t2[j]
26                s ← s + replace(t1[i], t2[j])
27                if t1[i] é uma folha
28                   s ← s + Σ_k^{t2[k]∈Cj} insert(t2[k])
29                elsif t2[j] é uma folha
30                      s ← s + Σ_k^{t1[k]∈Ci} delete(t1[k])
31                fi
32           else
33                s ← s+ RTDM(t1[i], t2[j], ε)
34          fi
35          M[i,j] ← min(d, i, s);
36       end
37    end
38    return M[m,n]
39    end
```

Figura 3.3: O algoritmo RTDM.

a recursão antes de se atingir as folhas das árvores. O primeiro corte, chamado *corte top-down*, está expresso nas linha 25. O segundo corte, denominado *corte bottom-up*, faz uso das classes de equivalência calculadas na primeira etapa do algoritmo e está expresso nas linhas 23 e 24. Finalmente, tem-se um ganho adicional quando deseja-se apenas saber se a distância *top-down* restrita é inferior a um dado limiar ϵ. Para isso basta que o algoritmo cesse a recursão ao encontrar mapeamentos com custo superior ao limiar em consideração. Esse corte, chamado *corte threshold*, pode ser visto nas linhas 21 e 22 da Figura 3.3.

3.3 Análise de Complexidade

A maioria dos autores de algoritmos de distância de edição em árvores mostra a ordem de complexidade de seus algoritmos apenas em função do número de vértices da árvore de entrada e algumas de suas características intrínsecas, tais como número de folhas, altura ou grau da árvore. Porém, pode-se fazer uma análise mais apropriada levando em consideração a saída dos algoritmos, uma vez que o desempenho de vários dos algoritmos está intimamente associado à diferença entre as árvores. Analisaremos agora, progressivamente, o algoritmo RTDM, até atingir um limite assintótico que faça jus ao desempenho do algoritmo.

Teorema 3.2. *O algoritmo RTDM tem custo de execução $O(n^2)$ e ocupa espaço $O(d^2h)$.*

Prova. A cada iteração do algoritmo, é definida uma matriz com tamanho $(d_1 d_2)$. No pior caso (todos os rótulos são iguais), o algoritmo é chamado recursivamente para todos os elementos da matriz, até que as folhas de uma das árvores sejam atingidas. São, portanto, executadas $(d_1 d_2)^{\min(h_1+h_2)}$ comparações.

Analisando o algoritmo, vê-se que o pior caso ocorre com árvores isomórficas com apenas os rótulos das folhas distintos. Isso acontece porque as chamadas recursivas só ocorrem na linha 33, e essa linha só é chamada quando temos um alinhamento de vértices com o mesmo rótulo, cujas sub-árvores não são idênticas (caso contrário o algoritmo cessaria a recursão na linha 24). Como o máximo de alinhamentos de vértices de mesmo rótulo acontece quando todas os filhos das sub-árvores analisadas são idênticos e em igual número, pode-se restringir a análise do algoritmo apenas à entradas com sub-árvores isomórficas que difiram apenas pelos rótulos das folhas, ou seja, $d_1 = d_2$ e $h_1 = h_2$. Portanto, substituindo $h = O(\log_d n)$, que é um resultado conhecido para árvores completas de grau fixo, é possível provar que o algoritmo tem custo de execução $O(n^2)$:

$$(dd)^{\log_d n} = O(n^2) \quad (3\text{-}1)$$

$$d^{2\log_d n} = O(n^2) \quad (3\text{-}2)$$

$$n^2 = O(n^2) \quad (3\text{-}3)$$

Como apenas uma matriz está em memória por vez para cada nível visitado da árvore, o algoritmo atinge o topo de utilização de memória ao alcançar as folhas da árvore, onde a ocupação de memória é $(d_1 d_2)\min(h_1, h_2)$. Como no pior caso $d_1 = d_2$ e $h_1 = h_2$, segue-se que a utilização de memória é d^2h e a prova está completa. □

A análise anterior desconsidera todos os cortes de recursão existentes no algoritmo. Façamos portanto uma nova análise considerando o *corte top-down*.

Teorema 3.3. *O algoritmo RTDM tem custo de execução $O((n-k)^2)$, onde $k = \nu(T, T')$ para o custo unitário.*

Prova. Primeiramente veremos como, a partir do pior caso definido na prova anterior, pode-se obter, por construção, um novo pior caso alterando algum rótulo das árvores de entrada.

Suponha o pior caso, dado por duas árvores isomórficas T e T' com todos os rótulos idênticos, exceto pelas folhas. Se alterarmos o rótulo de um vértice interno $t[i]$ de alguma das árvores, o algoritmo cessará a recursão para a sub-árvore $T[i]$, na linha 25 (corte *top-down*). Portanto, para manutenção do pior caso, devemos modificar um rótulo $t[i]$, tal que $|T[i]|$ é mínimo. Conclui-se, portanto, que para manutenção do pior caso para k rótulos modificados, os vértices alterados devem ser primeiro as folhas de uma árvore (já alterados), depois os pais das folhas e assim por diante.

Para árvores isomórficas com k rótulos modificados, de acordo com o pior caso definido acima, a distância de edição *top-down* restrita é $\nu(T, T') = k$, e o algoritmo analisa apenas os vértices-filho de rótulos idênticos (caso contrário a recursão cessa no corte *top-down*). Portanto, o algoritmo analisa apenas as superárvores de T e T' com rótulos idênticos e de tamanho $n - k$.

Como partimos de um árvore completa e os rótulos são sempre modificados em um caminhamento *bottom-up* (todas as folhas, os pais das folhas, etc.), a superárvore de rótulos idênticos é uma *árvore balanceada*. Para árvores balanceadas, têm-se mais uma vez que $h = O(\log_d n)$ e pode-se prosseguir com prova análoga à prova anterior. Conclui-se então que o algoritmo RTDM tem custo de execução $O((n-k))^2)$ e a prova está completa. □

O próximo passo é observar o ganho obtido pelo *corte bottom-up*. Esse corte possibilita cessar a recursão quando o algoritmo se depara com sub-árvores idênticas.

Teorema 3.4. *O algoritmo RTDM tem custo de execução $O((n-(k+s))^2)$ onde $k = \nu(T, T')$ para o custo unitário e $s = |I|$, onde $I = \{(x, y) \mid T[x] = T'[y]\}$.*

Prova. Assim como foi feito na prova anterior, pode-se obter o pior caso para o algoritmo por construção. Agora, porém, considera-se também que a recursão cessa para todas as sub-árvores idênticas (corte *bottom-up*). De forma análoga à prova anterior, pode-se provar que o algoritmo precisa analisar apenas a superárvore balanceada de tamanho $n - (k + s)$, onde s é o número de sub-árvores idênticas de T e T'. A análise segue como feito na prova anterior e obtém-se o custo $O((n-(k+s))^2$. □

Finalmente, faz-se necessário acrescentar o custo relativo à primeira etapa do algoritmo para obter o custo total do algoritmo RTDM. Com isso, atinge-se o limite assintótico $O((n-(k+s))^2 + n)$. A partir desse limite, pode-se observar que, ao compararmos árvores idênticas ($s = n$, $k = 0$), a segunda etapa roda em tempo constante e obtém-se a distância de edição *top-down* restrita com um número linear de comparações, executadas na primeira etapa. Se as árvores forem completamente diferentes ($s = 0$, $k = n$), mais uma vez a segunda etapa roda em tempo constante. Nesse caso, se optarmos por omitir a primeira etapa, o algoritmo como um todo roda em tempo constante. Infelizmente, k é saída da segunda etapa e s saída da primeira etapa, e, portanto, não é possível saber, a priori,

se é ou não benéfico fazer uso do algoritmo *IdenticalSubtrees* e a decisão deverá ser tomada com base em conhecimento da entrada. Para o sistema de extração de notícias da Web apresentado nesta dissertação, optou-se por usar sempre o algoritmo *IdenticalSubtrees*.

3.4 Análise Empírica

Na prática, o algoritmo é bastante veloz, por causa dos vários cortes da recursão. Exceto pelo pior caso, o custo será amortizado pelo corte nas linhas 23 a 25, que chamamos *corte bottom-up* ou pelo corte nas linhas 25 a 31, que chamamos *corte top-down*. Ademais, quando estamos interessados apenas em saber se a distância *top-down* se encontra em um dado limiar, o corte nas linhas 21 a 22 impede que o algoritmo inspecione caminhos que não podem levar a uma solução satisfatória. Na Figura 3.4, podemos ver uma comparação do algoritmo RTDM com a versão em memória do algoritmo de Chawathe [7].

Optamos por nos restringir a comparação ao algoritmo de Chawathe porque apenas algoritmos para o problema de distância de edição *top-down* têm desempenho comparável ao algoritmo RTDM, e todos os algoritmos dessa classe possuem ordem de complexidade quadrática. Embora existam algoritmos lineares para a distância de edição *bottom-up*, essa formulação do problema não é útil para a extração de dados da Web.

As curvas são mostradas através da ligação direta dos pontos medidos, e através de uma suavização de *bezier* das curvas resultantes. A curva com suavização é uma aproximação da média do tempo de execução do algoritmo dado um número infinito de medidas.

As curvas na Figura 3.4 refletem o impacto dos vários cortes utilizados no algoritmo RTDM. Embora ambos os algoritmos sejam quadráticos no pior caso, o algoritmo RTDM é sensível à entrada, e em nenhuma das comparações feitas, atinge seu limite superior. Como todas as árvores usadas para comparação dos algoritmos são páginas da Web[3], não é possível dizer definitivamente que o algoritmo RTDM obtém esse desempenho em qualquer cenário. Ainda assim, se construirmos uma entrada especificamente adaptada para sempre resultar no pior caso do algoritmo, seu desempenho seria o mesmo do algoritmo de Chawathe, exceto por uma variação constante.

3.5 Observações Finais

Uma observação útil em relação ao algoritmo RTDM é que uma alteraração trivial nas linhas 25 a 31 (corte *top-down*), resulta em um algoritmo que determina a distância *top-down* tradicional (não restrita). Como conseqüência, tem-se também um novo algoritmo para a distância *top-down*.

Outro aspecto interessante do algoritmo RTDM é sua flexibilidade, no que tange ao custo das operações de edição. Por exemplo, ele nos permite comparar uma dada árvore com um padrão de árvore com curingas de tamanho variável, uma característica que é explorada na estratégia de extração de dados da Web apresentada nesta dissertação. Esse problema é semelhante ao problema de casamento de expressões regulares em cadeias e uma formulação dele já foi tratada na literatura [55].

[3] As páginas Web utilizadas são as mesmas usadas nos experimentos de extração do Capítulo 4.

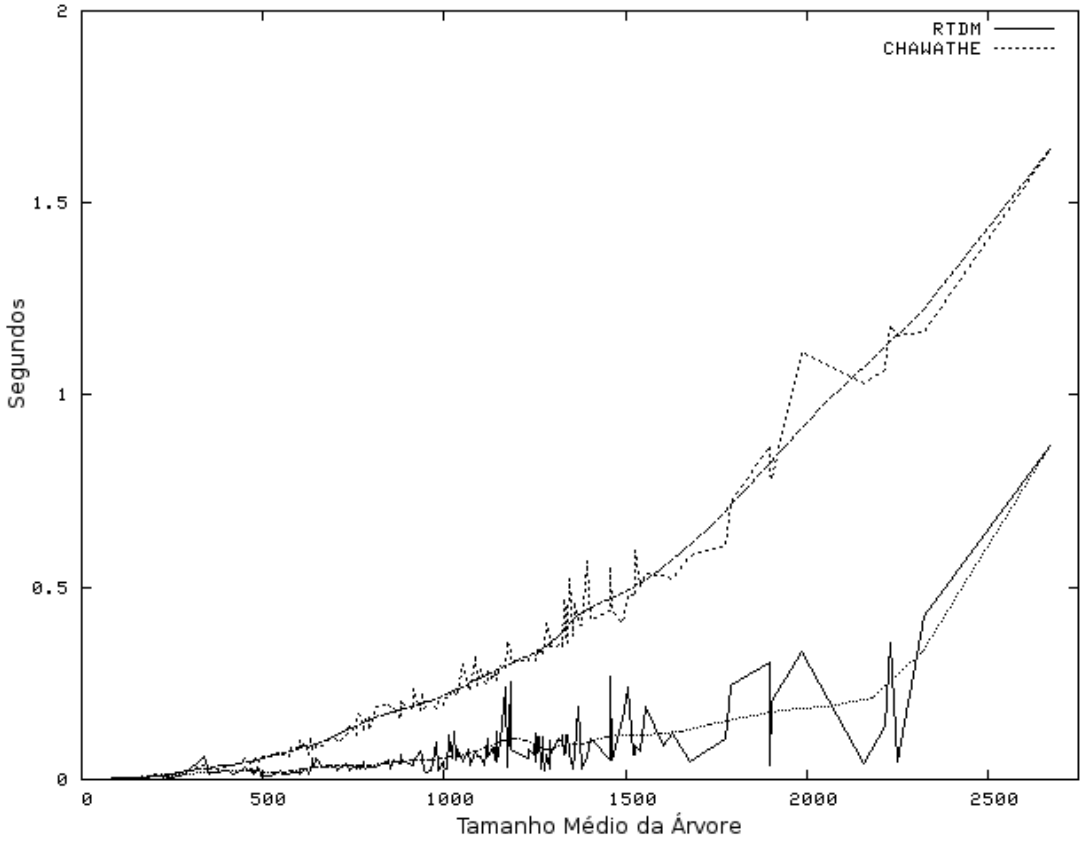

Figura 3.4: RTDM vs CHAWATHE

Capítulo 4

Extração de Notícias da Web

Neste capítulo, descrevemos como utilizar o algoritmo RTDM para criação de sistemas para extração de dados da Web. Embora a maior parte das idéias apresentadas seja de aplicação geral, nossa discussão é conduzida demonstrando a criação de um sistema de extração de notícias de jornais e revistas em formato eletrônico. Mostraremos como extrair o título e corpo (e possivelmente outros dados) das notícias das versões eletrônicas de grandes jornais e revistas brasileiras. Com essa informação podemos, por exemplo, enriquecer a informação provida por empresas de *clipping*. O sistema que implementamos tem alto grau de recuperação e precisão, trabalhando de forma completamente automática. Em nossos experimentos, atingimos a precisão média de 87,71% de sucesso. A seguir temos uma breve introdução ao contexto deste trabalho.

Grande parte dos trabalhos recentes utilizando distância de edição em árvores [21, 38, 52] tem como foco a extração de dados da Web. Isso não surpreende, uma vez que os documentos da Web são escritos em linguagens como HTML e XML, e essas linguagens são, essencialmente, a representação serializada de árvores, cujos vértices internos contêm marcações estilísticas e de estrutura, e as folhas contêm os dados exibidos na página.

Embora, nos primórdios da Web os documentos HTML fossem criados, em sua grande maioria, através de um processo manual, atualmente eles são gerados por programas de computador, que fazem modificações em uma árvore modelo, chamada *template*, acrescentando dados recuperados de um banco de dados. Essa observação é especialmente verdadeira quando nos restringimos ao universo de sítios de jornais ou revistas, onde uma grande quantidade de conteúdo é criada por profissionais de mídia e publicada através de ferramentas de *software*. O *template* provê a formatação e o estilo da página Web, enquanto as informações propriamente ditas são criadas pelos jornalistas e obtidas de um banco de dados. Na Figura 4.1 podemos ver exemplos de notícias que compartilham *templates* comuns.

O problema de extração de dados abordado consiste em, dado o conjunto de páginas de um sítio, obter o conteúdo do banco de dados utilizado para gerá-lo. A idéia que apóia o conjunto de algoritmos aqui discutido consiste em agrupar as páginas que compartilham um mesmo *template*, descobrir esse *template*, e então retirar os dados que foram acrescentadas ao *template*, provenientes do banco de dados.

Para completar a descrição do cenário onde estaremos trabalhando, é preciso que nos aprofun-

Figura 4.1: Diferentes notícias que compartilham *templates* comuns

demos um pouco mais no formato de sítios de jornais e revistas. A maioria desses sítios possui a seguinte organização: (a) uma página principal com as principais chamadas, (b) várias páginas de seções que mostram as chamadas agrupadas pelas áreas de interesse (por exemplo, esportes, tecnologia, etc.) e (c) páginas que contêm as notícias propriamente ditas, contendo o título, autor, data e corpo da notícia. Apenas essas últimas são de nosso interesse, pois são onde a informação relevante se encontra, enquanto as primeiras servem basicamente ao aspecto navegacional do sítio.

Esse cenário configura o problema de extração com registros únicos e múltiplas páginas. Ou seja, cada página contém uma única notícia e a nossa entrada é formada por várias páginas. O cenário oposto ocorre quando se tem uma única página, e vários registros, tais como livros e preços em uma livraria virtual. Outros trabalhos na literatura tratam desse problema [52] e os resultados aqui mostrados não podem ser utilizados para esse tipo de extração sem um esforço significativo de adaptação.

A estratégia adotada para extração dos dados pode ser dividida em quatro passos distintos: (1) agrupamento das páginas, (2) geração de padrões de extração, (3) identificação de dados e (4) rotulamento de dados. A Figura 4.2 ilustra esses passos.

Iremos agora detalhar cada um dos passos usados no processo de extração. É importante notar que o fundamento dos passos principais (agrupamento, geração de padrões e identificação de dados) é o algoritmo RTDM, descrito no capítulo anterior, aplicado com diferentes modelos de custo.

4.1 Agrupamento de Páginas

O primeiro passo do trabalho toma como entrada um conjunto de páginas recuperadas da Web e gera grupos de páginas que compartilham características estilísticas e de formatação, ou seja, criadas

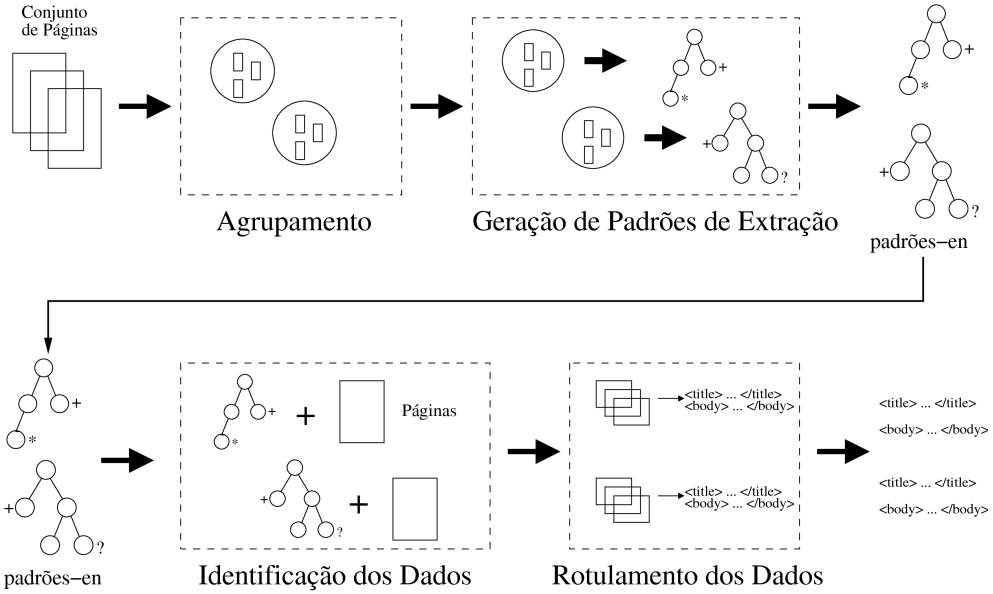

Figura 4.2: Principais passos do processo de extração.

a partir de um *template* comum. Cada um desses grupos será posteriormente generalizado em um padrão de extração para um dado *template*.

Para gerar os grupos de páginas, usamos técnicas tradicionais de agrupamento hierárquico [49], nas quais a medida de distância entre duas árvores é a distância de edição *top-down* restrita. Como não existem grupos pré-definidos, é adotado um limiar para determinar se dois grupos devem ser fundidos. Na implementação usada para os experimentos deste capítulo, tal limiar é dado por 80% de similaridade entre as árvores.

O modelo de custo usado nesse passo é o mais simples possível. Todas as operações têm custo unitário, exceto pela substituição de vértices com rótulos iguais, que tem custo zero. Outros trabalhos [35] sugerem o uso de um conjunto mais sofisticado de operações, porém os experimentos realizados mostram que esse modelo simples é efetivo o suficiente para os propósitos desejados. A saída desse passo é um conjunto de grupos de páginas que compartilham um mesmo *template*.

4.2 Geração de Padrões de Extração

No passo de geração de padrões de extração, cada um dos grupos criados é generalizado em um padrão de extração, que iremos chamar padrão de extração de nós ou *padrão-en*. Formalmente, um padrão-en é uma árvore definida como se segue.

Definição 4.1. *Um **padrão de extração de nós**, ou padrão-en, é uma árvore ordenada rotulada que pode conter vértices especiais chamados* curingas*. Todo curinga é necessariamente uma folha e pode assumir um dos seguintes tipos:*

- ***Single** (·) Um curinga que captura uma única sub-árvore e precisa ser consumido.*
- ***Plus** (+) Um curinga que captura uma floresta de sub-árvores irmãs e precisa ser consumido.*

- ***Option*** *($?$) Um curinga que captura uma única sub-árvore e pode ser descartado.*

- ***Kleene*** *($*$) Um curinga que captura uma floresta de sub-árvores irmãs e pode ser descartado.*

Um padrão-en pode ser pensado como uma expressão regular para árvores. No caso de expressões regulares, porém, pode-se ter curingas que capturam qualquer subconjunto do alfabeto de entrada, Σ. Já os curingas de um padrão-en sempre capturam qualquer elemento do alfabeto, ou seja, vértices com qualquer rótulo.

O objetivo, neste passo, é assegurar que cada curinga corresponda a um objeto rico em dados[1] na página Web. Curingas *single* e *plus* devem corresponder a objetos obrigatórios na página, como o título e corpo de uma notícia, enquanto curingas *option* e *kleene* devem corresponder a objetos opcionais, tais como listas de notícias relacionadas.

Um padrão-en *aceita* (ou casa com) uma dada árvore se existe um mapeamento com custo não infinito entre o padrão-en e a árvore em questão. Esse conceito e o modelo de custo associado a esse mapeamento serão formalmente definidos na próxima seção.

O objetivo neste passo é, dado um grupo de páginas, gerar o padrão-en que aceita todas as páginas do grupo. Conseqüentemente, as diferenças de conteúdo entre as páginas no grupo serão modeladas como curingas nos padrões-en gerados. Para gerar os padrões-en, iremos nos valer de uma *operação de composição*, definida conforme a seguir.

Definição 4.2. *Sejam T_x e T'_x padrões-en distintos. Então a* ***composição*** *de T_x e T'_x, $T_x \circ T'_x$ é um padrão-en T''_x tal que:*

- *Seja S o conjunto de árvores aceitas por T_x.*

- *Seja S' o conjunto de árvores aceitas por T'_x.*

- *Seja S'' o conjunto de árvores aceitas por T''_x.*

- *Então $S \cup S' \subseteq S''$.*

O processo de geração de um padrão-en consiste em iterar sobre todas as árvores que representam as páginas em um grupo e, incrementalmente, fazer a composição de cada uma com o padrão-en sendo criado. É importante notar que qualquer árvore pode ser vista como um padrão-en sem nenhum curinga, e, portanto, pode ser adotada como o padrão-en inicial para um grupo. No final do processo, tem-se um padrão-en que aceita todas as árvores no grupo.

Vejamos como o algoritmo RTDM é usado na implementação da operação de composição. Primeiro é preciso definir a operação de igualdade para vértices de um padrão-en.

Definição 4.3. *Seja $t[i] \in T_x$ e $t'[j] \in T'_x$. Então, $t[i] = t'[j]$ se e somente se:*

1. *$t[i]$ e $t[j]$ são curingas.*

2. *$t[i]$ e $t[j]$ não são curingas e seus rótulos associados são iguais.*

[1] Objetos ricos em dados são os objetos, representados em uma página Web, que possuem informação, potencialmente relevante, para o usuário.

Essa é a operação de igualdade que será usada pelo algoritmo RTDM. Como modelo de custo, damos o mesmo custo, 1, para qualquer operação de edição nas árvores. Dados dois padrões-en, T_x e T'_x, o algoritmo RTDM pode ser usado para obter um mapeamento $M_{T_x \leftrightarrow T'_x}$. A partir desse mapeamento, um padrão-en composto $T''_x = T_x \circ T'_x$ pode ser construído seguindo o algoritmo dado na Definição 4.4.

Definição 4.4. *Dados dois padrões-en T_x e T'_x, um mapeamento $M_{T_x \leftrightarrow T'_x}$ entre esses padrões, e uma função de composição de vértices $f(a, b)$, definimos o algoritmo de construção do padrão-en composto $T''_x = T_x \circ T'_x$ como se segue.*

1. *Se o vértice a não pertence ao mapeamento, então adicione a' a T''_x, onde $a' = f(a, ?)$.*
2. *Se $(a, b) \in M$, então adicione a' a T''_x, onde $a' = f(a, b)$.*

A função de composição de vértices $f(a, b)$ é definida a seguir.

Definição 4.5. *Sejam n, n_1 e n_1 vértices não curingas de um padrão-en. Definimos a função de composição de vértices $f(a, b)$, cuja ordem dos parâmetros não é relevante, da seguinte maneira:*

$f(*,*)$	$=$	$*$	$f(+,+)$	$=$	$+$	$f(.,.)$	$=$	$.$
$f(*,+)$	$=$	$*$	$f(+,.)$	$=$	$+$	$f(.,?)$	$=$	$?$
$f(*,?)$	$=$	$*$	$f(+,?)$	$=$	$*$	$f(.,n)$	$=$	$.$
$f(*,.)$	$=$	$*$	$f(+,n)$	$=$	$+$	$f(?,?)$	$=$	$?$
$f(*,n)$	$=$	$*$				$f(?,n)$	$=$	$?$

$f(n_1, n_2) = n_1$ *se n_1 e n_2 tem rótulos idênticos*

$f(n_1, n_2) = .$ *se n_1 e n_2 tem rótulos distintos*

A motivação por trás desse conjunto de operações é que vértices opcionais no *template* que o padrão-en está tentando modelar devem se manter opcionais, após compor o padrão-en com uma nova árvore, e quantificadores mais poderosos (por exemplo, *kleene* e *plus*) devem ser mantidos no padrão-en final. Finalmente, vértices não curingas mapeados para vértices não curingas com rótulos diferentes, na árvore sendo composta, devem resultar em novos curingas.

Um detalhe interessante é que alguns objetos ricos em dados nas páginas podem compreender várias sub-árvores irmãs, como o corpo de uma notícia, formado por vários parágrafos adjacentes. O propósito dos curingas *plus* e *kleene* é, exatamente, capturar tais objetos como entidades únicas.

Ainda sobre os curingas *plus* e *kleene*, podemos ver que se observarmos atentamente a função $f(a, b)$ acima, veremos que não há uma política de "promoção" de quantificadores, ou, em outras palavras, curingas *plus* ou *kleene* nunca serão gerados se não há curingas *plus* ou *kleene* na entrada da função. Esses curingas são criados durante um passo adicional após a composição de dois padrões-en. Esse passo adicional é bastante simples. Todo curinga seguido por uma série de curingas *option* deve ser convertido em um curinga para objetos de tamanho variável, ou seja, curingas *plus* ou *kleene*. Se o curinga que precede o conjunto de curingas *option* é um *single* ou *plus*, então o conjunto de curingas *option* e o curinga precedente devem ser convertidos em apenas um curinga *plus*. Se o curinga precedente é um *option* ou *kleene*, então a conversão deve resultar em um *kleene*. A Definição 4.6 formaliza a operação de promoção.

Definição 4.6. *Sejam os curingas single, plus, option e kleene, respectivamente denotados ., $+$, ? e $*$, a promoção de uma cadeia de curingas se dá de acordo com as regras seguintes.*

- $.??? \rightarrow +$
- $+??? \rightarrow +$
- $???? \rightarrow *$
- $*??? \rightarrow *$

A Figura 4.3 ilustra todo o processo de geração dos padrões-en, incluindo a "promoção" de um curinga. Na implementação usada para os experimentos deste capítulo, mesmo se os curingas forem separados por um máximo de 3 vértices não curingas, eles podem ser fundidos (incluindo os vértices não curingas) em um único curinga de tamanho variável (*plus* ou *kleene*).

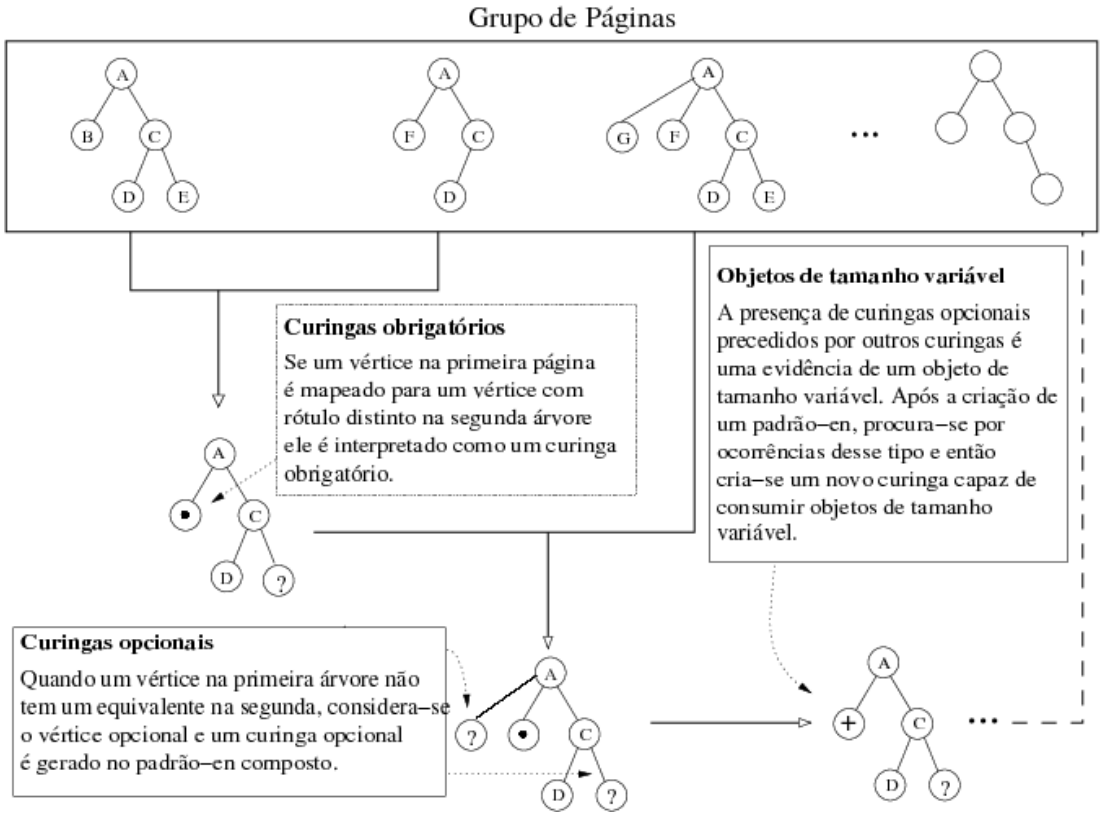

Figura 4.3: Criação de um padrão-en a partir de um grupo de páginas

A técnica para geração de extratores discutida nesta seção pode ser aplicada a qualquer domínio, desde que se configure o cenário de múltiplas páginas e registros únicos. A necessidade de múltiplas páginas é óbvia, uma vez que a geração consiste em comparar páginas com estruturas semelhantes. As razões do requisito de registros únicos, porém, são mais difíceis de ser compreendidas. Na verdade, basta que o número de registro em cada uma das páginas seja o mesmo. Caso contrário, iremos sempre modelar as sub-árvores contendo os registros adicionais como curingas opcionais. Esses curingas, porém, acabam por se posicionar na raiz dessas sub-árvores, o que pode compreender uma grande quantidade de elementos de formatação que são indesejados.

Um conjunto de páginas de resposta de mecanismos de busca, por exemplo, pode conter uma página com apenas uma resposta e várias outras com dez respostas. Ao aplicarmos o algoritmo de geração de extratores, apenas a primeira resposta seria corretamente mapeada para um curinga *single*, enquanto todo o restante da página seria mapeado para um curinga *kleene*, e então a estrutura da página não seria mais corretamente refletida no padrão-en.

O problema de múltiplos registros é minimizado pelo passo de agrupamento, pois as páginas com quantidades muito distintas de registros seriam direcionadas a grupos diferentes. Ainda assim, o extrator gerado nesses casos fatalmente irá conter erros, exceto se as páginas em cada grupo contiverem exatamente o mesmo número de registros.

Os padrões-en são uma aproximação da estrutura dos dados contidos em uma página. Este passo corresponde, portanto, ao sub-problema (2), de acordo com o Capítulo 1. No entanto, os padrões-en são capazes apenas de aproximar essa estrutura, e há ainda muito a se fazer até que seja possível inferir, com alto grau de precisão, a estrutura dos dados de um conjunto de páginas Web. Notadamente, esse problema foi provado NP-Completo em [50].

4.3 Identificação de Dados

No passo de identificação de dados, casa-se o conjunto de padrões-en gerados com o conjunto de páginas das quais se pretende fazer a extração. Para escolher o padrão-en mais adequado a uma dada página, e extrair os dados da mesma, mais uma vez recorre-se ao algoritmo RTDM, com um modelo de custos apropriado.

Diz-se que, em um dado mapeamento, se um vértice curinga no padrão-en é mapeado para um vértice na árvore HTML alvo, então o curinga *consome* o vértice. Os vértices consumidos por curingas representam as diferenças entre as páginas. Assim como foi feito por Crescenzi et. al. [12], neste trabalho pressupõe-se que as partes divergentes das páginas são exatamente os dados provenientes do banco de dados. Ou seja, as diferenças entre as páginas correspondem às suas porções ricas em dados.

Para se identificar os dados de uma páginas Web, faz-se um *casamento* entre um padrão-en e a página, que nada mais é que um mapeamento satisfazendo um conjunto de requisitos apropriado à identificação dos dados. Diz-se que, se existe um casamento entre um padrão-en e uma página Web, então o padrão-en *aceita* a página. A Definição 4.7 formaliza o conceito de casamento.

Definição 4.7. *Um **casamento** entre um padrão-en e uma árvore é um mapeamento M_x, tal que as seguintes regras são satisfeitas, na ordem dada:*

1. *Todo vértice não curinga pertencente ao padrão-en deve ser mapeado para um vértice idêntico na árvore alvo.*

2. *Todo vértice restante na árvore alvo deve ser consumido por um curinga.*

3. *Curingas single (.) devem consumir exatamente uma sub-árvore da árvore alvo.*

4. *Curingas plus (+) devem consumir ao menos uma sub-árvore da árvore alvo.*

5. *Curingas option ($?$) devem consumir uma única sub-árvore da árvore alvo, se possível.*
6. *Curingas kleene ($*$) devem consumir ao menos uma sub-árvore da árvore alvo, se possível.*
7. *Curingas plus ($+$) devem consumir tantas sub-árvore irmãs na árvore alvo quanto possível.*
8. *Curingas kleene ($*$) devem consumir tantas sub-árvore irmãs na árvore alvo quanto possível.*

A satisfação das regras 1, 2, 3 e 4 é suficiente para garantir que um padrão-en aceite uma árvore alvo. As regras 5 e 6 asseguram que o casamento é tão *justo* quanto possível, ou seja, se for possível utilizar um curinga opcional (*option* ou *kleene*) sem violar as regras anteriores, ele deve ser utilizado. As regras 7 e 8 são automaticamente satisfeitas, se as regras anteriores forem observadas, e estão declaradas para facilitar o entendimento do processo de casamento.

Para obtermos um casamento, usamos o algoritmo RTDM com a função de igualdade mostrada na Definição 4.8, e um modelo de custo apropriado. É interessante notar que o primeiro item da definição da função de igualdade é exatamente o contrário do usado no passo anterior, que pode ser visto na Definição 4.3. Esse é apenas um artifício para que os curingas apareçam no mapeamento como rótulos distintos, uma vez que, como veremos abaixo, adota-se, neste passo, o custo zero para um mapeamento com um curinga. Caso tivéssemos adotado a mesma definição de igualdade do passo anterior, ainda assim obteríamos o mesmo mapeamento, porém precisaríamos instanciar todo o mapeamento para encontrar os vértices consumidos por curingas. Com esse artifício, basta que instanciemos a porção do mapeamento que une vértices distintos.

Definição 4.8. *Seja $t[i] \in T_x$ e $t'[j] \in T'_x$. Então, $t[i] = t'[j]$ se e somente se:*

1. *$t[i]$ e $t[j]$ não são curingas.*
2. *$t[i]$ e $t[j]$ tem rótulos iguais.*

O próximo passo é definir o modelo de custo para que o mapeamento se comporte de acordo com os requisitos de casamento, dados na Definição 4.7. Podemos encontrar na Definição 4.9 o modelo que satisfaz essa exigência.

Definição 4.9. *Seja a um vértice em um padrão-en e seja b um vértice na árvore alvo. O modelo de custo para o algoritmo RTDM é definido da seguinte maneira:*

- ***Substituição***

  (A) a é um curinga $\rightarrow 0$.

  (B) do contrário $\rightarrow \infty$.

- ***Inserção***

  (C) Existe um ancestral de b que foi consumido por um curinga $\rightarrow 0$.

  (D) O antecessor de b foi consumido por um $* \rightarrow 0$.

  (E) O antecessor de b foi consumido por um $+ \rightarrow 0$.

  (F) do contrário $\rightarrow \infty$.

- ***Remoção***

  (G) $a =?$ *ou* $a = * \rightarrow 1$.

  (H) do contrário $\rightarrow \infty$.

O custo de substituição A garante que apenas curingas podem ser substituídos pelas sub-árvores que eles consomem. O custo de inserção C permite que sub-árvores completas sejam consumidas pelos curingas. Os custos D e E permitem aos curingas de tamanho variável consumir florestas de árvores irmãs. O custo de remoção de vértices G assegura que apenas curingas opcionais podem ser removidos, e associa um custo não nulo à remoção de um curinga, para que eles sejam preferencialmente cobertos pelo custo A. Finalmente, os custos B, F e H em conjunto garantem que o padrão-en deve *aceitar* a árvore alvo, ou o mapeamento terá custo infinito. Embora os custos C, D e E possam, a primeira vista, parecer de cálculo complicado, eles podem ser trivialmente implementados em tempo constante dentro do algoritmo RTDM. Para testar a aplicabilidade de qualquer um deles, é preciso apenas verificar se o vértice na árvore alvo está sendo inserido na posição (ou imediatamente após) um curinga do padrão-en. Esse modelo de custos garante que ou as condições na Definição 4.7 são satisfeitas, ou o mapeamento tem custo infinito.

Uma vez que o padrão-en foi casado, o processo de identificação dos dados é direto. Ambas as árvores (o padrão-en e a árvore HTML) são percorridas em pré-ordem e, para cada curinga encontrado no padrão-en, a passagem de texto[2] nos vértices consumidos pelo curinga é extraída da página HTML.

Quando a extração de dados está sendo feita, em um conjunto de páginas já agrupadas, e com padrões-en correspondentes, a escolha do padrão-en para extração é simples. Basta usar o padrão-en correspondente ao grupo onde a página foi colocada. Quando se deseja aproveitar os padrões-en já gerados para algum sítio para se extrair dados de uma nova página, é preciso escolher o padrão-en correto. Para tanto, casa-se todos os padrões-en do sítio com a página, e o padrão-en escolhido é aquele cujo casamento tem o menor custo. A Figura 4.4 ilustra o processo de casamento.

A identificação de dados corresponde ao sub-problema (1), de acordo com o Capítulo 1. Ironicamente, foi preciso que primeiro se obtivesse uma aproximação da estrutura dos dados, o que corresponde ao sub-problema (2), através da geração dos extratores, para que então fosse possível identificar os dados.

Assim como o passo de geração de extratores, o passo de identificação dos dados não assume nada a respeito da entrada, exceto que se tratam de páginas Web, e que elas possuem um número igual de registros. Todas as técnicas mostradas até então podem portanto ser aplicadas em diferentes problemas. Mecanismos de busca, por exemplo, podem se beneficiar da identificação dos dados, que lhes permite descartar as partes irrelevantes das páginas, tais como menus e elementos de navegação. Já as heurísticas de rotulamento de dados que serão mostradas na próxima seção são aplicáveis somente ao problema específico de extração de notícias.

[2]Uma passagem de texto é simplesmente o texto contido em uma sub-árvore, que será exibido ao usuário.

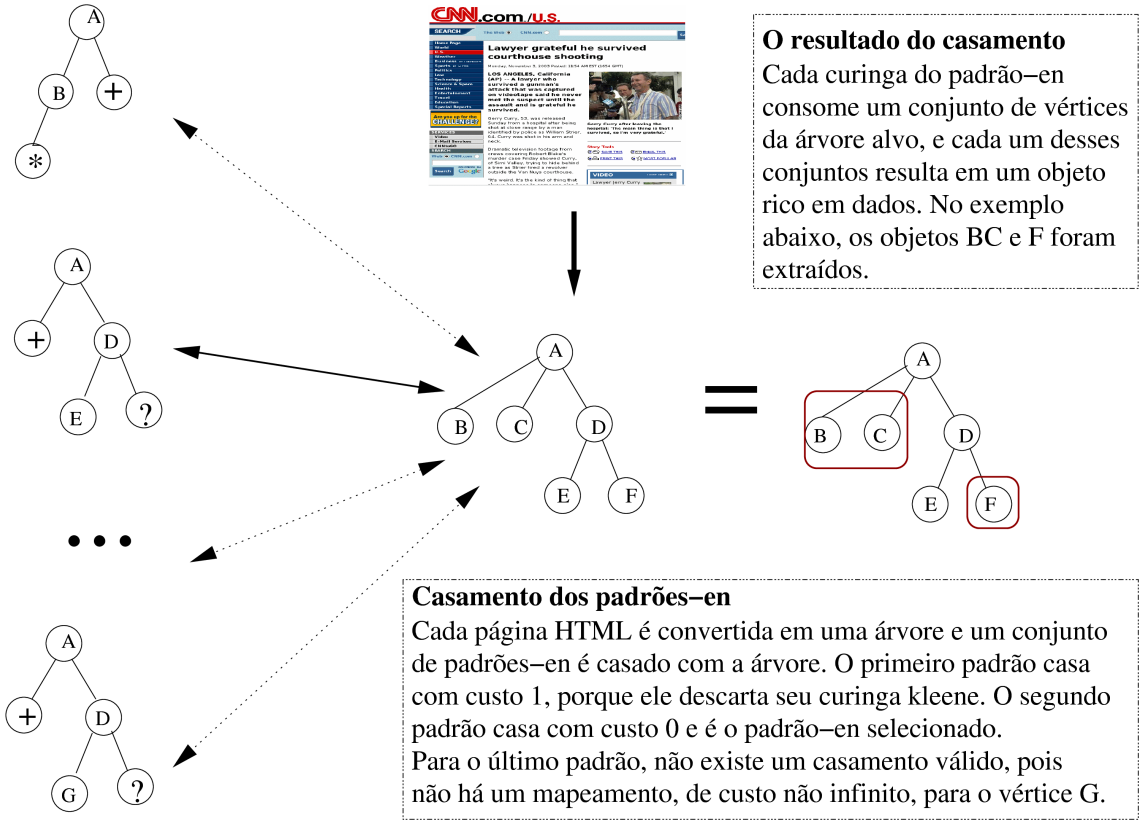

Figura 4.4: Casamento de padrões-en com uma página Web

4.4 Rotulamento de Dados

A saída do passo de casamento de dados é um conjunto ordenado de passagens de texto. Cada um dos elementos do conjunto corresponde a um conjunto de vértices consumidos por um curinga. Mais formalmente, pode-se definir a saída de um casamento como um conjunto $T = (t_1), (t_2), ..., (t_n)$ onde cada t_i é a passagem de texto recuperada pelo i-ésimo curinga do padrão-en.

O objetivo do passo de rotulamento de dados é selecionar de T as passagens t_i e t_j que correspondem ao título e ao corpo da notícia sendo extraída da página Web[3] Para atingir esse objetivo, são aplicadas heurísticas simples a T, como discutido abaixo. Dado um conjunto de passagens de texto $T = (t_1), (t_2), ..., (t_n)$, diz-se que:

- $length(t_i)$ é o número de termos (palavras) na passagem t_i.
- $\mid t_k \cap t_i \mid$ é o número de termos que ocorrem em t_k e t_i.
- t_i é o corpo de uma notícia se e somente se $length(t_i) > length(t_k)\ \ \forall\ 1 < k < n, k \neq i$ e $length(t_k) > 100$ (Heurística de rotulamento de corpo).
- t_j é o título de uma notícia se e somente se $1 \leq length(t_j) \leq 20$ e $\frac{|t_j \cap t_i|}{p_j - p_i} > \frac{|t_k \cap t_i|}{p_k - p_i}\ \forall\ 1 < k < i, k \neq j$ (Heurística de rotulamento de título).

[3]Embora outros dados relevantes como o nome do autor e data da notícia possam ser tratadas ortogonalmente, o foco deste trabalho concentra-se apenas na extração do título e corpo de notícias.

Em outras palavras, a passagem escolhida como o corpo da notícia é a mais longa passagem que contenha ao menos 100 palavras. A passagem escolhida para o título é aquela que tem entre 1 e 20 palavras, tem a maior interseção com o corpo da notícia, e é a mais próxima ao corpo. A intuição por trás da seleção de títulos é que a maior parte das vezes o título se encontra próximo ao corpo da notícia e seus termos aparecem no corpo da notícia.

Apesar da simplicidade das heurísticas, a estratégia de rotulamento é bastante efetiva, como poderá ser visto pelos resultados experimentais mostrados na próxima seção. Infelizmente, qualquer domínio, senão o de notícias, exige um conjunto diferente de heurísticas. Dentro do problema de extração de dados, o rotulamento dos dados faz parte do sub-problema de obtenção da semântica associada aos dados, denominado sub-problema (3) na introdução desta dissertação.

4.5 Resultados Experimentais

Os experimentos para validação da nossa abordagem foram feitos usando 4088 páginas HTML coletadas de 35 sítios brasileiros de notícias diferntes. Os sítios escolhidos são os mais populares veículos de mídia da imprensa brasileira, incluindo jornais de alcance nacional, agências de notícias, revistas e as principais publicações regionais. O sistema foi desenvolvido nas linguagens *C++* e *Python* e todos os experimentos foram executados em uma máquina *Pentium III* 700Mhz, com 128MB de memória principal. O sistema levou aproximadamente 4 horas para processar toda a entrada.

Os experimentos consistiram na análise da saída da totalidade do processo de extração. As notícias extraídas foram manualmente comparadas com as páginas HTML originais para verificar a correção e completeza dos resultados. A Tabela 4.1 mostra os resultados para todos os 35 sítios. Como se pode observar, extraiu-se corretamente uma média de 87,71% das notícias, enquanto 9,25% foram incorretamente extraídas e somente 3,04% não foram extraídas.

Durante os experimentos, foi verificado que o uso da distância de edição *top-down* restrita é realmente adequado para identificar as porções ricas em dados de páginas da Web. No passo de rotulamento, porém, ainda é difícil identificar precisamente o título das notícias. A maior parte dos erros foram devido a subtítulos e nomes de autores que foram extraídos como títulos.

É importante ressaltar que os resultados mostrados nas Seções 4.1, 4.2 e 4.3 correspondentes a agrupamento, geração de extratores e identificação de dados, aplicam-se a outros domínios, senão notícias e jornais eletrônicos, desde que se configure o cenário de múltiplas páginas com registros únicos. Apenas o passo de rotulamento deve ser modificado de acordo com a aplicação.

Sítio	Corretas	Incorretas	Não Extraídas	Total de Páginas
A notícia Joenville	83,95%	13,58%	2,47%	81
AOL Brasil	87,60%	12,40%	0,00%	121
Agência Estado	94,90%	4,08%	1,02%	98
Correio Brazilense	71,43%	11,90%	16,67%	119
Correio da Bahia	98,15%	1,85%	0,00%	54
DCI	96,55%	0,00%	1,72%	228
Diário de Natal	96,62%	0,00%	2,90%	206
Diário Grande ABC	100,00%	0,00%	0,00%	8
Diário do Maranhão	75,00%	25,00%	0%	48
Diário Popular	100,00%	0,00%	0,00%	85
Diário de Cuiaba	85,26%	12,82%	1,92%	154
Diário do Com. BH	92,31%	3,85%	3,85%	26
Estado de Minas	77,40%	21,47%	1,13%	177
Estado de São Paulo	84,33%	15,21%	0,46%	217
Folha de Pernam.	91,18%	1,47%	7,35%	68
Folha de São Paulo	77,78%	13,33%	8,89%	225
Gazeta Digital	88,17%	10,75%	1,08%	185
Gazeta Mercantil	87,01%	0,65%	12,34%	154
Hoje em Dia	90,91%	9,09%	0,00%	66
IDG Now	93,18%	2,27%	4,55%	44
ITWeb	96,88%	0,00%	3,13%	32
InvestNews	95,47%	0,00%	4,53%	329
Jornal da Tarde SP	90,57%	5,66%	3,77%	159
O Dia RJ	75,86%	22,07%	2,07%	144
O Globo	99,35%	0,65%	0%	307
Tribuna Santos	75,00%	22,58%	2,42%	123
Tribuna da Bahia	81,13%	15,09%	3,77%	53
Tribuna da Imprensa	90,63%	9,38%	0%	32
UOL	74,53%	23,58%	1,89%	106
Valor On Line	91,45%	4,27%	4,27%	117
Verdade On Line	82,61%	13,04%	4,35%	22
Vox News	80,00%	0,00%	20,00%	35
Yahoo	93,64%	0,91%	5,45%	208
Zero Hora	83,22%	16,11%	0,67%	149
Total	87,71%	9,25%	3.04%	4088

Tabela 4.1: Resultados obtidos para extração de notícias

Capítulo 5

Conclusões

Como primeiro resultado desta dissertação, temos uma revisão ampla de algoritmos de distância de edição em árvores existentes na literatura, que permite não só a introdução à área, mas também o conhecimento da enorme gama de opções de algoritmos, com diferentes aplicações.

Pudemos ver que a distância de edição em árvores é um problema de extrema relevância para diversas áreas, mas ainda há muito a ser feito, pois a maioria dos algoritmos conhecidos tem ordem de complexidade elevada. Não bastasse, conseguimos encontrar apenas um algoritmo para memória externa [7] e nenhum paralelizável. Também não pudemos encontrar algoritmos aproximados, embora já haja provas que tais algoritmos, supondo tempo polinomial, não podem ser construídos para várias formulações do problema, exceto se P = NP [25].

Nossa segunda contribuição é um novo algoritmo, para formulação restrita do problema de distância de edição em árvores, denominado RTDM. A descrição do algoritmo é completa, incluindo estudo de limites assintóticos, pseudo-código e resultados experimentais. Através do estudo da ordem de complexidade e de experimentos, pudemos ver que o algoritmo possui desempenho superior à maior parte dos algoritmos existentes até então na literatura. Apenas algoritmos da classe *bottom-up* têm custo inferior. Mas esses algoritmos, porém, não são apropriados para o problema de extração de dados da Web.

Finalmente, descrevemos uma estratégia para construção de sistemas de extração de dados da Web utilizando distância de edição em árvores, especificamente o algoritmo RTDM. Nossa estratégia é baseada em 3 passos principais, cada um deles de interesse independente. O primeiro passo permite agrupar páginas da Web com estrutura semelhante, e pode ser aplicado em qualquer contexto onde isso é desejado [32, 35]. O segundo passo é responsável pela construção dos padrões de extração, chamados padrões-en. Como esses padrões capturam uma aproximação da estrutura das páginas Web, também a construção dos padrões-en, é, per si, um resultado útil, como mostrado em [20, 50]. No terceiro, e último dos passos, os padrões de extração são usados para identificar os dados contidos nas páginas da Web. Os dados identificados, mesmo sem uma semântica associada, podem ser úteis em outros contextos [1]. Finalmente, mostramos um passo adicional para atribuição de semântica aos dados, com aplicação restrita ao domínio de revistas e jornais eletrônicos.

Para comprovar a eficácia de nossa estratégia, construímos um sistema de extração de notícias disponíveis na Web, cobrindo um total de 4088 páginas, distribuídas em um total 35 sítios diferentes,

escolhidos entre os principais veículos de mídia brasileiros. Posteriormente, nossa escolha pelo uso de distância de edições em árvores foi corroborada pela publicação de trabalhos independentes [21, 52], que compartilham muitas das idéias apresentadas nesta dissertação.

5.1 Trabalhos Futuros

Um amplo leque de trabalhos futuros pode ser explorado a partir dos resultados apresentados nesta dissertação. Primeiramente, podemos melhorar o passo de rotulamento dos dados, que é atualmente a parte mais frágil de nossa estratégia. Para tanto, já desenvolvemos técnicas simples e eficientes, como a exploração do texto de âncoras[1]. Embora existam trabalhos recentes na literatura objetivando obter soluções genéricas para esse problema, acreditamos que, ainda por um longo período, apenas soluções ad hoc serão capazes de prover resultados satisfatórios.

Em segundo lugar, podemos nos inspirar em trabalhos recentes para melhorar nossos resultados. Em especial, pretendemos fazer uso de algumas das idéias mostradas por Zhai e Liu em [52], notadamente o uso de informação visual para construção das árvores a partir das páginas Web.

Embora em teoria o código HTML, usado para descrevar as páginas, seja a serialização exata da árvore com os dados e a formatação da página, na prática podemos encontrar todo tipo de problema nos documentos HTML da Web, e os navegadores fazem um grande esforço para corrigir esses problemas e formatar uma página de maneira adequada para consumo dos usuários.

As bibliotecas de código público para geração das árvores também não são capazes de obter o mesmo resultado que os navegadores. Da mesma forma, não existem opções comerciais capazes de fazê-lo. Ainda que existam navegadores de alta qualidade com código público, ainda assim o processo de construção das árvores se encontra espalhado ao longo de todo código do navegador e é uma tarefa virtualmente impossível separá-lo em um código encapsulado, de fácil uso para outras aplicações.

Tendo isto em mente, Zhai e Liu propuseram construir as árvores a partir da API pública dos navegadores, ao invés da abordagem tradicional e ingrata, pela qual passamos, que é construir um processador HTML geral e com bons resultados. Ao usar a API pública de um navegador, se o navegador é capaz de construir a página Web, então ele pode contruir a árvore correta.

Além disso, ao se usar a API pública do navegador, torna-se possível obter informação visual sobre os elementos na árvore. Pode-se, por exemplo, ver a distância, em *pixels*, entre dois elementos da árvore. Como as páginas são construídas para consumo humano, essa informação é extremamente útil para determinar relações entre os elementos. Além disso, elementos não contíguos no código HTML, tais como células de uma tabela, tornam-se contíguos ao serem formatados para exibição pelo navegador.

Outra importante contribuição do trabalho de Zhai e Liu é a noção de alinhamento parcial. O alinhamento parcial caracteriza-se por postergar algumas decisões sobre o mapeamento entre duas árvores, até que mais árvores representativas da estrutura sendo analisada sejam vistas. Essa idéia pode ser aplicada com bons resultados tanto em nosso passo de agrupamento de páginas, quanto na geração dos extratores.

[1] Chama-se texto de âncoras o texto utilizado nos apontadores entre páginas Web

Visando novas aplicações, intencionamos utilizar a saída do passo de casamento, que são os dados identificados, para melhorar a precisão de mecanismos de busca. Esses mecanismos hoje são prejudicados pela grande quantidade de texto de navegação e formatação existentes nas páginas. A identificação dos dados relevantes nas páginas Web possibilita uma melhora significativa na precisão das respostas destes mecanismos, como já foi mostrado em [1].

Finalmente, como maior desafio a nossa frente, está a extensão da nossa estratégia para o cenário de múltiplos registros e páginas únicas. Esse é certamente um trabalho de longo prazo, mas que pode gerar excelentes frutos. A grande dificuldade nesse cenário é a existência de padrões que ocorrem um número variável de vezes dentro de uma página. Um exemplo de padrão desse tipo são as respostas de um mecanismos de busca. Acreditamos que a solução para esse problema envolve a combinação da distância de edição em árvores com técnicas de inferência de expressões regulares, e pretendemos explorar esse caminho em breve.

5.2 Considerações Finais

Cremos que as técnicas apresentadas nesta dissertação, em conjunto com idéias recentes mostradas na literatura [21, 52], são passos em um caminho que pode resolver de forma definitiva o problema de extração de dados da Web, e vislumbramos que essa solução também irá resolver o problema de se trazer estrutura ao conteúdo da Web. O alvo mais cobiçado, porém, é a Web Semântica. Infelizmente, embora Hogue e Karger já tenham apresentado um trabalho de extração de dados com distância de edição em árvores [21] diretamente voltado para a Web Semântica, achamos que as técnicas capazes de criar o rotulamento semântico dos dados automaticamente ainda estão significativamente distantes.

O trabalho de Hogue e Karger proporciona uma visão diferente do problema de extração de dados, contextualizando-o dentro da iniciativa da Web Semântica. Essa visão possibilita criar uma nova formulação para o problema, que dá destaque à sua relevância dentro dessa iniciativa, que é um dos principais desafios aos cientistas da computação atualmente. Exploramos esse caminho no apêndice desta dissertação. Acreditamos que essa é a descrição mais adequada para o problema, pois coloca em evidência sua importância na construção de uma Web Semântica que não implique na modificação de toda a estrutura que suporta a Web atualmente.

A última consideração a ser feita é que, tanto a distância de edição em árvores, quanto a extração de dados da Web, são problemas em aberto. Embora os resultados aqui apresentados tenham aplicação direta, ainda assim não se conseguiu obter uma solução plenamente satisfatória para nenhum desses problemas. Há ainda espaço para a melhoria do desempenho dos algoritmos de distância de edição em árvores, e a extração de dados é essencialmente um alvo móvel, dada a natureza extremamente dinâmica da Web.

Apêndice A

Extração de Dados e a Web Semântica

Como proposto por Tim Bernes-Lee, a Web Semântica promete "trazer estrutura para o conteúdo das páginas Web criando um ambiente onde agentes de software transitando de página a página possam executar tarefas sofisticadas para os usuários" [4]. Mais que estrutura, o conteúdo deve também ter associada *informação semântica que possa ser analisada por agentes de software*.

As tecnologias básicas necessárias para a Web Semântica já estão disponíveis há alguns anos. O núcleo central dessas tecnologias é formado por XML, XML Schema, RDF, XSL/XSLT e CSS. Como veremos, cada um desses componentes cumpre um papel na proposta de Tim Berners-Lee. Infelizmente, os produtores de conteúdo da Web continuam trabalhando com tecnologias antigas, tais como HTML e aplicações baseadas em bancos de dados relacionais. Além do fator inércia na adoção das novas tecnologias, outro aspecto que vem dificultando a materialização da Web Semântica é a sobrecarga de trabalho necessária para publicação de conteúdo em conformidade com as novas propostas.

Um argumento, ainda mais forte, que inibe os produtores de conteúdo é o modelo de negócios da Web. Muitos dos sítios da Web geram receita a partir de anúncios. A existência de agentes de *software* capazes de extrair e oferecer as informações na Web de forma conveniente aos usuários exime esses mesmos usuários da obrigação de visitar os sítios para obter seu conteúdo. Dessa forma, os anúncios não chegam aos usuários e nenhuma receita é gerada.

Essa observação leva a crer que, se a Web Semântica vier a se tornar real algum dia, o esforço necessário será fruto das tecnologias capazes de extrair os dados, sua estrutura e sua semântica das páginas Web, tais como elas existem agora. A Figura A.1 mostra a organização ideal da Web Semântica e o funcionamento da Web tradicional (ou da maior parte dela), na qual navegamos todos os dias.

Na Web tradicional, os dados que compõem o conteúdo de um sítio são, em geral, armazenados em um banco de dados relacional. Quando um usuário requisita uma página, a aplicação responsável pelo sítio busca os dados no banco e usa esses dados para preencher lacunas em um *template*. Um template nada mais é que o esqueleto de uma página Web, com locais previamente definidos onde serão inseridos os dados. O template armazena a informação de navegação comum a todas as páginas e de *apresentação* que serão usadas para exibir a página. O banco de dados contém os *dados* propriamente ditos, o esquema relacional que define a *estrutura* dos dados e os rótulos das colunas e tabelas,

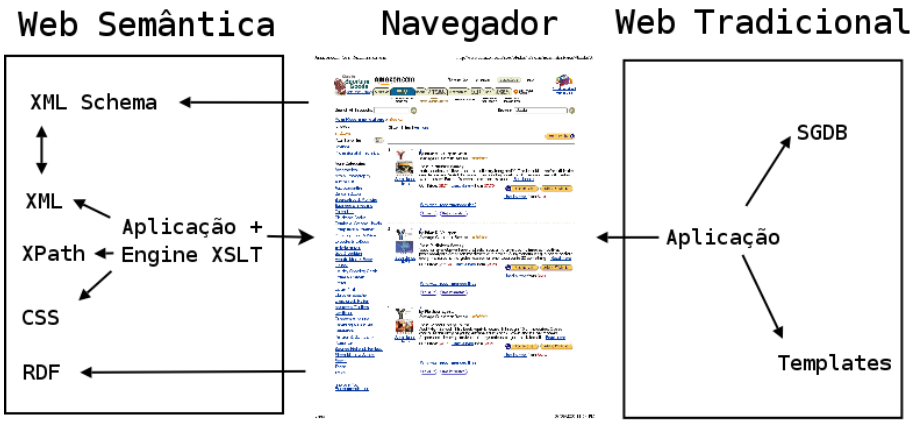

Figura A.1: Tecnologias da Web Semântica e da Web Tradicional

que são a informação de *semântica* dos dados. A *seleção* dos dados é feita através de consultas SQL disparadas pela aplicação.

A Web Semântica armazena as informações com uma granularidade significativamente maior: os *dados* estão nos arquivos XML; a *estrutura* dos dados é descrita pelo XML Schema; a informação para *apresentação* está nos arquivos CSS; as consultas ao banco de dados para *seleção* dos dados são substituídas por consultas XPath e a *semântica* dos objetos está armazenada em descritores RDF. A Figura A.2 mostra a relação entre as tecnologias da Web Semântica e da Web tradicional com conceitos que elas representam.

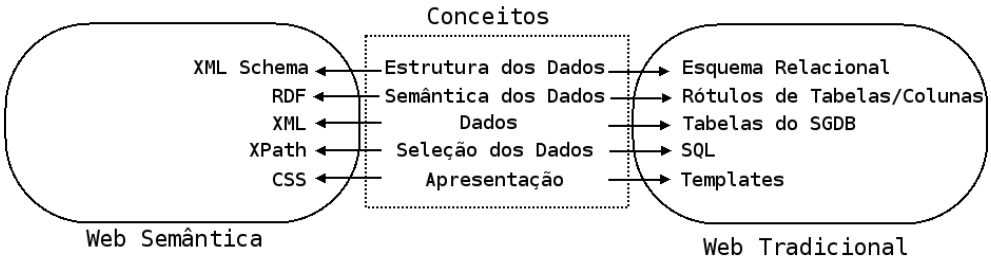

Figura A.2: Conceitos representados na Web Semântica e na Web tradicional

É importante ressaltar que a Figura A.2 é uma simplificação da relação entre os conceitos e as tecnologias. Na Web tradicional pode-se, por exemplo, encontrar dados distribuídos entre o banco de dados e a aplicação, a apresentação sendo gerada a partir de informação do banco de dados e, provavelmente, qualquer outra coisa que se puder pensar. Mas a simplificação mostrada é bastante confiável e serve, plenamente, aos propósitos de nossa discussão.

A partir dos conceitos acima, podemos descrever o problema de extração de dados da Web como, dado um sítio, descobrir os *dados* contidos em suas páginas, sua *estrutura* e, possivelmente, a *semântica* desses dados. Como vimos, essas informações correspondem aos arquivos XML, XML Schema e RDF na Web Semântica.

Se considerarmos que a informação de *apresentação* é irrelevante aos agentes de software da Web Semântica, e a *seleção* dos dados deve ficar a cargo dos próprios agentes, podemos então argumentar que a extração de dados da Web efetivamente satisfaz os requisitos necessários à infraestrutura da Web Semântica. Dadas essas considerações, definimos então os três objetivos da tarefa de extração:

- Obter os dados de um sítio da Web.
- Obter a estrutura dos os dados extraídos.
- Obter informação semântica para os dados extraídos.

Levando em consideração a definição acima, e observando os requisitos da Web Semântica, conclui-se então que o problema de extração de dados da Web e a criação da infra-estrutura da Web semântica são efetivamente o mesmo problema. Infelizmente, até o presente momento, não há nenhum trabalho na literatura capaz de resolvê-lo, em sua plenitude.

Referências Bibliográficas

[1] K. Ahnizeret, D. Fernandes, J. M. B. Cavalcanti, E. S. de Moura, & A. S. da Silva. Information retrieval aware web site modelling and generation. In *Proceedings of the 23th International Conference on Conceptual Modeling*, páginas 402–419, Xangai, China, 2004.

[2] D. T. Barnard, N. Duncan, & G. Clarke. Tree-to-tree correction for document trees. Technical Report 95, Queen's University, Ontario, Canada, 1995.

[3] R. Baumgartner, S. Flesca, & G. Gottlob. Visual web information extraction with lixto. In *Proceedings of the 27th International Conference on Very Large Data Bases*, páginas 119–128, Rome, Italia, 2001.

[4] T. Bernes-Lee. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. Harper San Francisco, San Francisco, 1999.

[5] P. Bille. Tree edit distance, alignment distance and inclusion. Technical Report TR-2003-23, University of Copenhagen, Copenhagen, Dinamarca, 2003.

[6] P. P. Calado, M. A. Gonçalves, E. A. Fox, B. Ribeiro-Neto, A. H. F. Laender, A. S. da Silva, D. C. Reis, P. A. Roberto, M. V. Vieira, & J. P. Lage. The Web-DL environment for building digital libraries from the web. In *Proceedings of the 3th ACM/IEEE-CS Joint Conference on Digital Libraries*, páginas 346–357, Washington, USA, 2003.

[7] S. S. Chawathe. Comparing hierarchical data in external memory. In *Proceedings of the 25th International Conference on Very Large Data Bases*, páginas 90–101, Edimburgh, Scotland, 1999.

[8] S. S. Chawathe & H. Garcia-Molina. An expressive model for comparing tree-structured data. Technical report, Standford University, San Francisco, USA, 1997.

[9] W. Chen. New algorithm for ordered tree-to-tree correction problem. *Journal of Algorithms*, 40 (2):135–158, 2001.

[10] W. W. Cohen & L. S. Jensen. A structured wrapper induction system for extracting information from semi-structured documents. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, Seattle, USA, 2001.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, & C. Stein. *Introduction to Algorithms, 2nd Edition*. The MIT Press/McGraw-Hill Book Company, Cambridge, Massachusetts, 2001.

[12] V. Crescenzi & G. Mecca. Automatic information extraction from large websites. *Journal of the ACM*, 51(5):731–779, 2004.

[13] V. Crescenzi, G. Mecca, & P. Merialdo. RoadRunner: Towards automatic data extraction from large Web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, páginas 109–118, Rome, Italia, 2001.

[14] A. S. da Silva. *Example-based Strategies for Extracting Semistructured Web Data*. Tese de Doutorado, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil, 2002.

[15] S. Dulucq & L. Tichit. RNA secondary structure comparison: exact analysis of the zhang–shasha tree edit algorithm. *Theoretical Computer Science*, 306(1-3):471–484, 2003.

[16] S. Dulucq & H. Touzet. Analysis of tree edit distance algorithms. In *Proceedings of the 14th Annual Symposium on Combinatorial Pattern Matching*, páginas 83–95, Morelia, México, 2003.

[17] R. Elmasri & S. B. Navathe. *Fundamentals of Database Systems, 2nd Edition.* Benjamin/Cummings, 1994.

[18] D. Florescu, A. Levy, & A. Mendelzon. Database techniques for the world-wide web: a survey. *SIGMOD Record*, 27(3):59–74, 1998.

[19] G. Grieser, K. P. Jantke, S. Lange, & B. Thomas. A unifying approach to HTML wrapper representation and learning. In *Proceedings of the 3th International Conference on Discovery Science*, páginas 50–64, London, England, 2000. Springer-Verlag.

[20] S. Grumbach & G. Mecca. In search of the lost schema. In *Proceedings of the 7th International Conference on Database Theory*, páginas 314–331, London, United Kingdom, 1999.

[21] A. Hogue & D. Karger. Thresher: Automating the unwrapping of semantic content from the world wide web. In *Proceedings of the 14th International Conference on World Wide Web*, Tokyo, Japan, 2005.

[22] C.-N. Hsu & C.-C. Chang. Finite-state transducers for semi-structured text mining. In *Proceedings of IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*, páginas 38–49, Stockholm, Sweden, 1999.

[23] C. Isert. The editing distance between trees. Technical report, Ferienakademie, Munich, Germany, 1999.

[24] J. Jansson & A. Lingas. A fast algorithm for optimal alignment between similar ordered trees. *Fundamenta Informaticae*, 56(1,2):105–120, 2003.

[25] T. Jiang, L. Wang, & K. Zhang. Alignment of trees - an alternative to tree edit. In *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, páginas 75–86, London, England, 1994.

[26] P. N. Klein. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms*, páginas 91–102, London, England, 1998.

[27] J. B. Kruskal. An overview of sequence comparison: time warps, string edits, and macromolecules. *SIAM Review*, 25(2):201–237, 1983.

[28] N. Kushmerick, D. S. Weld, & R. Doorenbos. Wrapper Induction for Information Extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, páginas 729–737, Osaka, Japan, 1997.

[29] A. H. F. Laender, B. Ribeiro-Neto, & A. S. da Silva. DEByE – Data Extraction by Example. *Data and Knowledge Engineering*, 40(2):121–154, 2002.

[30] A. H. F. Laender, B. Ribeiro-Neto, A. S. da Silva, & J. S. Teixeira. A Brief Survey of Web Data Extraction Tools. *SIGMOD Record*, 31(2):84–93, 2002.

[31] B. Liu, R. Grossman, & Y. Zhai. Mining data records in web pages. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining*, páginas 601–606, New York, USA, 2003.

[32] J.-K. Min, J.-Y. Ahn, & C.-W. Chung. Efficient extraction of schemas for XML documents. *Information Processing Letters*, 85(1):7–12, 2003.

[33] I. Muslea, S. Minton, & C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, 2001.

[34] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1): 31–88, 2001.

[35] A. Nierman & H. V. Jagadish. Evaluating structural similarity in XML documents. In *Proceedings of the 5th International Workshop on the Web and Databases*, Madison, USA, 2002.

[36] D. Quan, D. Huynh, & D. R. Karger. Haystack: A platform for authoring end user semantic web applications. In *Proceedings of the 2th International Semantic Web Conference*, Senibal Island, USA, 2003.

[37] D. C. Reis, R. B. Araújo, A. S. da Silva, & B. Ribeiro-Neto. A framework for generating attribute extractors for web data sources. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval*, páginas 210–226, Lisbon, Portugal, 2002. Springer-Verlag.

[38] D. C. Reis, P. B. Golgher, A. S. Silva, & A. H. F. Laender. Automatic web news extraction using tree edit distance. In *Proceedings of the 13th International Conference on World Wide Web*, páginas 502–511, New York, USA, 2004.

[39] T. Richter. A new measure of the distance between ordered trees and its applications. Technical Report 85166-CS, University of Bonn, Bonn, Germany, 1997.

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

[41] K. Tai. The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433, 1979.

[42] E. Tanaka & K. Tanaka. The tree-to-tree editing problem. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(2):221–240, 1988.

[43] G. Valiente. An efficient bottom-up distance between trees. In *Proceedings of the 8th International Symposium on String Processing and Information Retrieval*, páginas 212–219, Laguna de San Rafael, Chile, 2001.

[44] G. Valiente. *Algorithms on Trees and Graphs*. Springer-Verlag, Berlin, German, 2002.

[45] G. Valiente. Tree edit distance and common subtrees. Research Report LSI-02-20-R, Universitat Politècnica de Catalunya, Barcelona, Spain, 2002.

[46] R. A. Wagner & M. J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21 (1):168–173, 1974.

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

[48] J.-R. Wen, J.-Y. Nie, & H.-J. Zhang. Clustering user queries of a search engine. In *Proceedings of the 10th International Conference on World Wide Web*, páginas 162–168, New York, USA, 2001.

[49] P. Willett. Recent trends in hierarchic document clustering: a critical review. *Information Processing and Management*, 24(5):577–597, 1988.

[50] G. Yang, I. V. Ramakrishnan, & M. Kifer. On the complexity of schema inference from web pages in the presence of nullable data attributes. In *Proceedings of the 12th International Conference on Information and Knowledge Management*, páginas 224–231, New York, USA, 2003.

[51] W. Yang. Identifying syntactic differences between two programs. *Software Practice & Experience*, 21(7):739–755, 1991.

[52] Y. Zhai & B. Liu. Web data extraction based on partial tree alignment. In *Proceedings of the 14th International Conference on World Wide Web*, Tokyo, Japan, 2005.

[53] K. Zhang. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15(3): 205–222, 1996.

[54] K. Zhang & D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

[55] K. Zhang, D. Shasha, & J. T.-L. Wang. Approximate tree matching in the presence of variable length don't cares. *Journal of Algorithms*, 16(1):33–66, 1994.

[56] K. Zhang, R. Statman, & D. Shasha. On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3):133–139, 1992.